Sensor Planning in an Active Robotic Work Cell

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

Abstract

In this paper, we discuss techniques for extending the sensor planning capabilities of the "MVP" (Machine Vision Planning) system to include motion in a well-known environment. In a typical work cell, vision sensors are needed to monitor a task and provide feedback to motion control programs or to assess task completion or failure. In planning sensor locations and parameters for such a work-cell, all motion in the environment must be taken into account in order to avoid occlusions of desired features by moving objects and, in the case where the features to be monitored are being manipulated by the robot, to insure that the features are always within the camera's view. Several different sensor locations (or a single, movable sensor) may be required in order to view the features of interest during the course of the task. The goal is to minimize the number of sensors (or to minimize the motion of the single sensor) while guaranteeing a robust view at all times during the task, where a robust view is one which is unobstructed, in focus, and sufficiently magnified. In the past, sensor planning techniques have primarily focused on static environments. We present techniques which we have been exploring to include knowledge of motion in the sensor planning problem. Possible directions for future research are also presented.

1 Introduction

Recently, there has been much research in the field of sensor planning [1, 2, 4, 5, 7, 8, 12, 13]. The basic problem is that in setting up an automated system for monitoring some process, the effectiveness of the system can largely be determined by the locations, types and configurations of the sensors used. To manually determine these parameters on a case by case basis may not be cost efficient or accurate, and the resulting system may not be optimal in any sense. It may be better to have an automated system for determining the sensor locations and parameters for monitoring a given task.

To that end, many systems have been and are being developed which, based on geometric models of an environment and models of the sensors, can generate sensor locations and settings which provide a robust view of specific features so that the features are detectable, recognizable, measurable, or meet some other task constraints. In general, the sensors are cameras and a robust view implies that the camera must have an unobstructed view of the entire feature set, which must lie within the depth-of-field of the camera and must be magnified to a given specification. Sensor planning systems can then generate camera locations, orientations, lens settings (focus-ring adjustment, focal length, aperture), and in some cases lighting plans to insure a robust view of the features [10].

*This work was supported in part by DARPA contract N00039-84-C-0165, NSF grants DMC-86-05065, DCI-86-08845, CCR-86-12709, IRI-86-57151, IRI-88-1319, North American Philips Laboratories, Siemens Corporation and Rockwell International.

0-8194-0748-8/92/$4.00

Most of the methods presented to date have only addressed static environments such as would be found in a post-manufacturing inspection task (an exception is the VIO system of Niepold et. al. in [7]). The approach taken is to perform an off-line analysis of the geometric and optical constraints for a static environment and, via a generate-and-test or a synthesis approach, give one or more sensor location and parameter settings which are valid only for the specific static environment which was analyzed. When objects in the environment need to be moved for some reason, new sensor locations need to be computed off-line. This works well for quality-control or inspection tasks where, for example, parts can be fed to a specific location and orientation in the environment for inspection.

There are many instances where moving scenes may need to be monitored. Manufacturing or assembly tasks can be visually guided or monitored. Telerobotic operations need some form of sensor feedback. For these and other tasks, the sensor planning techniques presented to date would be inadequate due to the dynamic nature of the environments. But for these applications, it would still be better to have the robotic system remain in control of the sensor positions and settings to insure reliable monitoring rather than to require manual control over the sensors.

In this paper, we describe our most recent research in extending the MVP (Machine Vision Planning) System [12, 13] to plan sensor locations and settings for a changing environment. After a brief overview of the MVP system, the remainder of this paper will focus on an explanation of the constraints we impose on the environment, the theory behind our method, and a technique which helps to incorporate temporal reasoning into spatial problems. We also present examples which show this temporal reasoning working in conjunction with the MVP system. Finally, we present an overview of future work in this area, aimed at strengthening our ability to reason temporally as well as spatially for sensor planning and other problems in robotics.

2 Overview of MVP

A complete description of the MVP system is beyond the scope of this paper. For details, see [11, 12, 13]. In brief, MVP takes a constraint based description of the vision task requirements and synthesizes what has been termed a *generalized viewpoint*, which is an eight-dimensional vector incorporating sensor location, orientation, and lens parameters including aperture and effective focal length. The constraints MVP considers in determining viewpoints are depth-of-field, field-of-view, resolution, and unoccluded visibility.

MVP contains analytical relationships for the optical task constraints (resolution, focus, field-of-view), and uses 3-D solid geometric models of the environment to formulate visibility constraints. (The geometric models are limited to general polyhedra, both convex and concave; curved surfaces are not permitted.) The constraint equations can be thought of as defining hypersurfaces bounding feasible regions in the 8-dimensional parameter space of the generalized viewpoint. These constraints are combined in an optimization setting to produce a generalized viewpoint which meets all task constraints with as much margin for error in sensor placement and setting as possible (i.e., as far away from all hypersurfaces as possible). MVP, from a CAD description of the object to be viewed and its environment, can generate the visibility region for viewing the desired features. This region is calculated to be the total volume in space from which the features are viewable without obstruction. This volume is used in the optimization stage of MVP for finding the best viewpoint.

While MVP's synthesis approach makes it better suited to the extensions we present here than other less analytical approaches, there is no reason why the temporal extensions discussed here can not be applied to other sensor planning algorithms. In fact, it is our hope that these methods can be applied to 3-D planning problems in general.

3 The Introduction of Motion

In sensor planning problems, there is normally a well-defined set of target features which need to be monitored. These might correspond to a section of a part which has just come off the assembly line which the vision system might need to examine for defects. In an active environment, the feature set might correspond to a section of a

larger assembly being operated on, which we might need to monitor during the operation. For this work, we restrict ourselves to consider only the motion of obstacles in the environment, and not the motion of the target. We also assume complete knowledge of the environment in the form of 3-D geometric models. Finally, we assume knowledge of the motion of the obstacles in advance; unplanned motion may not take place.

The most important result of these limitations is that once we have a viewpoint[1] which is valid for a given instant in time t_n, it is guaranteed to be valid with respect to all *optical* constraints for all times t_m for $m > n$. This is fairly simple to show. Once we have a properly magnified and focused view, if neither the target nor the camera move, the object remains in proper focus with an unvarying magnification. The movement of obstacles means that only occlusion needs to be detected at later times.

One other point worth mentioning is that the static sensor planning problem does not require the camera to be attached to a robot. For the dynamic sensor planning problem, the camera must be movable from one viewpoint to another during the course of the task being monitored in order to maintain a robust view of the target.

To summarize, the exact problem we are dealing with is one in which an accurately movable camera is being used to monitor a task. In this task, the actual target we are monitoring does not move, but other objects in the environment, such as a robot arm, or other mechanical parts, move in a way which is known *a priori*. The problem is to find where to place the camera, and when and where to move the camera, so that at all times during the task, we have a good viewpoint for monitoring the task.

As an example, a spot-welding robot may be working on a stationary object (such as a car body) in a factory. There might be a vision system used to monitor the weld for defects and for accuracy. The sensor would need to be placed in such a way as to avoid occlusion by the many moving obstacles in the environment, such as the welding arm itself and any moving peripherals needed for this task. The target itself, that is the area to be welded, would remain stationary for this task.

4 Relation to Motion Planning

There is a very close relationship between the problem of sensor planning with moving obstacles and that of motion planning with moving obstacles. The essence of both problems is to find a representation which relates the temporal and spatial aspects of an object's motion. The essential difference is that while path planning needs to find a path which moves through a time-varying environment to reach a goal without hitting any obstacles, sensor-planning is searching for one or more generalized viewpoints which remain valid (unoccluded) for the duration of the task. The implications of this difference are that we can not envision planning a point that travels through configuration space or configuration space-time [3, 6]. This is because our "point" is actually a cone with its apex at the sensor and its base at the target polygon. We need to detect when obstacles will breach this viewing cone, not just detect the collision of a single point with a configuration space obstacle. The second difference is that in motion planning, we search for an unobstructed path. In sensor planning, the preference is to remain stationary. We are in search of a single viewpoint, if possible, which is valid during the entire operation, or, if that is not possible, as few viewpoints as possible.

5 The Naive Approach

One way to handle added complexities to a problem is to ignore them until they become an issue. Taking this approach to motion in sensor planning yields the following naive algorithm:

[1] Here, and elsewhere in this paper, when we refer to a *viewpoint* we are actually referring to the *generalized* viewpoint mentioned earlier.

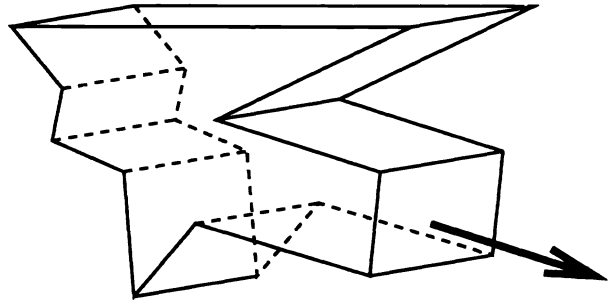

Figure 1: Polyhedron floating in space with direction vector

1. Compute a viewpoint for the initial state of the system, considering all obstacles in the environment as they are before any motion takes place.
2. At every time interval Δt, test the current viewpoint against the model of the changed environment.
3. If, at some instant t_n, the viewpoint is found to be invalid due to the movement of obstacles, compute a new viewpoint based on the current state of the model, and go back to step 2.

The algorithm can be run in advance, off-line, since the changes the environment goes through over time are known, so it would not even be necessary to have a function which evaluates or generates viewpoints in real time. The entire problem can be simulated, and the time intervals where the sensor needs to be moved and reset can be noted. During the process, the robotic system can pause, reposition the camera, and then continue its operation.

This approach has several major drawbacks:

- This algorithm is clearly not optimal in its use of the viewpoint evaluation function. In a task which takes $M \times \Delta t$ time to complete, the viewpoint needs to be evaluated exactly M times.
- The algorithm makes no attempt to reduce the number of sensor replacements required.
- A viewpoint is used up until the moment *after* it has become invalid, or at least up until the point at which the margin for error becomes very small. As an example, say the algorithm determines that at time t_n, the initial viewpoint is no longer acceptable. Although the initial selection of a viewpoint was chosen to have a large margin for error, this error margin only existed at time t_0; at some time before t_n, due to errors in sensor placement, etc, the viewpoint may be bad.
- The accuracy of this method is dependent upon the time interval Δt used. Δt is essentially the sampling frequency with which we test the environment. The viewpoint may become invalid between two samples and yet be valid at each sample. This behavior is not desirable.
- Knowledge of future positions of the obstacles is not used in this method. Given that the planning and evaluation functions are fast enough, there is no benefit to computing any information in advance. It is conceivable that at each time interval t_n, when we plan a new viewpoint, the very next motion (or the sequence of motions in the near future) make the new viewpoint invalid very quickly, perhaps even at time t_{n+1}. This is very bad; the sensor may need to be repositioned far to frequently.

The problem specification indicates that we have more information at our disposal than this solution uses. By using all available knowledge, that is, our knowledge of the temporal as well as the spatial aspects of the environment, we can hope to generate viewpoints which are valid (by a larger margin) for longer time intervals.

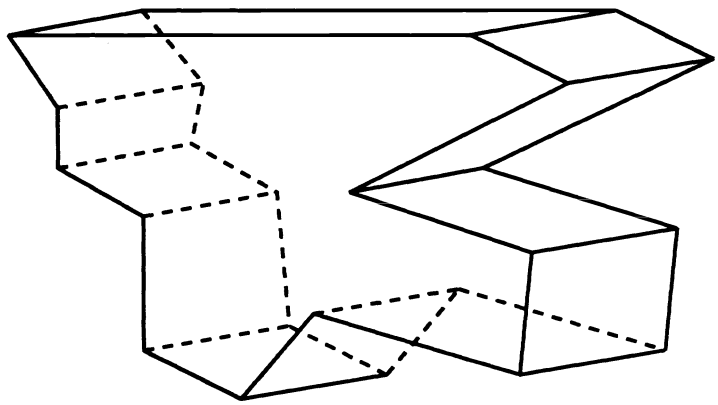

Figure 2: The unique minimal temporal object

6 Temporal Considerations

In order to make use of our knowledge of motion and time in the environment, we define a concept which relates the geometric orientation of an object, and its motion through space over a given time interval. The structure which embodies this relationship for a given object over a given time period is called a *minimal temporal object.* To illustrate this concept, note the polyhedron in figure 1. It is shown with a vector indicating its linear trajectory.

Definition 1 *A* minimal temporal object *is the set of all points through which a given object O passes during its motion over a given time interval T. The minimal temporal object representing the motion of O over T is notated as $\hat{\mathcal{T}}(T,O)$.*

Figure 2 shows the minimal temporal object for the polyhedron in figure 1 moving in the direction indicated. For a given object, and its trajectory and velocity over a given time period, there is one unique $\hat{\mathcal{T}}(T,O)$. The task of computing $\hat{\mathcal{T}}(T,O)$ is equivalent to computing the complete volume swept by O during its motion over T. For general objects moving in arbitrary paths, $\hat{\mathcal{T}}(T,O)$ may be exceedingly expensive to compute. This is why, in working within the sensor planning framework, we have been dealing with approximations to $\hat{\mathcal{T}}(T,O)$. We restrict ourselves to approximations of $\hat{\mathcal{T}}(T,O)$ which meet the following definition:

Definition 2 *A* temporal object *is defined as any volume which contains the set of all points through which a given object O passes during its motion over a given time interval T. A temporal object representing the motion of O over time interval T is notated as $\mathcal{T}(T,O)$.*

The most important consequence of this definition is that $\mathcal{T}(T,O)$ is necessarily contained within $\hat{\mathcal{T}}(T,O)$. Rough methods can be developed to compute $\mathcal{T}(T,O)$ and use them as approximations to $\hat{\mathcal{T}}(T,O)$ for planning. Note, that for any object moving through a given path, $\mathcal{T}(T,O)$ is not unique while $\hat{\mathcal{T}}(T,O)$ is.

Due to a result by Weld and Leu in [14], the volume formed by sweeping a polyhedral object along an arbitrary path is equivalent to the volume formed by sweeping each face along the same path and unioning these swept volumes together. The expensive portion of this algorithm (in the case of translational motion only) is in the unioning. A polyhedron of n faces requires n boolean unions to be swept via this method. Since, in the *Temporal Sensor Planning* algorithm (which follows), it may be necessary to compute swept volumes often, we have opted for a faster method of computing a $\mathcal{T}(T,O)$ as opposed to an accurate, but slower method for computing $\hat{\mathcal{T}}(T,O)$.

We use a simple algorithm to compute a $\mathcal{T}(T,O)$ given that T moves through a linear path. This is not particularly restrictive, since any arbitrary path can be approximated by a piecewise linear path, and the algorithm can be

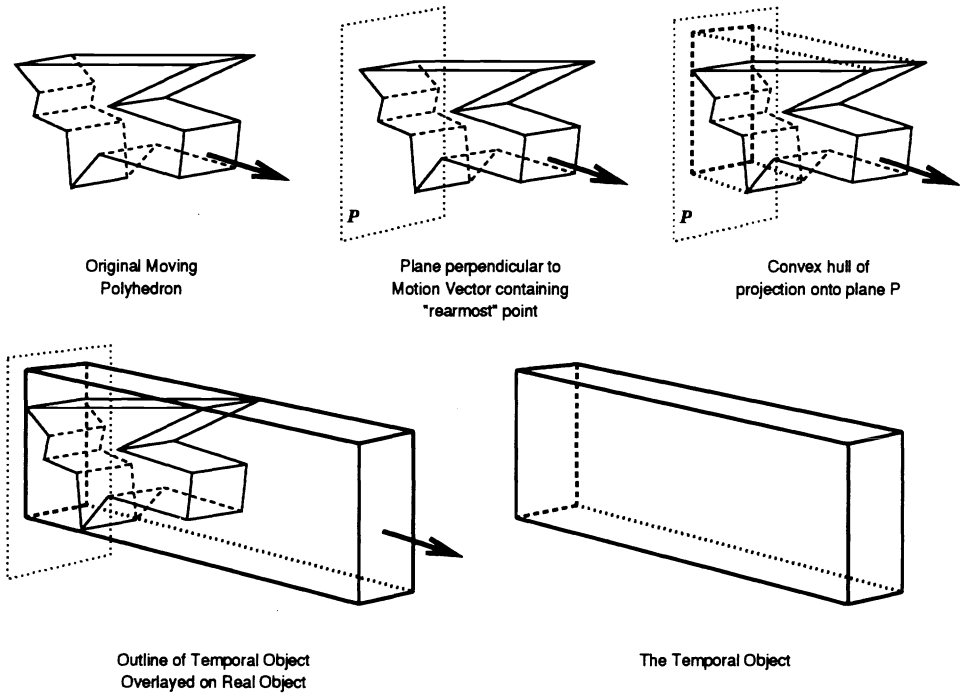

Figure 3: Generation of a Temporal Object

repeated over each linear segment of a path to compute the $\mathcal{T}(T, O)$ for the whole interval. Here, we describe the algorithm for computing $\mathcal{T}(T, O)$ when O is a polyhedron known to move in a linear path of length n in the $\vec{v}$ direction.

Generation of $\mathcal{T}(T, O)$

1. Calculate plane $\mathcal{P}$, which is the plane defined by the unit vector $\vec{v}$ and the point p of O in the extreme $-\vec{v}$ direction. That is, as O moves in the $\vec{v}$ direction, p is the point "furthest back," and $\mathcal{P}$ is the plane perpendicular to the trajectory of O, and containing p.
2. Project all vertices of O onto the $\mathcal{P}$ and take the convex hull of these points, creating a polygon s on $\mathcal{P}$.
3. $\mathcal{T}(T, O)$ is the right generalized cylinder with a linear axis parallel to $\vec{v}$, a constant cross-section, with a base on s and a height of n plus h, where h is the overall length of O in the $\vec{v}$ direction.

This algorithm is illustrated in figure 3.

Theorem 1 *Any volume created by the above algorithm necessarily contains* $\hat{\mathcal{T}}(T, O)$.

Proof:

1. By step 3 above, all perpendicular cross sections of $\mathcal{T}(T, O)$ are the same.
2. By step 2, the cross sections must be polygons wholly containing any perpendicular cross section of $\hat{\mathcal{T}}(T, O)$.
3. The overall length of $\mathcal{T}(T, O)$ is at least as long as that of $\hat{\mathcal{T}}(T, O)$ by the final step.

4. These three conditions show the total inclusion of $\hat{\mathcal{T}}(T,O)$ by $\mathcal{T}(T,O)$.

The key to using a temporal objects for sensor planning (or, in fact, for any collision avoidance problem) is that in planning around an obstacle given by $\mathcal{T}(T,O)$, you guarantee that you have avoided the actual obstacle O at any instant in interval T.

7 Temporal Objects in Sensor Planning

The essence of our approach to sensor planning around moving objects is to plan around the temporal objects generated from the original objects and their motion. The temporal objects, once calculated as regions in space, are treated as stationary objects for a static planning problem. In many cases, this is too restrictive. There may be one viewpoint which is valid for one portion of the time interval, and another viewpoint which is valid for another portion, yet there is no viewpoint which is valid for the entire interval. In cases such as this, our algorithm subdivides the interval in half whenever faced with a failure, and replans for the two halves independently.

More formally, this algorithm is described as follows. Assume we have a polygonal target τ which we wish to monitor during the time interval $T = [t_0, t_n]$. During T, there is a set of known obstacles O_0 through O_m, which move in known paths. The goal is to plan a single viewpoint valid for the entire interval, if such a point exists, or to determine a sequence of viewpoints.

Temporal Sensor Planning (*TSP*)

1. Compute $\mathcal{T}(T,O_i)$ for each of the m obstacles.
2. Use MVP to compute a viewpoint using the set of temporal objects spanning time interval T as the potential occluding bodies.
3. If MVP can successfully find a viewpoint which is valid in the presence of all of the temporal obstacles, it is guaranteed to be valid for any instant in T. If such a point is found with MVP, the algorithm terminates with a successful result: it has planned a single viewpoint.
4. If no such viewpoint is obtainable, we divide T into $T_1 = [t_0, t_{n/2}]$ and $T_2 = [t_{n/2}, t_n]$, and run the Temporal Sensor Planning algorithm on each subinterval.

The binary partitioning of the last step continues until we have found a valid viewpoint for all of T, or until we have divided into time intervals of some minimal preset length ϵ. If sub-intervals too small are reached, the determination is that the motion is too complex for this method to provide meaningful results. Typically, one chooses ϵ to be large enough so that it is feasible to stop and reset the sensor every ϵ interval, since, in the worst case, that is what might happen. If the *TSP* algorithm determines that the motion in the workcell is too complex to plan viewpoints, it might be an indication that the activity in the workcell is too complex to be monitored and that the task itself should be replanned.

It is important to note that in step 2 of the *TSP* algorithm, we rely on MVP to identify the fact that it has been unable to to find a valid sensor viewpoint. MVP itself relies on a nonlinear constrained optimization to compute a viewpoint. The failure of MVP to find a viewpoint does not guarantee that one does not exist. However, when a valid viewpoint is so well-hidden within the hypersurfaces of 8-dimensional parameter space that our optimization routine can not locate it, there is a very good chance that the viewpoint does not provide much of a margin for error. Therefore MVP's failure to find a viewpoint, while not a guarantee of the nonexistence of a *valid* viewpoint, is an excellent indication that there is no *good* viewpoint.

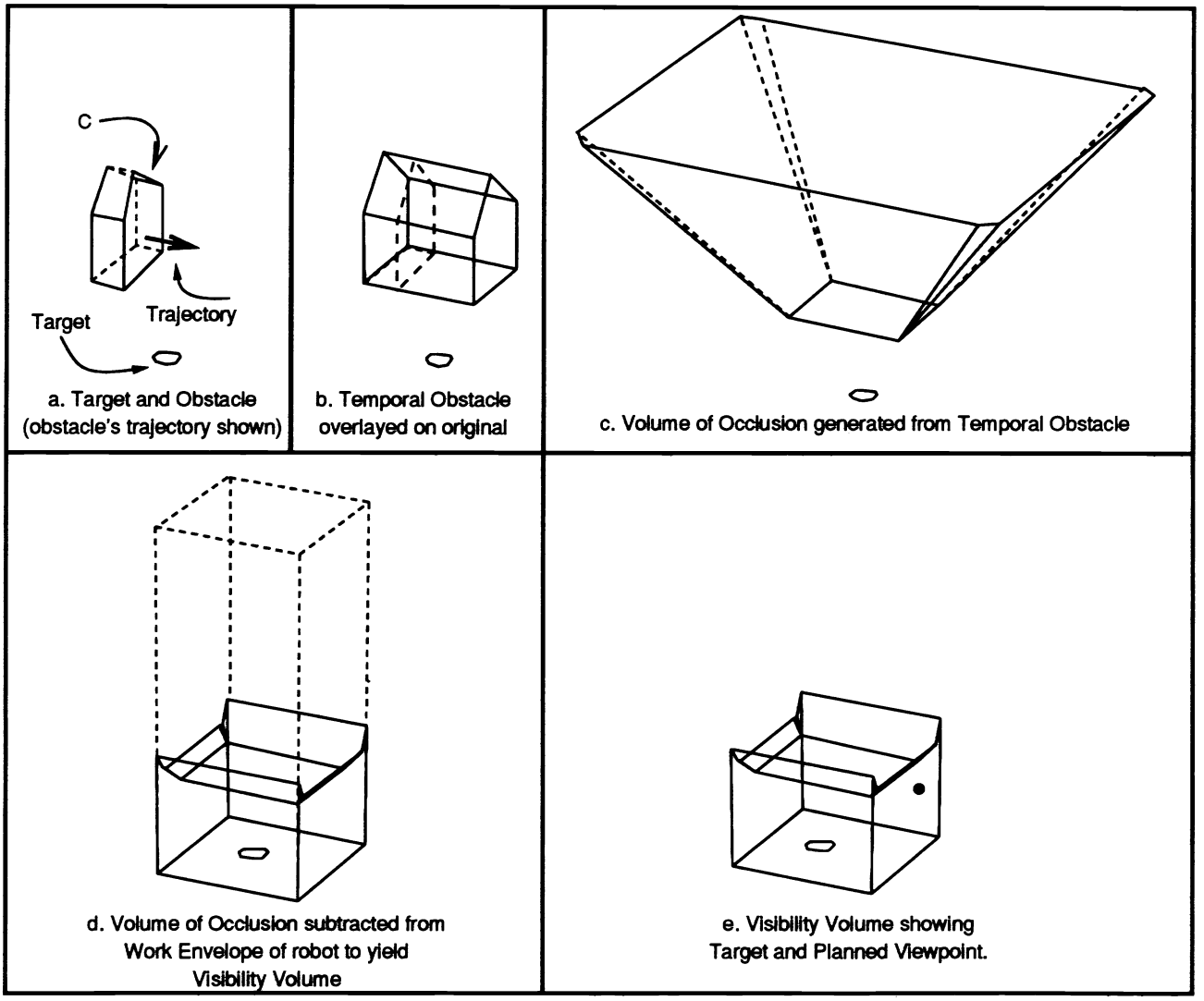

Figure 4: Simulation of *TSP* and MVP

8 Experimental Results

We have implemented *TSP* in conjunction with the MVP system, and have produced simulated results showing the effectiveness of *TSP* as a method of extending 3-dimensional planning algorithms to include the time domain. While only the visibility constraint of MVP is of concern to *TSP*, it is important to note that the other constraints do play a role in whether or not *TSP* can find a valid viewpoint. In these experiments, the sensor modeled is a typical CCD camera, and a resolution constraint of one pixel per 0.1 inches was used. The geometric models of the objects and environment as well as those of the temporal objects were calculated using the ACIS solid modeler [9].

In the first example, a polyhedral object C is the obstacle which moves in a linear trajectory above an octagonal target at a constant velocity over time interval T (figure 4a). Next, using the algorithm presented earlier, $\mathcal{T}(T, C)$ is generated (figure 4b). The corresponding volume of occlusion is also generated using the methods presented in [13] (figure 4c). Note that this volume represents an overapproximation to the volume in space from which, at some point in the interval T, the view of the target would be blocked due to C.

The volume of occlusion is subtracted from the work envelope of the robot carrying the camera to yield the visibility

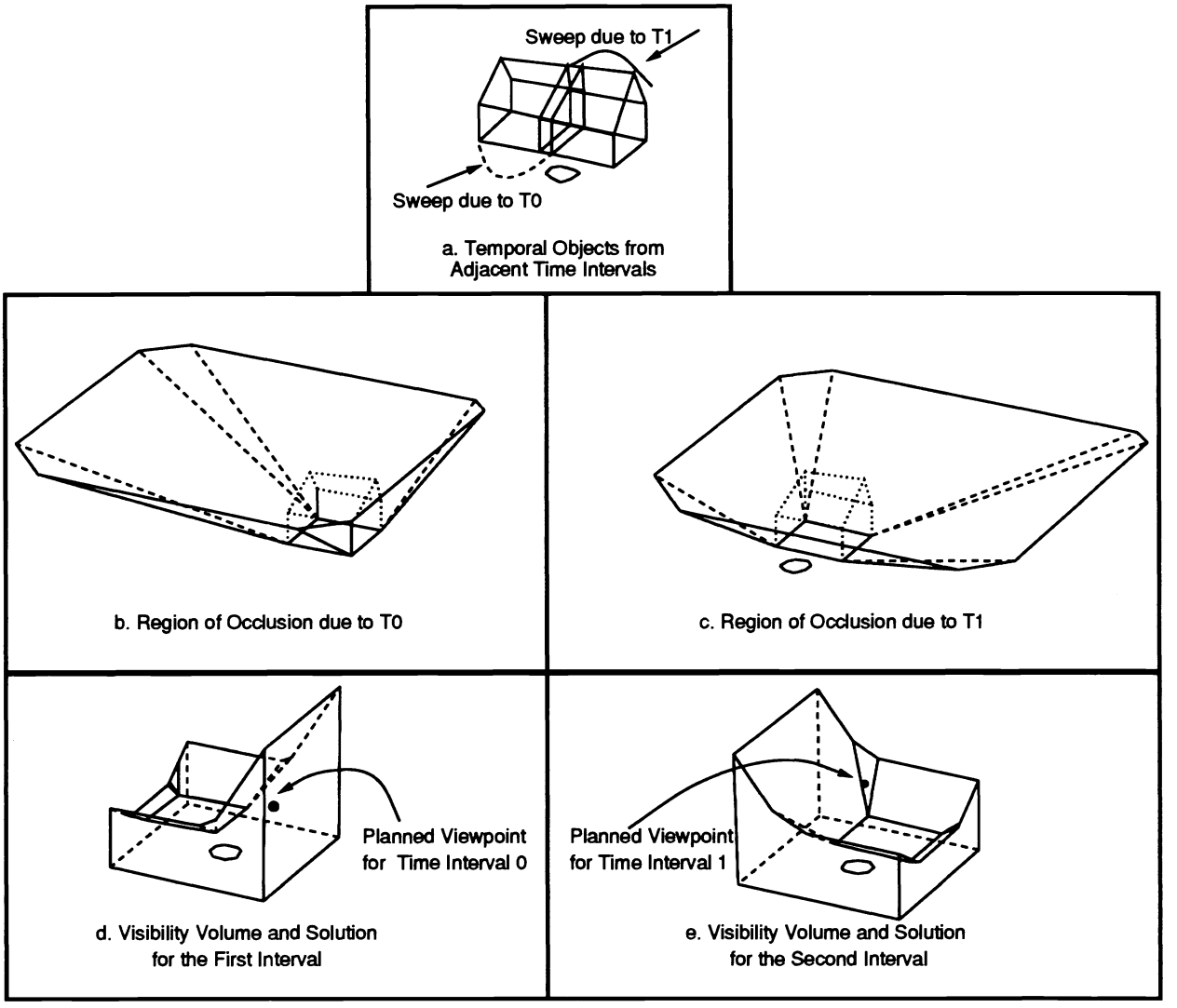

Figure 5: Simulation of *TSP* and MVP showing Binary Partition

volume (figure 4d). Note that any point in this volume is now guaranteed to have an occlusion-free view of the target region for the entire time interval T. This volume is used in the optimization portion of MVP to generate the shown viewpoint which is clearly valid for the entire motion of C. Note that in Figure 4e, the planned viewpoint is clearly within the visibility volume; the line of sight from the viewpoint to all points on the target does not violate the temporal obstacle.

In the next example, we lowered C so that MVP would be unable to find a viewpoint which maintained both visibility and focus over the entire time interval. The partition of the time interval produced two sub-intervals, both of which had valid viewpoints easily computed using MVP (see Figure 5).

9 Conclusions

This paper has presented two main ideas. First, we have presented work in extending geometric planning problems to include time and motion. The use of the temporal object makes the temporal components of the problem invisible

to any underlying geometric solver. This makes it a very useful notion; many other geometric planning problems may benefit from this concept.

Second, we have successfully extended our MVP system to plan sensor locations in a time-varying environment. This is notable in that to the best of our knowledge, motion has not been widely addressed in the sensor planning literature. This particular planning task is quite important to us and we plan on concentrating future work on improving the performance of MVP under various conditions, especially under more weakly constrained motion conditions.

For example, while the methods outlined above work well for the translational motion of known obstacles, they have yet to be used to model the motion of the target as well as the obstacles. The fact that we have constrained ourselves to moving obstacles has allowed us to ignore the effects of motion on all constraints other than visibility. Once we allow the target to move, resolution, focus, and field-of-view constraints must be dealt with in an environment which changes over time. Generating a temporal object for the target seems to be a reasonable approach, but it may not be sufficient. The problem of a moving target is inherently more complex than that of a moving obstacle.

Note that in sweeping a polyhedral obstacle along a linear trajectory, as we have done, the result is another polyhedral obstacle. Now note that a target consisting of a single point, when swept along a linear path, yields a line; a linear target swept along a linear path yields a polygon; a polygonal target swept linearly yield a polyhedra. Future work will explore how the planning algorithm can plan to view this temporal target, which is one dimension higher than the original target.

There may be other methods which may be used to plan sensor viewpoints around moving obstacles which take advantage of the fact that the visibility volume is the only constraint which changes through time. We may be able to examine how the actual obstacles move in relation to the current sensor's line-of-sight to the target to determine when replacement may be needed, for example.

Allowing motion in the sensor planning problem opens it up to many more subproblems which we hope to explore in future work. For example, it may be desirable to examine ways of ensuring that the series of viewpoints lay along a path which is accessible. We may also be exploring ways of generating a continuous path along which to move the sensor which maintain a particular view of a target.

References

[1] C. I. Connolly. The determination of next best views. In *Proceedings 1985 IEEE International Conference on Robotics and Automation*, 1985.

[2] C. K. Cowan and A. Bergman. Model based synthesis of sensor locations. In *Proceedings 1989 IEEE International Conference on Robotics and Automation*, December 1989.

[3] M. Erdmann and T. Lozano-Perez. On multiple mpoving objects. *Algorithmica*, 2(4):477–521, 1987.

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

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

[6] T. Lozano-Perez. Spatial planning: A configuration space approach. *IEEE Transactions on Computers*, 32(2):108–120, 1983.

[7] 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.

[8] S. Sakane, M. Ishii, and M. Kakiura. Occlusion avoidance of visual sensors based on a hand eye action simulator system: HEAVEN. *Advanced Robotics*, 2(2):149–165, 1987.

[9] Spatial Technology Inc. *ACIS Kernel Interface Guide.*

[10] Konstantinos Tarabanis. Sensor planning in computer vision: A review. Technical report, Columbia University Department of Computer Science, 1991.

[11] Konstantinos Tarabanis and Roger Y. Tsai. The MVP sensor placement and modeling system for machine vision. In *Proceedings SPIE Intelligent Robotic Systems Conference on Sensor Fusion IV: Control Paradigms and Data Structures*, Boston, MA, November 1991.

[12] Konstantinos Tarabanis, Roger Y. Tsai, and Steven Abrams. Planning viewpoints that simultaneously satisfy several feature detectability constraints for robotic vision. In *Proceedings Fifth Inernational Conference of Advanced Robotics*, 1991.

[13] Kostantinos Tarabanis, Roger Y. Tsai, and Peter K. Allen. Automated sensor planning and modeling for robotic vision tasks. In *Proceedings 1991 IEEE International Conference on Robotics and Automation*, April 1991.

[14] 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.