Contour Generators of Evolving Implicit Surfaces

Simon Plantinga †
Gert Vegter
Institute for Mathematics and Computing Science
University of Groningen

ABSTRACT

The contour generator is an important visibility feature of a smooth object seen under parallel projection. It is the curve on the surface which separates front-facing regions from back-facing regions. The apparent contour is the projection of the contour generator onto a plane perpendicular to the view direction. Both curves play an important role in computer graphics.

Our goal is to obtain fast and robust algorithms that compute the contour generator with a guarantee of topological correctness. To this end, we first study the singularities of the contour generator and the apparent contour, for generic views, and for generic time-dependent projections, e.g. when the surface is rotated or deformed. The singularities indicate when components of the contour generator merge or split as time evolves.

We present an algorithm to compute an initial contour generator, using a dynamic step size. An interval test guarantees the topological correctness. This initial contour generator can then be maintained under a time-dependent projection by examining its singularities.

Categories and Subject Descriptors

I.3.3 [Computer Graphics]: Picture/Image Generation—Line and curve generation; I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric algorithms, languages, and systems; G.1.0 [Numerical Analysis]: General—Interval arithmetic

General Terms

Algorithms, Theory

* Partially supported by the IST Programme of the EU as a Shared-cost RTD (FET Open) Project under Contract No IST-2000-26473 (ECG - Effective Computational Geometry for Curves and Surfaces)
† email: {simon,gert}@cs.rug.nl

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SM’03 June 16–20, 2003, Seattle, Washington, USA.
Copyright 2003 ACM 1-58113-706-0/03/0006 ...$5.00.

1. INTRODUCTION

An important visibility feature of a smooth object seen under parallel projection along a certain direction is its contour generator, also known as outline, or profile. The contour generator is the curve on the surface, separating front-facing regions from back-facing regions. This curve may have singularities if the direction of projection is non-generic. The apparent contour is the projection of the contour generator onto a plane perpendicular to the view direction. In many cases, drawing just the visible part of the apparent contour gives a good impression of the shape of the object. In this paper, we will not distinguish between visible and invisible parts of the contour generator. Stated otherwise, we assume the surface is transparent. Generically, the apparent contour is a smooth curve, with some isolated singularities. See Figure 1.

The contour generator and the apparent contour play an important role in computer graphics and computer vision. Rendering a polyhedral model of a smooth surface yields a jaggy outline, unless the triangulation of the surface is finer in a neighbourhood of the contour generator. This observation has led to techniques for view-dependent meshing and view-dependent refinement techniques, cf. [1].

Also, to render a smooth surface it is sufficient to render only the part of the surface with front-facing normals, so the contour generator, being the boundary of the potentially visible part plays a crucial role here. In non-photorealistic rendering [16] one just visualizes the apparent contour, perhaps enhanced by strokes indicating the main curvature directions of the surface. This is also the underlying idea in silhouette rendering of implicit surfaces [6]. In computer vision, techniques have been developed for partial reconstruction of a surface from a sequence of apparent contours corresponding to a discrete set of nearby projection directions. We refer to the book [9] for an overview, and for a good introduction to the mathematics underlying this paper. Other applications use silhouette interpolation [11] from a precomputed set of silhouettes to obtain the silhouette for an arbitrary projection. In computational geometry, rapid silhouette computation of polyhedral models under perspective projection with moving viewpoints has been achieved by applying suitable preprocessing techniques [4].

This paper presents a method for the robust computation of contour generators and apparent contours of implicit sur-
The apparent contour. It turns out that generically there are changes in the topology of the contour generator and its projection changes with time. We derive conditions that locate the trace of the contour generator, and their projections on the apparent contour, to be regular. Section 5 examines singularities under some time-dependent view, where the direction of projection is static. Then we pass to time-dependent views, where the direction of projection is dynamic. From a much more advanced mathematical point of view, where they are known under the names lips, beak-to-beak, and swallowtail bifurcation. See also [8]. For a nice non-mathematical description we refer to the beautiful book by Koenderink [15]. Also see Arnol’d [2] and Bruce [7] for a sketch of some of the mathematical details related to singularity theory. Arnol’d [2] contains some of the results of the paper in a complex analytic setting. Our approach is somewhere in-between the level of Koenderink’s book and the sophisticated mathematical approach. We use only elementary tools, like the Inverse and Implicit Function Theorem, and finite order Taylor expansions. These techniques are used to design algorithms, in the same way as the Implicit Function Theorem gives rise to Newton’s method.

Most curve tracing algorithms step along the curve using a fixed step size. See for example [6] or [14]. For a good approximation, the user has to choose a step size that is ‘small enough’ to follow the details of the curve. Some algorithms predict a dynamic step size, based on the local curvature. Both methods cannot guarantee a correct approximation to the curve. Also, these curve tracing algorithms assume there are no singularities. By examining the singularities before tracing the curve we can avoid them in the tracing process. We developed a condition based on interval analysis, that guarantees topological correctness of the traced curve.

In Section 2 we present the framework, and discuss criteria for a point on the contour generator and apparent contour to be regular. Section 3 examines singularities under some time-dependent view, for example when the viewpoint moves or when the surface deforms. In section 4 we explain the transformation to local models. For the implementation interval analysis is used. A brief overview can be found in section 5. The algorithm for computing the contour generator is explained in section 6.

2. CONTOUR GENERATOR AND APPARENT CONTOUR

Contour generators of implicit surfaces. To understand the nature of regular and singular points of the contour generator, and their projections on the apparent contour, we assume $S$ is given as the zero-set of a smooth function $F : \mathbb{R}^3 \rightarrow \mathbb{R}$, so $S = F^{-1}(0)$. Furthermore, we assume that 0 is a regular value of $F$, i.e., the gradient $\nabla F$ is non-zero at every point of the surface. The gradient vector $\nabla F(p)$ is the normal of the surface at $p$, i.e., it is normal to the tangent plane of $S$ at $p$. This tangent plane is denoted by $T_p(S)$. If $v$ is the direction of parallel projection, then the contour generator $\Gamma$ is the set of points at which the normal to the surface is perpendicular to the direction of projection, i.e., $p \in \Gamma$ iff the following conditions hold:

$$F(p) = 0$$
$$\langle \nabla F(p), v \rangle = 0.$$  \hfill (1)

For convenience, we assume throughout the paper that $v = (0, 0, 1)$. Then the preceding equations reduce to

$$F(x, y, z) = 0$$
$$F_z(x, y, z) = 0.$$  \hfill (2)

Here, and in the sequel, we shall occasionally write $F_z$ instead of $\frac{\partial F}{\partial z}(p)$. We also use notation like $F_x$ and $F_{xx}$, with a similar meaning.

We assume that $S$ is a generic surface, i.e. there are no degenerate singular points on its contour generator. Some functions can yield degenerate contour generators. For example, a cylinder has a two-dimensional contour generator for the view direction along its axis. Using a small perturbation, we can remove these degeneracies. In the case of the cylinder, the two-dimensional contour generator collapses to a one-dimensional curve. See for example [21].

We now derive conditions for the contour generator $\Gamma$ and the apparent contour $\gamma$ to be regular at a given point. Recall that a curve is regular at a certain point if it has a non-zero tangent vector at that point. The next result gives conditions in terms of the function defining the surface.

$$F(p) = 0$$
$$\langle \nabla F(p), v \rangle = 0.$$  \hfill (1)
2. Since \( \nabla \) consequence of this observation.

The necessity and sufficiency of condition (3) is a straightforward
iff the corresponding scalar coefficients are zero. The ne-
combination of these vectors in the right-hand side of (5) is zero
are linearly independent vectors. Therefore, the linear com-
(projection of points (\( x, y, z \))=0 , where, for \( F_3(p) \neq 0 , \))

2. A point \( p \in \gamma \) is a regular point of the apparent contour
if and only if

Here all derivatives are evaluated at \( p \). Since \( \nabla F(p) = (F_y, F_z)^T \neq 0 \), we see that \( (F_y, F_z) \) and \( (0, 0, 1) \) are
linearly independent vectors. Therefore, the linear combi-
ation of these vectors in the right-hand side of (5) is zero if
the corresponding scalar coefficients are zero. The neces-
sity and sufficiency of condition (3) is a straightforward
consequence of this observation.

2. Since \( \nabla F(p) \neq 0 \in \mathbb{R}^3 \), and \( F_y(p) = 0 \), we see that
\( (F_y(p), F_z(p)) \neq (0, 0) \). Assuming \( F_z(p) \neq 0 \), we get

Let \( p = (x_0, y_0, z_0) \). Then, the Implicit Function Theorem
yields locally defined functions \( \eta, \zeta : \mathbb{R}^2 \to \mathbb{R} \), with \( \eta(x_0) = y_0 \) and \( \zeta(x_0) = z_0 \), such that (2) holds iff \( y = \eta(x) \) and \( z = \zeta(x) \). The contour generator is a regular curve parametrized
as \( x \mapsto (x, \eta(x), \zeta(x)) \), locally near \( p \), whereas the apparent
contour is a regular curve in the plane parametrized as \( x \mapsto
(x, \eta(x)) \), locally near \( (x_0, y_0) \).

Singular points of contour generators. We apply the pre-
ceding result to detect non-degenerate singularities of con-
tour generators of implicit surfaces. This result will be ap-
plied later in this section, when we consider contour gen-
ators of time-dependent surfaces.

Again, let the regular surface \( S \) be the zero set of a \( C^3 \-
function \( F : \mathbb{R}^3 \to \mathbb{R} \), for which 0 is a regular value. We
consider the contour generator \( \Gamma \) of \( S \) under parallel pro-
jection along the vector \( v = (0, 0, 1) \). The equations for \( \Gamma \) are

We consider the contour generator as the zero-set of the
function \( F_z \), restricted to \( S \).

Corollary 2.2. Point \( p \) is a non-degenerate singular point
of \( \Gamma \) iff the following two conditions hold:

\[
F(p) = F_z(p) = F_{zz}(p) = \frac{\partial(F, F_y)}{\partial(x, y)}|_p = 0,
\]

and \( \Sigma(p) \neq 0 \), where, for \( F_3(p) \neq 0 , \)

\[
\Sigma(p) = -F_x^2F_{xxx}^2 + F_y^2F_{yy}^2 + 2F_xF_yF_{xyz}F_y - F_x^4F_{yy}^2 - 2F_x^2F_{xxy}F_yF_{zz} + 2F_x^2F_{xy}F_{yy}F_y - 2F_xF_yF_{xyz}F_y - F_xF_yF_{xyz}F_y^2F_{zz} - F_x^3F_{xxx}F_{yy} + F_x^4F_{yy}F_{zz}.
\]

Figure 2: (a) A local model of the surface at a fold point is \( x + z^2 = 0 \). Both the contour generator \( \Gamma \),
and the visible contour \( \gamma \), are regular at the fold
point, and its projection onto the image plane, re-
spectively. (b) A local model at a cusp point is \( x + yz + z^3 = 0 \). Here the contour generator is regular,
but the apparent contour has a regular cusp.

we expect that the regular points of the apparent contour
form a one-dimensional subset of the plane. Furthermore,
the singular points of the apparent contour are projections
of points satisfying an additional equation, viz. \( F_{zz}(x, y, z) = 0 \).

1 using a computer algebra system
and are therefore expected to be isolated. This is true for 
\textit{generic} surfaces. To make this more precise, we consider the 
set of functions $F : \mathbb{R}^3 \rightarrow \mathbb{R}$, satisfying 
\begin{equation} 
(F(x, y, z), F_x(x, y, z), F_{xx}(x, y, z), \Delta(x, y, z)) \neq (0, 0, 0, 0), 
\end{equation} 
(10) 
and 
\begin{equation} 
(F(x, y, z), F_z(x, y, z), F_{zz}(x, y, z), F_{zzz}(x, y, z)) \neq (0, 0, 0, 0). 
\end{equation} 
(11) 
If $F$ satisfies (10), then Proposition 2.1 tells us that the 
contour generator $\Gamma$ of $S = F^{-1}(0)$ under parallel projection 
on $v$ is a regular curve. Moreover, for a point $(x, y, z) \in \Gamma$ 
there are two cases: 
1. $F_z(x, y, z) \neq 0$; in this case the point projects to a 
regular point $(x, y)$ of the apparent contour $\gamma$. Such a 
point is called a \textit{fold point} of the contour generator. 
This terminology is justified by the \textit{local model} of the 
surface near a fold point, viz. 
\begin{equation} 
x + z^2 = 0. 
\end{equation} 
(12) 
See also Figure 2a. Here the contour generator is the 
y-axis in three space, so the apparent contour is the 
y-axis in the image plane. 
2. $F_z(x, y, z) = 0$; in this case the point projects to a 
singular point $(x, y)$ of $\gamma$. Such a point is called a \textit{cusp} 
point of the contour generator if, in addition to (10), 
condition (11) is satisfied, i.e., if both $\Delta(x, y, z) \neq 0$ 
and $F_{zz}(x, y, z) \neq 0$. In this case the surface has the 
following local model near the cusp point: 
\begin{equation} 
G(x, y, z) = x + yz + z^3 = 0. 
\end{equation} 
(13) 
See also Figure 2b. The local model $G$ is sufficiently 
simple to allow for an explicit computation of its 
contour generator and apparent contour: the former is 
parametrized by $z \mapsto (2z^3, -3z^2, z)$, the latter is a \textit{regular cusp} parametrized by $z \mapsto (2z^3, -3z^2)$. 

Intuitively speaking, a local model of the surface near a point is a \textquoteleft simple\textquoteright expression of the defining equation in suitably 
chosen local coordinates. Usually, as in the cases of fold 
and cusp points, a local model is a low degree polynomial, 
which can be easily analyzed in the sense that the contour 
generator and the apparent contour are easily determined. 
So far we have only considered parallel projection. The \textit{standard perspective transformation} \cite{13}, which moves the 
viewpoint to $\infty$, reduces perspective projections to parallel 
projections. By deforming the surface using this transformation, 
perspective projections can be computed by using 
parallel projection on the transformed implicit function. 

3. \textbf{EVOLVING CONTOURS} 

As we have seen, generic surfaces satisfy conditions (10) 
and (11), since violation of one of these conditions would 
correspond to the existence of a solution of four equations 
in three unknowns. However, \textit{evolving surfaces} depend on 
an additional variable, $t$ say. Time dependency is expressed 
by considering implicitly defined surfaces 
\begin{equation} 
S_t = \{(x, y, z) \in \mathbb{R}^3 \mid F(x, y, z, t) = 0\}, 
\end{equation} 
where $F : \mathbb{R}^3 \times \mathbb{R} \rightarrow \mathbb{R}$ is a smooth function of the space 
variables $(x, y, z)$ and time $t$. Generically we expect that 
exactly one of the conditions (10) and (11) will be violated 
at isolated values of $(x, y, z, t)$. For definiteness, we assume 
$(0, 0, 0, 0)$ is such a value. 

Violation of (10) corresponds to a singularity of the contour generator. In this case the implicit surfaces, defined by $F(x, y, z, 0) = 0$ and $F_z(x, y, z, 0) = 0$, are tangent at 
$(x, y, z) = (0, 0, 0)$, but the tangency is non-degenerate. Stated 
otherwise, the function $G : \mathbb{R}^3 \rightarrow \mathbb{R}$, defined by $G(x, y, z) = F_z(x, y, z)$, \textit{restricted} to the surface $S_0$, has a non-degenerate 
singularity at $(0, 0, 0)$. 

Generically, there are two types of \textbf{bifurcations}, 
corresponding to different scenarios for changes in topology of the 
contour generator. The \textit{beak-to-beak} bifurcation corresponds to 
the \textit{merging} or \textit{splitting} of connected components of the contour generator. Under some additional generic conditions (inequalities), a local model for this phenomenon 
is the surface, defined by 
\begin{equation} 
G(x, y, z, t) = x + (-y^2 + t)z + z^3, 
\end{equation} 
(14) 
Here the contour generator is defined by $x = 2z^3, -y^2 + 3z^2 = -t$. See also Figure 3. 

\begin{figure} 
\centering 
\includegraphics[width=0.8\textwidth]{beak_to_beak_bifurcation.png} 
\caption{The beak-to-beak bifurcation. With respect to the local model (14) the bifurcation corresponds to $t < 0$ (left), $t = 0$ (middle), and $t > 0$ (right). 
\label{fig:beak_to_beak_bifurcation}} 
\end{figure} 

Putting $G^t(x, y, z) = G(x, y, z, t)$, we check that $G^t$ satisfies (7) at $p = (0, 0, 0)$, and that $|\Sigma(p)| = -4$. (In fact, 
$G^t_z = 1$, so all higher order derivatives of $G^t_z$ vanish 
identically, so only the last term in the right hand side of (8) 
is not identically equal to zero.) Therefore, $G^t_z[S$ has a 
non-degenerate singular point of saddle type at $p$. According 
to the Morse lemma (see \cite{10} or \cite{17}), the level set of 
$G^t_z[S$ through $p$ consists of two regular curves, intersecting 
transversally at $p$, which concurs with Figure 3 (middle). 

A second scenario due to the violation of (11) is the \textit{lips} 
bifurcation, corresponding to the \textit{birth} or \textit{death} of connected components of the contour generator. Again, under some additional generic conditions a local model for this phenomenon 
is the surface, defined by 
\begin{equation} 
G(x, y, z, t) = x + (y^2 + t)z + z^3, 
\end{equation} 
(15) 
Here the contour generator is defined by $x = 2z^3, y^2 + 3z^2 = -t$. In particular, for $t > 0$ the surface $S_t$ has no connected 
component of the contour generator near $(0, 0, 0)$, for $t = 0$, 
the point $(0, 0, 0)$ is isolated on the contour generator, and 
for $t < 0$ there is a small connected component growing 
obt of this isolated point as $t$ decreases beyond 0. See also 
Figure 4.
As for the beak-to-beak bifurcation, we show that \( G^0 \) has a non-degenerate singular point at \((0,0,0)\), which in this case is an extremum.

Violation of (11) involves the occurrence of a higher order singularity of the apparent contour. Note, however, that in this situation the contour generator is still regular at the point \((x,y,z)\), cf. Proposition 2.1. Imposing some additional generic conditions a local model for this type of bifurcation is

\[
G(x,y,z,t) = x + yz + tz^2 + z^4 = 0. 
\]

Here the apparent contour is parametrized as \( z \mapsto (tz^2 + z^4, -2tz - 4z^3) \). See also Figure 5.

Then \( \Phi^\mu \), defined by \( \Phi^\mu(x,y,z) = \Phi(x,y,z,\mu) \), maps \( T^\mu \) to \( S^{\psi(\mu)} \), and preserves contour generators. Furthermore, the map \( h^\mu : \mathbb{R}^2 \rightarrow \mathbb{R}^2 \), defined by \( h^\mu(x,y) = h(x,y,\mu) \), maps the apparent contour of \( T^\mu \) onto that of \( S^{\psi(\mu)} \).

**Proposition 4.2.** If \( \Phi \) is an admissible local transformation from \( T \) to \( S \), locally near a point \( p \) on the contour generator of \( T \), where \( \Phi \) is of the form (18), then

1. \( \Phi \) maps \( T \) to \( S \), locally near \( p \) in \( S \);
2. \( \Phi \) maps the contour generator of \( T \) to the contour generator of \( S \), locally near \( p \);
3. \( h \) maps the apparent contour of \( T \) onto the apparent contour of \( S \), locally near the projection \( p \) to \( \mathbb{R}^2 \).

**Proof.** From (18) and (19) it is easy to derive

\[
F(\Phi(p)) = \psi(p) G(p), 
\]

\[
F(\Phi(p)) H_\mu(p) = \psi_\mu(p) G(p) + \psi(p) G_\mu(p). 
\]

Since \( \psi(p) \neq 0 \), and \( H_\mu(p) \neq 0 \), we conclude that \( G(p) = G_\mu(p) = 0 \) iff \( F(\Phi(p)) = F(\Phi(p)) = 0 \). \( \square \)
Example: local model at a truncated cusp point. We now illustrate the use of admissible transformations by deriving a local model for the class of implicit surfaces defined as the zero set of a function of the form:

\[ F(x, y, z) = a(x, y) + b(x, y)z + c(x, y)z^2 + z^3, \]

with \( a(0, 0) = b(0, 0) = c(0, 0) = 0 \), and

\[ \frac{\partial (a, b)}{\partial (x, y)} \bigg|_0 \neq 0. \]

Note that the local model \( x + yz + z^3 = 0 \), derived in Section 2 for a cusp point, belongs to this class. Our goal is to show that the latter is indeed a local model for all surfaces of the type (20). Since \( F(0) = F_x(0) = F_y(0) = 0 \), and

\[ \frac{\partial (F, F_x)}{\partial (x, y)} \bigg|_0 = \frac{\partial (a, b)}{\partial (x, y)} \bigg|_0 \neq 0, \]

we see that \( 0 \in \mathbb{R}^3 \) is a regular point of the contour generator of \( S \), whereas \((0, 0)\) is a singular point of the apparent contour, cf. Proposition 2.1.

As a first step towards a normal form of the implicit function \( F \), we can be certain that \( 0 \in \mathbb{R}^3 \) is a local diffeomorphism near \( 0 \). Let \( \varphi \) be its inverse, then, putting

\[ \Phi(x, y, z) = (\varphi(x, y), z - \frac{1}{2} c(\varphi(x, y))), \]

we get

\[ F \circ \Phi(x, y, z) = G(x, y, z). \]

In other words, the admissible transformation \( \Phi \) brings \( F \) into the normal form \( G \). In particular, it maps the surface \( T = G^{-1}(0) \) and its contour generator onto \( S = F^{-1}(0) \), and \( \varphi \) maps the apparent contour of \( T \) onto the apparent contour of \( S \).

5. INTERVAL ANALYSIS

One way to prevent rounding errors due to finite precision numbers is to use interval arithmetic. Instead of numbers, intervals containing the exact solution are computed. An inclusion function \( \square f \) for a function \( f: \mathbb{R}^m \to \mathbb{R}^n \) computes for each \( m \)-dimensional interval \( I \) (i.e. \( m \)-box) an \( n \)-dimensional interval \( \square f(I) \) such that

\[ x \in I \implies f(x) \in \square f(I) \]

An inclusion function is convergent if

\[ \text{width}(I) \to 0 \implies \text{width}(\square f(I)) \to 0 \]

where the width of an interval is the largest width of \( I \).

For example if \( f: \mathbb{R} \to \mathbb{R} \) is the square function \( f(x) = x^2 \), then a convergent inclusion function is

\[ \square f([a, b]) = \begin{cases} [\min(a^2, b^2), \max(a^2, b^2)], & a \cdot b \geq 0 \\ [0, \max(a^2, b^2)], & a \cdot b < 0 \end{cases} \]

Inclusion functions exist for the basic operators and functions. To compute an inclusion function it is often sufficient to replace the standard number type (e.g. double) by an interval type.

We assume there are convergent inclusion functions for our implicit function \( F \) and its derivatives, and will denote them by \( F \) (and similar for the derivatives). From the context it will be clear when the inclusion function is meant.

Interval arithmetic can be implemented using demand-driven precision. For the interval bounds, ordinary doubles (with conservative rounding) can be used for fast computation. In the rare case that the interval becomes too small for the precision of a double, a multi-precision number type can be used.

Interval Newton Method. For precision small intervals around the required value are used. Another use of interval arithmetic is to compute function values over larger intervals. If for an implicit surface \( F = 0 \) and a box \( I \) we have \( 0 \notin \square f(I) \), we can be certain that \( I \) contains no part of the surface. This observation can be extended to the Interval Newton Method, that finds all roots of a function \( f: \mathbb{R}^m \to \mathbb{R}^n \) in a box \( I \).

The first part of the algorithm recursively subdivides the box, discarding parts of space containing no roots. If the boxes are small enough a Newton method refines the solutions and guarantees that all roots are found. Solving

\[ f(x) + J(I)(z - x) = 0, \]

where \( x \) is the centre of \( I \), \( J \) is the Jacobian matrix of \( f \) and \( J(I) \) is the interval matrix of \( J \) over the interval \( I \), results in an interval \( Y \) containing all roots \( z \) of \( f \). This interval can be used to refine \( I \). Also, if \( Y \subset I \) there is a unique root of \( f \) in \( I \). See [12] or [19] for the mathematical details. A more practical introduction can be found in [20] or [21].

6. TRACING THE CONTOUR GENERATOR

Our goal is to approximate the contour generator by a piecewise linear curve. This initial approximation can then be maintained under some time-dependent view. To this end the singularities of the contour generator for an evolving view or surface can be precomputed using interval analysis. Since the topology doesn’t change between these singularities, the initial contour generator can be updated continuously, until we reach a time where a singularity arises. The local model at this singularity indicates how the topology has to be updated. See Section 4. For details we refer to the full version of this paper.

Note that for a singularity of the contour generator of a
time-dependent surface, we have
\[
\begin{pmatrix}
F(x, y, z, t) \\
F_1(x, y, z, t) \\
F_2(x, y, z, t) \\
\Delta(x, y, z, t)
\end{pmatrix} = 0.
\]
These singularities can therefore be considered as the zeroes of a function from \( \mathbb{R}^3 \) to \( \mathbb{R}^3 \). Using the Interval Newton Method, we can find all \( t \) for which a singularity occurs.

For the initial contour generator we can assume there are no singularities. The construction consists of two steps.

Firstly, for each component we have to find an initial point to start the tracing process. Interval analysis enables us to find points on all components of the contour generator. See below for details. These (regular) points serve as starting points for the tracing process.

Secondly, we trace the component by stepping along the contour generator. For each starting point we trace the component, by moving from a point \( p^i \) to the next point \( p^{i+1} \). We take a small step in the direction of the tangent to the contour generator, giving us points for the tracing process. Interval analysis enables us to find points on all components of the contour generator. See figure 9 for some results of the algorithm.

**Finding initial points.** A tangent vector to the contour generator at \( p \) can be found by computing
\[
w(p) = \nabla F(p) \wedge \nabla F_s(p) = \begin{pmatrix}
F_y F_z - F_z F_y \\
F_z F_x - F_x F_z \\
F_x F_y - F_y F_x
\end{pmatrix}.
\]
Since the components of the contour generator are bounded, closed curves, there are at least two points on each component where the \( x \)-component of \( w(p) \) disappears, i.e. where \( F_y F_z = 0 \).

Let \( R, S : \mathbb{R}^3 \to \mathbb{R}^3 \) be the functions
\[
R(p) = \begin{pmatrix}
F(p) \\
F_1(p) \\
F_2(p)
\end{pmatrix} \quad \text{and} \quad S(p) = \begin{pmatrix}
F(p) \\
F_1(p) \\
F_2(p)
\end{pmatrix}.
\]
The Interval Newton Method can find all roots of \( R \) and \( S \). These roots are used to create a list of (regular) initial points.

**Tracing step.** Let \( N(x) \) be the normalized vector field
\[
N(x) = \frac{\nabla F(x) \wedge \nabla F_s(x)}{\|\nabla F(x) \wedge \nabla F_s(x)\|}.
\]
For \( x \) on the contour generator, \( N(x) \) is a tangent vector at \( x \). From \( p^i \) we first move to \( q^i = p^i + \delta N(p^i) \), where \( \delta \) is the step size. To move back to the contour generator, we alternately move towards \( F = 0 \) and \( F_z = 0 \) by replacing \( q^i \) by
\[
\begin{cases}
q^{i+1} = q^i - \frac{F(q^i) \nabla F(q^i)}{\|\nabla F(q^i)\|^2} \quad \text{towards } F \\
q^{i+1} = q^i - \frac{F_2(q^i) \nabla F_2(q^i)}{\|\nabla F_2(q^i)\|^2} \quad \text{towards } F_z
\end{cases}
\]
until \( \|q^{i+2} - q^i\| \) is sufficiently small. The resulting point is the next point on the contour generator, \( p^{i+1} \). For this new point we perform the interval test (explained below) to determine whether \( p^{i+1} \) is a good approximation of the contour generator. If not, we decrease \( \delta \) (e.g. by setting it to \( \delta/2 \)), and repeat the tracing step from \( p^i \).

**Interval test.** For a fixed step size, there is always a possibility of accidentally jumping to another component of the contour generator, or of skipping a part (figure 6).

![Figure 6: Left: \( N(p^i) \) is a tangent to the intersection of \( F = 0 \) and \( F_z = 0 \). Right: a fixed step size can miss part of the contour generator.](image)

To assure that \( p^i p^{i+1} \) is a good approximation for the contour generator, first we construct a sphere \( S \) with centre \( p^i \), that contains \( p^{i+1} \). Then we take the bounding box \( B \) of \( S \):
\[
B = [p_x^i - \Delta, p_x^i + \Delta] \times [p_y^i - \Delta, p_y^i + \Delta] \times [p_z^i - \Delta, p_z^i + \Delta],
\]
where \( \Delta = \|p^{i+1} - p^i\| \). Over this box we compute the interval
\[
I = \langle N(p^i), N(B) \rangle,
\]
where \( N(B) \) contains all normalized vectors \( \frac{\nabla F(s) \wedge \nabla F_s(t)}{\|\nabla F(s) \wedge \nabla F_s(t)\|} \) with \( s, t \in B \).

**Lemma 6.1.** If \( I > \frac{1}{\sqrt{3}} \), (i.e. \( \forall i \in I: i > \frac{1}{\sqrt{3}} \)), then the part of the contour generator within \( S \) consists of a single component (fig. 7).

![Figure 7: The sphere containing the line segment, and its bounding box.](image)

**Proof.** We define \( G(x) = \langle x - p^i, N(p^i) \rangle \). The level sets of \( G \) are planes perpendicular to \( N(p^i) \). Let \( f : \mathbb{R}^3 \to \mathbb{R}^3 \) be the function
\[
f(x) = \begin{pmatrix}
F(x) \\
F_1(x) \\
G(x)
\end{pmatrix}.
\]
Suppose there are two points \( x \) and \( y \) of the contour generator in \( B \), lying in a plane perpendicular to \( N(p^i) \), i.e.
\[ f(x) = f(y) = (0, 0, \theta) \] for some \( \theta \). Then there are points \( s \) and \( t \) on \( xy \) where
\[
\begin{pmatrix}
\nabla F(s) \\
\nabla F_t(t) \\
(N(p')) (y - x)
\end{pmatrix} = \begin{pmatrix}
0 \\
0 \\
0
\end{pmatrix}
\]

For \( x \neq y \), we find that \( \langle \nabla F(s) \wedge \nabla F_t(t), N(p') \rangle = 0 \). Since \( s, t \in B \) this contradicts the interval test. Therefore, within box \( B \) each plane perpendicular to \( N(p') \) contains at most one point of the contour generator.

The interval condition \( I > \frac{1}{2} \sqrt{2} \) implies that the angle between \( N(p') \) and \( N(x) \) is at most \( \frac{\pi}{4} \). The contour generator lies in a cone \( C \) around \( N(x) \) (fig. 8) with top angle \( \frac{\pi}{2} \), for if it leaves the cone at point \( a \), then \( \langle N(p'), N(a) \rangle \) would be smaller than \( \frac{1}{2} \sqrt{2} \). It can only leave the sphere in \( S \cap C \). In this part of the sphere the contour generator can’t re-enter \( S \), because that would require an entry point \( b \) where \( \langle N(p'), N(b) \rangle < \frac{1}{2} \sqrt{2} \). Therefore, there is only a single connected component of the contour generator within sphere \( S \).}

![Figure 8: The contour generator lies within a cone.](image)

If the interval test succeeds, we can use the same sphere \( S \) to remove redundant points from the initial point list. If there are point in the list that lie within \( S \), they must be part of the component we are tracing, so they can be discarded.

To test whether the component is fully traced, we test if the initial point \( p^0 \) is contained in \( S \). If it is, testing
\[ \langle p^{i+1} - p^i, p^0 - p^i \rangle > 0 \]
tells us whether we’re done with this component (otherwise we just started the trace and are still moving away from \( p^0 \)).

The interval bound is valid if the centre of the cube is on the contour generator. Since \( p^0 \) is in general close to, but not on the contour generator, in practice we take a slightly larger bound. Also, \( \Delta \) should be slightly larger than \( \|p^{i+1} - p^i\| \), to prevent \( p^{i+1} \) from being too close to other parts of the contour generator outside \( S \).

### 7. CONCLUSION AND FUTURE RESEARCH

We presented a framework for the analysis of an implicit surface near regular and singular points of its contour generator and apparent contour, and also derived conditions for detecting changes of topology of these visibility features in generic one-parameter families of implicit surfaces.

We developed an algorithm to compute a topologically correct approximation of the initial contour generator. A dynamic step size, combined with an interval test, guarantees that no part of the contour generator is skipped.

We plan to extend this work by implementing a robust algorithm for maintaining the contour generator under time-dependent directions of projection or surfaces, in such a way that its topology is guaranteed.

### 8. REFERENCES


Implicit Function Theorem has a non-degenerate singular point. Assume that 0 is a regular value of \( F \) on \( S \).

APPENDIX

A. SINGULARITIES OF FUNCTIONS ON SURFACES

Non-degenerate singular points. Consider an implicit surface \( S = F^{-1}(0) \), where \( F: \mathbb{R}^3 \to \mathbb{R} \) is a \( C^2 \) function. We assume that 0 is a regular value of \( F \), so according to the Implicit Function Theorem \( S \) is a regular \( C^2 \)-surface.

Our goal is to determine conditions which guarantee that the restriction of a \( C^2 \) function \( G: \mathbb{R}^3 \to \mathbb{R} \) to the surface \( S \) has a non-degenerate singular point.

As for notation, the gradient of a function \( F: \mathbb{R}^3 \to \mathbb{R} \) at \( p \in \mathbb{R} \) will be denoted by \( \nabla F(p) \). Furthermore, the Hessian quadratic form of a function \( F: \mathbb{R}^3 \to \mathbb{R} \) at \( p \in \mathbb{R} \) will be denoted by \( H_F(p) \). Usually, we suppress the dependence on \( p \) from our notation, and denote this quadratic form by \( H_F \). With respect to the standard euclidean inner product its matrix is the usual symmetric matrix whose entries are the second order partial derivatives of \( F \). We denote partial derivatives using subscripts, e.g., \( F_x \) denotes \( \frac{\partial F}{\partial x} \), \( F_{xy} \) denotes \( \frac{\partial^2 F}{\partial x \partial y} \), etc.

**Theorem A.1.** Let \( F, G: \mathbb{R}^3 \to \mathbb{R} \) be \( C^2 \) functions, and let 0 be a regular value of \( F \). Let \( p \) be a point on the surface \( S = F^{-1}(0) \).

1. \( p \) is a singular point of \( G|S \) iff there is a real number \( \lambda \) such that

\[ \nabla G(p) = \lambda \nabla F(p). \] (22)

2. Furthermore, the singular point \( p \) is non-degenerate iff

\[ (H_G - \lambda H_F) | T_pS \] (23)

is a non-degenerate quadratic form, where \( \lambda \) is as in (22).

**Remark** The scalar \( \lambda \) in (22) is traditionally called a Lagrange multiplier.

**Corollary A.2.** The singularity \( p \) of \( G|S \) is non-degenerate off the \( 2 \times 2 \)-matrix \( \Delta \), defined by (24), is non-singular:

\[ \Delta = V^T \left( \begin{array}{ccc} G_{xx} & G_{xy} & G_{xz} \\ G_{yx} & G_{yy} & G_{yz} \\ G_{zx} & G_{zy} & G_{zz} \end{array} \right) - \lambda \left( \begin{array}{ccc} F_{xx} & F_{xy} & F_{xz} \\ F_{yx} & F_{yy} & F_{yz} \\ F_{zx} & F_{zy} & F_{zz} \end{array} \right) \right) V, \] (24)

where \( \lambda \) is the Lagrange multiplier defined by (22), and \( V \) is a \( 3 \times 2 \)-matrix whose columns span the tangent space \( T_pS \). Here all first and second order derivatives are evaluated at \( p \). Furthermore, \( G|S \), the singular point \( p \) is a maximum or minimum if \( \det(\Delta) > 0 \), and a saddle point if \( \det(\Delta) < 0 \). In particular, we may take \( V = X, V = Y, \) or \( V = Z \) if \( F_x(p) \neq 0 \), \( F_y(p) \neq 0 \), or \( F_z(p) \neq 0 \), respectively, where

\[ X = \left( \begin{array}{cc} F_y & F_x \\ -F_z & 0 \end{array} \right), \quad Y = \left( \begin{array}{cc} -F_x & 0 \\ F_z & 0 \end{array} \right), \quad Z = \left( \begin{array}{cc} -F_y & 0 \\ 0 & F_x \end{array} \right). \] (25)

**Proof of Theorem A.1.** 1. Saying that \( p \) is a singular point of \( G|S \) is equivalent to \( dG_p(v) = 0 \), for all \( v \in T_pS \). Since \( T_pS = \ker dF_p \), we see that this is equivalent to the existence of a scalar \( \lambda \) such that \( dG_p = \lambda dF_p \).

2. Since 0 is a regular value of \( F \), we have \( \nabla F(p) \neq 0 \). We assume that \( F_x(p) \neq 0 \), and argue similarly in case \( F_y(p) \neq 0 \) or \( F_z(p) \neq 0 \). Furthermore, assume that \( p = (0,0,0) \). According to the Implicit Function Theorem, there is a unique local solution \( x = f(y,z) \), with \( f(0,0) = 0 \), of the equation \( F(x,y,z) = 0 \). Implicit differentiation yields

\[ F_x f_x + F_y f_y = 0, \] (26)

\[ F_y f_x + F_z f_z = 0, \] (27)

where \( f_y \) and \( f_z \) are evaluated at \( (y,z) \), and \( F_x, F_y \) and \( F_z \) are evaluated at \( (f(y,z),y,z) \). Similarly,

\[ F_{xx} f_y^2 + 2F_{xy} f_x f_y + F_{yy} f_y + F_z f_{yz} = 0. \] (28)
Similar identities are obtained by differentiating (26) with respect to $y$, and (27) with respect to $z$.

Using $y$ and $z$ as local coordinates on $S$, we obtain the following expression of $G|S$ with respect to these local coordinates:

$$g(y, z) = G(f(y, z), y, z).$$

Differentiating this identity twice with respect to $y$ we obtain

$$g_{yy} = G_{xx} f_y^2 + 2G_{xy} f_y + G_{yy} + G_x f_{yy}.$$  \hspace{1cm} (29)

Since $F_x(p) \neq 0$, we solve $f_{yy}$ from (28), and plug the resulting expression into (29), to get

$$g_{yy} = (G_{xx} - \lambda F_{xx}) f_y^2 + 2(G_{xy} - \lambda F_{xy}) f_y + (G_{yy} - \lambda F_{yy}),$$  \hspace{1cm} (30)

where

$$\lambda = \frac{G_x}{F_x}$$

is the Lagrange multiplier, cf. (22). We rewrite (30) as

$$g_{yy} = \begin{pmatrix} f_y & 1 & 0 \end{pmatrix} \begin{pmatrix} H_G - \lambda F_x \end{pmatrix} \begin{pmatrix} f_y \\ 0 \end{pmatrix} = \frac{1}{F_x^2} \begin{pmatrix} F_y & -F_x & 0 \end{pmatrix} \begin{pmatrix} H_G - \lambda F_x \end{pmatrix} \begin{pmatrix} F_y \\ -F_x \\ 0 \end{pmatrix}$$

We similarly derive

$$g_{xz} = \frac{1}{F_x^2} \begin{pmatrix} F_y & -F_x & 0 \end{pmatrix} \begin{pmatrix} H_G - \lambda F_x \end{pmatrix} \begin{pmatrix} F_z \\ -F_x \\ 0 \end{pmatrix},$$

and

$$g_{zz} = \frac{1}{F_x^2} \begin{pmatrix} F_y & 0 & -F_x \end{pmatrix} \begin{pmatrix} H_G - \lambda F_x \end{pmatrix} \begin{pmatrix} F_z \\ 0 \\ -F_x \end{pmatrix}.$$  \hspace{1cm} (32)

Since the vectors $(F_y, -F_x, 0)$ and $(F_z, 0, -F_x)$ span the tangent space $T_p S$, we see that

$$\begin{pmatrix} g_{yy} \\ g_{xz} \\ g_{zz} \end{pmatrix} = \frac{1}{F_x^2} \Delta.$$  \hspace{1cm} (33)

**B. METABALLS**

One approach to modeling using implicit surfaces is to use metaballs. A single metaball consists of a density function around a single point. The function value at the point equals a given weight, and drops to zero at a given distance (the radius) from the centre. By adding individual metaballs and subtracting a threshold value, blobby objects can be joined smoothly.

For a single metaball with weight $w$ and radius $R$ we use the density function

$$D(P) = \begin{cases} w \left(1 - \left(\frac{r}{R}\right)^2\right)^2 & r < R \\ 0 & r \geq R \end{cases}$$

where $r$ is the distance from $P$ to the centre of the metaball.

Using threshold $T$, the implicit function for the metaball object is given by

$$F(P) = \sum_{i=1}^{n} D_i(P) - T.$$