Computing Swept Volumes for Sensor Planning Tasks

Steven Abrams   Peter K. Allen*
Center for Research in Intelligent Systems
Computer Science Department
Columbia University
New York, NY 10027

Abstract

This paper discusses the computation of the volumes swept by polyhedral objects moving through space and how these volumes can be used for sensor planning in active environments. It presents some new results in the computation of the approximate volumes swept by polyhedral objects as they undergo arbitrary motions. An algorithm is presented which approximates these volumes under a restricted class of motion, along with a discussion of how it can be extended to handle all types of motion. Results computed with this algorithm are also presented. Finally, we discuss methods for guaranteeing that the approximation is within some desired tolerance of the actual surface.

1 Introduction

As part of our ongoing research in sensor planning, we have been looking at methods of computing viewpoints which allow the monitoring of specified object features while other objects in the environment are moving [Abrams *et al.*, 1993, Abrams and Allen, 1991]. We have been pursuing methods based on temporal intervals, in which a portion of the task is selected to be monitored. The system computes the volumes swept by all moving objects during this interval and computes viewpoints which avoid occlusion by these swept volumes. Such viewpoints would be valid for the entire task interval. By examining appropriate time intervals, we break the *Dynamic Sensor Planning* problem down into a series of static problems.

The basic method used for computing viewpoints for each of these static problems is the Machine Vision Planning system developed earlier [Tarabanis *et al.*, 1994c, Tarabanis *et al.*, 1994d, Tarabanis *et al.*, 1994b, Tarabanis *et al.*, 1991], although any viewpoint planning system could be used, such as those presented in [Cowan, 1988, Cowan and Bergman, 1989, Cowan and Modayur, 1993, Hutchinson and Kak, 1989, Ikeuchi and Kanade, 1989, Niepold *et al.*, 1987]. A more complete survey of sensor planning systems can be found in [Tarabanis *et al.*, 1994a].

The general idea behind our approach to dynamic sensor planning is best described by the following algorithm:

Temporal Interval Search

1. Compute the volumes swept by all moving objects during the task interval T.
2. Use MVP to compute a valid, unoccluded viewpoint using these swept volumes as if they were actual objects.
3. If MVP can successfully find a viewpoint, use this viewpoint for the entire time interval T.
4. If no such viewpoint is obtainable, divide the time interval in half yielding $T_1 = [t_0, t_{n/2}]$. Go back to step 1 using interval T_1.
5. If the entire time interval T has been planned, we are finished. If not, go to step 1 using the remaining portion of the the original interval T.

Thus, a swept volume algorithm useful for this research must produce results fairly quickly (since it may be called often), and must produce polyhedral models of these volumes. As part of our previous research, we used a "stepwise union" algorithm to approximate the volume swept by a moving object.

*This work was supported in part by DARPA contract DACA-76-92-C-0007, NSF grants CDA-90-24735 and IRI-93-11877, IBM Corporation, Sun Microsystems, North American Philips Laboratories, Siemens Corporation and Rockwell International.

That is, we computed the union of the moving objects at each of n stationary positions along their trajectories to compute a polyhedral approximation. This was primarily to allow a proof-of-concept of the temporal interval search as a viable dynamic sensor planning method. The expense of this algorithm, (exponential in the number of steps taken), proved to be a major obstacle to further sensor planning experimentation. Therefore, we have embarked upon the present research task of robustly and efficiently computing the volumes swept by moving polyhedral objects.

2 Swept Volume Definitions

A "swept volume" is a fairly intuitive concept. Nevertheless, it is important to establish a formal definition so as to clarify exactly what it is that we will be computing. If Q is a trajectory through $\mathbf{R}^d$ which includes rotations, Q can be parameterised by t as

$$Q(t) = H(t) \tag{1}$$

where $H(t)$ is a $d+1$-dimensional homogeneous matrix whose skew factors are all zero and whose scale factor is 1, defining a position and orientation in $\mathbf{R}^d$. Such matrices are of the form

$$\begin{array}{ccc} & R & T \\ 0 & \ldots\ 0 & 1 \end{array} \tag{2}$$

Where R is a d-dimensional orthonormal rotation matrix and T is a $d \times 1$ translation vector.

For every t, this gives us a new translation and rotation. That is, we have a new position and orientation at every time t. We can now define M_H to be some set $M \subset \mathbf{R}^d$ transformed by the homogeneous matrix H, i.e.:

$$M_H = \{Hm : m \in M\} \tag{3}$$

We can now define the sweep of a set $M \subset \mathbf{R}^d$ over a set of transformations Q which includes both translations and rotations (notated $\mathcal{S}(M,Q)$) as:

$$\mathcal{S}(M,Q) = \bigcup_{H \in Q} M_H \tag{4}$$

Intuitively, we are dragging M through a path and at every point in the path, M may have a different orientation.

The cases which will be most interesting to us are those in which the transformations change continuously over time, since that is the way in which objects move — continuously. Because of nature of the analysis which we will be doing for sensor planning, we will be concerned with computing a boundary representation of the volume swept by M.

In the remainder of this paper, we will present a summary of previous work examining similar types of swept volume computations. We will then present some observations we have made about the class of surfaces which can arise in the types of sweeping in which we are interested. Next, we will present an algorithm which computes approximations to swept volumes for certain types of motion. Finally, we shall present some properties of the swept volumes which we hope will lead to an algorithm for robustly computing a polyhedral approximation of the volume swept, with a guaranteed error bound.

3 Related Sweeping Research

A number of previous researchers have examined the problem of computing swept volumes. Korein [1985] computes a polyhedral approximation to the swept volume of a polyhedron rotating about either an axis or radially rotating about a point. Kaul [1993] has developed algorithms for computing Minkowski sums of polyhedral objects. Since the Minkowski sum of an object and a straight line is exactly the volume swept out by moving that object along that path (while keeping the object in a fixed orientation), Kaul's methods can be used to compute translational sweeps robustly.

Wang and Wang [1986] discuss the modeling of the volume swept by a moving object (a *generator*), by considering the computation of the envelope surface of the family of surfaces defined by the moving boundaries of the object being swept. Martin and Stephenson [1990] extend this theory to include more general surfaces, also by using envelope theory. They explain how computer algebraic techniques can sometimes be used to obtain an implicit equation for the envelope of a moving surface. In addition, their technique will often require the employment of exponential algorithms in computer algebra for computing even simple sweeps.

Weld and Leu [1990] set forth some theory for computing the volume swept by a moving polyhedron. They show that the volume swept by a moving poly-

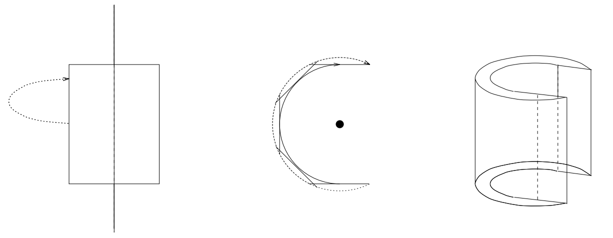

Figure 1: An example of Sliding Motion. A square is rotating about the axis in a circular trajectory. The top view shows the sweeping of the top edge of the square. The swept volume is bounded by two cylinders—the outer cylinder, defined by the moving vertical edges of the square, and the inner cylinder, defined by the envelope of the moving polygon's plane.

hedron A can be computed by unioning the volumes swept by the boundary elements of the polyhedron (i.e. its polygons) together with the polyhedron itself (at some position during the motion). (If it is possible to find two times i, j during the sweep where $A_i \cap A_j = \emptyset$, they show that it is sufficient to only union the volumes swept by the moving boundary elements.)

They continue to show that the volume swept by a polygon moving in 3-space is bounded by the surfaces swept by the moving boundary points of the polygon, the surface of the polygon at its initial and final positions, and, sometimes, by the volume swept by the interior points of the polygon. They observed that these points from a polygon's interior appear on the boundary of the swept volume only if there are successive instances of the polygon at times t and $t + \epsilon$ which intersect each other.

It appears that in order for these successive intersections to occur, the velocity of the polygon must have a zero component in the direction normal to the plane of the polygon for some points on the polygon. That is to say, if there is any point on the polygon at any time during the motion for which $v(x, y, z, t) \cdot \vec{n} = 0$ where $v(x, y, z, t)$ is the instantaneous velocity of point (x, y, z) at time t and $\vec{n}$ is the polygon's normal, then there will be points on the interior of the polygon which will appear on the boundary of the swept volume [Farouki, 1994]. This corresponds to, for example, translational motion in the plane of the polygon, or rotation of a polygon about an axis which intersects the polygon. We call this class of motion "sliding" motion, and give an example of this type of motion in figure 1. When there is sliding motion, one must look at the envelope of the moving polygon's plane to find the rest of the boundary. It is known that the envelope of a one-parameter family of planes is a developable surface. Portions of this surface will complete the boundary of the swept volume in cases which contain sliding motion.

Therefore, the boundary of the volume swept by a moving polygon consists of the polygon at it s initial and final positions, ruled surfaces defined by the motion of the polygon's edges, and, in the case of sliding motion, a portion of the developable surface formed by the envelope of the one-parameter family of planes of the moving polygon.

4 Swept Volumes for Sensor Planning

In the robot work-cell sensor planning research which we have been pursuing, the objects which move are typically articulated robot arms. This yields motion which is neither a pure translation, nor a pure rotation. When an articulated robot is moving, any point at the distal end of its last link is rotating about joint n's axis of rotation, which is itself rotating about the axis of joint $n - 1$ (which is rotating about the axis of joint $n - 2$, and so on). Therefore, the motions taking place in a robot work-cell can be quite complex. This would imply that the surfaces generated by each moving element of the robot model would also be very complex.

An exact computation of the swept volume will require intersecting all of the complex surfaces with each other. Further, due to the high degree of these surfaces, it appears that there would be no straightforward way to compute these intersections analytically, so they would need to be computed numerically. This is very expensive and has robustness problems due to floating-point error. Finally, the model-based sensor planning system we have developed requires polyhedral models, not curved surface models. Therefore, we have been examining methods for computing approximations to these surfaces.

5 Approximating The Swept Volume

Our research on computing approximate swept volumes has, thus far, focused on the approximation of

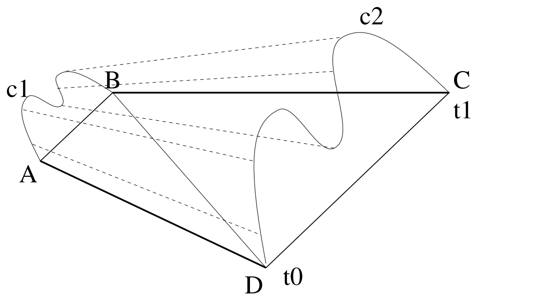

Figure 2: Approximating a patch of a ruled surface

the ruled surface generated by the moving edges of the polyhedron. Our basic approach has been to approximate the surface with a triangulated mesh, and do the intersections and graph-walking using only planar faces. In this section, we will present an algorithm for computing such an approximation.

We have seen that the swept surface defined by a moving edge is a ruled surface given by the equation

$$r(t, s) = p(t) + s \cdot v(t) \tag{5}$$

where the position and orientation of the ruling line segment is parameterised by t, and the position along the line segment is given by s. At parameter values t_0 and t_1, the rulings can be drawn, the corresponding end-points can be connected by line-segments, and a diagonal line can be drawn between one opposite pair of end-points, creating two triangles approximating the surface. See figure 2. When $t = t_0$, the ruling line $\overline{AD}$ is exactly on the surface. When $t = t_1$, line $\overline{BC}$ is exactly on the surface. As drawn, A and B are points with the same s parameter value (say, s_0), and C and D have a second s parameter value (s_1). Connecting A to B, C to D, and then D to B gives two triangles, $\triangle ABD$ and $\triangle DBC$. These triangles approximate the ruled surface on the parametric interval $([t_0, t_1], [s_0, s_1])$.

Each edge of each polygon of a polyhedral object to be swept can be stepped through its trajectory, creating a triangulated mesh approximating the ruled surface swept by each edge. Portions of these meshes will appear on the boundary of the swept volume. The boundary of the swept volume will also contain copies of each polygon of the object at its initial and final positions. If we ignore sliding motion, for the moment, we can compute a superset of the polygons which will form the boundary of the swept volume by collecting the approximating triangles and the initial and final polygons. Then, using standard solid modelling techniques, we can compute the arrangement of these polygons in space yielding a collection of cells (see, for example, [Edelsbrunner, 1987]). Most of these cells will be in the swept volume, but some of these cells may be voids within the swept volume. A complete algorithm for computing the actual swept volume would need to classify each cell to determine if it is a void or an interior cell.

An object which itself contains voids can clearly produce a swept volume which contains voids. For example, any hollow object which simply translates in a straight line will produce a swept volume with at least one interior void. However, there are motions of voidless objects which will produce voids in the swept volume. So in general, a method of classifying the resulting cells in the computed arrangement would be necessary.

5.1 Implementation

We have implemented the following algorithm for generating the swept volume $\mathcal{S}(P, T)$ for a polyhedron P and a trajectory T, such that no polygon of P undergoes "sliding motion" (as defined earlier) and such that no voids exist in the swept volume:

Polyhedral Sweep Algorithm

1. For each of the n polygons p_i of P, do the following:
   (a) Step each edge of p through the trajectory T using any step size Δt, connecting adjacent copies of each edge by forming triangles, as shown in figure 2. Call this set of triangles $\mathcal{F}$.
   (b) Add a copy of p at its initial and final positions to $\mathcal{F}$.
   (c) Set $\mathcal{F}$ is now a superset of the boundary of the actual volume swept. Compute the $\mathcal{A}$ = Arrangement($\mathcal{F}$).
   (d) Traverse the boundary of the infinite cell, i.e. the outer-most boundary of the $\mathcal{A}$. This is V_i, or $\mathcal{S}(p_i, T)$.
2. Compute $V = \cup_{i=1}^{n} V_i$.

In figure 3, we show a $5 \times 6 \times 7$ rectangular prism, turned so as to sit on one vertex. This cube is swept in a helical trajectory, given by:

$$H(t) = \begin{matrix} Rot(z, 0.1\,t\,\pi) & T(0, 8, \frac{t}{2}) \\ 0 \;\; \ldots \;\; 0 & 1 \end{matrix} \tag{6}$$

(where $Rot(z, \theta)$ implies a rotation about the Z axis of by θ, and and $T(x, y, z)$ implies a translation by (x, y, z)). The volume generated is shown in figure 4.

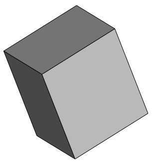

Figure 3: Rectangular prism to be swept

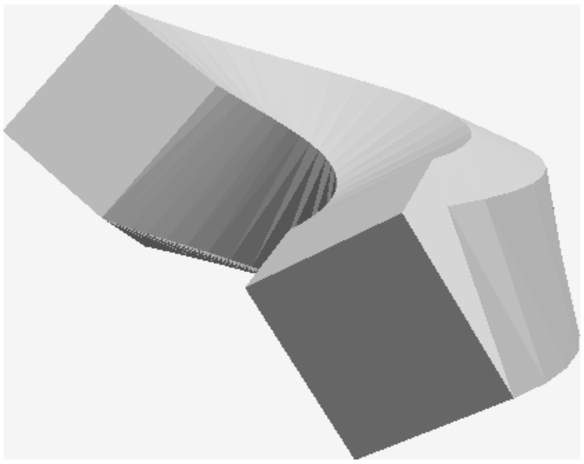

Figure 4: Volume generated from helical sweep

5.2 Handling Voids

The algorithm as we have implemented it will not compute swept volumes which contain voids. We now discuss some extensions to this algorithm which will handle both voids in the object being swept and voids induced in the result. When there are no voids in the output, the individual cells of the arrangement $\mathcal{A}$ are not important, because the outer-most boundary of $\mathcal{A}$ is the boundary of the swept volume. However, when there are voids, each cell of $\mathcal{A}$ must be examined and classified to determine if it is part of the volume or a void within the volume.

It is therefore important to examine the arrangements of polygons which tend to arise in these computations. First, we assert that each polygon (again, moving without "sliding"), when considered on its own, will create an arrangement which has no dangling faces (faces with edges of degree 1). This can be seen by looking at figure 5, which shows the three types of edges which will arise when computing the superset of the swept volume's boundary. Edges of type 1 (i.e. transformed copies of edges of the original polygon) will either be incident at one approximating triangle and a copy of the moving polygon (at the endpoints of the motion), or at two approximating triangles (at

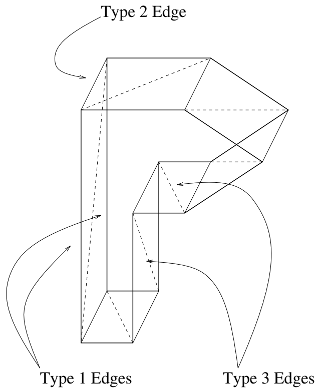

Figure 5: Examining a swept volume. The P-shaped polygon is translating "into the page."

intermediate steps in the motion). Edges of type 2 (i.e. edges which connect corresponding vertices at adjacent steps of the motion) will always be incident at the triangles which approximate the motion of the edges incident at a vertex. Finally, edges of type 3 (the edges connecting opposite vertices at adjacent steps of an edge's motion) will always be incident at the two triangles approximating the motion of an edge. Since every edge created is incident at two faces, there are no edges of degree one, and therefore no dangling faces.

By examining the nature of each cell's bounding faces, we may be able to classify the cell. For example, each face created as part of the sweep is a boundary of the swept volume of one element. If information regarding which side of each face contains the swept volume of its generating element, some cells may be easily classified as part of the volume. However, voids can not be classified in this manner since what might locally appear to be a void might be contained in the sweep of another element. Therefore, a combination of these local methods and global methods (such as ray-casting) will be required. The development of a cell-classification algorithm for the swept volume computation is still an open problem which is the subject of ongoing research.

However, if an algorithm can determine a priori that a void will not exist in the result, the most efficient method would be to traverse the outer boundary rather than to classify each of the cells which arise in the arrangement.

5.3 Incorporating Sliding Motion

We have implemented an extension to our **Polyhedral Sweep Algorithm** to include sliding motion.

We constructed a set of polygons consisting of each polygon p_i at each of its discrete positions through its trajectory, choosing the same step size that was used in step 1a. These polygons were added to the set $\mathcal{F}$. By doing this, we included an approximation to the developable surface formed by the moving polygons. Then, when these approximating polygons were needed on the outer boundary of the swept volume, the arrangement computation and boundary traversal algorithm would find these elements automatically. Figure 6 shows an example computed using this new algorithm. In this example, the same prism from the previous example was swept through a trajectory, where it was first translated by $(0, 10(2\sin(2\pi\frac{t}{10})) + 8, 0)$ and then rotated about the z-axis by $0.1\,t\,\pi + 2\,\pi\cos(2\,\pi\frac{t}{10})$.

Figure 6: Computed swept volume with sliding motion.

Unfortunately, we have empirically found that the arrangement computations (using both commercial and research geometric engines) are often not robust enough to handle the arrangement computations discussed in the first algorithm (due to floating-point error and tolerancing issues). Since this extension significantly increased the number of polygons to be arranged, this new algorithm tends to produce polygons whose arrangements can not be computed. Therefore, we have been examining methods of determining exactly when copies of a polygon being swept needs to be included in the set $\mathcal{F}$ of polygons to be arranged.

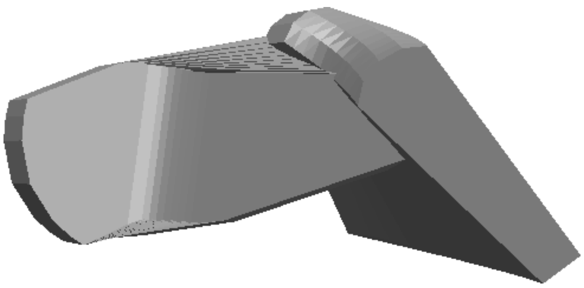

Figure 7: Rendering of swept volume for articulated robot motion.

Even in the cases for which an arrangement can not be computed, we are able to take the set of polygons $\mathcal{F}$ and graphically render them, displaying what the result should look like. Figure 7 shows a rendering of links 2 and 3 of a Puma 560 swept through a trajectory which involves simultaneous motion of the first three joints.

It is well-known that tolerancing and robustness issues make up the black art of geometric programming. Without resorting to special-purpose (and often computationally expensive) arbitrary-precision computational packages and the like, we hope to reach a level of performance acceptable for reasonably accurate approximations of swept volumes such as these, which we will need for our sensor planning work.

5.4 Determining the Step-Size

The last issue is to determine how large a step can be taken from t_0 to t_1 so that the triangulation is within some bound. Specifically, we want to choose the step such that for any point on the surface on the interval being approximated, the perpendicular distance from that point to one of the approximating triangles is guaranteed to be less than ϵ. Expressed another way, we want to guarantee that the Hausdorff distance between the surface patch and the two triangles must be less than ϵ.

Our initial observations indicate that the problem of computing the Hausdorff distance between a ruled surface patch and its approximating planes can be reduced to a small number of calculations involving the Hausdorff distance between a plane curve and an interpolating line segment. We are exploring this idea further and will be developing a technique based upon these observations for determining what step size can be used in the triangulation algorithm to guarantee that the approximated surface is within a specified tolerance of the actual surface.

6 Conclusion

The computation of swept volumes of polyhedra moving in arbitrary motions is still an open problem in computational geometry. In this paper we have presented our current research progress in the computation of swept volumes for sensor planning tasks, including experimental results from an algorithm which handles a restricted class of motion, as well as some discussion as to how the algorithm can be extended to handle general motions with a desired degree of accuracy. We shall be using these algorithms to compute viewpoints which are valid for given time-intervals in a robot work-cell which contains moving objects.

References

[Abrams and Allen, 1991] Steven Abrams and Peter K. Allen. Sensor planning in an active robotic work cell. In *Proceedings DARPA 1992 Image Understanding Workshop*, Boston, MA, November 1991.

[Abrams *et al.*, 1993] S. Abrams, P. K. Allen, and K. A. Tarabanis. Dynamic sensor planning. In *Proceedings DARPA 1993 Image Understanding Workshop*, Washington, DC, April 1993.

[Cowan and Bergman, 1989] C. K. Cowan and A. Bergman. Determining the camera and light source location for a visual task. In *Proceedings 1989 IEEE International Conference on Robotics and Automation*, May 1989.

[Cowan and Modayur, 1993] C. K. Cowan and B. Modayur. Edge-based placement of camera and light-source for object recognition and location. In *Proceedings 1993 IEEE International Conference on Robotics and Automation*, 1993.

[Cowan, 1988] C. K. Cowan. Model based synthesis of sensor locations. In *Proceedings 1988 IEEE International Conference on Robotics and Automation*, April 1988.

[Edelsbrunner, 1987] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*. Springer-Verlag, 1987.

[Farouki, 1994] R. T. Farouki. Personal communication, May 23 1994.

[Hutchinson and Kak, 1989] S. A. Hutchinson and A. C. Kak. Planning sensing strategies in a robot work cell with multi-sensor capabilities. In *Proceedings 1989 IEEE International Conference on Robotics and Automation*, December 1989.

[Ikeuchi and Kanade, 1989] K. Ikeuchi and T. Kanade. Modeling sensors: Towards automatic generation of object recognition programs. *Computer Vision, Graphics, and Image Processing*, 48:50–79, 1989.

[Kaul, 1993] Anil Kaul. *Computing Minkowski Sums*. PhD thesis, Department of Mechanical Engineering, Columbia University, 1993.

[Korein, 1985] James Korein. *A Geometric Investigation of Reach*. MIT Press, Cambridge, MA, 1985.

[Martin and Stephenson, 1990] R. R. Martin and P. C. Stephenson. Sweeping of three-dimensional objects. *Computer Aided Design*, 22(4):223–234, May 1990.

[Niepold *et al.*, 1987] R. Niepold, S. Sakane, and Y. Shirai. Vision sensor set-up planning for a hand-eye system using environmental models. In *Proceeding Soc. Instrum. Control Eng. Japan*, Hiroshima, Japan, July 1987.

[Tarabanis *et al.*, 1991] K. Tarabanis, R. Y. Tsai, and S. Abrams. Planning viewpoints that simultaneously satisfy several feature detectability constraints for robotic vision. In *Proceedings Fifth Inernational Conference of Advanced Robotics*, 1991.

[Tarabanis *et al.*, 1994a] K. Tarabanis, P. K. Allen, and R. Y. Tsai. A survey of sensor planning in computer vision. *IEEE Transactions on Robotics and Automation*, to appear 1994.

[Tarabanis *et al.*, 1994b] K. Tarabanis, R. Y. Tsai, and P. K. Allen. Analytical characterization of the feature detectability constraints of resolution, focus and field-of-view for vision sensor planning. *Computer Vision, Graphics, and Image Processing*, to appear 1994.

[Tarabanis *et al.*, 1994c] K. Tarabanis, R. Y. Tsai, and P. K. Allen. The MVP sensor planning system for robotic vision tasks. *IEEE Transactions on Robotics and Automation*, to appear 1994.

[Tarabanis *et al.*, 1994d] K. Tarabanis, R. Y. Tsai, and A. Kaul. Computing occlusion-free viewpoints. *IEEE Transactions Pattern Analysis and Machine Intelligence*, to appear 1994.

[Wang and Wang, 1986] W. P. Wang and K. K. Wang. Geometric modeling for swept volume of moving solids. *IEEE Computer Graphics and Applications*, 6(12):8–17, December 1986.

[Weld and Leu, 1990] John D. Weld and Ming C. Leu. Geometric representation of swept volumes with application to polyhedral objects. *International Journal of Robotics Research*, 9(5):105–117, October 1990.