Swept Volumes and Their Use in Viewpoint Computation in Robot Work-Cells

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 automatic computation of viewpoints for monitoring objects and features in an active robot work-cell. An important step in our algorithm for finding viewpoints is the computation of the volumes swept by polyhedral objects moving through space. A method for approximating these volumes for arbitrarily moving polyhedra is presented. Some swept volume results are presented, and methods for integrating these results into our automated Machine Vision Planning (MVP) system are discussed.

1 Introduction

Several researchers have focused on the computation of sets of positions, orientations, and optical settings for a camera (and, in some cases, for light sources) which will give satisfactory views of certain objects in a known scene. Each researcher has defined the phrase "satisfactory view" in his own terms, but the constraints most often considered are magnification (or resolution), focus, field-of-view, and occlusion. Examples of recent work in this field are in [4, 5, 6, 12, 13]. A complete survey of sensor planning systems can be found in [14]. The majority of the work has focused on sensor planning in static environments, i.e. where all of the objects are stationary, and is typically applied to automated inspection tasks. These systems can be used, for example, to automatically compute the viewpoints and optical settings needed to examine given features on a manufactured object to insure product quality. Our current research is aimed towards extending this work to environments in which objects move, such as a typical robot work-cell.

*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, North American Philips Laboratories, Siemens Corporation and Rockwell International.

1.1 Previous Research: MVP

Our previous research in this field has resulted in the development of the Machine Vision Planning (MVP) system [17, 18, 16, 15]. Briefly, MVP takes an optimization approach towards viewpoint computation. That is, it models each of the relevant constraints on a viewpoint (i.e. focus, resolution, field-of-view, and occlusion) as a function in the viewpoint parameter-space and optimizes a linear combination of these constraints. The basic idea is to obtain a viewpoint which is as far away from each of the constraints as possible. Such a viewpoint is then robust in terms of placement or calibration errors, since it meets all of the constraints as comfortably as possible.

An important part of MVP is an algorithm which computes visibility volumes. That is, given a polyhedral model of an object and of other objects in the environment, and given a list of features on that object, the MVP system can compute the locus of points in space (bounded by a polyhedron) from which the given features are visible. This algorithm is discussed at length in [18].

1.2 Adding Moving Objects

We have been extending MVP to function in an environment in which objects are moving [1, 2]. As an example, we may have a work-cell in which one or more robots are assembling an object. We may wish to automatically monitor this assembly task. Figure 1 shows the basic setup for such a system: two Puma 560 arms, able to operate in a work-cell, and a gantry robot, used for moving the camera through a computed trajectory. MVP, and the sensor planning systems referenced above, can not compute viewpoints in such an environment, because they do not handle moving objects. We call the problem of computing viewpoints in this environment *Dynamic Sensor Planning*.

Our approach to Dynamic Sensor Planning has been based on temporal intervals, in which the task is broken down into intervals, each of which is to be monitored by

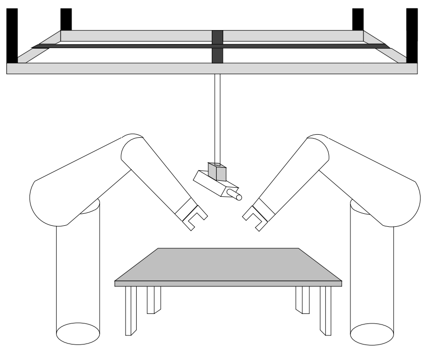

Figure 1: Gantry, encompassing Puma work-cells

a single viewpoint. To solve the occlusion problem, the system computes the volumes swept by all moving objects during this interval and, using the algorithms developed as part of MVP, computes viewpoints which avoid occlusion by these swept volumes. Such viewpoints are valid for the entire time interval. By similarly examining a number of time intervals, we break the Dynamic Sensor Planning problem down into a series of static subproblems.

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. Therefore, we have embarked upon the present research task of computing polyhedral approximations to the volumes swept by moving polyhedral objects.

In the remainder of this paper we shall define in precise terms the phrase "swept volume," describe our algorithm for computing polyhedral approximations to swept volumes, explain the issues involved in extending the MVP system to handle the temporal interval search, and describe the experimental test-bed which we are building for executing sensor-planning experiments.

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

$$Q \equiv H(t) \qquad (1)$$

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

$$\begin{array}{ccc} & R & & T \\ 0 & \ldots & 0 & 1 \end{array} \qquad (2)$$

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

This gives us 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.:

$$M_H = \{Hm : m \in M\} \qquad (3)$$

We can now define the sweep of a set $M \subset \mathbf{R}^d$ over an arbitrary trajectory Q (notated $\mathcal{S}(M, Q)$) as:

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

Intuitively, we are dragging M through a path, twisting and turning it as we go.

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.

3 Related Sweeping Research

A number of previous researchers have examined the problem of computing swept volumes, including Korein [10] (for rotating polyhedra), Kaul [9] (using Minkowski sums for translations), Wang and Wang [19] (using envelope theory), and Martin and Stephenson [11]

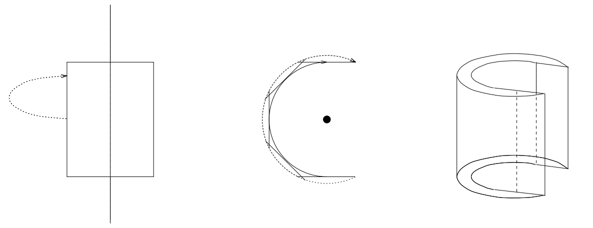

Figure 2: An example of Sliding Motion.

(using envelope theory and computer algebraic techniques). Of particular interest is Weld and Leu [20] who set forth some theory for computing the volume swept by a moving polyhedron. They show that the volume swept by a moving polyhedron 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 *ruled surfaces* swept by the moving edges of the polygon, copies 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. In these cases, portions of the envelope of the moving polygon's plane (a developable surface), will form the rest of the boundary.

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 [8]. 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 2. In this figure, a square is rotating about the vertical axis (not an axis in the

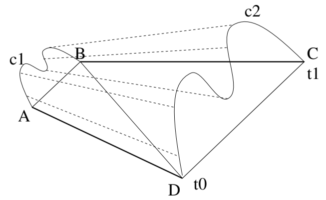

Figure 3: Approximating a patch of a ruled surface

plane of the square) 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 square's moving plane.

4 Approximating The Swept Volume

An exact computation of the volume swept by a moving polygon will require computing an intersection graph of the ruled surfaces, the developable surfaces, and the polygon at its initial and final positions. Once done, the intersection graph can be traversed to find the boundary of the swept volume. Computing the intersections of these surfaces is an expensive proposition with many robustness problems, and may not even be possible, depending on the types of motion undergone. Also, the sensor-planning algorithms which we have been using require polyhedral models of the swept volumes. Therefore, we have developed a method for computing polyhedral approximations to these volumes.

Our basic approach has been to approximate the ruled surface with a triangulated mesh, and to approximate the developable surface with copies of the polygon stepped through its trajectory. We then do the intersections and graph-walking using only the triangles in the mesh and the polygons of the moving object. In this section, we present an algorithm for computing such an approximation.

The surface swept 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 3. 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 corresponding vertices A and B with a line segment, corresponding vertices C and D with a segment, and then opposite vertices D and B, gives two triangles, $\triangle ABD$ and $\triangle DBC$. These triangles approximate the ruled surface on the parametric interval $([t_0, t_1], [s_0, s_1])$.

In this fashion, each edge of each polygon 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 the polygon object at its initial and final positions.

In addition, when sliding motion takes place, we need to include copies of the polygon undergoing sliding motion. We have seen that sliding motion takes place when some point on the polygon has a zero velocity in the polygon's normal direction. Looking at the polygon's motion at the same discrete steps as we looked at each edge above, we can determine if sliding motion took place between t and $t+\Delta t$. The following algorithm will compute $\mathcal{S}$, a set of faces which will be useful for approximating the developable surface formed by the interior of a moving polygon p when that polygon undergoes sliding motion:

Sliding Motion Test

1. Compute the motion of each vertex of p from t_i to t_{i+1} relative to the p's normal at time t_i. The vertex will be moving forward, backward, or in the plane of p. For each vertex v_j, let $D_{t_i}(v_j)$ be $+1$ (forward), -1 (backward), or 0 (in the plane), depending on its motion at t_i.

2. If there are any zero crossings, i.e. if $D(v_j) \neq D(v_k)$ for any v_j, v_k of p (such as if the face is rotating about an axis in its plane), or if there are any vertices v_l for which $D(v_l) = 0$, the face is undergoing sliding motion between t_i and t_{i+1}. Add copies of the face at both t_i and t_{i+1} to $\mathcal{S}$.

3. If for any vertex v_j of p, $D_{t_{i-1}}(v_j) \neq D_{t_i}(v_j)$, i.e. if any vertex has changed direction from the previous interval (relative to the face's plane, that is), the face is undergoing sliding motion at t_i. Add a copy of the face at t_i to $\mathcal{S}$.

This algorithm is run at every intermediate step t_i *between* the start time and the end time, (i.e. not at the end points of motion themselves).

The set of polygons $\mathcal{S}$ produced with the above algorithm, plus copies of the moving polygon at the start and end points of motion, plus the set of triangles generated to approximate the ruled surfaces from each edge, yield a new set of polygons, $\mathcal{F}$. $\mathcal{F}$ is a superset of the boundary of our approximation to the volume swept by a moving polygon. That is, some of the members of $\mathcal{F}$ may be partially or completely on the boundary, while some members may be partially or completely interior to the actual swept volume.

To compute the actual boundary of the swept volume, we can compute the arrangement of these polygons in space yielding a collection of cells (see, for example, [7]). Most of these cells will be in the swept volume, but some of these cells may be voids within the swept volume.

While voids inside the swept volume are not important for sensor-planning tasks (a camera placed within a void will unable to see anything outside of the void), 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. Clearly, an object which itself contains voids can 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. The union of all of the cells which pass the classification test then comprises the swept volume.

4.1 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 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 3. Call this set of triangles $\mathcal{F}$.

   (b) Add a copy of p at its initial and final positions to $\mathcal{F}$.

   (c) Run the **Sliding Motion Test** (above) on p_i, using the same step size Δt. Add all of the faces placed in $\mathcal{S}$ by this algorithm to $\mathcal{F}$.

   (d) Set $\mathcal{F}$ is now a superset of the boundary of the actual volume swept. Compute the set $\mathcal{A} =$ Arrangement$(\mathcal{F})$.

   (e) 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 $V = \cup_{i=1}^{n} V_i$.

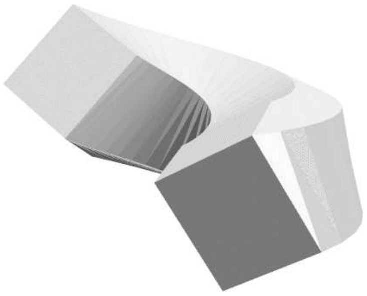

Figure 4: Volume generated from helical sweep

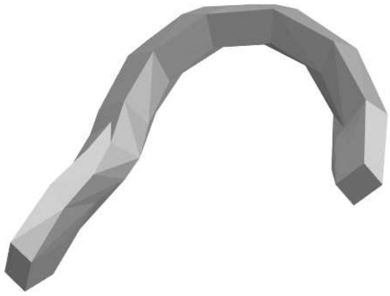

Figure 5: Computed swept volume with sliding motion.

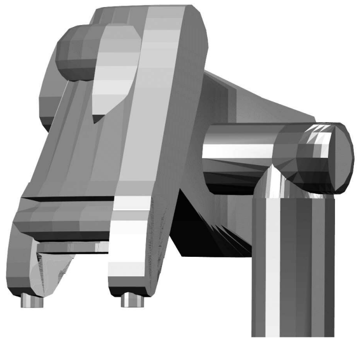

Figure 6: Rendering of swept puma model.

A rectangular prism with dimensions $5 \times 6 \times 7$, turned so as to sit on one vertex, is swept in the helical trajectory given by:

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

(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 5 shows another example which undergoes sliding motion. In this example, the same prism from the 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 $0.1\, t\, \pi + 2\,\pi \cos(2\,\pi \frac{t}{10})$.

4.2 Robustness Issues

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 above (due to floating-point error and related issues). We are exploring methods for improving the robustness of these algorithms. 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 6 shows a rendering of a Puma 560 swept through a trajectory in which the arm first moves up, then to the viewer's left, and then down.

5 Integration with MVP

Recall that MVP uses a constrained optimization for computing viewpoints. Some of the constraint equations are nonlinear; some are not continuously differentiable. Optimization of such functions is a very difficult problem. The optimization systems we have used (IMSL and LANCELOT [3]), have taken anywhere from seconds to minutes to perform the viewpoint computation and, at times, required manipulation of the weighs or starting points to produce results. This is certainly acceptable performance for one-shot off-line planning problems (such as those for which MVP is used). However, the basic Temporal Interval Search algorithm given in section 1 needs to automatically solve many static subproblems. The subproblems are generated automatically and it is not desirable to require human intervention to facilitate the optimization. Therefore, we have been exploring some changes to the MVP algorithms to make them more appropriate for dynamic sensor planning.

We have found that the exact formulation of each constraining equation has a profound effect on the optimization. We are analyzing each equation to find formulations which yield the most stable results. This process has led us to a better understanding of how each constraint effects the search for a robust viewpoint. Using this understanding, we are developing a heuristic search algorithm for the computation of sets of robust viewpoints.

6 Experimental Test-Bed

In order to demonstrate the usefulness and practicality of the sensor planning system as a whole, we will be running experiments in our robot work-cell. In these ex-

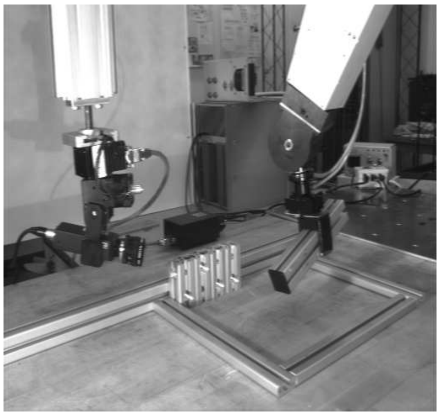

Figure 7: The experimental test-bed, showing a camera carried by the gantry robot and a Puma-560

periments, surveillance points and intervals will be computed by the dynamic sensor planning system. To realize these computed viewpoints, we have been constructing a sensor-positioning robot. This 5 degree-of-freedom Cartesian robot, having a work-space of roughly 1000 ft^3, carries a CCD camera in hand/eye configuration. It can accurately position the camera in and around our robot work-cell, thereby monitoring objects under manipulation by our two Puma 560's. The end-effector of the Cartesian robot is shown in our work-cell with one of the Pumas in figure 7. The current state of this robot is that the mechanical, electrical, and controlling software has all been completed. As soon as the hand/eye system has been calibrated, we shall continue with sensor planning and placement experiments.

References

[1] S. Abrams and P. K. Allen. Computing swept volumes for sensor planning tasks. In *Proceedings DARPA 1994 Image Understanding Workshop*, Washington, DC, November 1994.

[2] S. Abrams, P. K. Allen, and K. A. Tarabanis. Dynamic sensor planning. In *Proceedings 1993 IEEE International Conference on Robotics and Automation*, Atlanta, GA, May 1993.

[3] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. *LANCELOT A Fortran Package for Large-Scale Nonlinear Optimization*. Springer Series in Computational Mathematics. Springer-Verlag, New York, 1992.

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

[5] 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.

[6] 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.

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

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

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

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

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

[12] S. Sakane, R. Niepold, T. Sato, and Y. Shirai. Illumination setup planning for a hand-eye system based on an environmental model. *Advanced Robotics*, 6(4):461–482, 1992.

[13] S. Sakane, T. Sato, and M. Kakikura. Model-based planning of visual sensors using a hand-eye action simulator system: HEAVEN. In *Proceedings of the 3rd International Conference on Advanced Robotics*, pages 163–174, Versailles, France, October 1987.

[14] K. Tarabanis, P. K. Allen, and R. Y. Tsai. A survey of sensor planning in computer vision. *IEEE Transactions on Robotics and Automation*, 11(1), February 1995.

[15] 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.

[16] 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*, 59(3), May 1994.

[17] K. Tarabanis, R. Y. Tsai, and P. K. Allen. The MVP sensor planning system for robotic vision tasks. *IEEE Transactions on Robotics and Automation*, 11(1), February 1995.

[18] K. Tarabanis, R. Y. Tsai, and A. Kaul. Computing occlusion-free viewpoints. *IEEE Transactions Pattern Analysis and Machine Intelligence*, to appear 1995.

[19] 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.

[20] 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.