

Implicit Modeling of Swept Surfaces and Volumes

William J. Schroeder

William E. Lorensen

GE Corporate Research & Development

Steve Linthicum

GE Aircraft Engines

Abstract

Swept surfaces and volumes are generated by moving a geometric model through space. Swept surfaces and volumes are important in many computer-aided design applications including geometric modeling, numerical cutter path generation, and spatial path planning. In this paper we describe a numerical algorithm to generate swept surfaces and volumes using implicit modeling techniques. The algorithm is applicable to any geometric representation for which a distance function can be computed. The algorithm also treats degenerate trajectories such as self-intersection and surface singularity. We show applications of this algorithm to maintainability design and robot path planning.

Keywords: Computational geometry, object modeling, geometric modeling, volume modeling, implicit modeling, sweeping.

1.0 Introduction

A swept volume is the space occupied by a geometric model as it travels along an arbitrary trajectory. A swept surface is the boundary of the volume. Swept surfaces and volumes play important roles in many computer-aided design applications including geometric modeling, numerical control cutter path generation, and spatial path planning.

In geometric modeling swept curves and surfaces are used to represent extrusion surfaces and surfaces of revolution [1][2]. More complex geometry can be generated by using higher order sweep trajectories and generating surfaces [3]. In robot motion planning, the swept volume can be used to evaluate safe paths [4][5] or establish the footprint of a robot in a

workcell. Numerical control path planning uses swept volumes to show the removal of material by a tool [6].

Swept surfaces and volumes can also be applied to resolve maintainability issues that arise during the design of complex mechanical systems. The designer needs to be able to create and visualize the accessibility and removability of individual components of the system. Typical questions related to maintainability include:

- Can a mechanic remove the spark plugs?
- Is there room for an improved power supply?
- What is the impact on the maintenance of a system if new components are included?

In maintenance design, the swept surface of the part to be removed is called the removal envelope. The removal envelope is the surface that a component generates as it moves along a safe and feasible removal trajectory. A safe trajectory is one that a component can follow without touching other components of the system. A feasible trajectory is one that can be performed by a human. Lozano-Perez [7] calls the calculation of safe trajectories the *Find-path* problem. Although this trajectory may be automatically calculated, restrictions on the trajectory's degrees of freedom [8], or the geometry of the obstacles [9] prohibit practical application to real-world mechanical systems. The technique presented here assumes that a safe and feasible trajectory is already available. For our application, we rely on computer-assisted trajectory generation using commercial computer-aided design [10] or robot simulation software.

In the next section we review related work on swept volumes and implicit modeling. Then we describe the algorithm in detail along with a discus-

sion of error and time complexity. Examples from maintainability and robot motion simulation illustrate the effectiveness of the algorithm in application.

2.0 Background

Literature from two fields: spatial path planning and implicit modeling contribute to swept volume generation.

2.1 Path Planning

Weld and Leu [11] present a topological treatment of swept volumes. They show that the representation of a swept volume in R^n generated from a n -dimensional object is reduced to developing a geometric representation for the swept volume from its $(n-1)$ -boundary. However, they point out that the boundary and interior of the swept volume are not necessarily the union of the boundary and interior of the $(n-1)$ -boundary. That is, there may exist points on the boundary of the swept volume that are interior points of the $(n-1)$ -boundary of the swept volume. Likewise, boundary points of the object may contribute to the interior of the swept volume. This unfortunate property of swept volumes limits conventional precise geometric modeling to restricted cases. Martin and Stephenson [12] recognize the importance of implicit surface models for envelope representation but seek to provide a closed solution. They present a theoretical basis for computing swept volumes, but note that complicated sweeps may take an unrealistic amount of computer time.

Wang and Wang [6] present a numerical solution that uses a 3D z-buffer to compute a family of critical curves from a moving solid. They restrict the generating geometry to a convex set, an appropriate restriction for their numerical control application. Other numerical techniques include sweeping octrees[5]. Swept octrees produce very general results at the cost of the aliasing effects due to octree modeling.

2.2 Implicit Modeling

An implicit model specifies a scalar field value at each point in space. A variety of field functions are available depending on the application. Soft objects created for computer animation [13] use a field that is unity at a modeling point and drops to zero at a specified distance from the point. The variation of the fields is typically a cubic polynomial. Usually these fields are represented on a regular sampling (i.e., volume) as described by Bloomenthal [15].

Surface models are extracted using standard iso-surface techniques such as marching cubes [16]. If the field values are distance functions to the closest point on the model, offset surfaces can be created by choosing a non-zero iso-surface value [14].

3.0 Algorithm

The goal of the algorithm can be described as follows. Given a geometric model and a trajectory defined by a sequence of continuous transformations, generate the swept volume, the volume occupied as the model travels along the trajectory, and the swept surface, the boundary of the swept volume.

The swept surface algorithm is implemented in three basic steps (Figure 1).

1. Generate an implicit model from the original geometric model. We use implicit techniques that assign a distance value from each voxel to the surface of the object. The geometric model may be of any representational form as long as a distance function can be computed for any point. Common representations include polygonal meshes, parametric surfaces, constructive solid models, or implicit functions.
2. Sweep the implicit model through the workspace. The workspace is a volume constructed to strictly bound the model as it travels along the sweep trajectory. The sweeping is performed by repeatedly sampling the implicit model with the workspace volume as it is transformed along the sweep trajectory. The end result is a minimum distance value at each voxel in the workspace volume.
3. Generate the swept surface using the iso-surface extraction algorithm marching cubes. The value d of the iso-surface is a distance measure. If $d = 0$, the surface is the swept surface. For $d \neq 0$, a family of surfaces offset from the swept volume is created.

A detailed examination of each of these steps follows.

3.1 Generating the Implicit Model

This step converts the geometric model into a sampled distance function in a 3D volume of dimension (n_1, n_2, n_3) . We refer to this representation as the implicit model V_I .

Our approach is similar to Bloomenthal [15]. Any n -dimensional object in R^n (we assume here $n = 3$) can be described by an implicit function $f(p) = 0$ where the function f defines the distance of point

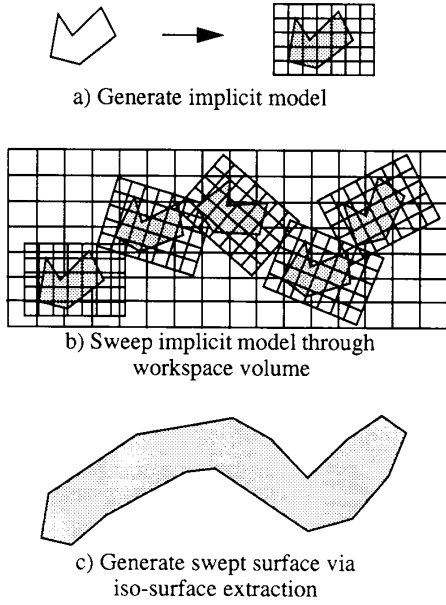


Figure 1. Overview of swept surface generation.

$p \in R^3$ to the surface of the object. To develop the implicit model we compute the function $f()$ in a 3D volume of dimension (n_1, n_2, n_3) . For geometric models having a closed boundary (i.e., having an inside and an outside), $f()$ can generate a signed distance; that is, negative distance values are inside the model and positive values are outside of the model.

Geometric representations often consist of combinations of geometric primitives such as polygons or splines. In these cases $f()$ must be computed as the minimum distance value as follows. Given the n primitives $G = (g_1, g_2, \dots, g_n)$ defining the n distance values (d_1, d_2, \dots, d_n) at point p , choose the minimum value.,

$$d = f(p) = \text{Min}(|d_1|, |d_2|, \dots, |d_n|)$$

For an implicit model, the minimum is equivalent to the union of the scalar field [15].

The implicit model can be generated for any geometric representation for which the distance function $f()$ can be computed. One common representation is the polygonal mesh. Then $f(p)$ is the minimum of the distances from p to each polygon (Figure 2). For more complex geometry whose distance function may be expensive or too complex to compute, the model can be sampled at many points, and then the points can be used to generate the distance function.

Figures 6 and 7 illustrate the generation of an

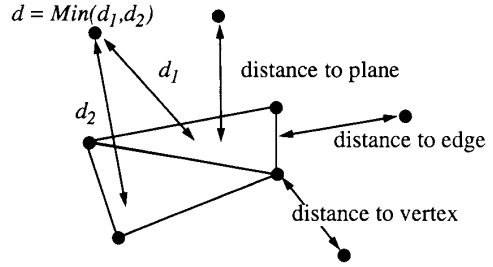


Figure 2. Computing distance function $f()$ for polygonal mesh.

implicit model from a polygonal mesh. Figure 6 shows the original mesh consisting of 5,576 polygons. The implicit model is sampled on a volume V_I at a resolution of 100^3 . An offset surface of the implicit model is generated using a surface value $d = 0.25$ in Figure 7. In this example a non-negative distance function has been used to compute the implicit model. As a result, both an outer and inner surface are generated. Any closed surface will necessarily generate two or more surfaces for some values of d if $f()$ is non-negative.

3.2 Computing the Workspace Volume

The workspace volume V_w is generated by sweeping the implicit model along the sweep trajectory, and sampling the transformed implicit model. V_w must be sized so that the object is strictly bounded throughout the entire sweep. One simple technique to size V_w is to sweep the bounding box of the geometric model along the sweep trajectory and then compute a global bounding box.

The sweep trajectory ST is generally specified as a series of transformations $ST = \{t_1, t_2, \dots, t_m\}$. Arbitrary transformations are possible, but most applications define transformations consisting of rigid body translations and rotations. (The use of non-uniform scaling and shearing creates interesting effects.)

The implicit model travels along the sweep trajectory in a series of steps, the size of the step dictated by the allowable error (see Discussion). Since these transformations $\{t_1, t_2, \dots, t_m\}$ may be widely separated, interpolation is often required for an intermediate transformation t' . We recover positions and orientations from the provided transformations, and interpolate these recovered values.

The sampling of the implicit model is depicted in Figure 3. The values in V_w are initialized to a large positive value. Then for each sampling step, each point in V_w is inverse transformed into the coordi-

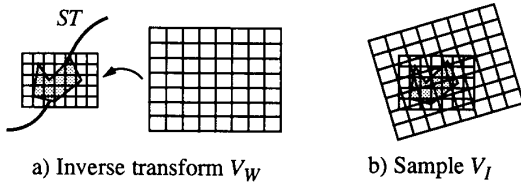


Figure 3. Sampling the implicit volume.

nate system of V_I . The location of the point within V_I is found and then its distance value is computed using tri-linear interpolation. As in the implicit model, the value of the point in V_w is assigned the minimum distance. The result is the minimum distance value at each point in V_w seen throughout the sweep trajectory.

An alternative workspace volume generation scheme calculates the distance function $f()$ directly from the transformed geometric model, eliminating the need for V_I entirely. Although this can be efficient when $f()$ is inexpensive to compute, we have found that in our applications the performance of the algorithm is unsatisfactory. Sampling the implicit model into the transformed workspace volume improves the performance of the algorithm significantly, since the sampling is independent of the number of geometric primitives in the original model, and the cost of tri-linear interpolation is typically much smaller than the cost of evaluating $f()$.

3.3 Extracting Swept Surfaces

The last step in the algorithm generates the swept surface from the workspace volume. Since V_w represents a 3D sampling of distance function, the iso-surface algorithm marching cubes is used to extract the swept surface. Choosing $d = 0$ generates the surface of the swept volume, while $d \neq 0$ generates offset surfaces. In many applications choosing $d > 0$ is desirable to generate surfaces offset from the swept volume by a certain tolerance.

As with any iso-surface extraction algorithm, the surfaces often consist of large numbers of triangles. Hence we often apply a decimation algorithm [17] to reduce the number of triangles representing the surface.

4.0 Discussion

4.1 Degenerate Trajectories

Degenerate trajectories, as described by Martin and Stephenson [12], are self-intersecting trajecto-

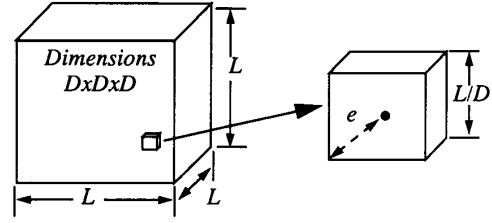


Figure 4. Sampling error in 3D volume.

ries or trajectories that result in surface singularities. Self-intersection occurs when the generating geometry intersects itself as it travels along the sweep trajectory. Surface singularities occur when non-volume filling geometry is created, for example when a plane travels perpendicular to its normal.

The algorithm presented here correctly treats degenerate trajectories. Conventional analytical techniques require sophisticated mathematical techniques that do not yet have general solutions. Our algorithm uses simple set operations that are immune to both types of degeneracies.

4.2 Error Analysis

Due to sampling errors and differences in representation, the generated swept surface is an approximation to the actual surface. Sampling errors are introduced at three points: 1) when creating V_I from the geometric model, 2) when sampling V_I into the workspace volume V_w , and 3) when incrementing the transformation along the sweep trajectory ST . Representation errors are due to the fact that the swept surface is a polygonal mesh, while the original representation can be combinations higher-order surfaces or even implicit functions. Here we address sampling errors only, since representational errors depend on the particular geometric representation, which is beyond the scope of this paper.

The sampling error bounds of the distance function in a 3D volume is the distance from the corner of a cube formed from eight neighboring voxels to its center (Figure 4). Without loss of generality we can assume that the volume size is length L , and that the volume dimensions are $n_1, n_2, n_3 = D$. It follows that the cube length is then L/D . The maximum sampling error e at a voxel is then

$$e \leq \frac{\sqrt{3}}{2} \left(\frac{L}{D} \right)$$

The error introduced by stepping along ST depends upon the nature of the transformation. Translations give rise to uniform error terms, but rotation gener-

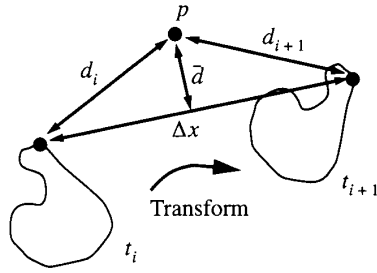


Figure 5. Approximating stepping error.

ates variable error depending upon distance from center of rotation. The stepping error can be estimated by linearizing the transformation and assuming that the maximum positional change of any point is Δx (Figure 5.). If the actual distance value to some point p in V_w is given as \bar{d} , and the distance functions are d_i and d_{i+1} at the points t_i and t_{i+1} along ST , then the maximum error occurs at $\Delta x/2$ and is given $\bar{d} = d_i^2 - \left(\frac{\Delta x}{2}\right)^2$, the error is bounded by

$$e = d_i - \sqrt{d_i^2 - \left(\frac{\Delta x}{2}\right)^2} \leq \left(\frac{\Delta x}{2}\right)$$

The value of Δx can be estimated by transforming the bounding box of the model and computing the displacement of the extreme points.

These error terms can be combined to provide an estimate to the total error:

$$e_{\text{tot}} \approx \frac{\sqrt{3}}{2} \left(\frac{L_I}{D_I} + \frac{L_W}{D_W} \right) + \frac{\Delta x}{2}$$

The terms L_I , L_W , are the lengths and D_I , D_W are the dimensions of the implicit and working volumes, respectively. These terms arise when creating V_I from the geometric model and when sampling V_I into the workspace volume V_w .

4.3 Complexity Analysis

The bulk of computation occurs when sweeping the implicit model through the workspace volume. Generating the implicit model and the final iso-surface are one-time events at the beginning and end of the algorithm.

Assuming that $f()$ can be computed in constant time for any point, the implicit model can be generated in time $O\langle D_I^3 \rangle$. If the computation of $f()$ is proportional to the number of geometric primitives n_p in the model, the time complexity is $O\langle n_p D_I^3 \rangle$. The complexity of sweeping is $O\langle n_s D_w^3 \rangle$ where n_s is the

number of steps in the sweep. The final step, iso-surface extraction, is a function of the size of the working volume $O\langle D_w^3 \rangle$.

4.4 Multiple Surface Generation

More than one connected swept surface may be created for certain combinations of geometry and iso-surface value d . As described previously if $f(p) \geq 0$ for all p , both an inner and outer surface may be generated. If the geometric model is non-convex, then multiple inside and outside surfaces may be created. There will be at least one connected outer surface, however, that will bound all other surfaces. In some applications such as spatial planning and maintainability design this result is acceptable. Other applications require a single surface.

One remedy that works in many cases is to compute a signed distance function $f()$ where values less than zero occur inside the object. This approach will eliminate any inner surfaces when $d > 0$. Generally this requires some form of inside/outside test which may be expensive for some geometric representations. Another approach to extract the single bounding surface is to use a modified form of ray casting in combination with a surface connectivity algorithm.

5.0 Results

Our initial application for swept surfaces was for maintainability design. In this application the swept surface is called a removal envelope. The removal envelope graphically describes to other designers the inviolable space required for part access and removal. We have since used the swept surfaces to visualize robot motion. Three specific examples follow. The first two examples use a trajectory planned using the McDonnell Douglas Human Modeling System.

5.1 Fuel/Oil Heat Exchanger

The part shown in Figure 6 is a fuel/oil heat exchanger used in an aircraft engine. Maintenance requirements dictate that it must be able to be replaced within 30 minutes while the aircraft is at the gate. The implicit model of the heat exchanger is shown in Figure 7. Figure 8 shows the resulting swept surface. Both the implicit model and the workspace volume were generated at a resolution of 100^3 . The sweep trajectory contained 270 steps. The swept surface, generated with an offset of $d=0.2$ inch, consists of 15,108 triangles.

Figure 9 shows the removal envelope positioned within surrounding parts. Of particular interest is the

(See color plates, page CP-4.)

pipng that has been rerouted around the removal envelope. The figure demonstrates the small clearances typical in complex mechanical environments. Because of tolerances in these systems, we generally choose to create offset swept surfaces. Currently we use no quantitative method to choose the offset value and rely on design experience.

5.2 Fuel Nozzle

Figure 10 shows another application of swept surfaces to maintainability design. A proposed design change to the aircraft engine pre-cooler (shown in turquoise) impacted the removal of the number 3 fuel nozzle. The removal path consists of many rotations and is self-intersecting at many points.

The fuel nozzle was modeled using 6,100 polygons. The implicit model was generated at a volume resolution 50^3 and the workspace volume at 100^3 . A total of 600 steps was taken along the sweep trajectory. The offset swept surface of 16,572 triangles was generated with a value $d=0.2$ inch.

5.3 Robot Motion

Swept surfaces are effective tools for visualizing complex robot motion. In Figure 11 the swept surfaces of the end effector, hand, wrist, and arm of the robot are shown simultaneously. Each robot part was sampled at 50^3 . The workspace volume was sampled at 100^3 . The total number of triangles representing the four surfaces was 681,800 using a distance value $d=1$ inch (over a workspace volume size $L_w = 140$ inches.)

6.0 Conclusion

We have demonstrated an algorithm that generates swept surfaces and volumes from any geometric representation for which a distance function can be computed. This includes such common forms as parametric surfaces, polygonal meshes, implicit representations, and constructive solid models. The algorithm treats complex trajectories including self-intersection and surface singularities. We have successfully applied the algorithm to a variety of complex applications including maintainability design and robot spatial planning.

Acknowledgments

Steve Rice of McDonnell Douglas assisted in the extraction of removal paths from the McDonnell Douglas Human Modeling System.

References

- [1] D. F. Rogers and J. A. Adams. *Mathematical Elements for Computer Graphics*. McGraw-Hill Publishing Co., New York, 1990.
- [2] M. E. Mortensen. *Geometric Modeling*. John Wiley & Sons, New York, 1985.
- [3] S. Coquillart. A control-point-based sweeping technique. *IEEE Computer Graphics and Applications* 7(11):36-45, November 1987.
- [4] T. Lozano-Perez. Spatial Planning: A configuration space approach. *IEEE Trans. on Computers* C-32(2):108-120, February 1983.
- [5] M. Celenk and W. Sun. 3D visual robot guidance in dynamic environment. *Proceedings of IEEE Int'l Conference on Systems Engineering*, pp 507-510, August 1990.
- [6] W. P. Wang and K. K. Wang. Geometric modeling for swept volume of moving solids. *IEEE Computer Graphics and Applications* 6(12):8-17, 1986.
- [7] T. Lozano-Perez and M. A. Wesley. An algorithm for planning collision-free paths among polyhedral objects. *Comm. of the ACM* 22(10):560-570, October 1979.
- [8] J. Lengyel, M. Reichert, B. R. Donald, and D. P. Greenberg. Real-time robot motion planning using rasterizing computer graphics hardware. *Computer Graphics* 24(4):327-335, August 1990.
- [9] R. A. Brooks. Solving the find-path problem by good representation of free space. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-132(3):190-197, March/April 1983.
- [10] McDonnell Douglas Human Modeling System Reference Manual. Report MDC 93K0281. McDonnell Douglas Corporation, Human Factors Technology. Version 2.1, July 1993.
- [11] J. D. Weld and M. C. Leu. Geometric representation of swept volumes with application to polyhedral objects. *Int'l J. of Robotics Research*, 9(5):105-117, October 1990.
- [12] R. R. Martin and P. C. Stephenson. Sweeping of three-dimensional objects. *Computer-Aided Design* 22(4):223-234, May 1990.
- [13] B. Wyvill, C. McPheeters, G. Wyville. Animating soft objects. *The Visual Computer* 2:235-242.
- [14] B. A. Payne and A. W. Toga. Distance field manipulation of surface models. *IEEE Computer Graphics and Applications*, 12(1):65-71, January 1993.
- [15] J. Bloomenthal. Polygonization of implicit surfaces. *Computer Aided Geometric Design*, 5(4):341-355, November 1988.
- [16] W. E. Lorensen and H. E. Cline. Marching cubes: a high resolution 3d surface construction algorithm. *Computer Graphics* 21(4):163-169, July 1987.
- [17] W. J. Schroeder, J. A. Zarge, and W. E. Lorensen. Decimation of triangle meshes. *Computer Graphics* 26(2):65-70, July 1992.

(See color plates, page CP-4.)