Computing Camera Viewpoints in an Active Robot Work Cell

Steven Abrams[*]
IBM T. J. Watson Research Center
Yorktown Heights, New York 10598, USA
abrams@watson.ibm.com

Peter K. Allen
Department of Computer Science
Columbia University
New York, New York 10027, USA
allen@cs.columbia.edu

Konstantinos Tarabanis
University of Macedonia
Thessaloniki, Greece
kat@macedonia.uom.gr

Abstract

This paper presents a dynamic sensor-planning system that is capable of planning the locations and settings of vision sensors for use in an environment containing objects moving in known ways. The key component of this research is the computation of the camera position, orientation, and optical settings to be used over a time interval. A new algorithm is presented for viewpoint computation which ensures that the feature-detectability constraints of focus, resolution, field of view, and visibility are satisfied. A five-degree-of-freedom Cartesian robot carrying a CCD camera in a hand/eye configuration and surrounding the work cell of a Puma 560 robot was constructed for performing sensor-planning experiments. The results of these experiments, demonstrating the use of this system in a robot work cell, are presented.

1. Dynamic Sensor Planning

This paper presents a system capable of planning the locations and settings of vision sensors for use in a dynamic environment. The system can plan the positions, optical settings, and movements of a camera needed to meet a set of task constraints in a modeled, dynamic environment. We call this type of sensor planning *dynamic sensor planning*, as compared with *static sensor planning*, in which neither the objects nor the sensors are moving.

[*]The research described in this paper was performed while this author was at the Columbia University Department of Computer Science.

The International Journal of Robotics Research
Vol. 18, No. 2, February 1999, pp. xxx-xxx,

It is interesting to note that while research in planning the *motion* of robots and using sensory feedback to monitor and control these motions abounds, what research there has been in planning sensing strategies has focused on environments in which there is no motion (Tarabanis, Allen, and Tsai (1995) offer a survey of sensor-planning research). It is our belief that an intelligent robot system capable of planning its own actions should be capable of planning its own sensing strategies.

The most general statement of the dynamic sensor-planning problem is:

> Given a model of an environment that includes models of the motion taking place, a set of features to be monitored, models of the sensors being employed, and a description of the vision-task constraints, produce a plan to use the sensors in such a way as to satisfy the task constraints.

However, there are a number of possible variations of this problem:

Surveillance planning. There are *N* different immobile sensors, each of which can be used for a different time interval. In this case, the goal of the system is to determine the best locations for these stationary sensors so as to provide valid viewpoints for all of the specified time intervals.

Snapshot planning. The constraints must be met for a specified set of features at any (unspecified) point in time. That is to say, the entire task need not be monitored, and it makes no difference when the image is taken, so long as the constraints for the image are met. The system must determine not only where to position the camera, but when to take an image.

Viewpoint path planning. There is only one movable sensor. The system needs to compute a continuous trajectory through the sensor's parameter space that ensures valid viewpoints through all of the desired time intervals.

Multicamera path planning. There is more than one mobile sensor, each of which can be used for a different time interval. This permits the sensor-planning system to compute paths through the sensor's parameter space that are only piecewise continuous: one sensor can be in use during time interval T_i, while another is being moved into place for time interval T_{i+1}.

The focus of this paper is *surveillance planning*: planning the positions of more than one sensor, each of which will be active for a different time interval, to guarantee that certain features can be robustly monitored during a robot task that is known a priori. There are two basic cases that must be dealt with separately in the surveillance planning problem. First is the case where the target objects, i.e., those features that must be viewed remain stationary, and other objects, such as the robot that is performing some operation on the stationary part, moves. This case arises in a variety of fixtured manufacturing tasks such as spray painting and spot welding. Second is the case where the targets to be viewed are moving as well. This case arises in teleoperation and in other types of manufacturing tasks such as pick-and-place, part insertion, etc.

The main difference between these two cases is that in the first case, if a viewpoint is found to be valid at some point during the task, it is guaranteed to be valid with respect to all *optical* constraints at all times during the task. This is because the functions defining the constraints only depend on the target-feature locations and the sensor parameters, and not on the positions or orientations of obstacles in the environment. This fairly obvious but important property allows us to ignore changes in the optical constraints over time and focus only on changes in the geometric parameters; i.e., the visibility constraint.

The second case is more difficult, because it requires an examination of how changes in the position and orientation of the target features affect the optical parameters, particularly focus and resolution. However, if the viewpoint is considered in terms of a coordinate frame attached to the feature set, the target can always be considered stationary, with the entire environment considered as moving. The only limitation is that the entire feature set must be moving as a single rigid body; i.e., features cannot move independently.

The subsequent sections of this paper discuss the work that we have done to date on solving the former case, and discuss some extensions that may be required to address the latter case. Figure 1 shows the basic experimental setup for our sensor-planning system: a robot arm, able to operate in a work cell, and a gantry robot, used for moving the camera through a computed trajectory.

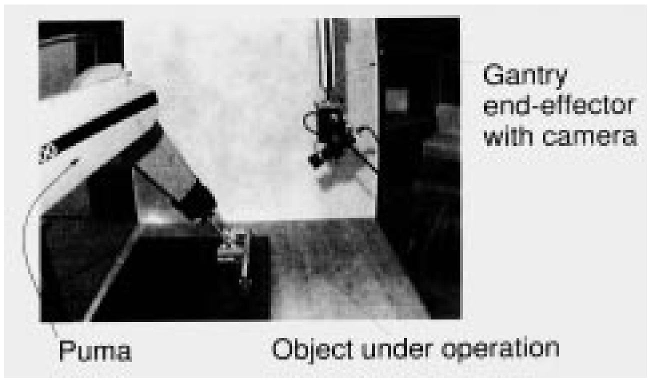

Fig. 1. Robot work cell for dynamic sensor planning.

2. Background Material and Prior Work

The bulk of the research in sensor planning for feature detectability has been concerned with the computation of sets of positions, orientations, and optical settings for a camera (and, in some cases, for light sources) that will give satisfactory views of a static scene. In the vision-sensor-planning literature, the constraints that have been considered important for judging the quality of a viewpoint are:

- *visibility*—the features must not be occluded by other objects in the scene;
- *field of view*—the features must be within the sensor's field of view;
- *resolution*—features of a specified minimum size must be resolvable by the sensor;
- *focus*—the target features must be properly focused; and
- *illumination*—there must be sufficient and appropriate lighting so that the features will be discernible in the image.

Sensor-planning systems have fallen into two basic categories. First, there are those methods that discretize the sensor's parameter space, generating candidate points that are tested against the constraints to see if they are acceptable. Those systems categorized as "generate-and-test" systems by Tarabanis and colleagues (Tarabanis, Allen, and Tsai 1995), such as the HEAVEN (Sakane, Ishii, and Kakikura 1987) and VIO (Niepold, Sakane, and Shirai 1987) systems, as well as the IVIS system (Tarbox and Gottschlich 1995), are examples that fall into this category.

Second are the techniques that take an analytical approach, directly computing valid viewpoints or sets of viewpoints from the constraints, referred to as "synthesis" methods in Tarabanis's survey. The machine vision planning (MVP) system (Tarabanis 1991; Tarabanis, Tsai, and Abrams 1991; Tarabanis, Tsai, and Allen 1994b, 1995; Tarabanis, Tsai, and Kaul 1996) and the sensor-planning research done at SRI (Cowan 1988; Cowan and Bergman 1989), for example, fall into this category. Two of the major contributions of the MVP system are directly used in the present research, specifically:

1. precise models of vision sensors and the equations that govern the optical constraints; and
2. methods of computing the visibility regions for convex and concave polygonal targets in static polyhedral scenes.

2.1. Feature-Detectability Constraints

It is now instructive to review the optical feature-detectability constraints, as they were formulated in the MVP system. (A more-detailed discussion with derivations can be found in the work of Tarabanis, Tsai, and Allen (1994a).) In the equations that follow, r_v is the position of the front nodal point of the lens, v is the unit vector along the optical axis, a is the diameter of the aperture of the lens, d is the distance from the back nodal point of the lens to the image plane, and f is the focal length of the lens. Figure 5 graphically illustrates the optical constraints of resolution, field of view, focus, and visibility.

2.1.1. Resolution

Considering only the *pixel resolution*, to ensure that every pair of points a distance l from one another is resolvable, one must ensure that each of these points images to a distinct pixel on the image plane.[1] Tarabanis found that the following equation defines the size of a linear feature in the image plane:

$$\frac{d\, l \, |\, [(r_a - r_v) \times u] \times v \,|}{((r_a - r_v) \cdot v)\, ((r_b - r_v) \cdot v)} \quad = \quad w, \tag{1}$$

where r_a and r_b are the endpoints of the linear feature to be viewed, u is the unit vector along the linear feature (from r_a to r_b), and l is the length of that feature; w is then the size of that feature in the image plane. If w is at least as large as the largest distance between pixels (i.e., the diagonal, for rectangular pixels), then every feature of length l is resolvable.

2.1.2. Focus Constraint

A thick-lens imaging system with a finite aperture is perfectly focused at a specific distance D (measured along the optical axis) given by the equation

$$D = \frac{afd}{a(d-f)}. \tag{2}$$

In practice, however, if a point images to a blur circle of a given size c, it is considered to be sufficiently in focus for a given application. For most computer-vision applications using CCD-based sensors and frame grabbers, c is chosen to be the size of 1 pixel, measured along its smallest side. This ensures that the blur circle of any feature point will be smaller than 1 pixel. The system is then sufficiently focused for a range of depths from D_1, the far limit of the depth of field, to D_2, the near limit. These limits are given by Krotkov (1989):

$$D_1 \quad = \quad \frac{afd}{a(d-f) - cf}, \tag{3}$$

$$D_2 \quad = \quad \frac{afd}{a(d-f) + cf}. \tag{4}$$

2.1.3. Field of View

Any feature that projects completely or partially outside of the sensor area is not within the sensor's field of view. So, for a camera with a rectangular sensor, the field-of-view boundary is a rectangular pyramid with its apex at the front nodal point of the lens.

For a rectangular image plane, the field-of-view angle depends on the direction in which it is measured (i.e., horizontally, vertically, or diagonally). However, sensor-planning systems generally consider the image plane to be symmetrical about the optical axis for the purposes of field of view, obviating the need to compute a third orientational parameter for the camera (the roll angle about the optical axis). The field-of-view angle of the camera, α, is then computed based on the length of the smaller side of the sensor area, $I_{\min}$. Therefore

$$\alpha \quad = \quad 2 \tan^{-1}(I_{\min}/2\, d). \tag{5}$$

For simplicity, when considering the field-of-view constraint, most sensor-planning systems aggregate the set of features to be imaged into the minimum sphere circumscribing the features (centered at r_s, of radius R_f), and then ensure that this sphere is entirely within the sensor's field of view. Looking at Figure 2, it can be seen that for a fixed camera orientation of v, the lens is constrained to lie in a cone. The apex of this cone is given by

$$r_k \quad = \quad r_s - \frac{R_f}{\sin(\frac{\alpha}{2})}\, v. \tag{6}$$

Moreover, the camera can never be positioned closer than $\frac{R_f}{\sin(\alpha/2)}$ to r_s, yielding the field-of-view limiting sphere shown in Figure 2.

3. Outline of the Approach

To recap, the exact problem we are discussing is one in which one or more cameras are 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

1. Although other constraints may come into play to determine the true resolution of the system, such as the quality of the optics, the lighting, and so forth, it is the pixel resolution with which we are concerned in the present research.

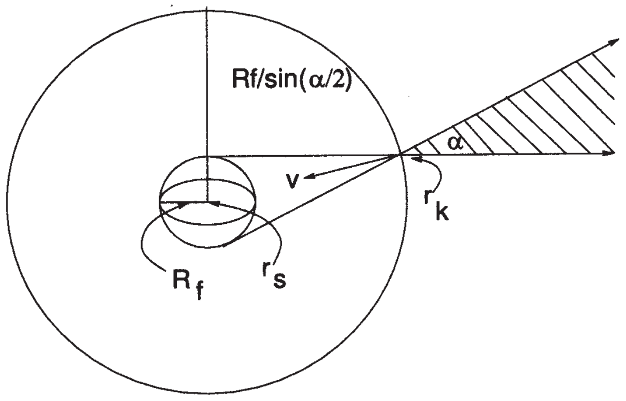

Fig. 2. The field-of-view constraint forces the camera to be positioned outside of the larger sphere, regardless of its orientation. However, for any specific orientation v, the camera is constrained to be positioned within a cone as shown.

mechanical parts, move in a way that is known a priori. The problem is to find where to place the camera(s), and when to use each camera, so that at all times during the task, we have a "good" viewpoint for monitoring the target features. This is typical of a robotic work cell, where objects that are to be assembled and inspected may be occluded by the motion of robotic hands and arms during the assembly process.

In this research, we are planning the positions and settings of the sensor, not the illuminator. Therefore, a viewpoint is considered "good" if the features are resolvable to the given specification, properly focused, completely contained within the camera's field of-view, and completely unoccluded.

Specifically, the system presented computes:

- *position*—the position in $\mathbf{R}^3$ of the sensor (actually the front nodal point of the lens);
- *orientation*—the pan and tilt components of the direction of the optical axis (the roll about the optical axis is not considered);
- *back nodal point to image-plane distance*—not the focal length of the lens, but rather the distance from the back nodal point of the lens to the image plane (for a given focal-length lens, this parameter primarily determines the focused distance of the lens, but also impacts the field of view and resolution);
- *aperture*—the size of the lens opening, which impacts the focus constraint and dictates the requisite illumination; and
- *temporal intervals*—owing to motion in the environment, the position, orientation, focus, and aperture settings computed may only be valid for a specific time interval. The system computes a number of points and their associated time intervals so that the entire feature set can be monitored during the entire task.

To accomplish sensor planning, the following data must be available to the system:

- *object models*—regular polyhedral models of the objects in the environment that indicate the target features to be monitored are needed to satisfy the visibility constraint;
- *motion models*—models of the motion taking place in the environment are needed to ensure that the visibility constraint is satisfied at all times;
- *sensor resolution*—the maximum distance between adjacent pixels on the image plane is needed to ascertain that the resolution constraint is met; and
- *sensor element size*—the minimum dimension of each pixel on the image plane is needed to meet the focus constraint; i.e., so that none of the feature points are blurry enough to cover more than 1 pixel.

Our basic idea is to find temporal intervals for which viewpoints can be computed, compute swept volumes representing the motion of the objects during those intervals, and then use a viewpoint- computation algorithm to compute the sensor positions, orientations, and optical settings that are valid for each interval. This approach, therefore, has three main components: swept-volume computation, viewpoint computation, and temporal decomposition.

3.1. Swept-Volume Computation

Given that we have an object O whose trajectory Q_T is known over a time interval T, we define $\mathcal{S}(O, Q_T)$ to be the volume swept out by O during T. For example, in Figure 3, if object O is rotating about point p at a rate of 90° per second, and T is 1 sec, then $\mathcal{S}(O, Q_T)$ is the swept region shown on the right.

Let V represent visibility volume for $\mathcal{S}(O, Q_T)$. Set V is the set of all points (in $\mathbf{R}^3$) that give views of the target which have no obstructions (due to O) for the entire time interval. A

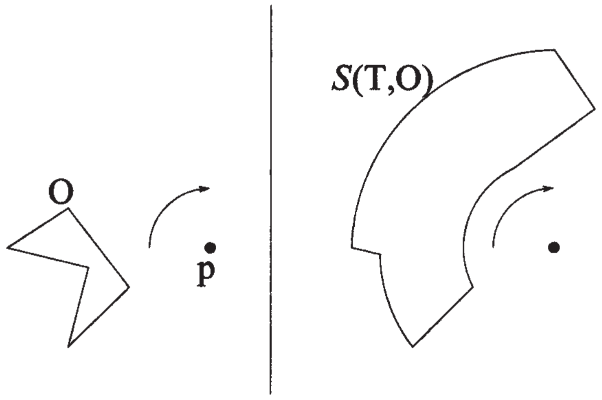

Fig. 3. Two-dimensional swept region example, generating $\mathcal{S}(O, Q)$.

2-D example is shown in Figure 4. Any point in the visibility region V may be used to monitor the target for the entire time interval T without fear of occlusion. Any point in the occluded region, at some moment during time interval T, will have an occlusion due to the moving object O. Therefore, we decided to use swept volumes as part of the process of finding viewpoints that will be valid for a given time interval.

A number of researchers have looked into the very interesting problem of computing the volumes swept by moving objects, including Hui (1994), Kaul (1993) Korein (1985), Martin and Stephenson (1990), Schroeder, Lorensen, and Linthicum (1994), Sourin and Pasko (1995), Wang and Wang (1986), and Weld and Leu (1990). However, for the purposes of this research, we found that the existing solutions all had a number of drawbacks, including unacceptable limitations on the classes of motion or moving objects, and impracticality of implementation. Therefore, we developed a new algorithm for computing polyhedral approximations of the volumes swept by arbitrarily moving polyhedra. The models are geometrically and topologically valid (i.e., regular solids in $\mathbf{R}^3$).

The algorithm is based on sampling and interpolation. To summarize, the moving object is stepped through its trajectory. At each step, a set of faces is created stretching from the edges at one position to the corresponding edges at the next position. These faces, along with copies of the faces of the moving object itself at certain points during its trajectory (where needed) are all collected into a set. This set is then processed so that the outer boundary of the swept volume can be computed. This algorithm is the subject of separate papers, and the interested reader should consult these earlier works (Abrams 1997; Abrams and Allen 1997) for details.

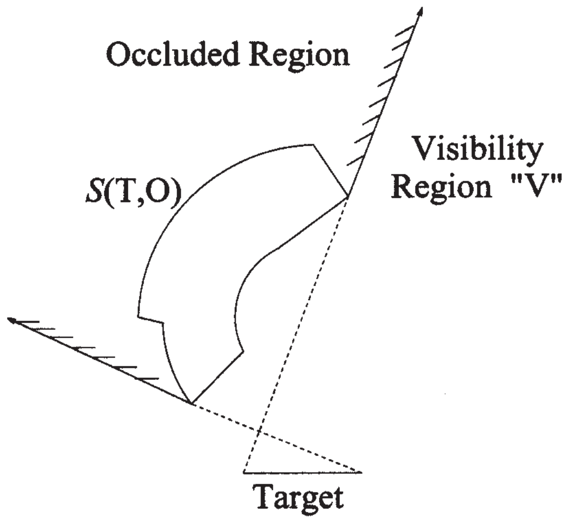

Fig. 4. Visibility region for $\mathcal{S}(O, Q)$.

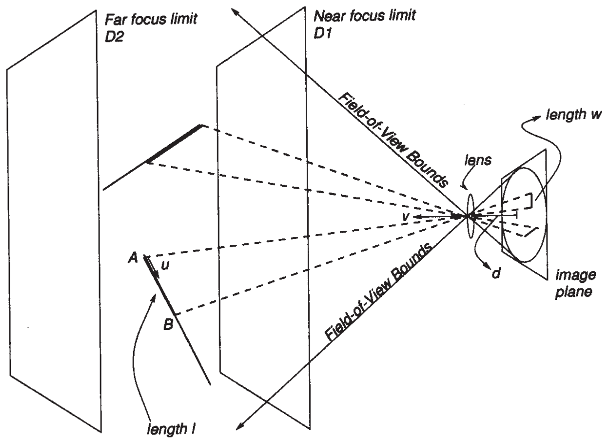

Fig. 5. Feature-detectability constraints. The linear features shown meet all of the optical constraints. They are within the field-of-view cone, between the depth-of-field limits, and project to appropriate sizes on the image plane.

3.2. Temporal Decomposition

The primary question is, “What makes a good temporal interval?” The intuitive answer is that, for dynamic sensor planning, a temporal interval is “good” if it is as large as it can be while still allowing the computation of a single robust viewpoint. This, of course, begs the question, “How can a good temporal interval be found?”

One rather obvious method is to leave it to *humans* to find good temporal intervals. Humans have higher task-level knowledge of the problem being solved, and can often do a very good job of dividing it into intervals, each of which should be monitored by a single viewpoint. However, our goal is to obtain an automated process for computing these viewpoints.

The method employed for the present research is a binary temporal interval search. Assume we have a polygonal target τ that we wish to monitor during the time interval $T = [t_0, t_n]$. During T, there is a set of known obstacles O_1 through O_m, which move in known paths. The goal is to plan a single viewpoint that is valid for the entire interval, if such a point exists, or to determine a sequence of viewpoints which, when executed at the appropriate times, allow the features to be monitored for the entire interval.

Temporal Interval-Search Algorithm

1. Compute the swept volume $\mathcal{S}(O_i, Q_T)$ for each of the m obstacles.
2. Attempt to compute a viewpoint using $\mathcal{S}(O_0, Q_T)$ through $\mathcal{S}(O_m, Q_T)$ as well as all stationary objects in the environment as the set of potentially occluding bodies.
3. If a viewpoint can be found successfully, use this viewpoint for the entire time interval T.

4. If no such viewpoint is obtainable, divide the time interval in half, yielding $T_1 = [t_0, t_{n/2}]$. If T_1 is shorter than the smallest allowable time interval, report a failure. Otherwise, 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 original interval T.

The algorithm essentially tries to quickly find the endpoint of an interval for which a single viewpoint can be found. In step 4, notice that we allow the user of the system to specify a minimum time interval. This prevents the system from examining infinitesimal intervals when no viewpoint can be found. It can also be used by the user to limit the temporal resolution of the system in preventing, for example, motions of the camera every second while monitoring a 2-hour task.

While there are refinements that could be made to this interval-search algorithm, the most dramatic improvement probably can come from human intervention. If there are clearly distinct intervals of the task to be monitored, the system will be more efficient if the user breaks them up in advance and feeds them, as separate tasks, into the sensor-planning system. As we discuss in Section 6, the temporal interval decomposition problem is an area open for investigation in the future.

4. Viewpoint Computation

In conducting dynamic sensor-planning experiments, we have found that the optimization approach used in MVP, while notable for its generality, has some drawbacks. Since some of the constraints are nonlinear and, at points, nondifferentiable, general-purpose optimization algorithms can have difficulty computing viewpoints. Convergence cannot be guaranteed, and the system is sensitive to the initial guess and the weights given to the constraints. These shortcomings triggered our search for new algorithms, starting with a review of the field.

The discretization approaches taken by some (such as Sakane et al. (1992) and Sakane, Sato, and Kakikura (1987)) are more straightforward than optimization (involving the search of a tessellated sphere), and can easily be shown to converge. However, they tend to ignore or assume certain imaging parameters (i.e., the distance from the viewpoint to the features is set to meet an overall magnification requirement, or the viewing orientation is assumed to be toward the center of the feature points), and the computational cost of finer tessellations can lead to very costly searches. The synthesis approach taken by Cowan and colleagues (Cowan and Kovesi 1988; Cowan and Modayur 1993) does not suffer from the computational expense of the generate-and-test methods. Their method of computing admissible domains in $\mathbf{R}^3$ for each constraint is also straightforward, but they too consider only a subset of the parameters and a subset of the solution space, thereby eliminating a number of potential solutions a priori.

Our goal was to develop a viewpoint-computation algorithm that has the benefits of MVP (i.e., uses analytical formulations for the constraints, avoids implicit assumptions about the viewing parameters, does not require the expense of generate-and-test, and solves for all viewing parameters), and of the other methods (i.e., straightforward search with demonstrable convergence).

4.1. A New Approach to Viewpoint Computation

Viewpoint Computation Algorithm

1. Compute a candidate set in $\mathbf{R}^3$. This set, called V_c, will be a superset of the projection of the solution set into $\mathbf{R}^3$.
2. Search the candidate set for points as far from the boundary of V_c as possible, finding a number of local maxima. These are potential solutions for the camera position.
3. Compute the camera orientation for the solutions found.
4. Compute the optical settings for the solutions found.
5. Verify the correctness of these solutions.

Overall, we are taking a decomposition-based approach toward solving this problem; that is, we are implicitly weighing certain constraints more than others. Focus is given the least consideration, since it can be handled at the latest stages by controlling the focus setting on the lens, the aperture, and if needed, the lighting. In contrast, visibility must be considered foremost, as the only way to give a previously occluded viewpoint an unobstructed view of the scene is to remove the obstacle or move the camera! As it turns out, the field of view and resolution constraints can be combined in an interesting manner and considered together. We shall see that these two constraints often directly oppose one another, and therefore in a decomposition-based approach such as this, should be considered simultaneously.

One constraint we have imposed upon ourselves in obtaining this candidate volume is to approximate no more than is necessary, and to assume nothing additional about the camera parameters. We are not interested in assuming, for example, a specific camera orientation. In our opinion, while there are clear benefits to working in three dimensions rather than in the higher dimensions of the camera parameter space,[2] discarding possible solutions to the problem in order to simplify the computations should be avoided.

Note that in this approach, the selection of optics (i.e., the focal length of the lens) is not made automatically. There are a number of reasons for this. To begin with, there is a trade-off between the distance to the feature points and the focal length.

2. The primary such benefit is the availability of 3-D computational tools.

That is, a very similar image is obtained using a short focal-length lens at a short distance as would be obtained using a longer lens from a greater distance. The difference is in the exaggerated perspective obtained with the shorter lens. Because the amount of perspective desired in the resulting image is highly dependent upon the application and has an impact on the resolution constraint, the selection of focal length is a choice best left to the user of the system. There is generally a discrete set of focal lengths available to a user, not a continuous range. The user knows which focal lengths are available, and can select one. Finally, there are a number of situations where selection of an inappropriate focal length prevents the computation of a viewpoint. As we shall see, these situations are readily noticeable and can be automatically determined.

4.2. Computing a Candidate Set: Merging Field of View and Resolution

Before discussing what V_c should look like, we will examine the merging of the field-of-view and resolution constraints. The field-of-view constraint, in essence, only prevents the camera from being placed too close to the features (i.e., a distance $\frac{R_f}{\sin(\alpha/2)}$ from the center of the sphere circumscribing the features). Beyond this distance, any position can satisfy the field-of-view constraint simply by adjusting the orientation. What we shall see in this section, however, is that the resolution constraint alone does not place a bound on the position of the camera. The sensor's position and orientation *together* determine the resolution constraint. We will show how resolution can be considered in conjunction with the field-of-view constraint to yield bounds on our candidate set in $\mathbf{R}^3$.

For the moment, let us examine the constraint imposed on the camera's position, assuming that the optical settings and orientation are kept constant. For any given viewing *direction* v and for each feature i, there is a field-of-view cone that constrains the position of the camera. The axis of this is the cone in the direction of v, and its apex is at point r_k. This was shown in Figure 2 and in eqs. (5) and (6). Technically, the half-angle of this cone is dependent upon d, the distance from the back nodal point of the lens to the image plane. However, under normal viewing conditions, d can be approximated by f, the focal length, without having a significant effect on the field of view.[3]

Further, for any given viewing direction and for each feature i (a linear feature of length l),[4] there is a surface S_{i_v} that constrains the position of the camera due to the resolution requirement. By plotting eq. (1), keeping all parameters constant (other than camera position), we can see this surface. Figure 6, for example, shows the surface constraining the position of the viewpoint for a vertical feature (i.e., $r_a = (0, 0, 0)$, $r_b = (0, 0, 1)$), viewed from a downward-vertical direction ($v_0 = (0, 0, -1)$,) such that $l = 1$ mm, and $w = 1$ pixel. Figure 7, showing the same plot for a different viewing orientation (v_1), has a very different shape. Again, although d appears in these equations, it can be approximated using f.

For any viewing orientation, the set of valid camera positions includes only those that are both *inside* the cone and *under* the resolution surface. Figure 8 shows the field-of-view cone superimposed on the resolution surface for viewing directions v_0 and v_1: each viewing orientation implies a single volume that constrains the camera's position.

Turning the camera's orientation *away* from a feature (while holding the camera's position constant) increases the resolution, albeit at the expense of field of view. If the camera is too far away to image the features with proper resolution, the camera can be rotated *away* from the feature, and the resolution will increase. However, one can only turn the camera away as much as the field-of-view constraint will permit.

So, in broad terms, the field-of-view constraint alone imposes a limit on how close the camera can be to any one feature (as a function of the orientation); and the resolution constraint alone imposes a constraint on how far the camera

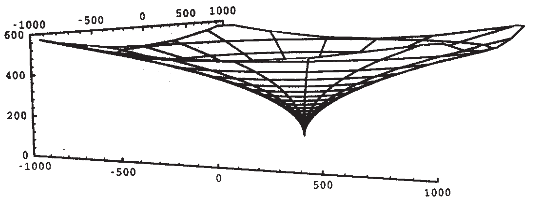

Fig. 6. When viewing a vertical linear feature of unit length at the origin, this surface illustrates the constraint on the camera's position if its orientation is fixed to be looking downward, i.e., $v_0 = (0, 0, -1)$.

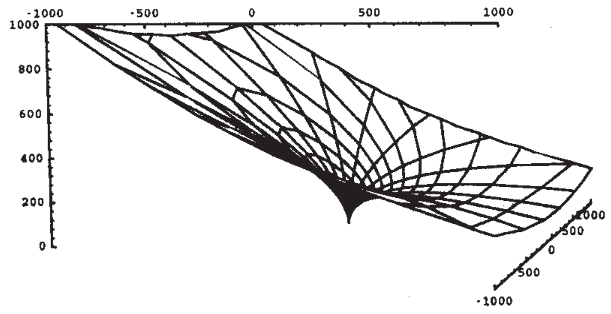

Fig. 7. When viewing a vertical linear feature of unit length at the origin, this surface illustrates the constraint on the camera's position if its orientation is fixed to be $v_1 = (-\sin(\frac{\pi}{8}), 0, -\cos(\frac{\pi}{8}))$.

3. When focused at infinity, $d = f$, and under normal viewing conditions, d does not differ from f by enough to affect this surface significantly. In any case, we will eliminate this approximation later.

4. Recall that l is the minimum feature size that must be resolvable.

can be from any one feature (also as a function of the orientation). Therefore, it should be possible to combine these constraints to yield a set of positions, all of which have the property that at least one orientation can be found such that both constraints are met. In fact, it is possible.

We have a method for computing a region in $\mathbf{R}^3$ called V_{FR_i}. Each point in V_{FR_i} has associated with it at least one orientation that satisfies both the field-of-view and resolution constraints for the linear feature i. Further, this volume contains all such points.[5]

By way of illustration, we computed the intersection of the regions bounded by resolution surfaces and the field-of-view cones to yield a single volume of feasibility for field-of-view and resolution constraints together, R_{i_v}. The shape of this volume is that of a cone "capped" by the resolution surface. These volumes are shown in Figure 9 for $v = v_0$ and $v = v_1$. A corresponding volume exists for every possible viewing orientation. Clearly, V_{FR_i} volume for this feature i must include the union of these volumes R_{i_v} for all viewing orientations.

5. This volume is a superset of the complete candidate volume V_c. Consideration of other constraints and features will further reduce the set.

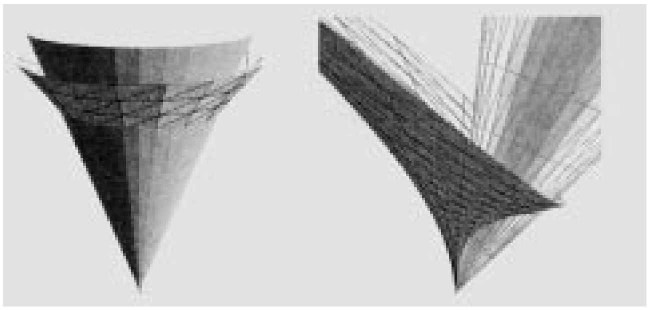
Fig. 8. Superimposition of the resolution and field-of-view constraints when $v = v_0$ (left) and $v = v_1$ (right).

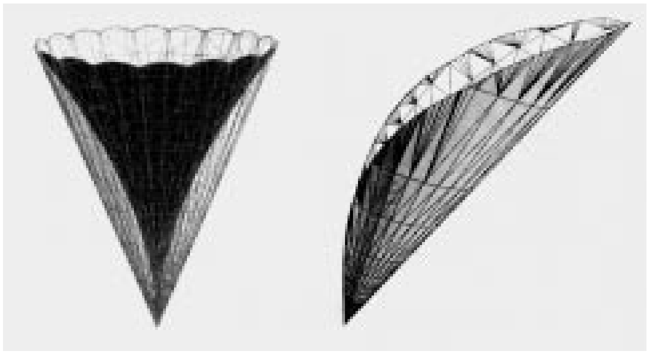
Fig. 9. Intersection of the resolution and field-of-view constraints for two viewing directions, v_0, v_1.

4.2.1. Resolution and Field of View in Two Dimensions

We first compute the region V_{FR_i} in two-dimensional viewing. We then extend these results to three dimensions; that is, the viewing orientation v is constrained to lie in the plane defined by the feature endpoints and the camera's entrance pupil. As before, the feature being viewed is a vertical linear feature at the origin. This figure can be produced by intersecting the 3-D constraints of Figure 9 with the plane containing the lens and the feature endpoints (in this case, the xz-plane). The figure shows the superposition of the field-of-view and resolution constraints for viewing the feature with an orientation of $v = v_1$. The intersection points between the constraints have been highlighted. The region R_{i_v}, in this situation, is the region inside the "V" and below the curve.

As we rotate the viewing angle in the plane, it turns out that these intersection points are constrained to lie on a circle. This can be seen in Figure 11, which shows eight pairs of field-of-view and resolution constraints, with the highlighted intersection points clearly lying on a circle. From a practical point of view, this means that if we were to select an orientation of the camera (in the plane), position the camera (in the plane) such that the feature is just at the edge of the camera's field of view, and back up until the feature just barely met the resolution criteria, the camera would always end up on a point on this circle, regardless of the chosen orientation.

To prove this, recall that the resolution constraint specifies the minimum image-space length w of a feature of a given object-space length l. The angle subtended by a feature of length w on the image plane (as seen from the back nodal point of the lens) depends on its position on the image plane (see Fig. 12). (If the image plane were a circular arc centered at the back nodal point, then the feature would subtend a constant angle, regardless of its position.) If the feature is imaged just at the edge of the field of view, we know its position. If the feature is imaged at the smallest tolerable size, we know that its size is w. Therefore, the intersection of the field-of-view and resolution constraints occurs when the feature is imaged at the minimum tolerable size at the limit of the field of view. At these points, the image of the feature always subtends a constant angle, say λ.

The feature itself, as seen from the front nodal point of the lens, must subtend the same angle as its image as seen from the back nodal point. Therefore, the points for which this condition holds true are exactly those points for which the feature subtends an angle of λ; these points lie on one of two circles. The two circles are reflections of one another across the line of the feature.

Specifically, what is the equation of this circle? The set of points for which a segment of length l subtends an angle of λ is given by a circle of radius R with a center on the perpendicular bisector of the segment at distance of H. Radius R and distance H are given by the following:

$$R = \frac{l}{2\sin(\lambda)}, \quad (7)$$

$$H = R\cos(\lambda). \quad (8)$$

These equations can be solved to yield (Abrams 1997):

$$R = \frac{c\,l}{2\,w\,\sin(\frac{\pi-\alpha}{2})}, \quad (9)$$

$$H = R\frac{-w^2+b^2+c^2}{2\,b\,c}, \quad (10)$$

where w is the length of the feature in the image, I is the width of the sensor, and d is the distance from the back nodal point to the image plane.

If we were to compute the union of all of these regions R_{i_v} for all viewing orientations (still constrained to the 2-D configuration), these circles would form part of the boundary of this union (and therefore the V_{FR_i} region). The other portion of the boundary would be shown in the lower-left corner of Figure 11. A similar graph, showing several field-of-view cones in 2-D, has been enlarged in Figure 10 to show this corner. In 2-D, the apexes of the field-of-view cones are constrained to lie on a circle, and this can be seen in the enlarged figure. The camera cannot be positioned closer than this circle, or the feature cannot be imaged within the camera's field of view at any orientation. Since we are, for the moment, considering only a single linear feature of length l, eqs. (5) and (6) give us that the radius of this circle is $\frac{l}{2\sin(\frac{\alpha}{2})}$, and it is centered at the midpoint of the edge.

So the region V_{FR_i} is bound by these circles: one small one, centered at the feature itself; and two larger ones, reflections of each other, given by eqs. (8)–(10). For any point inside this region, there exists at least one orientation that meets both the field-of-view and resolution constraints. Outside of this region, the field-of-view and resolution constraints cannot

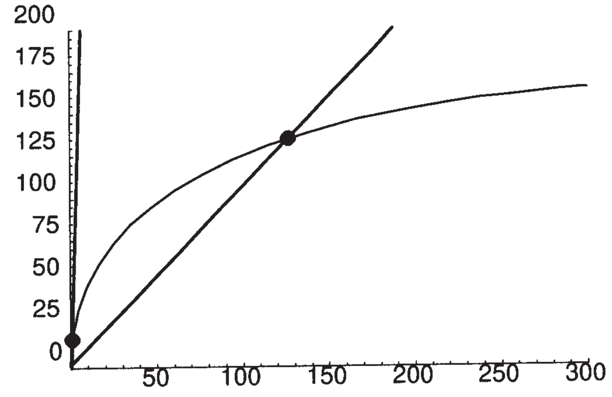

Fig. 10. The field-of-view cone and the resolution curve in 2-D, shown with the intersection points, for a given viewing direction.

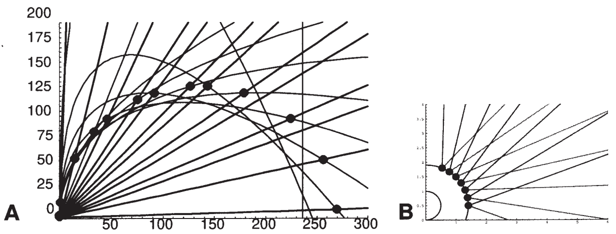

Fig. 11. (a) Field-of-view cones and resolution curves for several viewing orientations, in 2-D, shown with intersection points. (b) The enlargement of the lower-left corner of (a), showing the apexes of the cones sitting on a circle. The inner circle in (b) is the circle circumscribing the feature to be viewed.

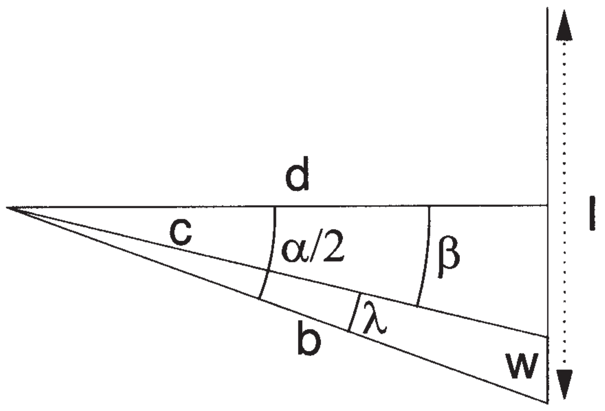

Fig. 12. The segment with length w in the image plane is imaged at the field-of-view limit, and subtends an angle of λ, as seen from the back nodal point of the lens.

simultaneously be met, since rotating the lens such that the segment is large enough results in the segment being imaged outside of the camera's field of view.

Referring again to Figure 11, there is another circle that can be seen, apparently formed by the envelope of the resolution curves (as described by Tarabanis, Tsai, and Allen (1994a). This envelope bounds the region in which the resolution constraint is met for *any* camera orientation, while the larger circle (formed by the intersection points) described here bounds the region in which *some* camera orientation will simultaneously satisfy both the field of view and the resolution. The inner circle is the region used by Cowan and Kovesi (1988) for their resolution constraint. It can be seen that by dropping Cowan's assumptions about the camera's orientation, additional regions are admitted to the candidate set.

4.2.2. Resolution and Field of View in Three Dimensions

To better understand what is happening in three dimensions, imagine a coordinate system set up by the feature's endpoints and the camera's front nodal point, as illustrated in Figure 13. That is, let A be the origin, AB be the z-axis, and r_0 (the nodal point) be on the xz-plane, to the positive x- side.[6] The camera's orientation can then be expressed as (θ, ϕ), where θ is the angle the optical axis makes with the y-axis (after being projected into the xy-plane) and ϕ is the angle that the optical axis makes with the z-axis. The two-dimensional example we have been using corresponds to setting θ to $\frac{\pi}{2}$, keeping the optical axis in the xz-plane.

Rotations away from the feature (in either ϕ or θ) improve resolution at the expense of field of view. However, the resolution changes by an amount that depends on the direction of rotation. The resolution improves more quickly by rotating ϕ (staying in the plane defined by the feature and the camera position). This can be seen in that eq. (1), where as ϕ approaches $\frac{\pi}{2}$, the minimum resolvable feature (l) is proportional to $\sin^2(\phi)$, while as θ approaches $\frac{\pi}{2}$, l is proportional to $\sin(\theta)$.[7]

However, the field-of-view constraint is only concerned with the magnitude of the angle that the optical axis makes with the feature set, not the direction. That is, as far as the field-of-view constraint is concerned, the combination of ϕ and θ must be such that the field-of-view constraint is still met.

Recall that in two dimensions, the goal was to find a constraint on the position of the camera that bounds the set of points for which the field-of-view and resolution constraints can simultaneously be met. In order to do this in three dimensions, we now see that it is sufficient to consider rotations in ϕ, leaving $\theta = \frac{\pi}{2}$. For if the camera is rotated such that the feature is at the field-of-view limit using a rotation where $\theta \neq \frac{\pi}{2}$, with the camera positioned such that the resolution constraint is precisely met, the camera can be further rotated in ϕ by setting $\theta = \frac{\pi}{2}$ and increasing the size of the feature; that is, as far as resolution is concerned, rotations in ϕ buy you more than rotations in θ. Field of view, on the other hand, imposes a maximum rotation regardless of direction. Therefore, there are positions for which the only way to satisfy both field-of-view and resolution constraints is to rotate the lens in ϕ, not θ. There are no points for which the converse is true. To find a V_{FR_i}, which includes exactly the set of candidate points for which there exists an orientation that satisfies both field-of-view and orientation constraints, we therefore leave $\theta = \frac{\pi}{2}$ and consider only rotations of ϕ.

The boundary of the positional constraint, then, is equivalent to taking the 2-D region V_{FR_i} and sweeping it about the line defined by the edge feature (which passes through the larger circle). This volume has the shape of a torus that has lost its hole (because its major radius is smaller than its minor radius), which has had a sphere subtracted from its center. The toroidal shape results from rotating the two larger, symmetric circles of the 2-D case about the axis of the feature; the sphere subtracted from the center results from rotating the smaller circle, which limits the position of the camera due to field of view alone. The major radius of this torus is H, and its minor radius is R, as defined in eqs. (8)–(10). The radius of the subtracted sphere is $\frac{l}{2\sin(\alpha/2)}$, from the field-of-view constraint eqs. (5) and (6) above.

Therefore, to complete the volume V_{FR_i}, we subtract a sphere from the center of the "pinched torus." For any point inside this volume, an orientation can be found that satisfies both the field-of-view and resolution constraints (but not necessarily the same one). For any point outside this volume, no such orientation exists. Figure 14 shows a cutaway view of this volume for a vertical linear feature.

The discussion thus far has been restricted to handling linear features of length l. To handle segments longer than l, we consider a segment of length l at each of the endpoints of the actual feature and intersect their corresponding candidate volumes (similar to the technique used by Cowan and Bergman (1989)). If and only if each of these segments is imaged to

6. This assumes, of course, that A, B, and r_0 are not collinear. If they are, the entire feature images to a dot, and the resolution constraint is not met.
7. For situations where the camera is not positioned on the x-axis of the specially constructed coordinate system, this analysis still holds, but a different ϕ is used. In these cases, ϕ is $\frac{\pi}{2}$ minus the angle between the viewing direction vector v and the segment connecting the front nodal point to the feature's distal endpoint.

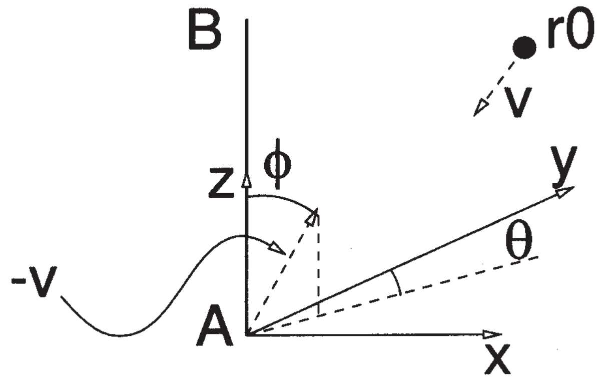

Fig. 13. The coordinate system setup. r_0 is in the xz-plane. Setting θ to $\frac{\pi}{2}$ corresponds to the "two-dimensional" viewing configuration.

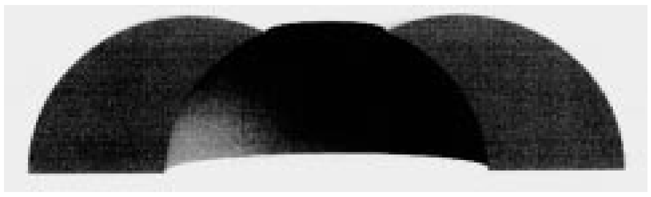
Fig. 14. Candidate volume for one linear feature, considering only field-of-view and resolution constraints. The figure has been cut away twice to show the inside, but the actual volume is symmetric about the two cutaway planes.

the appropriate resolution, then the resolution constraint for the entire feature is satisfied. Therefore, we position a V_{FR_i} at each end of the feature and compute their intersections. In addition to this, however, we must compute a field-of-view limiting sphere for the entire feature set; that is, in computing a constraint on the position of the camera, it is important to ensure that both ends of the feature can be imaged with proper resolution while in the camera's field of view. This is handled, as is the case in other sensor-planning research, by computing a single sphere of radius $\frac{R_f}{\sin(\frac{\alpha}{2})}$ and subtracting it from the candidate set.[8]

4.2.3. *Visibility*

For a feature to be detectable by a vision sensor, it may not be occluded by other objects. Using algorithms explained in detail by Tarabanis, Tsai, and Kaul (1996), we construct a polyhedral volume containing all points from which the target features are unoccluded. This volume, and therefore the visibility constraint, is independent of the optical parameters and the camera orientation. Tarabanis and colleagues describe two methods; we use the "decomposition-based" approach, where every face on the target and the occluding bodies are decomposed into convex pieces. Then, the visibility algorithm is run on every convex target/occluding-object pair. This generates volumes of occlusion, which correspond to the "shadow volumes" that would be caused by the occluding object if the target were an area light source.

4.3. Computing a Candidate Set: Merging All Constraints

To find the total feasible volume V_c, the volumes V_{FR_i} are computed for every feature i and intersected with each other, yielding one feasibility volume V_{FR}. Fortunately, many geometric modeling systems are capable of representing and manipulating tori and spheres, allowing these intersections to be performed analytically. Then V_{FR} is intersected with the visibility volume for the feature set.[9] From this result, the single field-of-view limiting sphere is subtracted, yielding a single set of feasible points in $\mathbf{R}^3$; namely, V_c. The boundary of V_c, therefore, consists of spherical, toroidal, and polygonal components.

At this point, there are several ways to determine that the lens selection may have been inappropriate. If the field-of-view limiting sphere is so large that nothing remains when it is subtracted from V_{FR}, this indicates that the focal length of the lens is too short for the features selected. Essentially, the constraints are saying that for the resolution constraint on any individual feature to be met, the camera must be placed so close to the feature set that the field-of-view constraint for all other features cannot be met. A longer focal-length lens should therefore be selected.

On the other hand, if subtracting the field-of-view limiting sphere from the visibility volume yields an empty volume, this indicates that the focal length is too long. In this case, the field-of-view constraint forced the camera to be so far away from the feature set that the features were occluded. If neither of these conditions holds, then we have a reasonable assurance that the lens selection is appropriate to the task at hand.

It is still, however, possible that V_c is empty or very small. In particular, the intersection of the visibility volume with V_{FR} may yield a very small or empty set. If this is a static sensor-planning problem, this is an indication that the feature set is too large to be monitored entirely from one viewpoint. There may be no single point from which the entire set is visible, while meeting the other constraints. If, however, this is a dynamic sensor-planning problem, the other possibility is that the task interval is too large to be monitored entirely by one viewpoint. The goal then is to position the camera at an appropriate point in V_c and compute an orientation for which all features will be unoccluded within the sensor's field of view, and properly resolvable.

4.4. Searching within the Candidate Set

Once a candidate set, V_c, has been computed, we need to find a viewpoint within this set. The optimality criteria used in MVP for computing a single viewpoint was distance. That is, it attempted to find a viewpoint (in the full eight-dimensional sensor-parameter space) as far away from the constraining surfaces as possible. In this new formulation, when we are searching a region in $\mathbf{R}^3$, this same criteria can be used while avoiding the global optimization difficulties encountered in MVP. We use a gradient-based search algorithm to maximize the distance between the viewpoint and the boundary.

There are a number of easily computable starting points for this search. For example, a point on the surface of each spherical patch may be used, or a point on the surface of each "capping" toroidal surface may be used. In fact, a point on any of the boundary surfaces can be used as a starting point, and the search will converge to a local minima inside the volume. If desired, the quality of the result may be improved by performing several optimizations, each starting from a different boundary surface, and the best overall result can be used.

Computing the orientation can be a bit more difficult, however. Recall that in the computation of the candidate set, we have ensured that for each feature there exists an orientation satisfying all constraints. However, we have not ensured that there exists a single orientation that satisfies all constraints for all features simultaneously. Therefore, a gradient-based

8. Note that if $H+R \geq \frac{R_f}{\sin(\frac{\alpha}{2})}$, then there are no valid viewpoints: resolution and field of view have, essentially, fought each other to the death.

9. Alternatively, each of the occlusion volumes can individually be subtracted from V_{FR}. This often yields greater performance and robustness over first unioning the occlusion volumes, then complementing this to form a visibility volume, and then intersecting the visibility volume with V_{FR}. The results are, however, equivalent.

search can be used to find a viewing orientation that satisfies all constraints for all features.

This search has a starting point easily available to it; namely, a point oriented toward the center of the sphere encompassing the points. Under the simplified field-of-view constraint used, this is the optimal orientation for field of view. However, once we have reached the stage of computing the final orientation, we discard this simplified field-of-view constraint. To do this, a *minimal circumscribing cone* is computed about the feature points. This cone has its apex at the computed camera position, and it has the smallest cone angle (γ) that still encompasses the feature points. The axis of this cone is the true optimal orientation for the field of view. Further, $\frac{\alpha}{2} - \frac{\gamma}{2}$ is exactly the amount of room we have to "adjust" the orientation before the field-of-view constraint is violated.

From here, the search algorithm computes the resolution of each feature. For each feature for which the resolution constraint is not met, it computes an adjustment that will improve the resolution of these features while still maintaining field of view. This adjustment can take the form of a constrained optimization, such as was used in MVP, but only optimizing the orientation subject to the field-of-view and resolution constraints. This process iterates until either it is "stuck" (i.e., reusing already-discarded viewing orientations, or oscillating between subsets of features), until no adjustment direction can be found, or until it converges on a successful orientation. Clearly, this optimization may not converge, as there may not be any orientation that satisfies all constraints for all features at the computed position. In such a case, one of the other locally minimal positions should be used.

In cases where none of the locally optimal positions yield successfully computed orientations, there are a number of possibilities. First of all, it has not been proven that this implies that no solution exists. It is, however, a strong indication that no *good* solution exists. It may well be an indication that the sensor is not up to the task. A bit of common-sense fault analysis will certainly help. If the visibility volume is extremely small, perhaps the feature set should be partitioned. If the V_{FR_i} volumes barely overlap, this indicates that a denser sensor array should be used, improving the resolution. It would then be wise to select a different lens or camera and recompute.

4.5. Computing the Optical Parameters

Once a single viewpoint and orientation have been successfully computed, we are left with the task of computing d and a to ensure that the features are in focus.

It seems to make little sense to compute the focus-related lens parameters in a system in which lighting is not planned. That is because the depth of field of the optical system can be changed simply by adjusting the aperture. The smaller the aperture, the larger the depth of field will be. Therefore, given sufficient lighting (or a sufficiently sensitive sensor), one can reduce the aperture as needed (in the limit, reducing the lens to a pinhole), thereby increasing the system's depth of field (in the limit, to an infinite depth of field), bringing the features into focus. Rather than computing absolute focus and aperture settings in the absence of lighting information, we take a different approach.

Any given position and orientation of the lens imposes a depth-of-field requirement: the aperture and focus must be set such that the near DOF limit is no farther away than the nearest feature point, and the far DOF limit is no closer than the farthest feature point. Of interest, then, are the values for d and a that will just barely satisfy the depth-of-field constraint.

We can therefore compute an optimal value for d, called d_{optimal} and the upper bound for a, called $a_{\max}$. Values $a_{\max}$ and d_{optimal} combine in such a way that $a_{\max}$ is the largest possible aperture for which there exists a value for d such that the entire feature set is within the camera's depth of field. That value for d is d_{optimal}. Then $a_{\max}$ can be used in an illumination-planning system (one of the planned extensions of the current system) to ensure that the correct amount of light is present to obtain an appropriate response from the sensor, given an aperture of no more than $a_{\max}$:

$$\begin{aligned} D_1 - D_2 &= (r_f - r_c) \cdot v, \\ D_1 &= r_f, \end{aligned}$$

we find that d and $a_{\max}$ are given by

$$d_{\text{optimal}} = \frac{2\, D_{\max}\, f (D_{\max} - D_f)}{2D_{\max}(D_{\max} - f - D_f) + f\, D_f}, \quad (11)$$

and

$$a_{\max} = \frac{2\, c\, D_{\max}(D_{\max} - f - D_f) + f\, c\, D_f}{f\, D_f}, \quad (12)$$

where

$$\begin{aligned} D_{\max} &= (r_f - r_v) \cdot v, \\ D_f &= (r_f - r_c) \cdot v. \end{aligned}$$

For a CCD imaging system, we use $c = 1$ pixel (measured across its minimum dimension) to ensure that no feature point is blurred more than 1 pixel.

Once the imaging parameters are computed, the resolution and field-of-view constraints are recomputed using the actual d to ensure that the viewpoint has not been invalidated with respect to these constraints. Since the viewpoint was chosen to be as far away from these boundaries as possible, and since these boundaries do not change significantly as the lens is focused, the validity of the viewpoint should not, in general, change. First, notice that this only *improves* the resolution, so there is no cause for concern; the field-of-view constraint must, however, be rechecked.

What can be done if this underestimate causes the field-of-view constraint to be violated? There are a number of options. First, one can conclude that the lens is inappropriate for the vision task; a longer focal-length lens can be selected. Second, the entire process can be iterated, using the new, known value of d instead of the approximation of f. While this will result in a different viewing position, the camera will be forced back to bring the features into the field of view. Third, an attempt can be made to back the camera up along v until the field-of-view constraint is met. The ability to do this, however, may be hampered by the resolution and/or visibility constraints, so these constraints need to be checked.

Alternatively, we can compute the smallest value of d for which the features are completely within the camera's field of view, and solve eqs. (3) and (4) for a. This is the maximal aperture that will bring the features into focus. Essentially, this equates to intentionally defocusing the features to sufficiently "zoom out" the image, and then closing down the aperture as needed to bring the features back into focus.

However, the purpose of the optimization is to ensure that we are as far from the bounds as possible, thereby avoiding this situation arising in the first place. In fact, in the sensor-planning experiments described below, the computed orientation and optical parameters were always found to be sufficient, and none of these additional corrective measures were needed.

5. Experimental Results

With the preceding discussion as a foundation, we can now go on to describe the complete viewpoint-computation algorithm.

Viewpoint Planning Algorithm

1. Compute the occlusion volumes V_{occ} for all features to be inspected.
2. Compute the volumes V_{FR_i} (the positional constraints for the combined field of view and resolution) for every feature i.
3. Compute the overall volume V_{FR} as the intersection of all V_{FR_i}.
4. Subtract the field-of-view limiting sphere for the entire feature set from this volume.
5. Clip this region to the half-space corresponding to the "front" of each target face.
6. Subtract volume V_{occ} from this region, yielding the overall candidate volume V_c.
7. Optimize to find a position (or positions) at least locally optimal within V_c.
8. For each solution point p, compute a circumscribing cone with its apex at p containing the entire feature set. Use the axis of this cone for the camera orientation v. Verify that this satisfies the resolution constraint. If it does not, optimize to find an orientation that does.
9. Compute the optimal focus and maximum aperture for the position(s) and orientation(s) using eqs. (11) and (12).
10. Verify that the position, orientation, and optical settings are all valid using the actual value of d computed, rather than the approximation of $d = f$.

A number of real and simulated experiments using this algorithm were run to verify the algorithm's ability to correctly plan viewpoints in dynamic scenes. Two real experiments are presented here. (Simulations and additional experiments are available in an earlier work (Abrams 1997).) In each of these experiments, the object being viewed was placed in the work cell of a Puma 560 robot, and a motion of the Puma was programmed. Above and around this robot, a 5-DOF Cartesian robot having a work space of approximately $1,000\text{ft}^3$ was built. This gantry robot was equipped with a Sony XC77 CCD camera in a calibrated hand/eye configuration. An overview of this work cell is shown in Figure 1.

This hand/eye setup was calibrated simply by measuring the camera and lens mount, yielding a calibration only accurate to within approximately a few millimeters. Further, the calibration of the two robots relative to one another was also done via several measurements rather than through a rigorous calibration procedure, yielding only a rough calibration. While a rough calibration is useful for demonstrating the robustness of a computed viewpoint, if the sensor-planning techniques described here are to be used in applications requiring more precise or demanding camera positioning, calibration methods such as those described by Tsai (1989) or Willson and Shafer (1993) should be used.

In each of the experiments, the objects to be used were first modeled using the ACIS geometric modeling system (Spatial Technologies 1995). The objects were positioned in the Puma's work space, and an operation on or around the objects was programmed using the teach pendant. Points during each motion were recorded, and the joint angles at these positions, along with a kinematic model of the Puma, were used to compute the swept volumes of the Puma's motion. These swept volumes, along with the models of the object, sensors, and task constraints, were used as input to the viewpoint-computation algorithm. When necessary, temporal interval decomposition was performed to yield shorter time intervals, and the process was repeated to compute viewpoints for each time interval. The sensor parameters and time intervals produced were finally used to program the gantry and create a coordinated motion plan.

5.1. Experiment 1

Figure 15 shows the scene as it is set for our first example. Here, we see a model of the Puma poised over a fixtured mechanical part. The end effector on the Puma is a long, slender tool tip. The Puma makes a pass over the object so that the tool follows the contour of the groves in the front of the part, simulating a gluing or welding application. The two white strips, corresponding to the two straight segments of the front-most grooves on the part, are the features to be viewed during this task, yielding eight linear features (the edges bounding these two rectangular strips). The minimum resolvable feature length is set at 1 mm, and an 8.5-mm lens is used for this task. The goal is to compute a viewpoint from which the target features can be monitored while the Puma is moving through its task.

We begin by computing the volume swept by the Puma as it performs this task. This volume can be seen in Figure 16. (Notice, for example, that the end effector is no longer a pointed tool, but the volume swept by that tool.) Then, we compute the visibility volume for the rectangular features.

The volumes $V_{i_{FR}}$ are computed for each feature. These regions are intersected for all features, forming the candidate set V_{FR}. Then, V_{FR} is intersected with the visibility volume, forming the feasibility volume V_C, shown in Figure 16.

In this example, a search algorithm employing a single start point was used. This search yielded the viewpoint $(-310.71, 805.19, -235.87)$. For this experiment, the orientation is computed so as to maximize just the field-of-view computation, yielding an orientation of $v = (0.1264, -0.1584, -0.9792)$. The optical parameters are then computed using eqs. (11) and (12), yielding $d = 8.94$ mm and $a_{\max} = f/1.16$. The field-of-view and resolution constraints are recomputed, and found to be satisfied with these parameters.

To verify this viewpoint, the camera is moved into the computed position, and an image is taken. The camera is manually set to the computed focus value, and its aperture is set to be smaller than the maximum prescribed aperture (significantly smaller, actually, to approximately $f/5.6$). The resulting image is shown in Figure 17.

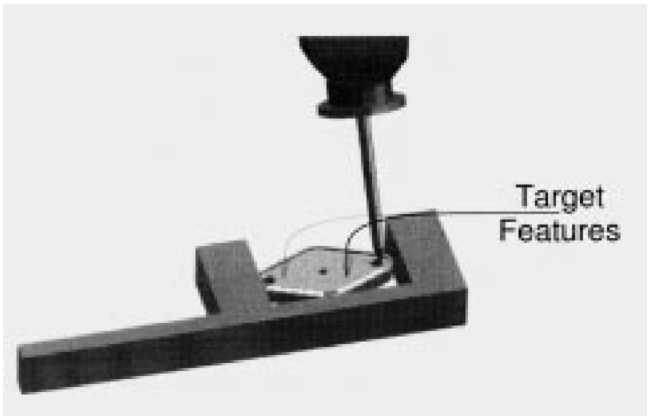

Fig. 15. The robot and the object to be imaged.

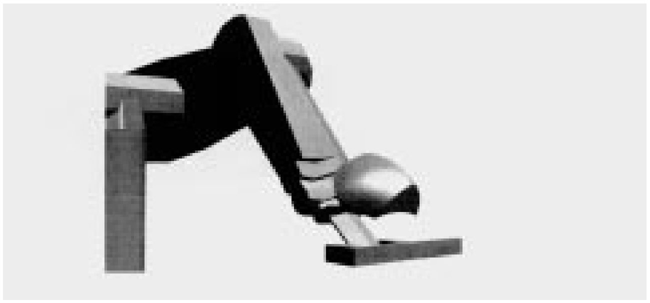
Fig. 16. The merged candidate volume V_c, embodying all task constraints, along with the swept robot model and the fixtured object.

5.2. Experiment 2

In this task, the Puma is working in and around an electronics assembly. A circuit board is fixtured in front of the Puma, and the Puma moves through the fixture. The sensor-planning system is asked to maintain a robust view of the top surfaces of several chip sockets during this motion. Figure 18 shows the circuit board held in a fixture. Specifically, a rectangular box encompassing the two rows of three sockets along the back of the circuit board is used as the target feature.

Figure 19 shows the Puma positioned within the fixture. The Puma, with a parallel-jaw gripper for an end effector instead of the stylus used in the previous experiment, moves through an L-shaped trajectory, forward along one side of the fixture and across the front of the circuit board. The volume swept by the Puma moving through this motion is shown in Figure 20. The sensor-planning system is asked to compute a viewpoint that monitors the sockets using a 12.5-mm lens, with a minimum resolvable feature length of 1 mm, while the robot undergoes this motion.

This motion is quite restrictive in that it does not permit the computation of any useful viewpoints to cover the entire task interval. In fact, other than a couple of small inaccessible

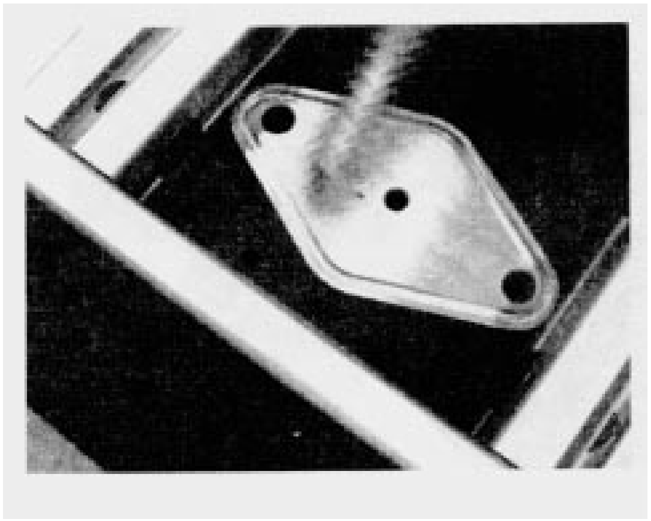
Fig. 17. Image taken from the computed viewpoint while the probe is in motion.

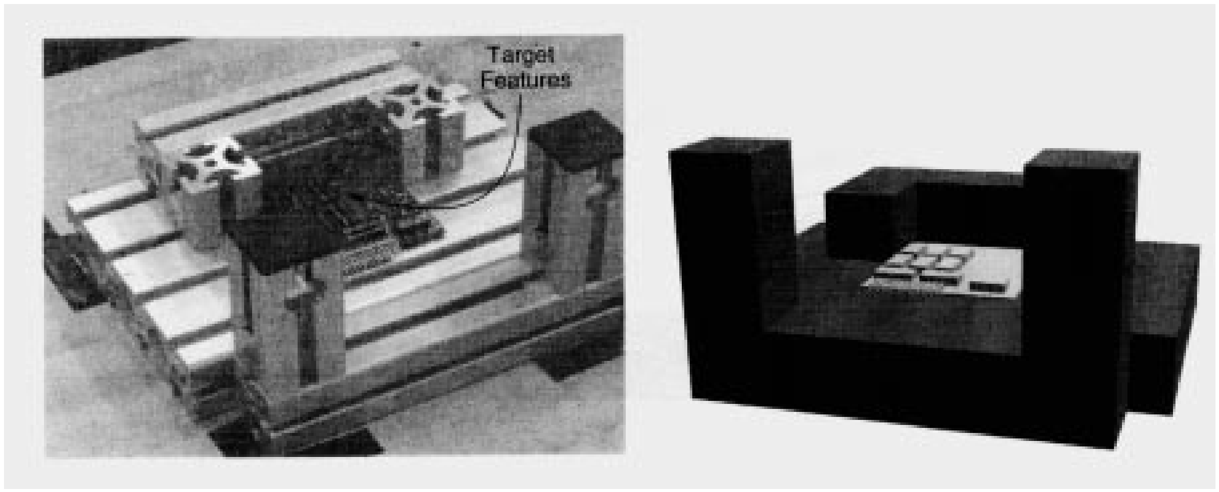

Fig. 18. The fixtured circuit board, along with its model, used for Experiment 2.

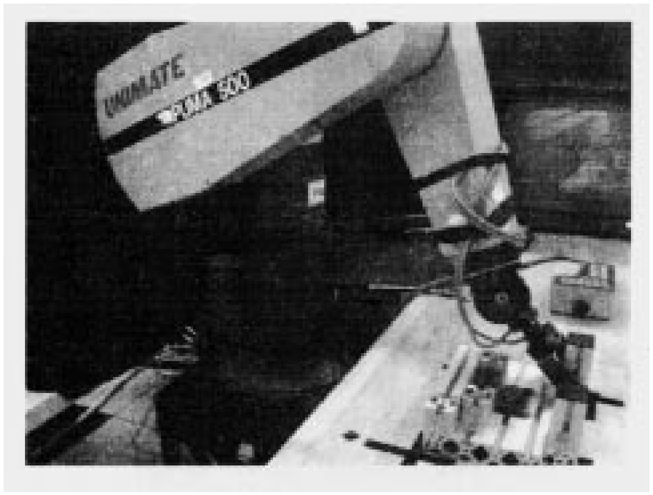

Fig. 19. The Puma is poised in position for the motion.

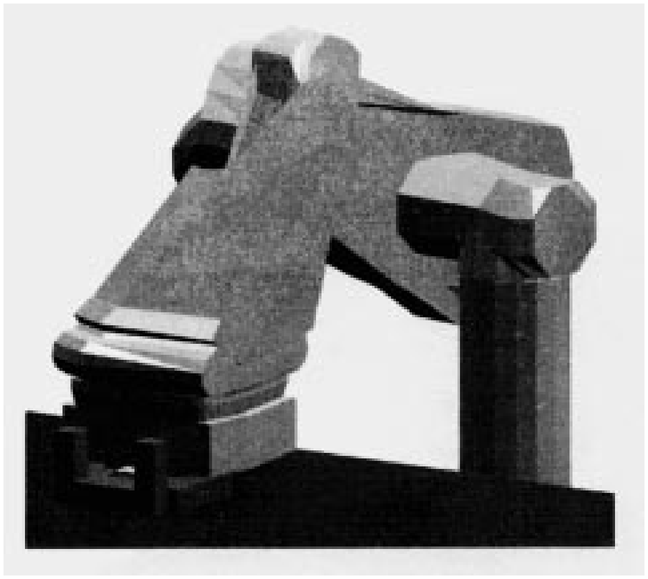

Fig. 20. The computed volume swept by the Puma as it moves through the entire trajectory. It moves forward along the side of the circuit board, and then across the front of the circuit board, behind the pillars.

"slivers" of regions right up against the fixture, the V_c set is empty. Therefore, we were forced to use a temporal decomposition. Recall that in cases where the entire time interval cannot be monitored by one viewpoint, we divide the task into two equal halves and attempt to compute a viewpoint for each half. This task gets divided into the "forward" motion and the "across" motion; the corresponding swept volumes are shown in Figures 21 and 24.

The V_c volume computed for the first time interval is shown in Figure 22. Volume V_c has one connected component, and two starting points were computed for the positional optimization: one central to the inner spherical surface, and one central to the torroidal surface. Of these, one converges in a local minima in a very tight area of the volume, close to one of the posts on the object—so close, in fact, that positioning the camera there would cause a collision between the gantry camera body and the post. A goal of future research is to

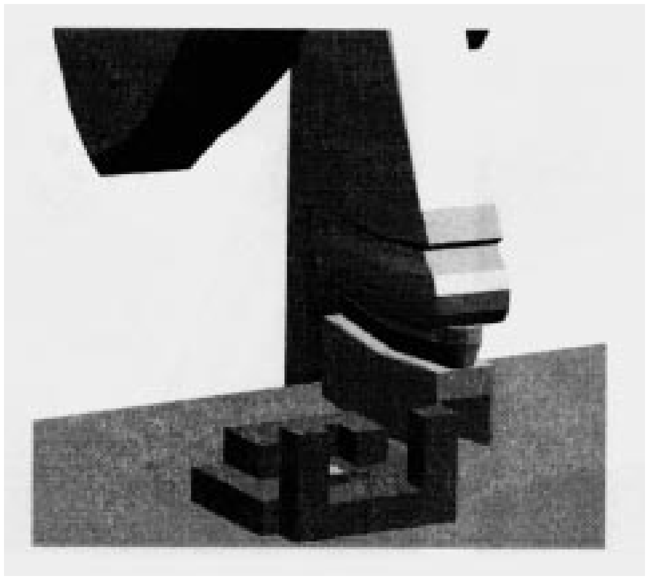

Fig. 21. The computed volume swept by the Puma as it moves through the first half of the trajectory.

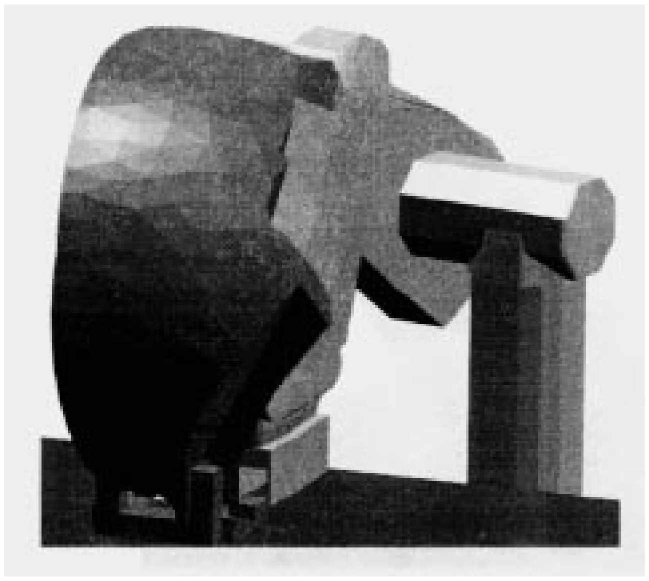

Fig. 22. The merged candidate volume V_c, embodying all task constraints during the first task interval.

include these positioning constraints into the system as well. The other answer is shown graphically in Figure 23.

Having successfully handled the first time interval, we proceed to the second. The volume swept by the Puma during this interval is shown in Figure 24; the candidate set V_c for this experiment is shown in Figure 25. This set actually has four connected components. There is the large one clearly visible, and three much smaller ones—precisely, those "sliver" components present during the whole task interval. Again, these are in extremely close proximity to the fixture, and nearly insignificant in volume. The viewpoint-computation algorithm then produces six viewpoints— two of the searches could not converge, due to the tiny size of the connected components—of which four are too close to the fixture for collision-free positioning, and one is under the Puma (inaccessible to our gantry). Therefore, we have one result, shown graphically in Figure 26.

Having now completed the planning for the entire task, the Puma is moved through its task, pausing at each temporal break for the gantry to reposition itself to the next computed viewpoint. The images from these viewpoints while the robot task is in motion are shown in Figures 23 and 26.

Several features of this experiment are worth noting. First, notice the importance of using multiple starting points for the search. Due to the number of local minima in the volumes and additional unaccounted-for constraints (such as the accessibility of a point by the gantry), it is desirable to have a number of solutions from which to choose. Also, notice that during the first interval, the camera is oriented nearly perpendicular to the plane of the target features (the normal is (0,0,1), and the viewing direction is (0.0697, −0.0265, −0.9972)). Therefore, the features all project to nearly the same distance along the optical axis, and so the depth of field required is almost zero. This explains the absurdly high value for the maximal aperture of 150.669 mm, or approximately f/0.083. Also, the relative orientation of the optical axis and each of the feature edges is nearly identical, yielding nearly identical minimum-resolvable feature lengths.

Compare this with time-interval 2, where the camera has a more off-axis view of the plane: there is a real constraint on the aperture of around f/1.17 (still not very restrictive), and the minimum resolvable feature lengths are quite different for each of the edges, although still satisfying the constraints.

6. Conclusions

This paper has discussed the problem of dynamic sensor planning for vision sensors in an active robot work cell. It has described the problem, detailed the constraints, and outlined the various forms of the problem. Further, it has presented an approach for solving a specific variant of this problem, called the surveillance planning problem, and presented experimental results that demonstrate the usefulness of the approach. With only one exception (Niepold, Sakane, and Shirai 1987), none of the sensor-planning systems we have found in the literature discuss sensor planning when objects are moving.

The viewpoint-planning algorithm presented in this paper has some benefits over previous methods. Some of the earlier work relies on simplifications of the constraints or assumptions about certain imaging parameters. This eliminates possible solutions from consideration. Other systems use expensive nonlinear constrained optimizations, which are very sensitive to both the initial guess used and the weights assigned to the individual constraints. Still others rely on dis-

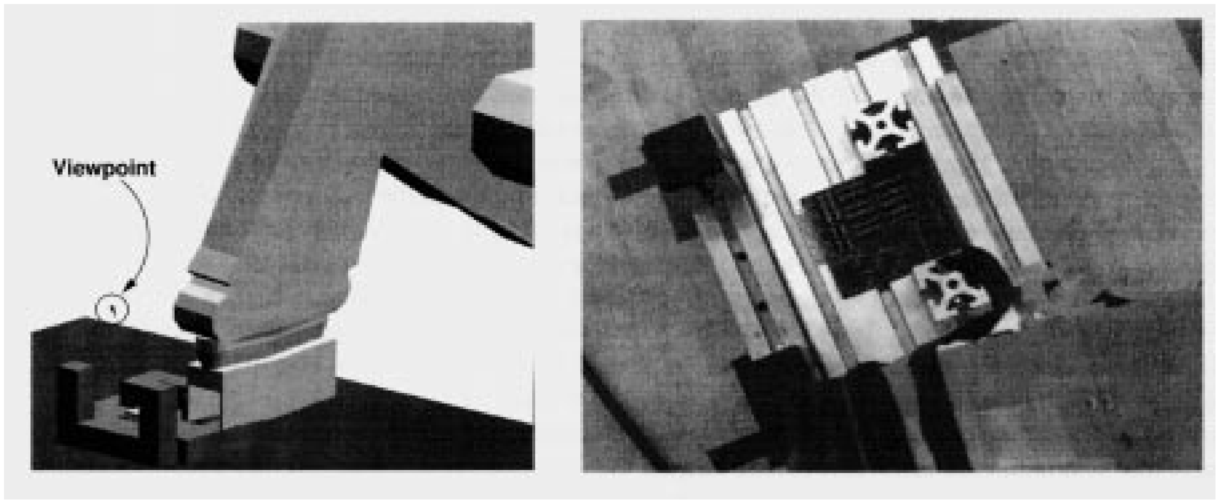

Fig. 23. The small cone illustrates the computed camera position and orientation (left). The image taken from that position is shown (right).

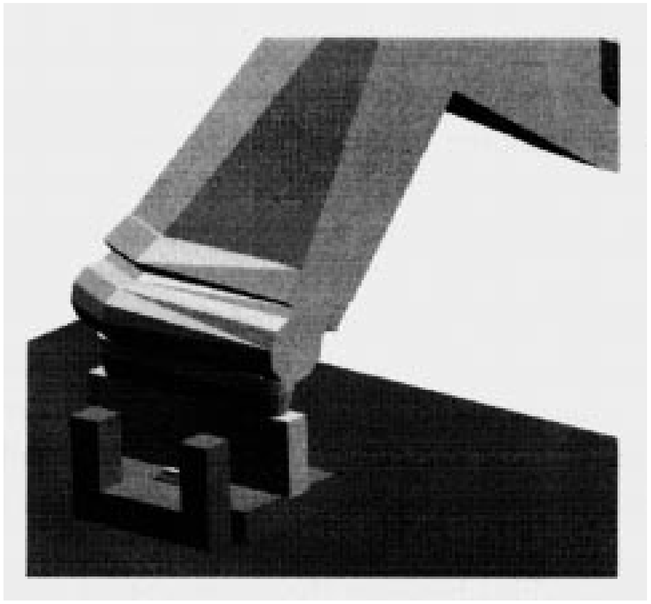

Fig. 24. The computed volume swept by the Puma as it moves through the second half of the trajectory.

cretized approaches and tessellated search spaces, which can be very costly when finer discretizations are used.

By carefully examining the relationships between the constraints, a new way of integrating the field-of-view and resolution constraints was found. It is this integration that permits the computation of feasibility regions in three dimensions. These feasibility regions differ from those used, for example, by Cowan and Kovesi (1988), in that all feasible points are included in the regions, without an implicit assumption of the camera's orientation.

This three-dimensional region can be searched independently of the other camera parameters; subsequently, we can solve for these other parameters. Therefore, our method increases the size of the searchable region over previous methods, without resorting to a full-scale optimization of all parameters. Once a position and orientation have been found, the computation of d_{optimal} and the upper bound for a, a_{max} allow for a synergy between the sensor and illumination planners.

The primary limitation in the viewpoint-computation algorithm is that it may be unable to find a viewpoint that satisfies the resolution constraint for all features; that is, the optimization of the orientation is not guaranteed to converge, even if there is a solution. Since none of the examples that were run required an optimization of the orientation at all, it is difficult to know exactly under what circumstances the optimization will fail, or to know what steps should be taken by the viewpoint-computation algorithm in this eventuality.

The temporal interval decomposition is, clearly, the coarsest aspect of the method, and, therefore, the area most open to future work. Before attacking this problem in more depth, one must answer the question, "What makes an interval good?" This is a difficult question, which we leave for future researchers to answer.

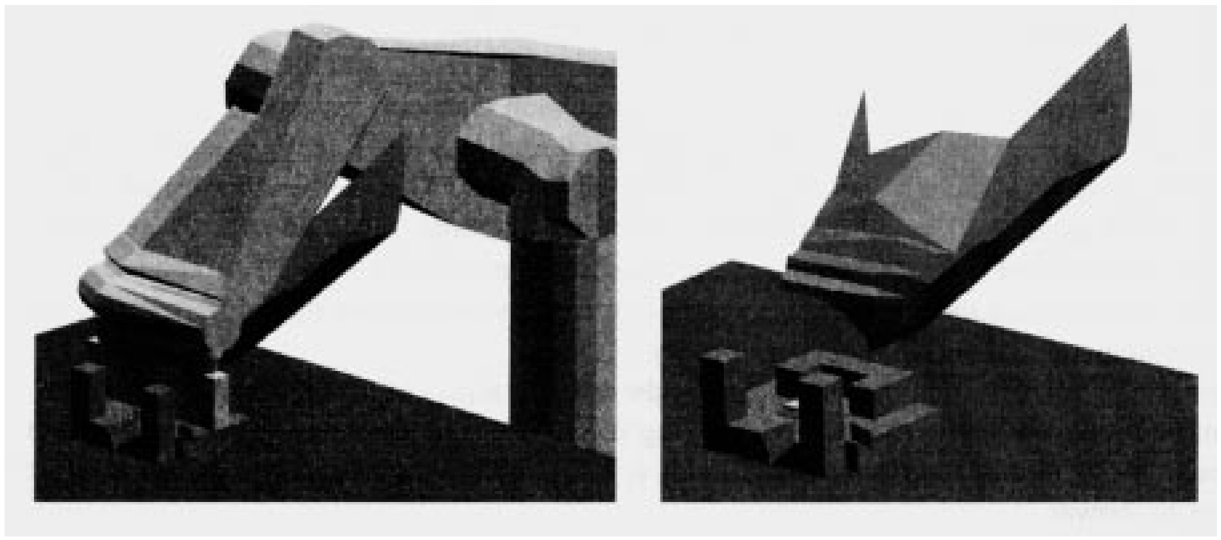
Fig. 25. The merged candidate volume V_c for the second task interval, with and without the Puma.

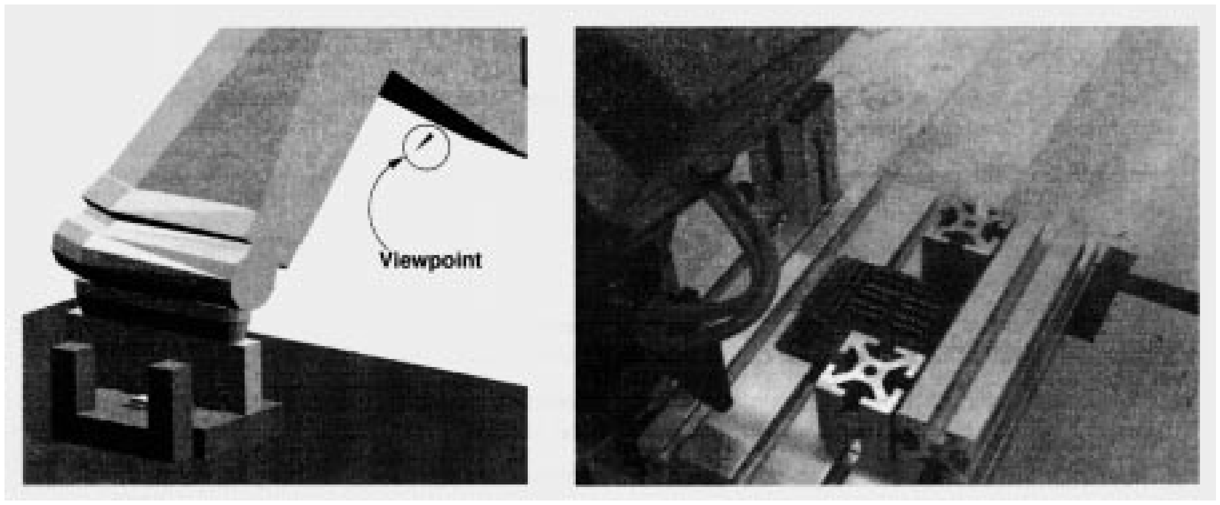

Fig. 26. View showing one of the computed viewpoints and orientations as a cone (left); and the image taken from this viewpoint (right).

As the examples showed, there are constraints that come into play in the real-world placement of the sensor which are currently not modeled. Constraints such as collision, accessibility, and range of operation of the sensor (i.e., limitations in the pan and/or tilt angles available) need to be handled for the system to become truly automated.

Acknowledgments

Peter Allen's work has been supported in part by DARPA contract DACA-76-92-C-0007, an ONR MURI Grant, NSF grants CDA-90-24735 and IRI-93-11877, and Toshiba Corporation.

References

Abrams, Steven, 1997. Sensor planning in an active robot work cell. Ph.D. thesis, Columbia University, New York.

Abrams, Steven, and Allen, Peter K. 1997. Computing swept volumes. Technical Report, Department of Computer Science, Columbia University, New York.

Cowan, C. K. 1988. Model-based synthesis of sensor locations. *Proc. of the 1988 IEEE Intl. Conf. on Robot. and Automat.* Los Alamitos, CA: IEEE.

Cowan, C. K., and Bergman, A. 1989. Determining the camera and light source location for a visual task. *Proc. of the 1989 IEEE Intl. Conf. on Robot. and Automat.* Los Alamitos, CA: IEEE.

Cowan, C. K., and Kovesi, P. D. 1988. Automatic sensor placement from vision task requirements. *IEEE Trans. Pattern Analysis Machine Intell.* 10(3):407–416.

Cowan, C. K., and Modayur, B. 1993. Edge-based placement of camera and light source for object recognition and location. *Proc. of the 1993 IEEE Intl. Conf. on Robot. and Automat.* Los Alamitos, CA: IEEE.

Hui, K. C. 1994. Solid sweeping in image-space application in NC simulation. *Visual Comp.* l(6):306–316.

Kaul, Anil. 1993. Computing Minkowski sums. Ph.D. thesis, Department of Mechanical Engineering, Columbia University, New York.

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

Krotkov, E. P. 1989. *Active Computer Vision by Cooperative Focus and Stereo.* Springer-Verlag.

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

Niepold, R., Sakane, S., and Shirai, Y. 1987 (Hiroshima, July). Vision sensor setup planning for a hand-eye system using environmental models. *Proc. of the Soc. Instrum. Control Eng. Japan*.

Sakane, S., Ishii, M., and Kakikura, M. 1987. Occlusion avoidance of visual sensors based on a hand-eye action simulator system: HEAVEN. *Adv. Robot.* 2(2):149–l65.

Sakane, S., Niepold, R., Sato, T., and Shirai, Y. 1992. Illuminatiion setup planning for a hand-eye system based on an environmental model. *Adv. Robot.* 6(4):461–482.

Sakane, S., Sato, T., and Kakikura, M. 1987 (Versailles, October). Model-based planning of visual sensors using a hand-eye action simulator system: HEAVEN. *Proc. of the 3rd Intl. Conf. on Adv. Robot.*, pp. 163–174.

Schroeder, William J., Lorensen, William E., and Linthicum, Steve. 1994 (Washington, DC, October). Implicit modeling of swept surfaces and volumes. *Proc. of the IEEE Visualization Conf.* Washington, DC: IEEE.

Sourin, A., and Pasko, A. 1995 (Salt Lake City, UT, May 17-19). Function representation for sweeping by a moving solid. *Proc. of the Third Symp. on Solid Modeling and Applications*. New York: ACM Press, pp. 383–391.

Spatial Technologies, Inc. 1995. *ACIS Programmer's Guide, 1.7 ed.*. Spatial Technologies, Inc. 2425 55th St., Boulder, CO.

Tanabanis, Konstantinos. 1991. Sensor planning and modeling for machine vision tasks. Ph.D. thesis, Department of Computer Science, Columbia University, New York.

Tanabanis, Konstantinos, Allen, Peter K., and Tsai, Roger Y. 1995. A survey of sensor planning in computer vision. *IEEE Trans. Robot. Automat.* 11(1).

Tanabanis, Konstantinos, Tsai, Roger Y., and Abrams, Steven. 1991. Planning viewpoints that simultaneously satisfy several feature-detectability constraints for robotic vision. *Proc. of the Fifth Intl. Conf. on Adv. Robot.*

Tanabanis, K., Tsai, R. Y., and Allen, P. K. 1994a. Analytical characterization of the feature-detectability constraints of resolution, focus, and field of view for vision-sensor planning. *Comp. Vision Graphics Image Proc.* 59(3).

Tanabanis, Konstantinos, Tsai, Roger Y., and Allen, Peter K. 1994b. Analytical characterization of the feature-detectability constraints of resolution, focus, and field of view for vision-sensor planning. *Comp. Vision Graphics Image Proc.* 59(3).

Tanabanis, Konstantinos, Tsai, Roger Y., and Allen, Peter K. 1995. The MVP sensor planning system for robotic vision tasks. *IEEE Trans. Robot. Automat.* 11(1).

Tanabanis, Konstantinos, Tsai, Roger Y., and Kaul, A. 1996. Computing occlusion-free viewpoints. *IEEE Trans. Pattern Analysis Machine Intell.* 18(3).

Tarbox, G. H., and Gottschlich, S. N. 1995. IVIS: An integrated volumetric inspection system. *Comp. Vision Image Understanding* 61(3):430–144.

Tsai, Roger Y. 1989. A new technique for fully autonomous and efficient 3-D robotics hand-eye calibration. *IEEE Trans. Robot. Automat.* 5(3):345–358.

Wang, W. P., and Wang, K. K.. 1986. Geometric modeling for swept volume of moving solids. *IEEE Comp. Graphics Applications* 6(12):817.

Weld, John D., and Leu, Ming C. 1990. Geometric representation of swept volumes with application to polyhedral objects. *Intl. J. Robot. Res.* 9(5):105–117.

Willson, R. G., and Shafer, S. A. 1993 (New York, June 15-17). What is the center of the image? *Proc. of the 4th IEEE Conf. on Comp. Vision and Pattern Recognition.* Los Alamitos, CA: IEEE.