Contouring 1- and 2-Manifolds in Arbitrary Dimensions

Joon-Kyung Seong
School of Computer Science
Seoul National University, Korea
swallow@3map.snu.ac.kr

Gershon Elber
Computer Science Dept.
Technion, Israel
gershon@cs.technion.ac.il

Myung-Soo Kim
School of Computer Science
SNU, Korea
mskim@cse.snu.ac.kr

Abstract

We propose an algorithm for contouring k-manifolds $(k = 1, 2)$ embedded in an arbitrary n-dimensional space. We assume $(n-k)$ geometric constraints are represented as polynomial equations in n variables. The common zero-set of these $(n-k)$ equations is computed as a 1- or 2-manifold, respectively, for $k = 1$ or $k = 2$. In the case of 1-manifolds, this framework is a generalization of techniques for contouring regular intersection curves between two implicitly-defined surfaces of the form $F(x, y, z) = G(x, y, z) = 0$. Moreover, in the case of 2-manifolds, the algorithm is similar to techniques for contouring iso-surfaces of the form $F(x, y, z) = 0$, where $n = 3$ and only one $(= 3 - 2)$ constraint is provided. By extending the Dual Contouring technique to higher dimensions, we approximate the simultaneous zero-set as a piecewise linear 1- or 2-manifold. There are numerous applications for this technique in data visualization and modeling, including the processing of various geometric constraints for freeform objects, and the computation of convex hulls, bisectors, blendings and sweeps.

1. Introduction

Given a scalar field $F(x, y, z)$ in a three-dimensional space, the problem of constructing its iso-surfaces $F(x, y, z) = c$ has important applications in implicit surface modeling as well as in volume visualization. The Marching Cubes [15] algorithm and Dual Contouring techniques [6, 11] have been used quite successfully in solving this problem.

An iso-surface is essentially the zero-set of one equation in three variables. In computational science and engineering, we often encounter a situation where one needs to solve a set of m polynomial equations in n variables:

$$F_i(x_1, x_2, \cdots, x_n) = 0, \quad i = 1, \cdots, m.$$

Although solving a simultaneous system of equations is a familiar problem, no sufficiently robust and efficient algorithm has yet been developed for equations of a general type.

For the special case where $m = 2$ and $n = 3$, the problem is reduced to that of intersecting two implicit surfaces:

$$F_1(x, y, z) = F_2(x, y, z) = 0.$$

Computing a surface-surface intersection (SSI) is known to be one of the most challenging problems in geometric computation, especially when the two surfaces are both either parametric or implicit [9]. Because of the difficulty in dealing with the general situation, grid-based techniques such as Marching Cubes or Dual Contouring are a reasonable approach to approximating the zero-set. The precision of the solution is, however, limited by the grid size employed in the computation.

In this paper, we consider the following more general problem:

$$F_i(x_1, x_2, \cdots, x_n) = 0, \quad i = 1, \cdots, n-k, \;\; k = 1, 2,$$

where the zero-set of these $n - k$ simultaneous equations generates a 1- or 2-manifold, respectively, for $k = 1$ or $k = 2$ in an n-dimensional space.

Each cubic cell in a three-dimensional grid (a 3-cube) has six face-adjacent neighborhoods. Dual Contouring techniques utilize this connectivity. In the case of an n-cube (a hyper-cube in n-dimensions), there are $2n$ adjacent cells, each of which shares an $(n-1)$-cube with the given n-cube. Although this may look quite complicated, the situation is surprisingly similar to the three-dimensional case.

The key observation in this paper is that an n-cubic cell containing a 1-manifold has only two neighborhood cells connected on the 1-manifold, except for some degenerate cases. Based on this simple observation, we can extend the Dual Contouring approach from three dimensions to arbitrary n-dimensional spaces. A similar argument can be applied to a 2-manifold. An n-cubic cell on a 2-manifold has four such neighborhood cells.

A naive extension of the Dual Contouring method, however, finds difficulty in handling degenerate cases, which

occur where the 1- or 2-manifold solution manifold passes through vertices, edges, or $(n-k)$-cells, for some $k > 1$. Since the Dual Contouring method is based on the connectivity between face-adjacent solution points, we need to devise a new scheme for handling degenerate cases. The problem becomes even worse if one examines higher dimensional spaces given that there is a significantly higher chance of getting degeneracies. In this paper, we propose a tangent space-based scheme to resolve the degenerate cases. Using the tangent space, we can locally approximate the solution manifold. We first pre-classify all the abnormal solution cells degeneracies have a likelihood of occuring. The tangent space at these cells gives the correct information for a connection.

The presented algorithm is also based on a general approach for constructing the solution manifold. Elber and Kim [5] approximated the solution set by fitting a 1- or 2-manifold to a set of discrete points. Thus, the parameterization of the solution manifold usually depends on the original curve or surface in [5]. One needs to devise a problem-specific way to parameterize the solution manifold if one tries to fit the solution set using a set of discrete points. Bisector surfaces, for instance, are fitted following the parameterization of one of the given freeform surfaces [4]. An extreme case may require special treatment in the construction of the solution manifold. In this paper, we devise a general algorithm for contouring a 1- or 2-manifold that is independent of both the specific problem and the particular case.

One may wonder whether this problem might also be solved using a Marching n-Cubes approach. In the case of high dimensional spaces, the Marching Cubes approach becomes more complicated than Dual Contouring. Consider the case of a 2-manifold contouring. Note that there are 2^n vertices in an n-cube. Moreover, each vertex has n edges and each edge is shared by two vertices; thus, we also have $n2^{n-1}$ edges in an n-cube. In addition to that, we have to deal with $(n-2)$ scalar fields $F_i(x_1, \cdots, x_n)$, for $i = 1, \cdots, n-2$. It is more difficult to extend the Marching Cubes approach to an arbitrary dimension because of its exponential complexity growth. This also holds for a similar piecewise linear method in the area of numerical continuation methods [1, 2].

The rest of this paper is organized as follows. In Section 2, we briefly review previous results. In Section 3 we discuss five illustrative examples where each geometric constraint problem can be reduced to that of solving a system of non-linear equations. In Section 4, we present the Dual Contouring technique that generates the contouring of a 1- or 2-manifold in an n-dimensional space. Experimental results are shown in Section 5. Finally, in Section 6, we conclude.

2. Related Previous Work

The Marching Cubes algorithm [15] generates discrete points on the iso-surface (i.e., where the edges of each cube intersect the surface). The Extended Marching Cubes algorithm [13] adds some additional cell-interior points that capture sharp features of the iso-surface. As already mentioned, these methods are not applicable to higher-dimensional spaces: as the number of dimensions n increases, the number of vertices and edges of an n-cube increases exponentially and the Marching Cubes approach becomes very complex.

Dual Contouring methods [6, 11] are based on the connectivity of adjacent cubes through common faces. The 'duality' of this approach to Marching Cubes was originally observed in the duality of their triangulations of an iso-surface. In an n-dimensional setting, we can also observe a different type of duality: as the number of constraints increases, the dimension of the constraint manifold decreases. The Marching Cubes method deals with the constraints, whereas the Dual Contouring method considers the constraint manifold itself. Since we are considering low-dimensional manifolds, the Dual Contouring approach is more appropriate for our problem. The Marching Cubes approach might, however, be useful in applications involving a small number of constraints, where each constraint is defined by a large number of variables.

Lane and Riesenfeld [14] proposed a subdivision-based approach for the computation of roots of univariate polynomial functions, using the Bernstein-Bézier basis function. Nishita et al. [17] introduced a Bézier clipping technique that can very efficiently compute the common roots of two bivariate Bézier functions. Looking at higher dimensions, Sherbrooke and Patrikalakis [21] presented an approach to solving a set of multivariate polynomial equations given in Bernstein-Bézier form. These techniques can be applied where the number of equations is the same as the number of variables (i.e., when the solution sets are discrete).

Sederberg and his colleagues [19, 20] introduced the concept of a normal cone and a surface bounding cone, which can guarantee that the intersection curve of two surfaces has only a single branch. By interpreting the zero-set computation as the search for an intersection between multivariate implicit hyper-surfaces, Elber and Kim [5] adapted these tools to higher dimensions.

Hyper-surface bounding cones guarantee that the implicit hyper-surfaces intersect each other transversally in the domain and thus ensure that each domain contains only one connected component of the solution set. Once this condition is met, a simple subdivision scheme provides an efficient way to generate a dense distribution of discrete points across the solution set. Elber and Kim [5] approximated the solution set by fitting a hyper-curve or hyper-surface to

these points. In this paper, we consider a complete solution where we connect the solution points in a correct topology. This connectivity information is useful in progressively improving the precision of the solution by local refinements (i.e., without repeating the global curve or surface fitting).

3. Solving Geometric Constraints

A large variety of geometric problems can be reduced to the solution of a system of multivariate polynomial equations. Kim and Elber [12] surveyed the general paradigm of reducing geometric constraints to a set of non-linear equations. In this section, we present a few examples to illustrate our motivation for this work.

3.1. Computing Convex Hulls

Given a concave freeform surface or a set of freeform surfaces, we consider the problem of computing their convex hulls. A point $S(u,v)$ on a rational surface is contained in the boundary of the convex hull if and only if the surface S is completely contained on one side of the tangent plane $T(u,v)$. Thus, by computing bi-tangent planes of the given surfaces, we can determine developable surfaces that contribute to the boundary of the convex hull [22]. Computing a bi-tangent plane can be resolved by solving the following three equations in four variables u, v, s, t:

$$\begin{aligned} F(u,v,s,t) &= \langle S(u,v) - S(s,t), N(u,v)\rangle = 0, && (1)\\ G(u,v,s,t) &= F_s(u,v,s,t) = 0, && (2)\\ H(u,v,s,t) &= F_t(u,v,s,t) = 0, && (3) \end{aligned}$$

where F_s and F_t represent the s- and t-partial derivates of $F(u,v,s,t)$, respectively. Having three polynomial equations in four variables, we get a 1-manifold as the common zero-set. Two examples of computing bi-tangent planes are shown in Figure 6.

3.2. Computing Perspective Silhouettes of a General Swept Volume

Consider the silhouette curves of a general swept volume. Let O be a three-dimensional object bounded by a freeform surface $S(u,v)$, and let $A(t)$ denote an affine transformation represented by a 4×4 matrix. Each instance of the moving object O^t under the affine transformation $A(t)$ touches the envelope surface of the swept volume along a characteristic curve K^t, the curve K at time t. The characteristic curve K is a curve that contributes to the envelope surface. Moreover, the same instance of the moving object has its silhouette curve, S^t, from a viewing position $\vec{P}$, on the boundary of the moving object. The intersection $K^t \cap S^t$ contributes to the silhouette of the general swept volume. We formulate the problem as the following system of two polynomial equations in three variables [23][16]:

$$\begin{aligned} &F(u,v,t)\\ &= \left| A'(t)[S(u,v)] \;\; A(t)\left[\frac{\partial S}{\partial u}(u,v)\right] \;\; A(t)\left[\frac{\partial S}{\partial v}(u,v)\right]\right|\\ &= 0, \qquad (4)\\ &G(u,v,t)\\ &= \left\langle A(t)[S(u,v)] - \vec{P}, A(t)[N(u,v)]\right\rangle = 0. \qquad (5) \end{aligned}$$

Having two polynomial equations in three variables, the common zero-set generates 1-manifolds. Figure 7 shows two examples of computing the silhouette curves of a general swept volume.

3.3. Computing Bisectors

Consider the bisector surface of two rational surfaces $S_1(u,v)$ and $S_2(s,t)$. Each point (x,y,z) on the bisector surface is at an equal (orthogonal) distance from $S_1(u,v)$ and $S_2(s,t)$; Elber and Kim [4] reduced the problem of computing the bisector surface into that of solving two polynomial equations in four variables:

$$\begin{aligned} F_1(u,v,s,t) &= 0,\\ F_2(u,v,s,t) &= 0. \qquad (6) \end{aligned}$$

Figure 1 shows an example of a bisector surface between two freeform surfaces. Elber and Kim [4] approximated the solution set by fitting a hyper-surface to a set of discrete solution points. However, near the self-intersecting region, the fitted surface is not sufficiently accurate. Figure 8 shows examples of surface-surface bisectors that are generated by applying the contouring algorithm presented in this paper.

3.4. General Sweep Computation

The swept volume of a three-dimensional object O under an affine transformation $A(t)$ is given by $\cup_t A(t)[O]$. Assuming $a \le t \le b$, the boundary surface of the swept volume consists of some patches of $A(a)[S(u,v)]$ and $A(b)[S(u,v)]$, together with the envelope surface, which is the set of points $A(t)[S(u,v)]$ that satisfy Equation (4). Figures 9 shows two examples of a general sweep. Joy and Duchaineau [10] computed the boundary of a swept volume using a Marching Cube algorithm in xyz-space. The envelope surface is usually more complicated in xyz-space than its counterpart in the uvt-space; thus it is much easier to deal with the problem in the parameter space.

3.5. Blending of Two Freeform Surfaces

Given two freeform surfaces $S_1(u,v)$ and $S_2(s,t)$, we consider the construction of a smooth blending surface be-

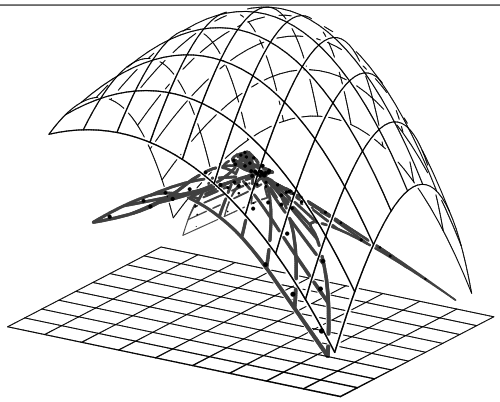


Figure 1. Bisectors between two freeform surfaces. The bisector surface is approximated by fitting a surface to a set of discrete solution points, the result of solving a set of Equations (6). Thus, the bisector is not sufficiently accurate near the self-intersecting regions. Compare this result with Figure 8(b).

tween the two surfaces. The potential method of Hoffmann and Hopcroft [8] computes a smooth blending surface between two implicit surfaces. We apply a similar technique to the blending of two parametric surfaces, where a blending surface is determined by the following system of four equations in six variables, u, v, s, t, α_1, and α_2:

$$S_1(u,v) + \alpha_1 N_1(u,v) = S_2(s,t) + \alpha_2 N_2(s,t), \quad (7)$$

$$\alpha_1^2 + \alpha_2^2 + 1 - 2\alpha_1 - 2\alpha_2 = 0, \quad (8)$$

where N_i is the normal of a surface S_i, for $i = 1, 2$, and Equation (7) represents three scalar constraints. Having 4 constraints, the resulting zero-set generates a 2-manifold in a 6-dimensional space. Figure 10 shows two examples of constructing a smooth blending surfaces.

4. Contouring Methods

Given a set of multivariate polynomial or rational equations represented as B-spline functions, Elber and Kim [5] isolated the n-cubic cells that contain the zero-set of these equations by recursively subdividing the B-spline functions, in all dimensions. The subdivision process continues until reaching a given maximum depth of subdivision. At the end of this subdivision step, we bound a 1- or 2-manifold using a set of n-cubic grid cells of the same size. Figure 2 shows one simple example of the result of this subdivision stage, where the solution space is a 2-manifold.

The centroid of each n-cubic grid cell that results from the subdivision stage is projected onto the 1- or 2-dimensional solution manifold using a numeric step,

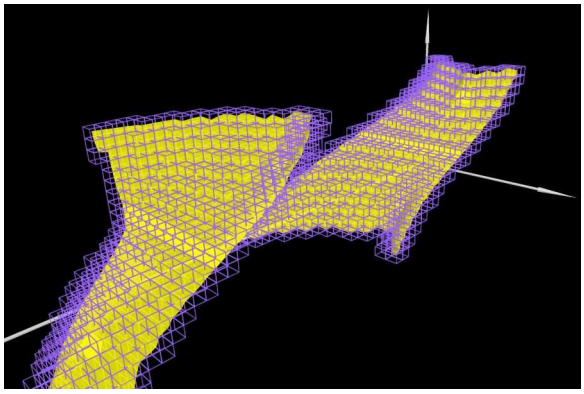


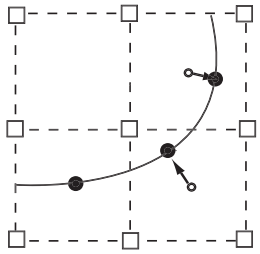
Figure 2. The result of the subdivision stage of the solver is a set of n-cubes that intersects the 2-manifold solution set.

where the Newton-Raphson is applied in n-dimensional space. The projected points are laid on the 1-manifold solution curve or on the 2-manifold solution surface with high precision. After the projection, the points are connected to other solution points in adjacent cells. In this section, we present a new algorithm for connecting the solution points in the 1- and 2-manifolds. Note that a discrete solution point only has a list of face-adjacent solution points. Basically, we extend the Dual Contouring method from three dimensions to arbitrary n-dimensional spaces.

4.1. 1-Manifold Solution

If a solution cell has only two face-adjacent cells, it is straightforward to connect the solution points to form a union of 1-manifold polylines. However, there are also cases where the original 1-manifold solution curve nearly passes through vertices, edges, or $(n-k)$-cells, for some $k > 1$. Figure 3 shows an analogy of this situation in the 2-dimensional case. In Figure 3(b), the solution points $\mathbf{p}$ and $\mathbf{q}$ can be connected through an adjacent cell, either $\mathbf{r}$ or $\mathbf{s}$. We call these cubes $\mathbf{r}$ and $\mathbf{s}$ *abnormal* since their projected solution points lie outside the cube. This degenerate case occurs since the B-spline subdivision maintains the set of valid n-cubic domains conservatively. In this case, the solution points $\mathbf{p}$ and $\mathbf{q}$ have more than two face-adjacent cells. Thus, the union of the remaining n-cubic cells may generate a thick volume surrounding the 1-manifold solution set. We need to eliminate some redundant cells so that the solution set can be properly represented as a union of 1-manifold polylines.

The problem becomes more difficult when we consider higher dimensional spaces since the extra di-



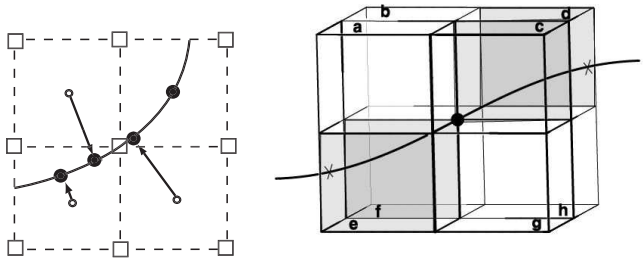
(a) (b)

Figure 3. (a) The zero-set typically passes through the faces of cubic cells; but (b) it can also pass through a vertex to produce two redundant solutions r and s.

mension introduces a higher probability of yielding degeneracies. Figure 4 compares the problem of tracing a curve in 2-dimensional and 3-dimensional grids. In the case of Figure 4(a), two solution cells are connected through one face-adjacent *abnormal* cell. However, in the 3-dimensional case, a pair of face-adjacent *abnormal* cells connects two solution cells (see Figure 4(b)). It is thus much more difficult to deal with the degenerate cases in higher dimensions than in lower dimensions. In Section 5, we compare the probability of getting degeneracies according to the dimensionality of the problem. Since we are considering a 1-manifold solution in arbitrary dimensions, we need to devise an algorithm for handling these complex degeneracies.

To resolve the degenerate cases, we apply different schemes to n-cubic solution cells depending on their classification. We classify a cube as a *normal* one if a numerically improved solution point lies inside the cube. Otherwise, we treat a cube as an *abnormal* one. In Figure 3(b), two solution points $\mathbf{p}$ and $\mathbf{q}$ are *normal* cubes, and $\mathbf{r}$ and $\mathbf{s}$ are classified as *abnormal* ones. When a *normal* cubic cell has a face-adjacent *normal* cube, we connect them by an edge.

When we finish contouring along face-adjacent *normal* cubes, we may end up with 1-manifold polylines with dangling normal cubes at their ends, each with only one face-adjacent normal cube. Two such nearby polylines are connected through a cluster of *abnormal* cells, which forms a thick volume connecting the two polylines. Recall that *abnormal* cubic solution cells are produced when the 1-manifold solution curve passes through vertices, edges, or $(n-k)$-cells, for some $k > 1$. The *abnormal* cubes are



(a) (b)

Figure 4. The connecting problem becomes considerably more complex as one examines higher dimensions.

redundant solutions. We need to eliminate most of them, while using the cluster information to connect two nearby polylines smoothly. For this purpose, we construct a tangent line to the 1-manifold solution curve in $\mathcal{R}^n$ at each *abnormal* solution point. Then, we take an average of the tangent lines and locally approximate the solution manifold. We then parameterize the dangling *normal* solution points using the tangent line to connect them.

In Figure 4(b), we have a cluster of six *abnormal* solution cells near the *normal* solutions $\mathbf{e}$ and $\mathbf{d}$. These two solution points $\mathbf{e}$ and $\mathbf{d}$ are dangling after we finish contouring along face-adjacent *normal* cubes. We compute the centroid of six *abnormal* solution points and project it into the 1-manifold curve to get an optimal solution point. We then compute a tangent line at the solution point to connect two dangling *normal* solutions through the intermediate solution point (See Figure 5). In summary, a cluster of *abnormal* solution cells gives topological information on how to connect the dangling *normal* solution points and make a bridge between the two dangling *normal* solutions. In the case of 1-manifolds, the connection for two dangling solution points is trivial. Furthermore, a similar approach using an appropriate tangent space works for 2-manifolds or higher dimensional manifolds.

The tangent line to the 1-manifold solution curve in $\mathcal{R}^n$ is spanned by a single vector denoted v_1. This vector should be orthogonal to the normal space of the 1-manifold solution set. This normal space is spanned by the gradients of all the constraints,

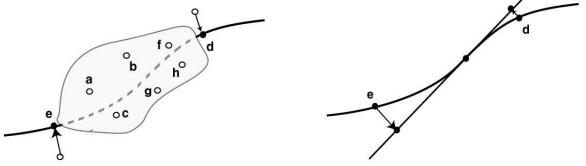
$$\mathcal{G} = \{\nabla F_i(\mathbf{u}) \mid i = 1, \cdots, n-1\},$$

where $\mathbf{u} = (x_1, \cdots, x_n)$ is a point projected onto the 1-manifold. Let $\{\varphi_i \mid i = 1, \cdots, m\}$, $(m \leq n-1)$, denote a set of linearly independent unit vectors generated from the set $\mathcal{G}$ by applying the Gram-Schmidt orthogonalization process [7]. Now, from a standard basis $\{e_i\}$ of $\mathcal{R}^n$, we con-

struct a set of vectors

$$\bar{e}_i \quad = \quad e_i - e_i(\varphi_1^T\varphi_1 + \cdots + \varphi_m^T\varphi_m),\ i = 1, \cdots, n,$$

each of which is the projection of e_i to the space orthogonal to $\mathcal{G}$. We select $(n-m)$ independent vectors $\bar{e}_i$ of the largest magnitude as $v_1, \cdots, v_{n-m}$, which span the space orthogonal to $\mathcal{G}$. In contouring a 1-manifold solution, we select a single vector $\bar{e}_i$ of the largest magnitude and denote it as v_1. The overall procedure is summarized in **Algorithm1**.



(a) (b)

Figure 5. (a) *Abnormal* solution cells make a cluster near the dangling *normal* solution cell. (b) A tangent line provides connection information for two dangling solution points.

4.2. 2-Manifold Solution

The algorithm for contouring a 1-manifold solution can easily be extended to the tessellation of a 2-manifold in arbitrary dimensions. If a solution cell has only four adjacent cells, the connection of solution cells into a 2-manifold is straightforward. When a *normal* cubic cell has four face-adjacent *normal* cubes, we construct four triangles by connecting them. We do not take into account *abnormal* cubes when we are dealing with the *normal* cube. Then, similarly to Section 4.1, we get polygons with dangling *normal* cubes at their ends, which have less than four face-adjacent *normal* cubes. These dangling *normal* cubes must have a set of face-adjacent *abnormal* cubes. A cluster of the *abnormal* solution cells will fill the gap between two disconnected polygons in a similar way to the cluster of *abnormal* cubes as discussed in Section 4.1.

In the contouring of a 2-manifold surface, we construct a tangent plane to the 2-manifold in $\mathcal{R}^n$ and locally approximate the solution set. Using the tangent plane we connect a set of dangling *normal* cubes with a triangular mesh that fills the gap in-between them. That is, we reduce the problem of tessellating the 2-manifold in $\mathcal{R}^n$ to that of triangulating on a hyper-plane in the degenerate case.

The tangent plane in the 2-manifold solution in $\mathcal{R}^n$ is spanned by two independent vectors denoted v_1 and v_2. These vectors can be computed in a similar way to the case of 1-manifolds. Here, we select two independent vectors, v_1 and v_2, of the largest magnitude among the vectors $v_1, \cdots, v_{n-m}$, which span the space orthogonal to $\mathcal{G}$. **Algorithm1** also summarizes this overall procedure.

When the normal vectors span a vector space of a dimension lower than $(n-2)$, the constraint manifold thickens and may become a 3-manifold or have an even higher dimensionality. It is very difficult to deal with this degenerate case reliably. An analogy can be found in trying to intersect two almost overlapping freeform surfaces, a situation that is problematic for many SSI algorithms. Even in this difficult case, the Dual Contouring approach would provide a reliable solution to the extraction of a k-dimensional constraint manifold, for $k > 2$. The k-dimensional tangent spaces play a role similar to the one the tangent planes in the 2-manifold case play.

5. Experimental Results

Figure 6 shows two examples of freeform surfaces and their convex hull, each of which was computed by solving a system of Equations (1) – (3). In these figures, 1-manifold curves that determine the boundary curves of the convex hull patches are shown in bold lines and the convex hull patch is shown in light lines. Recall that we have to solve three polynomial equations in four variables for this problem. The perspective silhouette curves of a general swept volume can also be computed by using the contouring algorithm proposed in this paper. Figure 7 shows two examples of computing the 1-manifold silhouette curves of an envelope surface, which is the result of solving Equations (4) and (5).

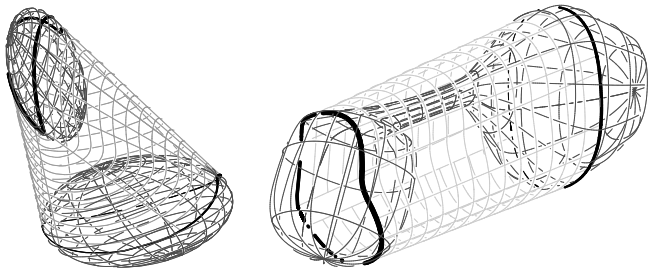


Figure 6. Two examples of convex hulls of freeform surfaces, the results of solving a system of Equations (1) – (3).

Algorithm 1
Input:
$\mathcal{F}_i, i = 1, ..., n-k,\ k = 1, 2$, having $(n-k)$ multivariate rational constraints in n variables;
τ , tolerlance of subdivision process;
Output:
$\mathcal{M}$, a k-manifold approximation in the parameteric space of $\mathcal{F}_i$;
Begin
(1) $\mathcal{S} \Leftarrow$ **ZeroSetSubdiv**($\mathcal{F}_i, \tau$);
for each n-cube $c \in \mathcal{S}$ **do**
Apply the Newton-Raphson method to project the solution point in c into the solution manifold;
Classify whether c is *normal* or *abnormal*;
end
for each n-cube $c \in \mathcal{S}$ **do**
if c is *normal* **then**
Search for all face-adjacent connections;
for every connected *normal* cubes, generate a line/triangle;
if c has face-adjacent *abnormal* cell **then**
(2) $\mathcal{D} \Leftarrow \mathcal{D} \cup \{c\}$;
end
end
for each dangling *normal* cube $d \in \mathcal{D}$ **do**
$\mathcal{N} \Leftarrow$ face-adjacent *normal* cubes at d;
$\bar{\mathcal{N}} \Leftarrow$ face-adjacent *abnormal* cubes at d;
for all *abnormal* connections **do**
$\mathcal{N} \Leftarrow \mathcal{N} \cup \{$face-adjacent *normal* cubes$\}$;
$\bar{\mathcal{N}} \Leftarrow \bar{\mathcal{N}} \cup \{$face-adjacent *abnormal* cubes$\}$;
Recursive upto depth of $(n-1)$ layers;
end
Compute a centroid of a set $\bar{\mathcal{N}}$;
$V \Leftarrow$ basis vector(s) which span the tangent space at the centroid;
Parameterize all the *normal* cubes $n \in \mathcal{N}$, over the tangent space spanned by V;
Construct a polyline/triangles from the set $\mathcal{N} \cup \{$centroid$\}$;
for all *normal* cubes in $n \in \mathcal{N}$ **do**
Delete n from $\mathcal{D}$;
end
end
return a set of polylines/triangles;
End.

Note.
(1) A function, **ZeroSetSubdiv**, in step (1) of **Algorithm1** computes a set of discrete solution points satisfying all the constraints $\mathcal{F}_i, i = 1, \cdots, n-k$.
(2) $\mathcal{D}$ represents a set of dangling *normal* cells.

the set of Equations (6). The bisector surface may have self-intersections in regions of high curvature. Figure 8(b) shows such a case. Compare the result with Figure 1. In this problem of computing bisectors, we are dealing with two equations in four variables.

The 2-manifold boundary of a swept volume can also be computed using the technique proposed in this paper. A sweep envelope boundary surface can be extracted by solving Equation (4). Figure 9(a) shows the flying motion of a boomerang and Figure 9(b) shows its corresponding envelope surface of the swept volume. A more complicated example is shown in Figure 9(c) and Figure 9(d).

A smooth blending surface between two parametric surfaces can be computed by solving Equations (7) and (8). Figure 10 shows two parametric surfaces and their smooth blending surfaces. This problem is 6-dimensional.

As mentioned in Section 4, the probability of getting degeneracies becomes considerably higher as one examines higher dimensions. Table 1 compares the number of *normal* and *abnormal* solutions according to the dimension of the problem. Since the problem of computing blending surfaces has dimension six, it has a higher chance of producing *abnormal* solution points than that of computing bisector surfaces or sweep surfaces. The complexity of the solution manifold also affects the probability of getting degeneracies. Bisectors in Figure 8(b) have self-intersections at high curvature regions of the surfaces. Thus, they have a significantly higher rate of having *abnormal* solutions than other examples of computing bisector surfaces.

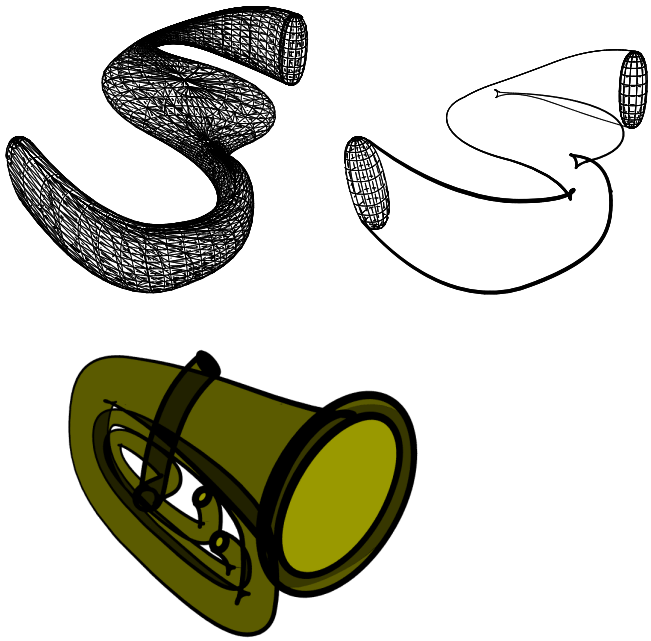


Figure 7. Perspective silhouette of a general swept volume, the result of solving Equations (4) and (5).

	Normal solution	Abnormal solution	% %
Bisector			
Fig 8(a)	444	306	40.8%
Fig 8(b)	484	1088	69.2%
Fig 8(c)	735	281	27.6%
Sweep			
Fig 9(b)	570	182	24.2%
Blending			
Fig 10(a)	1054	1530	59.2%
Fig 10(b)	918	1465	61.4%

Table 1. The number of solution points in the experimental results.

6. Conclusions and Future Work

We have shown that the Dual Contouring technique can be effectively adapted to representing 1- and 2-dimensional solution manifolds embedded in an arbitrary n-dimensional space. The extension to the n-dimensional space is simple for regular cases. Nevertheless, degenerate cases may produce thick volumes in the approximation. The problem becomes more serious in higher dimensions. Even in these cases, we can properly approximate the curves and surfaces by classifying the solution cubes into two groups and constructing a tangent space to the solution manifold at the degenerate solution point. We have also demonstrated this technique in solving practical problems arising from geometric modeling and constraint solving.

In the current work, we do not consider an adaptive generation of polylines or triangles according to the shape of the 1- or 2-manifold. By directly analyzing the curvature of the manifold, we may generate a better quality of approximation to the solution. In future work, we also plan to investigate a similar approach for general k-manifolds, for $k \geq 3$. The general solution is useful, for example, in dealing with degenerate cases that arise from tangential intersections among high-dimensional hyper-surfaces.

Acknowledgements

This work was partially supported by European FP6 NoE grant 506766 (AIM@SHAPE) and partially by the Israeli Ministry of Science Grant No. 01-01-01509.

References

[1] E.L. Allgower and S. Gnutzmann. An algorithm for piecewise linear approximation of implicitly defined two-dimensional surfaces. *SIAM J. Numer. Anal.*, Vol. 24, pp. 452–469, 1987.

[2] E.L. Allgower and K. Georg. Numerical Continuation Methods: An Introduction. Springer Verlag, Berlin, Heidelberg, 1990.

[3] G. Elber and M.-S. Kim. Bisector Curves of Planar Rational Curves. *Computer-Aided Design*, Vol. 30, No. 14, pp. 1089–1096, 1998.

[4] G. Elber and M.-S. Kim. A Computational Model for Nonrational Bisector Surfaces: Curve-Surface and Surface-Surface Bisectors. *Proc. of Geometric Modeling and Processing 2000*, Hong Kong, pp. 364–372, April 10-12, 2000.

[5] G. Elber and M.-S. Kim. Geometric Constraint Solver Using Multivariate Rational Spline Functions. *Proc. of ACM Symposium on Solid Modeling and Applications*, Ann Arbor, MI, pp. 1–10, June 4–8, 2001.

[6] S. Gibson. Constrained Elastic SurfaceNets: Generating Smooth Surfaces from Binary Segmented Data. In: MICCAI. Springer-Verlag, Berlin, 1998.

[7] G. H. Golub and C. F. Van Loan. Matrix Computation. The John Hopkins University Press, Baltimore and London, Third Edition, 1996.

[8] C. Hoffmann and J. Hopcroft. *The Potential Method for Blending Surfraces and Corners*. Geometric Modeling, in Gerald Farin (ed.), Philadelphia, SIAM Publications, 1987.

[9] J. Hoschek and D. Lasser. *Fundamentals of Computer Aided Geometric Design*. A K Peters, 1993.

[10] K. Joy and M. Duchaineau. Boundary Determination for Trivariate Solid. *Proc. of Pacific Graphics 99*, Seoul, Korea, pp. 82–91, October 5-7 1999.

[11] T. Ju, F. Losasso, S. Schaefer, and J. Warren. Dual Contouring of Hermite Data. In *Proceedings of SIGGRAPH 2002*, pp. 339–346, 2002.

[12] M.-S. Kim and G. Elber. Problem Reduction to Parameter Space. *The Mathematics of Surfaces IX (Proc. of the Ninth IMA Conference)*, R. Cipolla and R. Martin (eds), Springer, London, pp. 82–98, 2000.

[13] L. Kobbelt, M. Botsch, U. Schwanecke, and H.-P. Seidel. Feature-sensitive Surface Extraction from Volume Data. In *Proceedings of SIGGRAPH 2001*, pp. 57–66, 2001.

[14] J. Lane and R. Riesenfeld. Bounds on a Polynomial. *BIT*, Vol. 21, pp. 112–117, 1981.

[15] W. Lorensen and H. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. In *Proceedings of SIGGRAPH 1987*, pp 163–169, 1987.

[16] R. Martin and P. Stephenson. Sweeping of Three-Dimensional Objects. *Computer-Aided Design*, Vol. 22, pp. 223–234, 1990.

[17] T. Nishita, T. Sederberg, and M. Kakimoto. Ray Tracing Trimmed Rational Surface Patches. *Computer Graphics*, Vol. 24, No. 4 (Proc. of ACM SIGGRAPH 90), pp. 337–345, August 1990.

[18] H. Pottmann and J. Wallner. *Computational Line Geometry*. Springer-Verlag, Berlin, 2001.

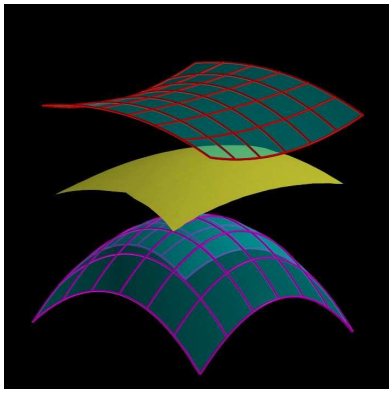
[19] T. Sederberg and R. Meyers. Loop Detection in Surface Patch Intersections. *Computer Aided Geometric Design*, Vol. 5, No. 2, pp. 161–171, 1988.

[20] T. Sederberg and A. Zundel. Pyramids that Bound Surface Patches. *Graphical Models and Image Processing*, Vol. 58, No. 1, pp. 75–81, 1996.

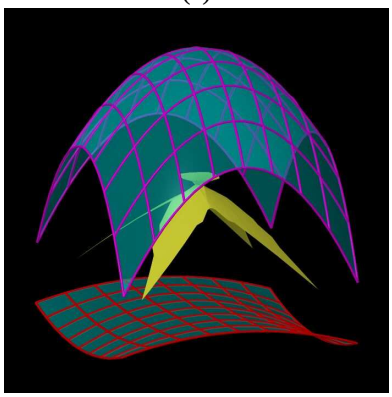
[21] E. C. Sherbrooke and N. M. Patrikalakis. Computation of the Solutions of Nonlinear Polynomial Systems. *Computer Aided Geometric Design*, Vol. 10, No. 5, pp. 379–405, 1993.

[22] J.-K. Seong, G. Elber, J.K. Johnstone, and M.-S. Kim . The Convex Hull of Freeform Surfaces. *Computing*, Vol. 72, No. 1, pp. 171–183, 2004.

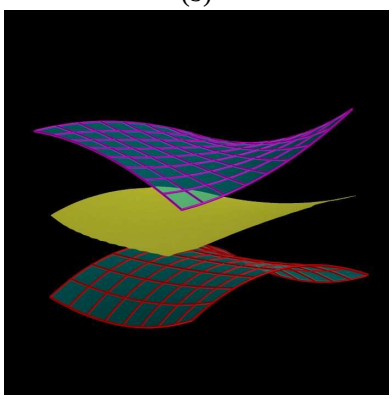
[23] J.-K. Seong, K.-J. Kim, M.-S. Kim, and G. Elber. Perspective Silhouette of A General Swept Volume. *The 5th Korea-Israel Bi-National Conference on Geometric Modeling and Computer Graphics*, Seoul, Korea, pp. 97–101, October 2004.



(a)

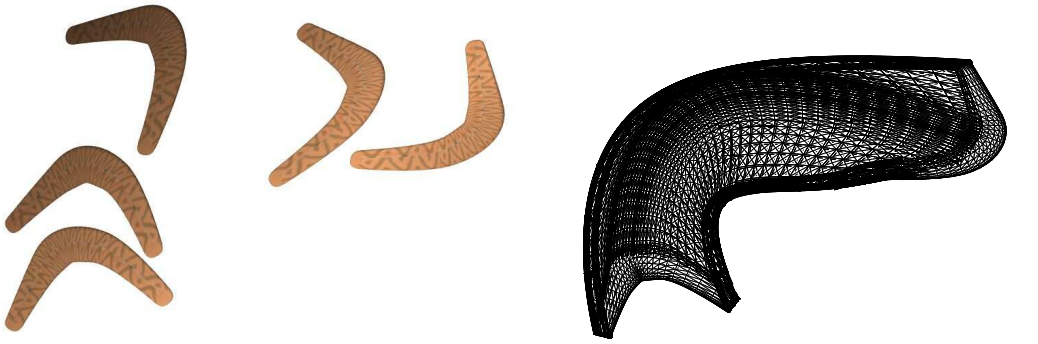


(b)

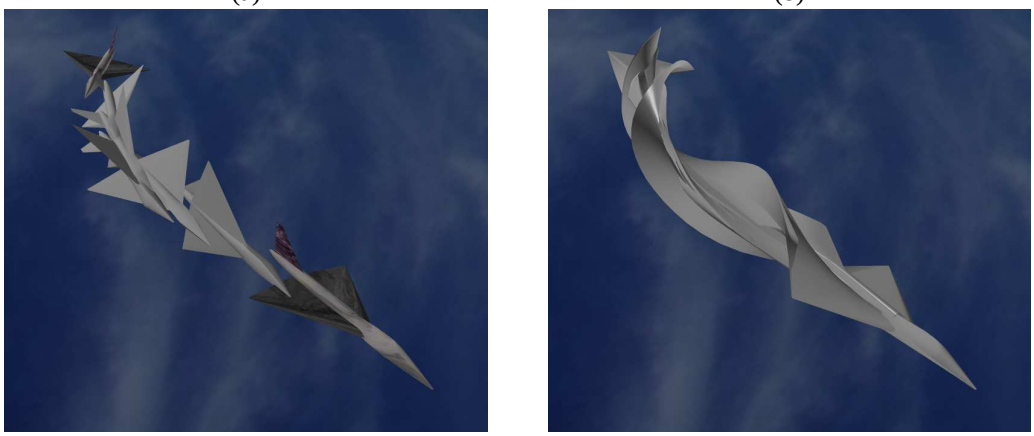


(c)

Figure 8. A few examples of bisectors between two freeform surfaces in $\mathbf{R}^3$, result of solving a set of Equations (6). Compare (b) with Figure 1, especially near the self-intersecting region.

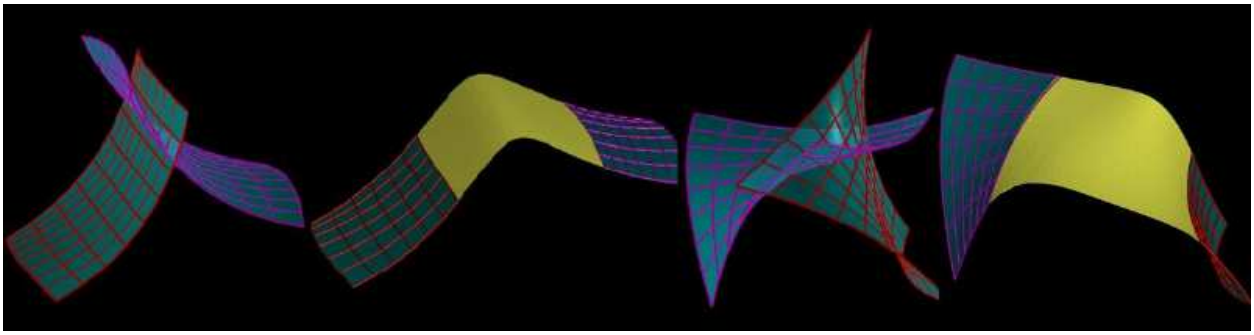


(a) (b)



(a) (b)

Figure 9. (a) Flying motion of a plane and (b) the corresponding sweep envelope surface, the result of solving Equation (4).



(a) (b)

Figure 10. Smooth blending surface between two parametric surfaces, the result of solving Equations (7) and (8).