Preprint: Clip Art Rendering of Smooth Isosurfaces

Accepted pending revision for IEEE Transactions on Visualization and Computer Graphics
3rd Revision, submitted January, 2007

Matei Stroila, Elmar Eisemann and John C. Hart, *Member, IEEE*

***Abstract*— Clip art is a simplified illustration form consisting of layered filled polygons or closed curves used to convey 3-D shape information in a 2-D vector graphics format. This paper focuses on the problem of direct conversion of smooth surfaces, ranging from the free-form shapes of art and design to the mathematical structures of geometry and topology, into a clip art form suitable for illustration use in books, papers and presentations.**

We show how to represent silhouette, shadow, gleam and other surface feature curves as the intersection of implicit surfaces, and derive equations for their efficient interrogation via particle chains. We further describe how to sort, orient, identify and fill the closed regions that overlay to form clip art. We demonstrate the results with numerous renderings used to illustrate the paper itself.

***Index Terms*— particle systems, non-photorealistic rendering, line art drawing**

I. Introduction

COMPUTER GRAPHICS is largely the study of computational tools for visual communication, which often relies on rendering — the ability to display a computer representation of a geometric object. Photorealism focuses on physical modeling and simulation but does not always lead to better shape recognition [22]. Non-photorealistic and artistic rendering techniques instead utilize the knowledge of human perception, long understood by artists and designers, to automatically construct illustrations that convey an object's shape, often quite simply with minimal rendering primitives. Such abstraction is furthermore an important tool for comprehension and memorization [27].

This paper focuses on a specific style of NPR called *clip art*, by which we mean a small number of layered, filled polygons or closed curves, as shown in Fig. 1. Such clip art representations are commonly available in a wide variety of authoring tools for articles, books and presentations. Clip art libraries are generated by designers specially trained to use this medium to convey the essence of a shape despite its minimal geometry. This paper focuses on automating this process to directly create clipart renderings from smooth free-form surfaces.

We generate clip art by sampling surface feature curves in object space, connecting these samples into closed feature curves, and projecting and ordering these curved into the 2-D layered filled polygons or closed curves that form the clip-art output. This vector output format eases the scaling and placement of figures in a document, and is significantly less memory-consumptive than raster images.

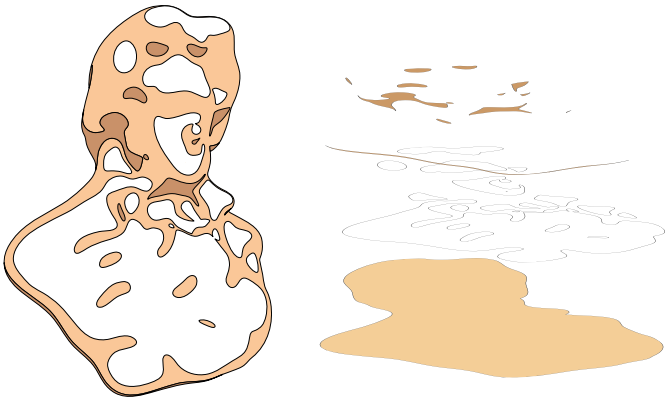

Fig. 1. An implicit Beethoven bust (left) converted into clip art consisting of layered, closed, 2-D curves (right) indicating silhouettes, shadows and highlights.

Our output appears similar to cartoon rendering which typically alters the shading of a rendered polygonal mesh to produce flat surfaces. Cartoon rendering often focuses on mesh shading, producing either a raster image or a 2-D polygon soup of reshaded mesh triangles. In either case such cartoon shading produces larger output files than our clip art approach. Section II further compares this clip art approach to other NPR methods.

Another alternative is to trace contours or feature lines in a rendered image, which is simpler and supports a wider variety of models (anything that can be rendered). However, the foreshortening inherit in any viewing projection can compress large surface regions near the silhouette, leading to numerical imprecision, raster artifacts and noise along the very feature curves most important to effective illustration. As demonstrated in Fig. 2, tracing contours in object space produces watertight region boundaries whose resolution is limited only by the number of feature-curve samples used (which grows linearly as the image resolution increases). Though we do not pursue this further here, these object space contours better facilitate further output stylization than those extracted from a rendered image.

We sample surface feature curves with a a surface-constrained particle system [28]. Previous particle-based NPR approaches [13], [25] tracked each feature curve as surface-

Dept. of Computer Science, University of Illinois, Urbana-Champaign
M. Stroila is currently at NAVTEQ and can be contacted at matei.stroila@gmail.com
E. Eisemann is a student at ARTIS-GRAVIR/IMAG-INRIA and can be contacted at elmar.eisemann@inrialpes.fr. ARTIS is a team of the GRAVIR/IMAG laboratory, a joint effort of CNRS, INRIA, INPG and UJF.

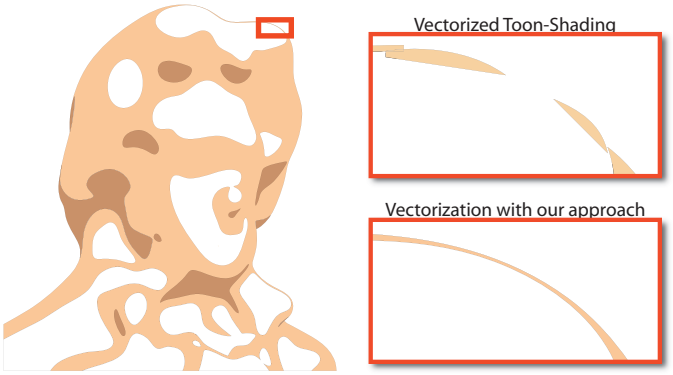

Fig. 2. Contouring a rendered image (left) can be inaccurate near the silhouette whereas tracking feature curves in 3-D (right) more accurately tracks contours.

surface intersection (SSI), using two simultaneous but independent particle-on-surface constraints. Section III shows that these independent constraints can generate particles in equilibria away from both surfaces, and devises a new two-surface particle constraint that correctly samples these SSI feature curves. While this constraint-based approach is not guaranteed to accurately detect and trace all contours, it works well enough to produce illustrations.

Using this new SSI particle constraint, Section IV reviews how various surface feature curves including silhouettes, shadows[1], gleams and parabolic points can be represented as the intersection of implicit surfaces. It also extends this SSI approach to include suggestive contours, and introduces a simpler suggestive contour approximation justified with visual comparison.

These feature curve particles connect to form closed chain loops which project to delineate view-plane polygonal regions. Section V shows how to carefully manage the curve orientation and visibility of these polygonal regions to produce the layered filled polygons or closed curves that form the vector graphics format of clip art output.

Section VI shows that the system runs fast enough to render simple clip art (consisting of thousands of curve sample points) in real time, which allows the user to orient objects and position lighting to achieve an effective illustration. Section VII concludes with ideas for further research.

II. Related Work

Our work focuses on the direct non-photorealistic rendering of the isosurfaces of smooth (C^3) functions by extracting various closed contour loops to generate layers of filled 2-D image curves. This work builds on the wealth of existing approaches to NPR, many of which are summarized in surveys [1], [19].

[1]These shadow contours divide regions facing toward and away from a light source. We do not yet consider the detection and generation of cast shadow contours.

A. Smooth Surface Illustration

Hertzmann and Zorin [17] also studied the illustration of smooth surfaces, but instead of general isosurfaces they applied a smooth interpolant to an input surface mesh that cleverly caused the contour generator to avoid vertices that lead to some degenerate cases they demonstrated. Elber [10] likewise relied on a polygonal approximation of a smooth isosurface to construct an initial uniform distribution of points that were projected onto the isosurface via gradient flow. These points were then swept into strokes along the surface to generate line-art textured renderings of the smooth isosurface. Burns *et al.* [6] track silhouettes of volumetric isosurfaces, capitalizing on the voxel grid's fixed resolution for an efficient step size, and using a stochastic neighborhood test to detect silhouettes. Our work focuses on the direct extraction of closed silhouette and feature curves from a smooth implicit surface, though some of our results are demonstrated on implicit surfaces constructed from the embedding of a surface mesh, using a B-spline space interpolant.

B. Direct Isosurface Illustration

Bremer and Hughes [5] directly tracked the contour generator of an implicit surface via predictor-corrector Euler integration of its tangent vector (a so-called "marching" method), seeded by intersecting a random interior ray to find the surface which was then traversed in a random direction to find the silhouette. Foster *et al.* [13] similarly traced isosurface contours using seed points found by the surface constrained "floater" particles of Witkin and Heckbert [28]. Su and Hart [25] reprogrammed these "floater" particles to adhere to silhouettes, by constraining them to both the implicit surface and the view-tangency isosurface, but when computed independently, these dual constraints can counteract each other leading to errors. Section III derives the correct two-surface constraint.

C. Surface-Surface Intersection

As mentioned in the last paragraph, our surface illustrations are built on the correct tracking of the intersection of two implicit surfaces. Surface-surface intersection is a classic problem of computer-aided geometric design attacked with a wide array of approaches [18]. Because of the difficulty in their surface navigation, implicit-implicit intersections are often computed through space subdivision (a "lattice" approach). For example, Suffern & Balsys [26] use interval subdivision to extract and plot silhouette and other surface intersection contours on implicit surfaces. Plantinga and Vegter [24] use interval arithmetic to trace the contour generator with a dynamic step size, thus guaranteing its correct topology. Our intersection contour tracking using chained dual-surface "floater" particles is more similar to a marching approach, though one whose sample points evolve in parallel instead of in serial. In particular our particle tracking of contours is based entirely on geometry and, unlike Plantinga and Vegter [24], is not guaranteed to find and properly trace all contours, especially near visual events.

D. Dynamic Surface Illustration

Many techniques for real-time extraction of silhouettes recompute their projection from scratch [23]. Kalnins *et al.* [20] studied the coherence of stylized silhouettes, using the previous silhouettes to seed a new extraction of the next frame's silhouettes. The particles we use to track the silhouette work similarly for dynamic surfaces or views. We do not yet explore the detection and correction of contour topology, and rely on the purely geometric algorithm used to connect particles into a closed contour to reconnect contours after a visual event.

III. Curve Adhesion

In this section, we present our extension of the surface adhesion behavior of Witkin and Heckbert [28] to a curve adhesion behavior.

The condition that n moving particles, $\mathbf{x}_i(t)$ lie on the intersection curve of two smooth surfaces defined implicitly by $F=0$ and $G=0$ is given by

$$F_i(t) \equiv F(\mathbf{x}_i(t)) = 0, \tag{1}$$
$$G_i(t) \equiv G(\mathbf{x}_i(t)) = 0. \tag{2}$$

Differentiating with respect to time yields $2n$ linear dynamic constraint equations

$$C_{Fi}(\mathbf{v}_i) \equiv \nabla F_i^T \mathbf{v}_i + \phi F_i = 0, \tag{3}$$
$$C_{Gi}(\mathbf{v}_i) \equiv \nabla G_i^T \mathbf{v}_i + \phi G_i = 0. \tag{4}$$

of the particle velocities $\mathbf{v}_i = d\mathbf{x}_i/dt$, where the additional "feedback" terms (scaled by constant ϕ) are added for numerical stability to bring particles $\mathbf{x}_i$ back to the zero surface if they stray.

In fact it is precisely this feedback term that causes particles to lie off the intersection curve (between curves) when using two independent particle-on-surface dynamic constraints. A dynamic constraint removes forces that would move a particle off its current isovalue contour, so a particle's implicit surface function value remains constant, though not necessarily the desired value of zero in this case. These "feedback" terms are spring-like penalty constraints that nudge particles to a desired isovalue, whereas the dynamic constraint keeps them there. Particles can thus fall between isovalue contours when opposing feedback terms cancel.

Particle curve adhesion is then achieved by solving the following constrained optimization problem

$$\mathbf{v} = \arg\min \left\{ \frac{1}{2}\sum_{i=1}^{n} |\mathbf{v}_i - \mathbf{V}_i|^2 \;\middle|\; \begin{array}{l} C_{Fi}(\mathbf{v}_i) = 0 \\ C_{Gi}(\mathbf{v}_i) = 0 \end{array}, \forall i = \overline{1,n} \right\}. \tag{5}$$

where $\mathbf{V}_i$ are the desired particle velocities, set to a mutual particle repulsion force to evenly distribute the particles [28].

Following the Lagrange multipliers method, the gradient of the objective function with respect to the minimizing variables $\{\mathbf{v}_i\}$, should be equal to a linear combination of the gradients of the constrain functions

$$\mathbf{v}_i - \mathbf{V}_i + \lambda_i \nabla F_i + \mu_i \nabla G_i = 0. \tag{6}$$

Substituting the velocities computed from (6) into (3) and (4), we obtain the system

$$\lambda_i |\nabla F_i|^2 + \mu_i \nabla F_i^T \nabla G_i = C_{Fi}(\mathbf{V}_i), \tag{7}$$
$$\lambda_i \nabla G_i^T \nabla F_i + \mu_i |\nabla G_i|^2 = C_{Gi}(\mathbf{V}_i). \tag{8}$$

Then, we compute the Lagrangian multipliers λ_i and μ_i from the above system of equations and plug them in the following velocity update equations, which rewrite (6) as

$$\mathbf{v}_i = \mathbf{V}_i - \lambda_i \nabla F_i - \mu_i \nabla G_i. \tag{9}$$

Working out the details, it follows that if the two surfaces are not tangent at $\mathbf{x}_i$ then the linear system for the Lagrange multipliers has solutions

$$\lambda_i = \left(\frac{C_{Gi}(\mathbf{V}_i)}{|\nabla G_i|^2} - \frac{C_{Fi}(\mathbf{V}_i)}{\nabla F_i^T \nabla G_i} \right) \Big/ \left(\frac{\nabla F_i^T \nabla G_i}{|\nabla G_i|^2} - \frac{|\nabla F_i|^2}{\nabla F_i^T \nabla G_i} \right),$$
$$\mu_i = \left(\frac{C_{Fi}(\mathbf{V}_i)}{|\nabla F_i|^2} - \frac{C_{Gi}(\mathbf{V}_i)}{\nabla F_i^T \nabla G_i} \right) \Big/ \left(\frac{\nabla F_i^T \nabla G_i}{|\nabla F_i|^2} - \frac{|\nabla G_i|^2}{\nabla F_i^T \nabla G_i} \right).$$

An important remark is that the Lagrange multipliers λ_i and μ_i are not just $C_{Fi}(\mathbf{V}_i)/|\nabla F_i|^2$ and $C_{Gi}(\mathbf{V}_i)/|\nabla G_i|^2$. These would be the expressions obtained by solving separately the two surfaces' adhesions, which lack the interacting terms containing the dot product $\nabla F_i^T \nabla G_i$. Therefore, by only superposing two surface adhesion solutions, the particles would not converge to the correct curve. Figure 3 depicts this situation. Realize in particular that the surface in this example is relatively simple and smooth.

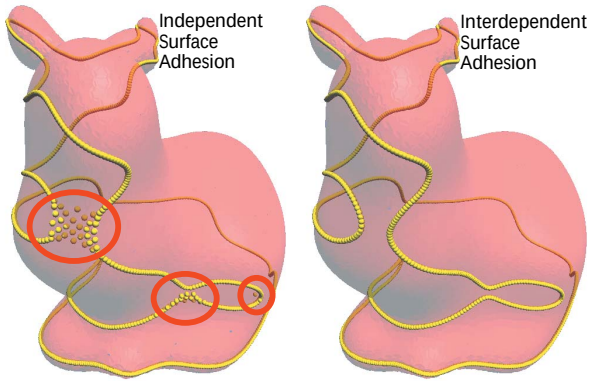

Fig. 3. Particles separately constrained to two surfaces (here the shown implicit surface and the implied shadow-contour surface) can find off-surface equalibria (left). A two-surface simultaneous dynamic constraint contain additional terms such that particles correctly adhere only to the surface-surface intersection curve (right).

IV. Feature Contours

Many of the feature curves on implicit surfaces used in computer graphics can be expressed as the intersection of the implicit surface with a second implicit surface. For example, an implicit shape's silhouette in object space (more formally its contour generator) is the curve found at the intersection of the shape's implicit surface and the implicit surface formed by the dot product of the shape function's gradient and the view vector. This section defines such implicit intersection curves

for silhouettes, shadows, parabolic points, suggestive contours and gleams.

We start with a few basic definitions and notations. Let $\mathbf{x} \in \mathbb{R}^3$ be the position vector of an arbitrary point located on the implicit surface $S = F^{-1}(0)$, for a smooth function $F : \mathbb{R}^3 \to \mathbb{R}$. We assume that 0 is a regular value of F, so that the surface S is smooth. We also assume that the function F has C^3 differentiability (the third order partial derivatives exist and are continuous).

In order to support both orthographic and perspective view projections, and both point and directional light sources, we let $\mathbf{c} = (c_1, c_2, c_3, c_h) = (\vec{c}, c_h)$ be the homogeneous coordinates of the position vector of the camera (the viewpoint), and let $\mathbf{l} = (\vec{l}, l_h)$ be the homogeneous position vector of the light source. We denote by $\vec{v}_c = \vec{c} - c_h\mathbf{x}$ the view direction vector, and by $\vec{v}_l = \vec{l} - l_h\mathbf{x}$ the light direction vector.

A. Silhouette Contours

A point $\mathbf{x} \in S$ is on the *contour generator* of a smooth surface if the view vector is tangent to the surface at $\mathbf{x}$, or equivalently, it is orthogonal to the surface normal at $\mathbf{x}$. For an implicit surface $F^{-1}(0)$, the surface normal direction is the support line of the gradient $\nabla F(\mathbf{x})$. The contour generator can hence be regarded as the curve given by the intersection of the implicit surface with the *silhouette surface*, which is defined implicitly as the zeroset of

$$F_{sil} = \nabla F(\mathbf{x})^T \vec{v}_c. \tag{10}$$

The normal of the silhouette surface is given by the gradient

$$\nabla(F_{sil}) = \nabla(\nabla F(\mathbf{x})^T \vec{v}_c) = \mathrm{H}F(\mathbf{x})\vec{v}_c - c_h \nabla F(\mathbf{x}) \tag{11}$$

where $\mathrm{H}F(\mathbf{x})$ is the Hessian matrix of F at $\mathbf{x}$. A silhouette surface for a torus is demonstrated in Figure 4.

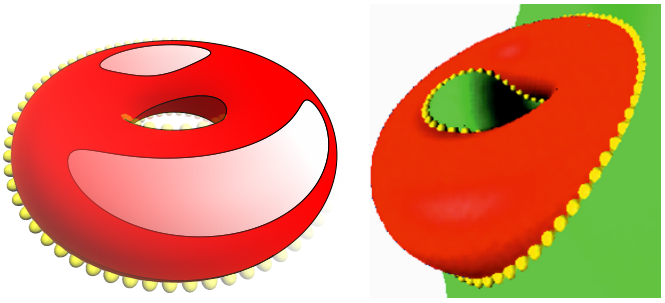

Fig. 4. The apparent contour (yellow) of the view along vector $\mathbf{v}$ (left) is the projection of the contour generator (yellow) shown from a side-view (right) at the intersection of the implicit surface $F = 0$ (red) and the (implicit) silhouette surface $\nabla F \cdot \mathbf{v} = 0$ (green).

B. Shadow Contours and Isophotes

The *shadow contour* of a free-form object is defined as the set of points on the object surface where the surface normal is perpendicular to the vector from the light source. As such, we have to use the light position vector instead of the camera position vector in all the formulas of the previous section.

As in the contour generator case, the shadow can be regarded as the curve defined by the intersection of the implicit surface with the *shadow surface*, which is as well implicitly defined as the zeroset of

$$F_{shad} = \nabla F(\mathbf{r})^T \vec{v}_l. \tag{12}$$

Such shadow curves are demonstrated in Figure 5.

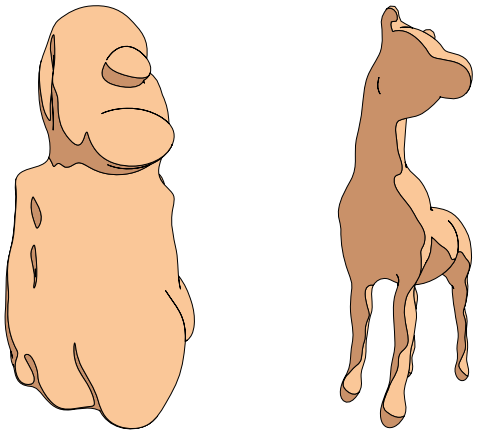

Fig. 5. Shadow and silhouette contours combined with shading on Moai (left) and Horse (right).

The shadow curve definition can also be extended to a family of isophote curves defined by different fixed acute angles between the surface normal and the light vector. This can be seen as a quantized Lambertian reflectance, as shown in Fig. 6.

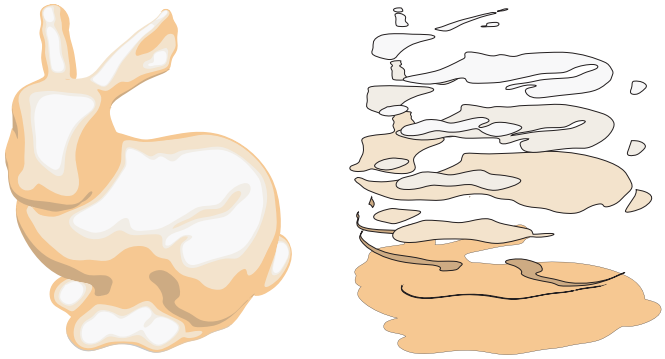

Fig. 6. An interpolated volumetric Stanford bunny, displayed as clip art consisting of isophotes of constant diffuse reflectance (left) stacked from individual isophote layers (right).

C. Gleams and Reflection Lines

Gleams (or more precisely specular highlights) are intended to be the reflections of light sources on the surface of an object, though graphics tends to use a reflectance approximation which yields round specular lobes from a point light source. Specular reflections are an important visual cue as they reveal details in a given shape, and designers commonly use *reflection lines* resulting from the reflection of light strips to assess the aesthetic quality of a shape, e.g. a car body.

For rendering such gleams and reflection lines in "clip art" style illustrations, we describe the specular contour as the boundary of the regions where the strength of the specular

reflection of light lies above a certain threshold. Following Blinn's specular reflectance formulation [3], let the *halfway* vector be $\mathbf{h} = \vec{v}_l/||\vec{v}_l|| + \vec{v}_c/||\vec{v}_c||$ and $\mathbf{n}$ be the normal vector to the surface at $\mathbf{x}$. Then, the specular contour is the set of points on the surface where $\mathbf{h}$ and $\mathbf{n}$ form a fixed angle α

$$C_{spec} = \{\mathbf{x} \in S | \angle(\mathbf{h}, \mathbf{n}) = \alpha\}. \tag{13}$$

The specular contour is then the intersection of the implicit surface with the specular surface, with the latter defined as the zeroset of the function

$$F_{spec} = \nabla F(\mathbf{x})^T \mathbf{h} - \arccos(\alpha) \|\nabla F(\mathbf{x})\| \|\mathbf{h}\|. \tag{14}$$

Figure 7 demonstrates these specular contours.

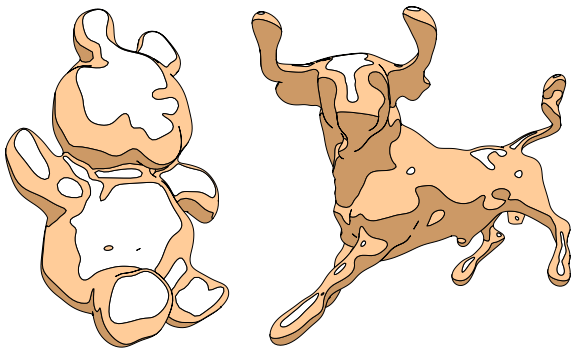

Fig. 7. Specularities convey important shape information in flat shaded areas. The sketchy appearance of the teddy (left) is still apparent in the clip art. The bull (right) contains lots of details that are faithfully transmitted, nevertheless the cornes have been smoothed as they would represent singularities.

D. Suggestive Contours

DeCarlo *et al.* [9] extended the apparent contour (the visible projection of the contour generator) to include additional shape cue feature curves called *suggestive contours*. These suggestive contours are built on the concept of radial curvature as described by Koenderink [21]. The *radial curvature* at a point $\mathbf{x}$ is the normal curvature of the surface in the direction given by the projection of the view vector, $\vec{v}_c$, onto the tangent plane at $\mathbf{x}$. We denote this projection vector by $\mathbf{w}$. The radial curvature is view dependent and undefined whenever $\vec{v}_c$ is parallel to $\mathbf{n}$.

For an implicit surface, the radial curvature at the point $\mathbf{x}$ is

$$k_r(\mathbf{x}) = \frac{1}{|\mathbf{w}|^2} \mathbf{w}^T \, \mathrm{H}F(\mathbf{x}) \mathbf{w}. \tag{15}$$

The suggestive contour is the set of points on the surface at which the radial curvature k_r is 0, and the directional derivative of k_r in the direction of $\mathbf{w}$ is positive. Thus, at a point of the suggestive contour, $\mathbf{w}$ points in one of the asymptotic directions of the surface (zero curvature directions).

The set of points on the surface at which the radial curvature k_r is 0 forms a set of closed loops on the surface. These loops are cut at the points where the view vector is tangent to the surface.

The *apparent contour* is defined as the visible portions of the contour generator projected onto the image plane. When viewed from one of its tangent directions, the contour generator forms a cusp and the apparent contour stops in an *ending contour*. Consequently, here the radial curvature is zero implying that the apparent contours and the suggestive contours meet. Suggestive contours and apparent contours line up with geometric tangent continuity in the image plane.

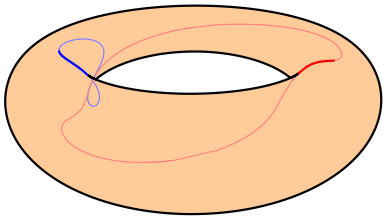

Fig. 8. Suggestive contours meet the visible contour tangentially, extending it to provide additional surface cues. The right suggestive contour (red, right) occurs where radial curvature is zero. The left suggestive contour (blue, left) approximates and simplifies radial curvature by not projecting the view vector onto the surface tangent plane. (A right "blue" curve and left "red" curve have been removed for illustration).

Although in general the radial curvature is defined only at the surface's points, for implicit surfaces, $k_r(\mathbf{x})$ can be regarded as a real function over three-space. Hence, the suggestive contour can be determined by intersecting the implicit surface with the radial curvature zeroset, restricting the result to points where the $\mathbf{w}$-directional derivative of k_r is positive.

The application of this definition of suggestive contours to implicit surface particle dynamics is costly. The gradient of radial curvature is quite complex,

$$\left.\begin{aligned} \nabla k_r(\mathbf{x}) = \tfrac{2\mathbf{w}^T \mathrm{H}F\mathbf{w}}{|\mathbf{w}|^4} \Big(\mathbf{w} - 2\tfrac{(\nabla F^T \vec{v}_c)(\nabla F^T \mathbf{w}) \mathrm{H}F \nabla F}{|\nabla F|^4} + \\ \tfrac{(\nabla F^T \mathbf{w})(\mathrm{H}F\vec{v}_c - \nabla F) + (\nabla F^T \vec{v}_c)\mathrm{H}F\mathbf{w}}{|\nabla F|^2} \Big) + \\ \tfrac{1}{|\mathbf{w}|^2} \Big(\partial^3_{\cdot,i,j} F w^i w^j - \\ 2\tfrac{((\mathrm{H}F\mathbf{w})^T \nabla F)(\mathrm{H}F\vec{v}_c - \nabla F) + (\nabla F^T \vec{v}_c)\mathrm{H}F\mathrm{H}F\mathbf{w}}{|\nabla F|^2} - \\ 2\mathrm{H}F\mathbf{w} + 4\tfrac{(\nabla F^T \vec{v}_c)((\mathrm{H}F\mathbf{w})^T \nabla F)\mathrm{H}F\nabla F}{|\nabla F|^4} \Big). \end{aligned}\right\} \tag{16}$$

We instead present a different definition for suggestive contours that is simpler with similar visual results. In particular the contours still align with the ending contours. In fact for any curve on the surface that crosses a cusp, its projection has a tangent that aligns with the tangent of the apparent contour at the intersection point [9]. The only exception occurs when the curve locally projects to a point, that is to say, the tangent of the space curve aligns with the view direction.

Instead of applying the Hessian to the projection of the view vector onto the tangent plane, $\mathbf{w}$, we apply it directly to the view vector. We construct a simpler suggestive contour surface as the zeroset of

$$F_{sug}(\mathbf{x}) = \vec{v}_c^T \, \mathrm{H}F(\mathbf{x}) \vec{v}_c \tag{17}$$

but only display its portions where

$$\nabla F_{sug}(\mathbf{x})^T \vec{v}_c > 0 \tag{18}$$

(which could be construed as a half-space whose boundary forms yet a third implicit surface).

The gradient of this newly defined implicit function takes the much simpler form (shown componentwise)

$$\partial_k F_{sug}(\mathbf{x}) = \sum_{i,j} (\partial^3_{k,i,j} F(\mathbf{x}) v^i_c v^j_c) - 2(\mathrm{H}F(\mathbf{x})\vec{v}_c)_k. \qquad (19)$$

The similarity to the original definition also comes from the fact that in the ending contour the two curves actually coincide because the view direction lies in the tangent plane and thus the missing projection has no influence. These simpler suggestive contours are demonstrated in Figures 9 and 10.

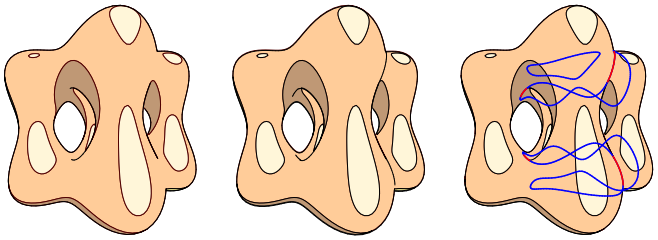

Fig. 9. Algebraic hollow cube (left) with suggestive contours (middle, dotted) which are segments of our simplified radial curvature contours (right).

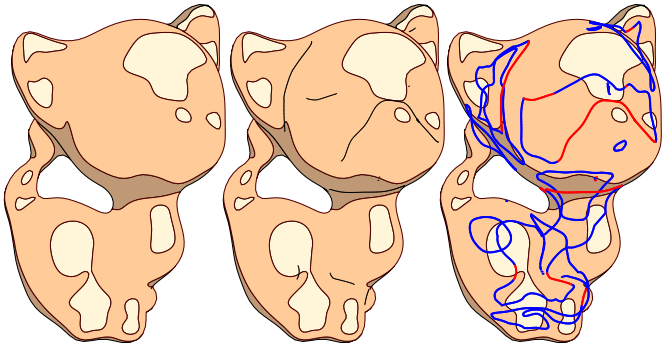

Fig. 10. Kitty without (left) and with suggestive contours (middle) generated from simplified radial curvature contours (right).

E. Parabolic Contours

The parabolic contours consist in surface points where the Gaussian curvature is zero. They can also be regarded as intersections of implicit surfaces using the Gaussian curvature formula for an implicit surface, which is defined over the entire embedding space [14]

$$k_G(\mathbf{x}) = \frac{\nabla F(\mathbf{x})^T \mathrm{H}F(\mathbf{x})^* \nabla F(\mathbf{x})}{\|\nabla F(\mathbf{x})\|^4} \qquad (20)$$

where $\mathrm{H}F(\mathbf{x})^*$ is the adjoint of the Hessian matrix (the adjoint of a matrix is the transpose of its cofactor matrix). Because we are interested in its zeroset, we can focus on the numerator of this expression which makes the calculation simpler and more stable.

The parabolic contours contain the critical points of the silhouette and shadow contours and so indicate where they change topology under different viewing/lighting conditions. Figure 13 demonstrates such a change in the shadow contours of an illuminated squash.

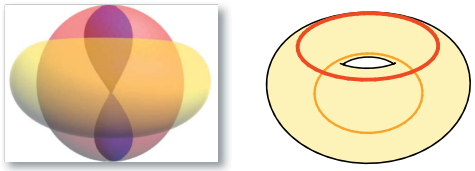

Fig. 11. A surface of parabolic points defined as the zeroset of an implicit Gaussian curvature formula that for the torus is a sphere surrounding an hourglass (left). Only the sphere intersects the torus, yielding two circles (right).

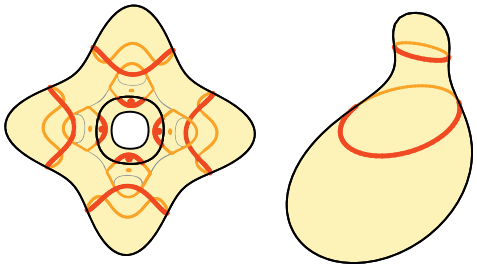

Fig. 12. Silhouette (black) and parabolic contours (red) on the hollow cube (left) and squash (right).

V. Constructing Clip Art

The clip art construction process, diagrammed in Fig. 14 consists of converting a shape specification using implicit surfaces into a depth-ordered list of layered closed filled curves in a format that can be displayed by any of a variety of vector graphics renderers. Our approach to clip art construction relies on the Witkin-Heckbert method of sampling and displaying an implicitly defined shape using self-organizing particles constrained to the implicit surface [28]. We use a fully programmable version of this framework [25] to implement new behaviors to interrogate the feature curves needed to convert implicit surfaces into clip art closed loops.

We begin with an implementation of "floater" particles [28], altered to sample space curves instead of surfaces. One of these alterations replaces the surface adhesion behavior that keeps particles on an implicit surface with the surface-intersection curve adhesion behavior described in Section III. We also alter the particle repulsion behavior, ordinarily used to distribute particles evenly over a surface, to achieve a force equilibrium using only two "curve" neighbors instead of six "surface" neighbors when particles settle on the surface. As is standard, we also use a grid spatial data structure to accelerate particle neighborhood queries [16].

In addition to simple repulsion, these curve particles need to properly connect with their neighbors to form closed loops.

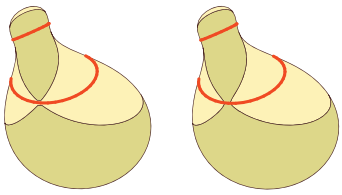

Fig. 13. Shadow contours on the squash passing through a critical point. The critical points of the shadow contours are also parabolic points.

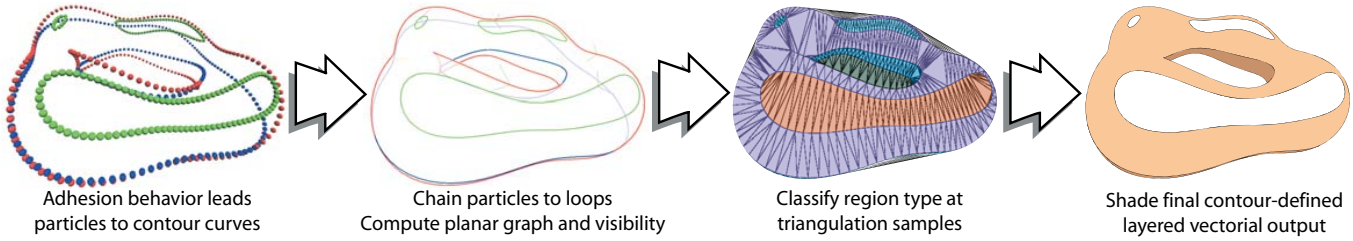

Fig. 14. We convert an implicit surface specified by a function into clipart specified by layered filled 2-D polygons by using particles to find surface feature curves and determining how those closed spacecurves should be layered and shaded when projected to 2-D.

Previous implementations simply connect a particle with its two nearest neighbors which works well most of the time but can prove unstable in some situations such as sharp turns. We instead rely on the differential geometry of the curve, constructing a tangent vector from the cross product of the two implicit surfaces whose intersection defines the curve. We use this information to better predict a particle's two neighbors, which have to lie in different half-spaces, as defined by the tangent.

Given one or more closed space curves, we then need to project them into simple closed curves in the image plane. We first project the space curves into possibly overlapping image curves. These intersections must be transverse and we use perturbations to avoid tangential crossings when they occur. We then use the CGAL computational geometry library [4] to construct an *arrangement* of intersecting curves [12]. This arrangement is a planar graph where each vertex corresponds to an intersection point of the projected curves, and in our case, each edge maintains an ordered list of particles on the corresponding curve segment connecting the intersection points. This arrangement allows the overlapping space curves to be decomposed into simple closed image curves joined at the intersection points.

The remainder of this section focuses on the ordering and shading of these simple closed image curves, to produce a direct clip art depiction of the implicit surface.

A. Visibility

Given a collection of closed image curves depicting closed space curves on an implicit surface, we first need to identify the visible segments of the image curves. For particles other than those on the contour generator, we use the surface normal to cheaply identify back facing curve segments. For the remaining curve segments, we determine the visibility of an individual particle by casting a single ray intersected against the implicit surface. Visibility can also be efficiently propagated along the curve segment.

We propagate visibility along the contour generator using Appel's hidden line algorithm [2] originally derived for polygonal meshes and more robustly updated for smooth surfaces [23], parameteric surfaces [11], and implicit surfaces [5]. We similarly extend Appel's algorithm to implicit surfaces and this section shows how to march along the particle chains and use their relationship with the implicit surface's differential geometry to overcome otherwise difficult cases. In particular, we use the depth order and radial curvatures of the two particles surrounding a cusp to precisely determine the change in quantitative invisibility. These extensions are necessary because visibility cannot be propagated in the exact same way as it is for polygonal meshes. (This is why Burns et al. [6] rely on a cast ray per silhouette voxel.)

We seek to decompose the curve into segments of constant quantitative invisibility. In the case of closed loops, this is guaranteed if we do not cross behind other silhouette edges or the radial curvature does not change sign [17].

For each loop in the arrangement graph mentioned earlier, we apply the following procedure. First, we pick randomly one of its edges and assign a *relative QI* of zero. (We don't know the actual quantitative invisibility yet, but we need a reference). Then, we propagate the *QI* along the loop edge by edge, changing it only when we encounter a vertex of higher degree (global occlusion) or when its radial curvature changes sign (local occlusion).

For global occlusion, we can assume we look down the z-axis and that the vertex in the planar graph corresponds to a point P on a contour loop for which we want to propagate visibility. All global intersections that have an influence on the *QI* have to be situated between P and the viewpoint. If the curve overlapping P is ordinary (no cusp), then the *QI* changes by two (intuitively, due to the front- and back-facing part of the interloping surface). If the normal of the interloping silhouette agrees with the orientation of our current loop, we decrement the *QI* by two, otherwise we increment. In practice this normal comes from the two closest particles on the occluding silhouette.

For a global occlusion where the occluding silhouette contains a cusp that overlaps our current loop, the *QI* remains unchanged, unless the current loop is tangent to the cusp, and in this rare case the *QI* changes by two.

Local occlusions occur at cusps, where the radial curvature changes sign. We use the depth of the sample particles near the cusp to determine its effect on the *QI*. Let z_i and z_{i+1} be the depths of the two particles (ordered from the loop orientation) on each side of the cusp. If $z_i < z_{i+1}$ then we increment *QI*, otherwise we decrement it. This relation between radial curvature and the effect of a cusp on *QI* can be proved as follows. Intuitively, within a neighborhood of a silhouette point, the surface is locally above or below the view rays depending on the radial curvature sign. This restriction of the surface flips at a cusp which signals an increase or decrease in depth complexity.

More rigorously, since the surface is smooth there exists a

surface neighborhood B of the cusp diffeomorphic to $\mathbb{R}^2$ that maps the contour generator to the x-axis. Let C be any smooth surface curve in this neighborhood that (1) connects a point N on the near silhouette branch in front of the cusp and a point F on the far silhouette branch behind cusp, (2) maps via the diffeomorphism to the $y > 0$ region except at its endpoints where $y = 0$ (C does not cross the contour generator), and (3) there exist surface neighborhoods $B_N, B_F \subset B$ about N and F whose intersections with the radial planes at those points coincides with C.

Restating Lemma 2.3 of Elber and Cohen [11] a surface curve that does not intersect the contour generator has constant *local QI*. Such is the case for the interior of C.

Assume $\kappa_r > 0$ at N and let point $P \in C \cap B_N$. If $P = N$ (the only silhouette point in $C \cap B_N$) its (relative) $QI = 0$ (it is visible) locally in this neighborhood. For points close to N the local $QI = 1$ because of $\kappa_r > 0$. Locally the curve remains inside of the face of the local arrangement, which implies that the QI remains constant in the interior of C (by the above lemma). Locally we find that the QI of F equals the QI of points nearby in $C \cap B_F$ (this time due to $\kappa_r < 0$). We conclude that the local QI for F has to be 1. The proof for $\kappa_r < 0$ follows similarly.

Once we have obtained the relative QI for the curve, we ground it to an absolute QI through a single ray query. We choose to perform this ray query on a point on the segment with the largest (relative) QI.

B. Shading

While the apparent contour outlines the object, we still need to fill in these outlines to visually disambiguate regions especially among complex combinations of specularities and shadows. While one could output triangles in a vector graphics format, they increase the size of the output with non intuitive components.

We instead seek to output clip art consisting of layers of filled closed polygons defined by projections of the visible portions of various surface contours. This section focuses on identifying these regions and constructing consistently oriented planar polygons.

Given a consistently oriented contour generator (based on the tangent) on a smooth surface, one can correctly fill its apparent contour using its winding number. This simplicity does not extend to the filling of shadows, specularities and other closed surface regions. Figure 15 demonstrates a scene where complete knowledge of all contour information does not help in identifying the proper fill regions. The grey dashed lines indicate two different configurations that lead to identical visible silhouette and shadow contours.

As long as no occluding contours occur, the contours' orientation plus the particles' visibility suffice to shade shadows (and specularities) correctly. One problem that arises is, that surface curves align with silhouettes making the construction of the planar graph more difficult. Nevertheless these intersections occur at visibility events. For a correct planar graph we can connect the last visible particle to the closest silhouette and rely on ray tracing in the case of doubt. We orient the

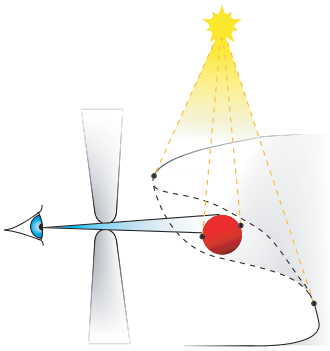

Fig. 15. Contour curves and visibility information are insufficient to deduce shading. The dashed lines indicate two possible surfaces that share the same contour lines (silhouette and shadow) but one hides the small sphere whereas the other does not.

apparent contour and the visible surface contours such that their interiors lie on the left. Shadowed areas are then indicated by resulting cycles (shadow in the bottom right of Figure 16). In the presence of occluding contours this approach fails (shadow in the middle of the same figure).

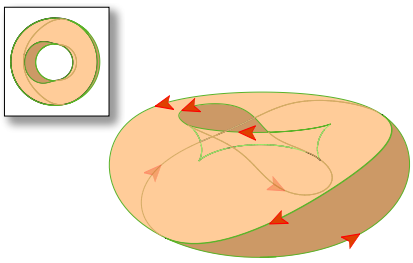

Fig. 16. Correctly oriented silhouette and shadow contours form closed cycles in the planar graph as long as no occluding contours appear.

Still each face of the planar graph of visible contours can be categorized (e.g. gleam, illuminated, shadowed or void). We can determine the category by casting a single eye ray that pierces each region, though we find multiple piercing eye rays help avoid occasional errors due to numerical instabilities in tight regions such as near the apparent contour. These region piercing rays should not pass too closely to region boundaries and we explored several approaches to choose these rays: random sampling, skeletal points and the barycenters of a constrained Delaunay triangulation. The latter best balanced speed and accuracy, and consistently classified over 95% of the regions in our experiments.

VI. Results

We implemented our particle-based contour illustration system using the Wickbert surface and particles system library (`http://www.uiuc.edu/goto/wickbert`), a user-extensible open-source cross-platform C++ library [25] based on the surface constrained particle systems of Witkin and Heckbert [28]. Wickbert implements a framework for programming particle systems using building-block interchangeable attributes, behaviors and shaders. Curve adhesion was implemented as a new particle behavior, and new particle

shaders were devised to organize and display closed polyline curves.

Using this system, we can manipulate (zoom, rotate) quartic surfaces in real-time using up to 4000 particles for feature contours rendering, on an Intel T2500 2.0GHz processor with 2.00GB of memory.

Timings Table (System: Intel T2500 2.0GHz, 2.00 GB of RAM)									
Model	Particles							Clip art	
	Cont. Type	# Particles	Implicit	Visibility	Repulsion	Adhesion	Total	Plan. Map	CDT
			ms	ms	ms	ms	ms	s	s
Hollow Cube (Algebraic)	Sil.	531	1.09	4.49	1.54	0.09	7.21	1.92	0.13
	Shad.	529	1.08	2.10	1.48	0.10	4.76		
	Spec.	246	0.50	2.40	0.44	0.06	3.40		
	Sugg.	390	1.01	3.31	0.94	0.98	6.24		
Beethoven Bust (Volumetric)	Sil.	1010	33.00	41.00	4.97	0.19	79.16	6.92	0.42
	Shad.	1197	43.53	21.36	6.45	0.19	71.53		
	Spec.	1119	40.09	39.88	5.72	0.12	85.81		
	Sugg.	1024	31.10	31.74	5.39	40.37	108.60		
Beethoven Bust (RBF, 504 ctrs.)	Sil.	225	40.00	106.56	0.41	0.05	147.02	1.20	0.11
	Shad.	306	62.51	94.65	0.86	0.09	158.11		
	Spec.	204	39.28	91.55	0.41	0.06	131.30		
	Sugg.	493	94.98	254.86	1.86	103.62	455.32		

TABLE I

TIMING BREAKDOWN PER TASK ACROSS ALGEBRAIC, VOLUMETRIC AND RBF IMPLICIT SURFACES.

Table I shows the timings for three different models and respectively three different implicit representations: algebraic (quartic), cubic B-spline interpolation of a 32^3-grid volume and radial basis function interpolation of 504 scattered data points.

The Implicit column measures the total time per screen-update iteration to evaluate the implicit surface function and its derivative. Direct evaluation of RBF's remains slow, though speedups exist via the fast multipole method [7], [8]. However, spline interpolation of RBF's sampled in a voxel grid results in a smooth surfaces that supports a much faster evaluation. The local coefficients for the splines are recalculated per particle per frame. Storing these coefficients per particle could further accelerate performance at the expense of a larger memory footprint.

For proof of concept we computed a ray-cast visibility per-particle, measured in the Visibility column. While this was the most expensive per-particle operation, it can be reduced by propagating quantitative invisibility.

The Repulsion timing indicate the total time per screen-update iteration to compute inter-particle repulsion. We used a simple $O(n^2)$ algorithm to find distances between each particle and every neighbor, which sufficed for the 1-D contours used for illustration. We would use an $O(n)$-time grid locality acceleration structure for depiction with surface particles.

The Adhesion column likewise computes the surface-surface intersection adhesion described in Sec. III, which is inexpensive except for suggestive contours, because of the significantly greate amount of arithmetic required. This additional arithmetic includes some numerical finite difference approximations of higher derivatives that were not implemented by default in our implicit surface system, which require many additional implicit surface function evaluations.

These per-particle computations are nevertheless dominated by the significantly more expensive planar map construction, needed to correctly fill the closed contour output. This is followed by a less expensive constrainted Delauney triangulation (CDT). Both operations exceed the time expected of an interactive system, so we rely on the particle-based contour display for interactive posing, lighting and manipulation, and compute the planar map and CDT when exporting to a portable vector graphics format (SVG).

The resulting clipart generated by the the system was used to illustrate the various sections of the paper. The clip art displayed is polygonal except for Figures 1 and 6, where we replaced the polygons with closed filled spline curves. Splines are smoother and can more compactly represent curved regions than could polygons, but care must be taken when fitting to avoid introducing overlap between neighboring regions.

Our vector clip art resolution depends on the number of particles used, and particle sampling density can be curvature dependent, whereas raster processing operates at a fixed resolution. Our particle approach renders an implicit directly, efficiently focusing its effort on contour generation whereas raster processing must sample the entire visible surface, throwing away most of the interior per-pixel results. Finally the resulting curves do not suffer from occlusion problems and are three dimensional loops, which is of particular importance for the usability for NPR stylized line drawings [15].

VII. CONCLUSIONS AND FUTURE WORK

In this paper we presented a new particle-based method for the direct conversion of smooth surfaces (represented as implicit surfaces) into clip art (vectorial graphics with closed curves). We developed a new particle behavior designed to track curves at the intersection of two implicit surfaces, and showed how to represent various feature curves as such, including a new, more efficient version of suggestive contours.

The result is a useful tool for generating clip art for illustration of free-form shapes and visualization of the differential geometry of smooth curves and surfaces. Because our system is built on the Witkin-Heckbert surface particles [28], an additional feature is that it supports the direct manipulation of the displayed shape by dragging the particles in the displayed contours. Other sculpting modifications are also possible.

Although this paper's work focuses on the creation of static clip art, the particle system inheres a certain temporal coherence. Several algorithms use the old silhouettes as a seed for detecting the new ones [6], [23]. In our approach silhouettes are moved directly by particles which keep their relative positions as long as no critical point is encountered. This is a result of the optimization scheme implemented in the silhouette adhesion behavior that finds the silhouette as a one dimensional variety. For slow view changes, the particles maintain the loop structure once it is established. We did not notice changes in the particles relative neighbors as long as smooth silhouettes are concerned.

Several opportunities for future work exist:

- Simple particle tracing is not guaranteed to accurately trace the contour, nor to find all contours. The addition of interval or Lipschitz methods based on analysis of the implicit surface's defining function could provide such a guarantee [24].

- Our shadow contours separate regions facing toward and away from a light source. We trace rays to determine contour visibility and one could use the same approach to add cast-shadow contours.
- We plan to investigate contour control through camera motion and shape deformation as a tool for automatically enhancing the communication ability of this display form.
- More complicated shading patterns, like gradients, would be simple to integrate in our output for improved visual effects.
- We currently only manually determine that splines fit to the contours do not overlap significantly.
- We are interested in the correct contour topology maintenance across a visual event.
- The clip-art approach traces three dimensional loops that allow the integration of our system into, for example, the framework of Grabli *et al.* [15] where the user can create an unlimited variety of line drawings based on a scripted shading language.

In other words, we have achieved the goal of producing portable vector graphics clip art from smooth implicit surfaces, but in the process opened up many new directions in need of further investigation.

REFERENCES

[1] Wang Ao-yu, Tang Min, and Dong Jin-xiang. A survey of silhouette detection techniques for non-photorealistic rendering. In *3rd Intl. Conf. on Image and Graphics (ICIG'04)*, pages 434–437, 2004.

[2] Arthur Appel. The notion of quantitative invisibility and the machine rendering of solids. In *Proc. 22nd Natl. Conf.*, pages 387–393, 1967.

[3] James F. Blinn. Models of light reflection for computer synthesized pictures. *Proc. SIGGRAPH, Computer Graphics*, 11(2):307–316, 1977.

[4] CGAL Editorial Board. *CGAL-3.2 User and Reference Manual*, 2006.

[5] David J. Bremer and John F. Hughes. Rapid approximate silhouette rendering of implicit surfaces. In *Proc. Implicit Surfaces*, pages 155–164, 1998.

[6] Michael Burns, Janek Klawe, Szymon Rusinkiewicz, Adam Finkelstein, and Doug DeCarlo. Line drawings from volume data. In *Proc. SIGGRAPH*, pages 512–518, 2005.

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

[8] J. C. Carr, R. K. Beatson, B. C. McCallum, W. R. Fright, T. J. McLennan, and T. J. Mitchell. Smooth surface reconstruction from noisy range data. In *Proc. GRAPHITE*, pages 119–ff, 2003.

[9] Doug DeCarlo, Adam Finkelstein, Szymon Rusinkiewicz, and Anthony Santella. Suggestive contours for conveying shape. *ACM TOG*, 22(3):848–855, 2003.

[10] Gershon Elber. Line art illustrations of parametric and implicit forms. *IEEE TVCG*, 4(1):71–81, 1998.

[11] Gershon Elber and Elaine Cohen. Hidden curve removal for free form surfaces. In *Proc. SIGGRAPH*, pages 95–104, 1990.

[12] Eyal Flato, Dan Halperin, Iddo Hanniel, Oren Nechushtan, and Eti Ezra. The design and implementation of panar maps in cgal. *J. Exp. Algorithmics*, 5:13, 2000.

[13] K. Foster, P. Jepp, B. Wyvill, M. C. Sousa, C. Galbraith, and J. A. Jorge. Pen-and-ink for BlobTree implicit models. *CG Forum*, 24(3):267, 2005.

[14] Ron Goldman. Curvature formulas for implicit curves and surfaces. *CAGD*, 22(7):632–658, 2005.

[15] Stéphane Grabli, Emmanuel Turquin, Frédo Durand, and François Sillion. Programmable style for npr line drawing. In *Rendering Techniques (Proc. EGSR)*, pages 33–44. ACM Press, 2004.

[16] Paul Heckbert. Fast surface particle repulsion. Tech Report CMU-CS-97-130, 1997.

[17] Aaron Hertzmann and Denis Zorin. Illustrating smooth surfaces. In *Proc. SIGGRAPH*, pages 517–526, 2000.

[18] J. Hosche and D. Lasser. *Fundamentals of Computer Aided Geometric Design*. AK Peters, 1993.

[19] Tobias Isenberg, Bert Freudenberg, Nick Halper, Stefan Schlechtweg, and Thomas Strothotte. A Developer's Guide to Silhouette Algorithms for Polygonal Models. *IEEE CG&A*, 23(4):28–37, 2003.

[20] Robert D. Kalnins, Philip L. Davidson, Lee Markosian, and Adam Finkelstein. Coherent stylized silhouettes. *(Proc. SIGGRAPH) ACM TOG*, 22(3):856–861, 2003.

[21] J J Koenderink. What does the occluding contour tell us about solid shape? *Perception*, 13(3):321–30, 1984.

[22] Jan J. Koenderink. *Solid shape*. MIT Press, 1990.

[23] Lee Markosian, Michael A. Kowalski, Daniel Goldstein, Samuel J. Trychin, John F. Hughes, and Lubomir D. Bourdev. Real-time nonphotorealistic rendering. In *Proc. SIGGRAPH*, pages 415–420, 1997.

[24] Simon Plantinga and Gert Vegter. Computing contour generators of evolving implicit surfaces. *ACM Trans. Graph.*, 25(4):1243–1280, 2006.

[25] Wen Y. Su and John C. Hart. A programmable particle system framework for shape modeling. In *SMI '05: Proceedings of the International Conference on Shape Modeling and Applications 2005 (SMI' 05)*, pages 114–123, Washington, DC, USA, 2005. IEEE Computer Society.

[26] Kevin G. Suffern and Ronald J. Balsys. Rendering the intersections of implicit surfaces. *IEEE CG&A*, 23(5):70–77, 2003.

[27] Holger Winnemoeller, Sven C. Olsen, and Bruce Gooch. Real-time video abstraction. *ACM TOG*, 25(3):1221–1226, 2006.

[28] Andrew P. Witkin and Paul S. Heckbert. Using particles to sample and control implicit surfaces. In *Proc. SIGGRAPH*, pages 269–277, 1994.