Proceedings of the 1996 IEEE
International Conference on Robotics and Automation
Minneapolis, Minnesota - April 1996

Computing Camera Viewpoints in a Robot Work-Cell

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

Konstantinos A. Tarabanis
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598

Abstract

Automatically planning a camera viewpoint for tasks such as inspection in an active robot work-cell is a difficult problem. This paper discusses new methods for computing viewpoints which meet the feature detectability *constraints of focus, field-of-view, visibility, and resolution. A theoretical outline of the method is presented, followed by experimental results and a discussion of future work.*

1 Introduction

In an effort to increase the efficiency and autonomy of the manufacturing process, several researchers have focused on developing automatic strategies for inspecting manufactured parts using a variety of sensors. Most often, the sensors used are vision sensors, and the systems developed compute sets of positions, orientations, and optical settings for a camera (and, in some cases, for light sources) which will give satisfactory views of certain objects in a known scene.

The constraints most often considered are resolution (or magnification), focus, field-of-view, and occlusion. Examples of work in this field are in [5, 6, 7, 8, 10, 11, 12]. Other systems, such as [13], use a combination of vision and touch sensors to perform the inspection. A more complete survey of sensor planning systems can be found in [14].

Our earlier research in this field resulted in the development of the Machine Vision Planning (MVP) system [15, 16, 17, 18]. Briefly, MVP takes an optimization approach towards viewpoint computation. The basic idea is to obtain a generalized viewpoint, i.e. a point in the parameter-space of the optical system, which is as far away from each of the constraints as possible. Such a viewpoint is then robust in terms of placement, model, or calibration errors, since it meets all of the constraints as comfortably as possible.

*This work was supported in part by DARPA contract DACA-76-92-C-0007, NSF grants CDA-90-24735 and IRI-93-11877, and an ONR MURI Grant

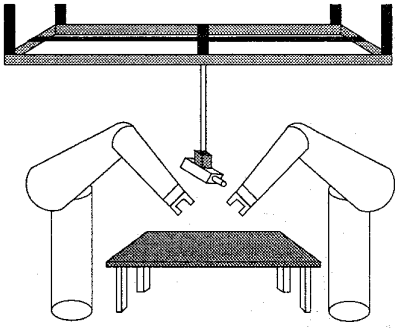

Figure 1: *Viewpoint Placement in a Robot Work-Cell*

One feature that all of the above mentioned systems have in common is that they are restricted to functioning in a static environment. It is interesting to note that while there has been an abundance of research in planning the *motion* of robots, the trajectories of cutting tools, and other activities in an intelligent manufacturing work-cell, research in planning sensor strategies has focused on environments in which there is no motion. Therefore, we have been extending MVP to function in an environment in which objects are moving [1, 2, 3].

As an example, we may have a work-cell in which one or more robots are assembling an object. We may wish to automatically monitor this assembly task. Figure 1 shows the basic setup for such a system: two robot arms, able to operate in a work-cell, and a gantry robot, used for moving the camera through a computed trajectory. We call the problem of computing viewpoints in this environment *Dynamic Sensor Planning*.

Our approach to Dynamic Sensor Planning has been to break the task down into intervals, each of which is to be monitored by a single viewpoint. To solve the problem of occlusion over time, the system computes the volumes swept by all moving objects during each interval and computes viewpoints which avoid occlusion by these swept volumes, while meeting all of the optical constraints. Such

0-7803-2988-4/96 $4.00 © 1996 IEEE

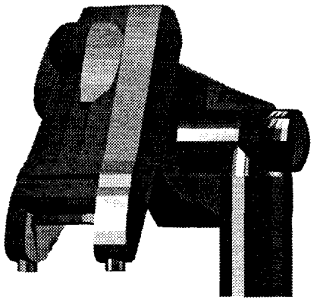

Figure 2: *Rendering of swept puma model.*

viewpoints are valid for the entire time interval. By similarly examining a number of time intervals, we break the Dynamic Sensor Planning problem down into a series of static subproblems.

Our recent results in computing swept volumes [2] has made this interval-based approach practical. As an example, we can now compute the volume swept by a Puma robot moving through a prescribed trajectory, such as the volume shown in figure 2.

We have found that dynamic sensor planning makes much greater demands on the underlying viewpoint computation engine than static sensor planning did. Because of this, we are investigating new methods for computing viewpoints. The focus of this paper, then, is on our latest results in viewpoint computation and, as such, is applicable to both the static and dynamic sensor planning problems.

2 Revisiting Static Sensor Planning

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, non-differentiable, convergence can not 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.

We found that the generate-and-test approaches taken by some (such as [11, 12]) are more straightforward (involving the search of a tessellated sphere) and can easily be shown to converge. However, they tend to ignore or take implicit certain imaging parameters (i.e. the distance from the viewpoint to the features is set to meet a magnification requirement or the viewing orientation is assumed to be towards the center of the feature points). The computational cost of finer tessellations can lead to very costly searches, however. The synthesis approach taken by Cowan et al. [5, 6] does not suffer from the computational expense of the generate-and-test methods. Their method of computing admissible domains in $\mathcal{R}^3$ for each constraint is also straightforward, but they also consider only a subset of the parameters.

We have been developing a viewpoint computation algorithm which 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 provable convergence).

3 Feature Detectability Constraints

Before proceeding to the development of our viewpoint computation algorithm, we will briefly review the feature detectability constraints used by our system. (A more detailed discussion with derivations can be found in [16].) In the equations which 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.

3.1 Resolution

Pixel resolution can be used to determine the minimum feature size resolvable by the system. That is, to ensure that every feature of size l is resolvable, one must ensure that it images to at least two pixels on the image plane. This locus is given by:

$$\frac{|[(r_a - r_v) \times u] \times v|}{((r_a - r_v) \cdot v)((r_b - r_v) \cdot v)} - \frac{w}{dl} \geq 0 \qquad (1)$$

where r_a and r_b are the end points of the linear feature to be viewed, u is the unit vector along the linear feature (from r_a to r_b), l is the length of the linear feature, and w is the minimum length of the feature in the image.

3.2 Focus Constraint

An imaging system is perfectly focused at a specific distance D (measured along the optical axis) given by the equation

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

In practice, however, if a point images to a blur circle of a given size c, it is considered sufficiently in focus for a given application. The system is then 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 (see also [9]).

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

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

Since these distances are measured along the optical axis, the corresponding constraints are given by:

$$(r_c - r_v)\cdot v - D_2 \geq 0 \quad (5)$$

$$D_1 - (r_f - r_v)\cdot v \geq 0 \quad (6)$$

where r_c is the feature point closest to r_v and r_f is the feature point farthest from r_v.

3.3 Field-of-View

Most CCD cameras have a field-of-view limited by the size of the sensor area and the focal length of the lens. Any feature which, due to the optics, projects all or partially outside of the sensor area is not within the sensors field of view. The locus which satisfies the field of view constraint for a set of features enclosed by a circumscribing sphere of radius R_f and center r_s, is given by the following equation:

$$(r_k - r_v)\cdot v - |r_k - r_v|\cos(\alpha/2) \geq 0 \quad (7)$$

Here, α is the field-of-view angle of the sensor. Since sensor planning systems generally consider the image plane to be symmetrical about the optical axis for the purposes of field of view, α is computed based on the length of the smaller side of the sensor area, I_{min}. Therefore, $\alpha = 2\tan^{-1}(I_{min}/2\,d)$. Finally, $r_k = r_s - R_f/\sin(\alpha/2)\,v$.

3.4 Visibility

In order for a feature to be detectable by a vision sensor, it may not be occluded by other objects. Using algorithms explained in detail in [19], 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.

4 Viewpoint Computation

The complexity of the constraints and the difficulties of solving for all parameters together led us to the decomposition-based approach described here. To compute the viewpoint and viewing parameters, first we will be computing a volume $V_c \subset \mathcal{R}^3$ which constrains the *position* of the camera. This volume will be the projection into

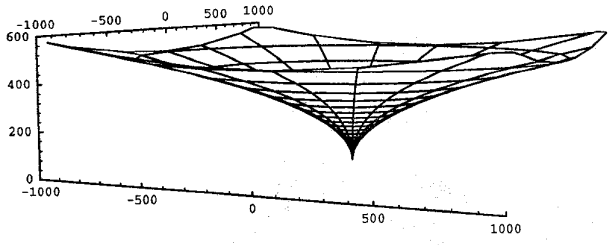

Figure 3: *Resolution surface,* $v = (0, 0, -1)$

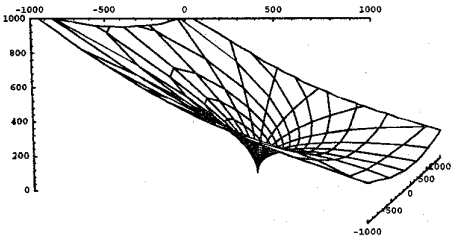

Figure 4: *Resolution surface,* $v = (-\sin(\frac{\pi}{8}), 0, -\cos(\frac{\pi}{8}))$

$\mathcal{R}^3$ of the entire solution set. Once the candidate set is computed, a specific position within V_c will be computed, followed by a camera orientation. Finally, an optimal value for the focus and a maximum allowable value for the aperture are computed. We will be using lens with a fixed focal-length f, whose aperture a and focus distance d have yet to be computed.

4.1 Computing a Candidate Set in $\mathcal{R}^3$

Before discussing what V_c should look like, we will examine the merging of the field-of-view and resolution constraints. For any given viewing direction v and for each feature i, there is field-of-view cone C_{i_v} (given by equation 7) which constrains the position of the camera. 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.[1].

Further, for any given viewing direction and for each feature i (a linear feature of length l), there is a surface S_{i_v} which constrains the position of the camera due to the resolution requirement. Figure 3, 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 vertical direction ($v_0 = (0, 0, -1)$, such that $l = 1$ mm, and $w = 1$ pixel. The corresponding picture for a viewing

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

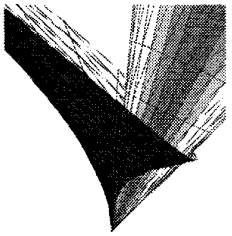

Figure 5: *Resolution & Field-of-View constraints, $v = v_1$.*

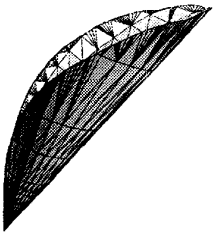

Figure 6: *Intersection of constraints, $v = v_1$.*

direction of $v_1 = (-\sin(\frac{\pi}{8}), 0, -\cos(\frac{\pi}{8}))$, shown in figure 4, has a very different shape. Again, d appears in the equation for this surface but can be approximated using f.

Figure 5 shows the field-of-view cone superimposed on the resolution surface for viewing direction v_1. For any viewing orientation, the set of valid camera positions includes only those which are both *inside* the cone and *under* the resolution surface.

One property of these constraints is that turning the cameras orientation *away* from a feature (while holding the cameras position constant) increases the resolution, albeit at the expense of field-of-view. To illustrate this, imagine the following extreme case. The camera is positioned somewhere in space, and there is a linear feature to be viewed. (For simplicity, let us assume that the feature endpoints and the camera's entrance pupil are not colinear.) Draw an imaginary line from one of the feature endpoints point to the camera's entrance pupil. Rotate the camera about its entrance pupil away from the feature until the optical axis is perpendicular to this line. Now, the feature will image to an infinite size, but be completely outside of the camera's field of view[2]. Therefore, if the camera is too far away to image the features with proper resolution, the camera can be turned 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.

These observations led us to compute a region in $\mathcal{R}^3$ in which there are known to be orientations that satisfy both the field-of-view and resolution constraints. This volume will be called V_{FR}. The complete candidate volume, V_c, is a subset of V_{FR} because we have not yet considered other constraints, which will further reduce the set.

By way of illustration, we computed the intersection of the regions bounded by S_{i_v} and C_{i_v} (for one viewing orientation) 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. This volume is shown in figure 6 for $v = v_1$. A corresponding volume exists for every possible viewing orientation. The V_{FR} volume for this feature i (called $V_{i_{FR}}$) must include the union of these volumes R_{i_v} for all viewing orientations.

[2]Unless, of course the camera has greater than 180 degrees of field-of-view.

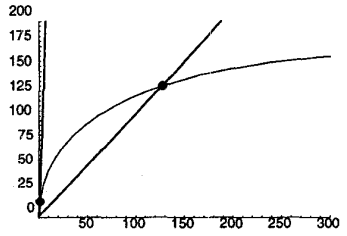

Figure 7: *Field-of-View cone and Resolution curve, in 2D, shown with intersection points.*

To better illustrate this, see figure 7. In this figure (for simplicity), we are limiting ourselves to a 2 dimensional viewing configuration. As before, the feature being viewed is a vertical linear feature at the origin. This figure can then be produced by intersecting the 3D constraints of figure 6 with the plane containing the lens and the feature end points (in this case, the X-Z 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.

It can be shown [4] that as we rotate the viewing angle in the plane, the intersection points are constrained to lie 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 meets the resolution criteria, the camera would always end up on a point on this circle, regardless of the chosen orientation. That these intersection points all lie on a circle can more clearly be seen in figure 8 which shows eight pairs of field-of-view and resolution constraints, with the highlighted intersection points.

It can further be shown that 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), this circle

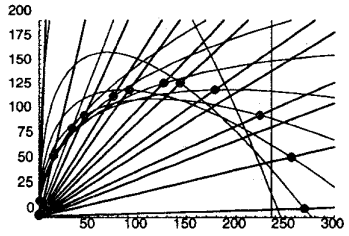

Figure 8: *Field-of-View cones and Resolution curves, in 2D, shown with intersection points.*

Figure 9: *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.*

would form part of the boundary of this union (and therefore the $V_{i_{FR}}$ region).

Interestingly, figure 8 also illustrates a smaller circle, formed by the envelope of the resolution curves (as described by Tarabanis in [16]). 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 satisfy both field-of-view and resolution simultaneously. The inner circle is the region used by Cowan [5] 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.

Thus far we have described the constraint on how far from the feature the camera can be. The constraint on how close the camera can be is simply the circle on which the apex of the field-of-view cone is constrained to lie.

Moving into 3-D, it can be shown that taking 2-D region $V_{i_{FR}}$ and sweeping it about the line defined by the edge feature (which passes through the larger circle) yields the volume $V_{i_{FR}}$. This volume has the shape of a torus that has lost its hole (because its major radius is smaller than its minor radius) and has had a sphere subtracted from its center. The major radius of this torus is given by:

$$r_{min} = \frac{l}{8\sqrt{\frac{d^2\,w^2}{(A)\left(A-4\,I_{min}\,w+4\,w^2\right)}}} \quad (8)$$

The minor radius of the torus is given by:

$$r_{maj} = r_{min}\cos(\lambda) \quad (9)$$

where

$$A = 4\,d^2 + I_{min}^2 \quad (10)$$

$$\lambda = \arccos\left(\frac{A - 2\,I_{min}\,w}{\sqrt{A}\sqrt{A - 4\,I_{min}\,w + 4\,w^2}}\right) \quad (11)$$

The radius of the sphere is $R_f / \sin(\alpha/2)$, from the field-of-view constraint above. A complete derivation of this volume is beyond the scope of this paper, but can be found in [4].

Figure 9 shows a cutaway view of this volume for a vertical linear feature, at the center of the volume. The discussion thus far has been restricted to handling linear features of length l. To handle linear features longer than l, it is sufficient to consider a segment of length l at each of the endpoints of the actual feature and intersect their corresponding candidate volumes.

In order to find the total feasible volume V_c, the volumes $V_{i_{FR}}$ 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 tori and spheres, allowing these intersections to be performed analytically. V_{FR} is then intersected with the visibility volume for the feature set, yielding a single set of feasible points in $\mathcal{R}^3$, namely, V_c. The boundary of V_c, therefore, consists of spherical, toroidal, and polygonal components.

We can position the camera at any point in V_c and compute an orientation for which each feature will be unoccluded, within the sensors field-of-view, and properly resolvable.

4.2 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 within MVP for computing a single viewpoint was to find one as far away from the constraining surfaces as possible. In this new formulation, when we are searching a region in $\mathcal{R}^3$, 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. Similarly, a gradient-based search can be used to find a viewing orientation which satisfies all constraints for all features.

4.3 Computing the Optical Parameters

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

(the focus setting) does not make much sense without computing at least an upper bound on the aperture, since for any value of d, a can be decreased (in the limit, reducing the lens to pinhole) to bring the features into focus. In our system, we compute both d and the upper bound for a, called a_{max}. a_{max} can then 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}.

Recall that focus is essentially concerned with ensuring that the farthest and nearest feature points are within the depth-of-field limits of the optical system. By simultaneously solving the equalities

$$D_1 - D_2 = (r_f - r_c) \cdot v \quad (12)$$

$$D_1 = r_f \quad (13)$$

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

$$d = \frac{2\,D_{max}\,f(D_{max} - D_f)}{2D_{max}(D_{max} - f - D_f) + f\,D_f} \quad (14)$$

$$a_{max} = \frac{2\,c\,D_{max}(D_{max} - f - D_f) + f\,D_f}{f D_f} \quad (15)$$

where:

$$D_{max} = (r_f - r_v) \cdot v$$

$$D_f = (r_f - r_c) \cdot v$$

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

Once the imaging parameters are computed, the resolution and field-of-view surfaces are recomputed using the actual d to insure 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.

5 The Complete Planning Procedure

Given that we now have the basic viewpoint computation algorithm, we can go on to describe the overall dynamic sensor planning procedure.

Dynamic Sensor Planning Procedure

1. Compute the volumes swept by all moving objects during the time interval to be monitored.
2. Compute the visibility volumes $V_{i_{vis}}$ for all features to be inspected.
3. Compute the volumes $V_{i_{FR}}$ (the positional constraints for field-of-view and resolution combined) for every feature i.
4. Compute the overall candidate volume V_c as the intersection of all $V_{i_{FR}}$ and $V_{i_{vis}}$ volumes with each other.
5. Optimize to find a position at least locally optimal within V_c.
6. Optimize to find an orientation at least locally optimal for this position.
7. Compute the optimal focus and maximum aperture for this position and orientation using equations 14 and 15.
8. Verify that the position, orientation, and optical settings are all valid using the actual value of d computed, rather than the approximation of $d = f$.

Several aspects of this algorithm require additional comment. To begin with, it is possible that the volume V_c will be empty. In this case, obviously, there is no viewpoint which will satisfy all constraints for all features. The feature set needs to be divided into subsets, each of which will be viewed by a different viewpoint. The *automatic* subdivision of the feature set will be the subject of future research.

Next, are the optimization steps. The optimization of step five solves only for position and can use a standard gradient-search technique from an easily computable starting point. Any point on any of the boundary surfaces, can be used as a starting point and the search will converge to a position 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.

The optimization of step six solves only for orientation and has an excellent starting point available to it, namely the optimal orientation for the field-of-view constraint: oriented towards the center of the sphere encompassing the points. This optimization may not converge, however, as there may not be any orientation which satisfies all constraints for all features. A complete search algorithm is being developed which is guaranteed to find a solution if one exists.

Finally, the verification of the viewpoint using the actual parameters is an interesting question. Recall that d was approximated by f in the computation of the $V_{i_{FR}}$ volumes. The actual computed value of d is somewhat greater than f, depending upon how close the camera is to the features. This means that the camera's field-of-view is somewhat

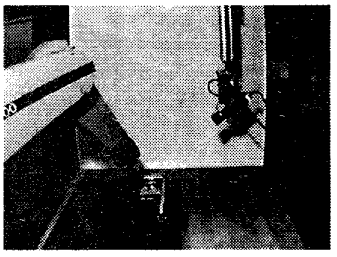

Figure 10: *Overview of the Experimental Setup*

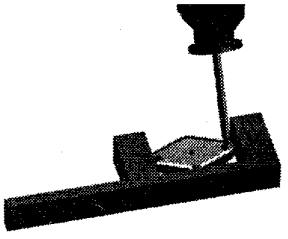

Figure 11: *The robot and the object to be imaged*

smaller than we allowed for, but the resolution is somewhat better. If the verification fails, the lens selected may be inappropriate for the task at hand, or the feature set may be too large to be viewed from a single viewpoint. We are exploring methods of determining what the problem is and how to solve it automatically.

6 Experimental Results

We have built an experimental test-bed consisting of a 5 degree-of-freedom Cartesian robot, having a work-space of roughly 1000 ft^3, carrying a CCD camera in hand/eye configuration. It can position the camera in and around our robot work-cell, and therefore can be used for static and dynamic sensor planning experiments. The end-effector of the gantry and the Puma robot performing the task are shown in figure 10.

The example task used is illustrated by figure 11, showing a model of the end-effector of a Puma robot poised over a fixtured part. The Puma will be making 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 8 linear features (the edges bounding these two rectangular strips). An 8.5 mm lens on a Sony XC77 CCD camera is used. The requirements for this task are that a length of 1 mm (on each of the features) must image to at least 1 pixel on the image plane.

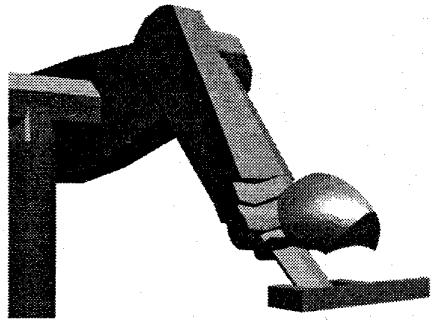

Figure 12: F_v, *including all features and constraints*

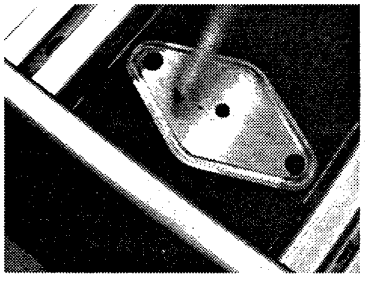

Figure 13: *Image taken from the computed viewpoint*

Since this is a dynamic sensor planning problem, we begin by computing the volume swept by the Puma as it performs this task. This volume can be seen in figure 12. (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}. V_{FR} is then intersected with the visibility volume, forming the feasibility volume F_v, shown in figure 12.

The gradient-based search algorithm is then applied to this volume, yielding the viewpoint (-310.7117, 805.1865, -235.8659). For this experiment, rather than using gradient search, the orientation was 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 equations 14 and 15, yielding $d = 8.9399$ mm and $a_{max} = f/1.1610$.

The gantry robot is moved into place and the image is taken while the Puma is performing its task. 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 13.

7 Conclusions

In this paper we have presented a new algorithm for computing viewpoints, useful for automated model-based inspection of either static objects or objects under operation in a robot work-cell. It makes use of our previous work in formulating the optical and geometric constraints on a viewpoint as well as more recent work in computing the volumes swept by an object in motion. We presented experimental results where the system computed a viewpoint, orientation, and the associated optical settings for observing features under operation in a robot work-cell.

There are still some issues requiring additional research, starting with the search technique used for the finding orientation once a position has been computed within V_c. As it stands now, we can not determine if there exists a single orientation which works for all features and all constraints from the computed position. We are currently exploring algebraic methods which will help us determining this, as well as guide our search algorithm for finding an optimal orientation. Also being explored is the automatic subdivision of the feature set (if there are no solutions valid for the entire set), and the computation of a better focal length if the given focal length is deemed inappropriate.

References

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

[2] S. Abrams and P. K. Allen. Swept volumes and their use in viewpoint computation in robot work-cells. In *Proceedings IEEE 1995 International Symposium on Assembly and Task Planning*, Pittsburgh, PA, August 1995.

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

[4] Steven Abrams. Merging sensor planning constraints. Technical report, Columbia University Department of Computer Science, 1996. In preperation.

[5] C. K. Cowan and P. D. Kovesi. Automatic sensor placement from vision task requirements. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(3):407–416, May 1988.

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

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

[8] Katsushi Ikeuchi and Jean-Christophe Robert. Modeling sensor detectability with the VANTAGE geometric/sensor modeler. *IEEE Transactions on Robotics and Automation*, 7(6):771–784, December 1991.

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

[10] Scott O. Mason and Armin Grun. Automatic sensor placement for accurate dimensional inspection. *Computer Vision and Image Understanding*, 61(3):454–467, May 1995.

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

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

[13] Tarek M. Sobh, J. Owen, C. Jaynes, M. Dekhil, and T. C. Henderson. Industrial inspection and reverse engineering. *Computer Vision and Image Understanding*, 61(3):468–474, May 1995.

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

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

[16] K. Tarabanis, R. Y. Tsai, and P. K. Allen. Analytical characterization of the feature detectability constraints of resolution, focus and field-of-view for vision sensor planning. *Computer Vision, Graphics, and Image Processing*, 59(3), May 1994.

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

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

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