APIC'95, El Paso, Extended Abstracts, A Supplement to the international journal of Reliable Computing

Rendering Swept Volumes Using Interval Methods

Wolfgang A. Sommerer[1] and Monika Kerbl[2]

I. Introduction

Most manufacturing processes are based on the principle of *material removal.* Different process types exist, each of them having its importance in diverse fields of the industry.

One process type in common use is milling. This process uses a rotating tool which is moved along some path to produce the desired product shape. Other processes that must be mentioned here are turning (where a moving tool removes material from a rotating part) and sheet punch, to mention just a few. In case of the milling process the motion of the tool can be performed in three, five or more axes (i.e., degrees of freedom).

These manufacturing processes have one common feature: material removal through movement of some tool.

Most of the machines in use today are *programmable*, i.e., have a controlling unit attached which directs the tool according to the program that describes the desired part (this program is called an *NC program*). Different methods for programming these controllers exist.

In almost common use today are CAD/CAM systems which semiautomatically create NC programs by analyzing the part shape. Since these tools allow the easy creation of complex programs (i.e., thousands of source lines), program validation has become a major problem.

Traditionally, program validation is done by stepwise execution of the program on the target machine under close control by an operator. The possible damage done to the machine by an undetected program error can be severe. Also, the cost involved in using machines which are part of the normal production cycle are significant, thus making alternative approaches to program validation attractive.

One such approach is program simulation by solid modeling [1]. The machining process is simulated by performing Boolean set difference operations between the workpiece and the swept tool volumes.

The authors are with the [1]Department of Computer Science, University of Linz, Austria, and with the [2]XSoft EDV-Software Entwicklungsges.m.b.H., Vienna, Austria.

Different data structures and algorithms have been proposed for representing and analyzing swept volumes [2, 3, 4, 5]. Due to the inherent complexity of the problem to be solved, these approaches made trade-offs either in generality or time-efficiency. We present a general, time efficient solution which makes use of interval analytic methods for rendering swept volumes.

II. Swept Volume

In a 2D or 3D space, a *swept volume* is described as the totality of all points that belong to the trace of a generating object moved along some path.

Figure 1 shows a simple example of a sweep operation in Euclidean 2D space. The generating object moves along the path, indicated by the dashed arc, from position 1 to position 4. The swept area is the set of all points covered by the generator during its movement along the given path.

In Euclidean 3D space, we can describe a solid by the implicit equation [6]:

$$f(x,y,z) \leq 0, \quad f : R^3 \to R. \tag{1}$$

Given a coordinate transformation function

$$m : R^3 \times \tau \to R^3, \tag{2}$$

($\tau = [t_S, t_E]$) that describes the motion of the solid, the resulting swept volume can be defined as follows:

$$\{m((x_0,y_0,z_0),t) \,|\, t \in \tau \wedge f(x_0,y_0,z_0) \leq 0\}. \tag{3}$$

Let us denote the *inverse* transformation by w; in other words, we define $w((x,y,z),t)$ as the point (x_0,y_0,z_0) for which $m((x_0,y_0,z_0),t) = (x,y,z)$. In terms of w, we can describe the following criterion for checking whether a given point $\vec{r} = (x,y,z)$ belongs to the swept volume:

To apply this criterion, we must consider the set of all the values $f(x_0,y_0,z_0)$ for all points (x_0,y_0,z_0) from the set

$$S(\vec{r}) = \{w((x,y,z),t) \,|\, t \in \tau\}. \tag{4}$$

Then:

- If there exists a point (x_0, y_0, z_0) from $S(\vec{r})$ for which $f(x_0, y_0, z_0) < 0$, then the given point $\vec{r}$ is *in* the swept volume.
- If the following two conditions are true:
  - there exists a point (x_0, y_0, z_0) from $S(\vec{r})$ for which $f(x_0, y_0, z_0) = 0$, and
  - for all points $(x_0, y_0, z_0) \in S(\vec{r})$, we have $f(x_0, y_0, z_0) \geq 0$,

  then the given point $\vec{r}$ is *on* the surface of the swept volume.
- If for all $(x_0, y_0, z_0) \in S(\vec{r})$, we have $f(x+_0, y_0, z_0) > 0$, then the given point $\vec{r}$ is *out*, i.e., outside the swept volume.

III. Rendering Swept Volumes

A. The above-described criterion cannot be applied directly

Rendering algorithms approximate objects and scenes so that they can be displayed on raster graphics display devices. In particular, a rendering algorithm computes color and other information (e.g. depth values, etc.) for every pixel of the screen or image, thus generating a shaded image of the object. Unfortunately, a computer algorithm which renders a swept volume cannot apply the above-described criterion directly because an infinite number of points would have to be classified for every pixel in the image.

B. Interval methods

Interval arithmetic methods [4, 9] extend the scope of implicit forms from point checking to range checking [6]. This allows the development of efficient algorithms which navigate along the envelope surface. Namely, assume that we are given a box $X \times Y \times Z$, where $X = [x_A, x_B]$, $Y = [y_A, y_B]$, and $Z = [z_A, z_B]$. Then:

- This box contains an element of the swept volume's surface if and only if

$$0 \in [f([w((X, Y, Z), \tau)])], \tag{5}$$

  where:
  - $[w(X, Y, Z, \tau)]$ denotes the range of the function $w(x, y, z)$ when $x \in X$, $y \in Y$, $z \in Z$, and $t \in \tau$, and
  - $[f(I)]$ denote the range of the function $f(x, y, z)$ when $(x, y, z) \in I$.
- Similarly, this box is *in* the swept surface iff

$$[f([w((X, Y, Z), \tau)])] \subseteq (-\infty, 0).$$

- The given box is *out*side the swept surface iff

$$[f([w((X, Y, Z), \tau)])] \subseteq (0, \infty)$$

.

Using interval computations, we can get an (outer) interval estimate $[e^-, e^+]$ for the desired interval $J = [f([w((X, Y, Z), \tau)])]$.

- If $e^+ < 0$, then $J \subseteq [e^-, e^+] \subseteq (-\infty, 0)$ and therefore, the box is *in*.
- If $e^- > 0$, then $J \subseteq [e^-, e^+] \subseteq (0, \infty)$ and therefore, the box is *out*.
- If $e^- \leq 0 \leq e^+$ (i.e., if $0 \in [e^-, e^+]$), then this can mean one of the two things:
  - the desired interval J may also contains 0; in this case, the given box intersects with the surface of the swept volume; or
  - the desired interval $J \subseteq [e^-, e^+]$ may not contain 0 at all; the fact that $0 \in [e^-, e^+]$ is in this case caused by the fact that interval computations often *overestimate*.

We will use this idea to render the swept volume.

The smaller the intervals that we start with, the better are the estimates produced by interval computations. Therefore, if the resulting interval contains 0, then, to make the results more accurate, we must consider smaller boxes. We use this idea in the following reasonably simple divide and conquer algorithm.

C. The general description of the proposed algorithm

Our algorithm belongs to the category of ray casting algorithms [8], and consists of two phases:

- In the first phase, a ray, which is perpendicular to the view plane, is cast through every pixel in the image and the entry point of each ray (i.e., the Z value of the first entry into the object) is calculated.
- In the second phase, pixel color intensity is determined by calculating surface gradient values. This can be easily performed by fitting an algebraic surface (typically linear) through neighboring points [7].

In the remainder of this paper, we will concentrate on the calculation of depth values given an implicit representation.

D. Ray casting

Let us first describe how *ray casting* is done for a single ray. For a given view, we cast a ray through any discrete point (i.e., pixel) of the image (see Fig. 2). The ray is perpendicular to the view plane and bounded by preset start and end values. Transforming the ray from view space (x, y) to world coordinates (x, y, z) results in a rectilinear region of the world space, which is used for classification via formula (5).

- If the result of the classification of the ray is *in* or *out*, we discard the ray, because it doesn't cross the envelope surface.
- Otherwise, the ray is divided into two sub-rays, and the above-described interval classification algorithm is applied to each sub-ray.

This process of classifying and dividing is continued until the length of the ray is sufficiently small.

Since we are looking for the minimum depth value, the front sub-ray (i.e., the part closer to the observer) of the division is examined first. If we detect the envelope surface to cross this part, the process is continued for this part (i.e, the second part of the current ray is discarded). Otherwise, the second part is examined.

E. Classifying regions in 3-D space

E.1. A problem

Classifying a *region* of Euclidean 3D space against a swept volume is not a trivial task. The straightforward solution would be the direct application of formula (5) (i.e., to inverse transform the region for τ and use the result for classification w.r.t. the generator).

The problem with this approach is that the shape generated by an inverse transformation is usually curvi-linear. So, if we approximate the resulting set by a parallelepiped, we are likely to extend the generating object. Figure 3 shows a case where this straightforward algorithm, when applied to a region that is located completely outside of the swept volume, results in *on*, because the inverse transformation for τ generates a region which extends into the generating object.

E.2. Our solution to this problem: main idea

Our approach to overcome this problem and achieve reliable classifications is structured as follows:

1. Determine if the moving object enters the region to be classified during the time interval.
2. Determine if there exists a point in time where the moving object covers the region completely.

If the result of the first query is "no", the examined region is completely outside of the swept volume. Otherwise, the second question must be answered.

- If the answer to that question is also "no", then the region is only partially covered by the swept volume, thus containing part of the envelope surface.
- If the answer to the second query is "yes", the region is completely inside the swept volume.

E.3. Our solution to this problem: details

To answer the above-described two questions, we have to check the existence of certain critical points in time. We define these critical points as follows (see also Fig. 4):

1. t_1: generator enters region.
2. t_2: generator covers region completely for the first time.
3. t_3: generator covers region completely for the last time.
4. t_4: generator leaves region.

The algorithm for checking the existence of t_1 (and t_4 as well) is as follows:

1. Inverse transform the region for τ and classify the result w.r.t. the generator.
2. If the result of the classification is *on*, subdivide in t, and repeat steps 1, 2, and 3 by substituting the result of the subdivision for τ or, if the time interval grew sufficiently small, terminate and take the lower bound (in case of searching for t_4, the upper bound) of the interval as the solution.
3. If the result is *in*, the lower bound (or the upper bound in case of searching for t_4) of the time interval is the solution.

In case of searching for t_1, intervals that are earlier in time must be checked first. If the search is for t_4, the strategy has to be reversed.

Finding t_2 and t_3 is done by identifying either a subregion of the region to be classified which is not covered in τ, or a subinterval of τ where the region is completely covered.

- If we find a subregion, then the region is partially covered by the swept volume.
- If we have identified a subinterval of τ, this means that the region is completely covered by the swept volume.

IV. Conclusion

Due to the increasing usage of computer based manufacturing technology in the shop floor as well as in the design department, simulation of manufacturing processes will become more important because it increases reliability and saves expensive machine time. Complete simulation (i.e., a simulation able to represent and process all important parameters) of these processes is still an unsolved problem because the technology in use today is based on simplifying assumptions (e.g. restrictions regarding tool shapes).

We have presented a general, time efficient solution to a basic requirement of such a simulation system. Although rendering swept volumes is not sufficient for performing Boolean set operations on solids, the extension of the algorithm to find all entry and exit points of the ray is relatively easy.

The ideas presented in this paper are currently being implemented as part of a system which allows the verification of complex NC programs. We based this system on object oriented methods using C++ as our implementation language of choice.

References

[1] M. Mäntylä, **An Introduction to Solid Modeling**, Computer Science Press, Rockville, MA, 1988.

[2] W. P. Wang, K. K. Wang, "Geometric Modeling for Swept Volume of Moving Solids", *IEEE Computer Graphics and Applications*, December 1986, pp. 8-17.

[3] J. Menon, R. J. Marisa, J. Zagajac, "More Powerful Solid Modeling through Ray Representations", *IEEE Computer Graphics and Applications*, May 1994, pp. 22-35.

[4] J. M. Snyder, **Generative Modeling for Computer Graphics and CAD**, Academic Press, San Diego, CA, 1992.

[5] T. van Hook, "Real-Time Shaded NC Milling Display", *Proceedings of SIGGRAPH'86*, Vol. 20, No. 4, August, 1986, pp. 15-20.

[6] T. Duff, "Interval Arithmetic and Recursive Subdivision for Implicit Functions and Constructive Solid Geometry", *Computer Graphics*, July 1992, pp. 131–137.

[7] J. D. Foley, A. van Damm, S. K. Feiner, J. F. Hughes, **Computer Graphics**, 2nd ed., Addison-Wesley, Reading, MA, 1990, pp. 471 ff.

[8] J. D. Foley, A. van Damm, S. K. Feiner, J. F. Hughes, **Computer Graphics**, 2nd ed., Addison-Wesley, Reading, MA, 1990, pp. 701 ff.

[9] R. E. Moore, **Interval Analysis**, Prentice Hall, Englewood Cliffs, NJ, 1966.