Exploiting Visual Constraints in the Synthesis of Uncertainty-Tolerant Motion Plans

Armando Fox and Seth Hutchinson

Abstract—**We introduce visual constraint surfaces as a mechanism to effectively exploit visual constraints in the synthesis of uncertainty-tolerant robot motion plans. We first show how object features, together with their projections onto a camera image plane, define a set of visual constraint surfaces. These visual constraint surfaces can be used to effect visual guarded and visual compliant motions (which are analogous to guarded and compliant motion using force control). We then show how the backprojection approach to fine-motion planning can be extended to exploit visual constraints. Specifically, by deriving a configuration space representation of visual constraint surfaces, we are able to include visual constraint surfaces as boundaries of the directional backprojection. By examining the effect of visual constraints as a function of the direction of the commanded velocity, we are able to determine new criteria for critical velocity orientations, i.e. velocity orientations at which the topology of the directional backprojection might change.**

I. Introduction

TO PERFORM effectively in real-world settings, robots must be able to plan and execute tasks in the presence of uncertainty. Typical sources of uncertainty in a robotic work cell include limited sensing accuracy, errors in robot control, and discrepancies between geometric object models and physical objects (including the parts to be manipulated and the robot itself). Because of this, the application of robotic technology to manufacturing problems has typically been restricted to situations in which uncertainty can be tightly controlled (for example, by using specialized fixturing devices).

The problems associated with uncertainty in robotic systems have long been the subject of research in the robotics community. A number of planning systems have been developed that characterize uncertainty by systems of constrained variables [6], [22], [32], [39]. These variables are propagated through transformations that represent the effects of actions on uncertainty in the planner's world description, providing the planner with worst case estimates at each step of the plan. There are two primary disadvantages to this approach: 1) the planner always assumes the worst, propagating upper bounds even when actual bounds are likely to be much tighter, and 2) the planner must know *a priori* the effects of manipulation actions and sensing on the uncertainty in the description of the work cell. There is no mechanism for revising behavior at execution time to account for dynamically occurring uncertainties.

An alternative to *a priori* consideration of worst case error is to use sensory feedback to adapt task execution to the actual state of the world that the robot encounters. Force-based control can be used to effect guarded [42] and compliant motions [28], [31], [35], [41], making assembly actions robust by exploiting physical constraints imposed on motion by the geometry of the workspace. To date, the most successful planning systems that use this approach comprise the preimage family of planners [13], [15], [26], [27]. The key to the preimage planning paradigm is the transformation of physical constraints into geometric constraints, which can be expressed as C-surfaces, i.e., sets of points in the configuration space where the manipulator makes contact with a physical surface. Thus, equipped with a set of equations that govern motion and friction in the configuration space, preimage planners are capable of developing motion plans that are tolerant of uncertainties in the manipulator's position (represented by an error ball in the configuration space), its trajectory (represented by an error cone), and even in part dimensions (represented by added dimensions in the configuration space [13], [14]).

Informally stated, a preimage of a goal is the set of points from which a commanded motion is guaranteed to reach and terminate recognizably in the goal. Because preimages are often difficult to compute, Erdmann [15] introduced *backprojections* as a means of usefully approximating preimages by separating goal reachability from goal recognizability.

The primary limitation of force control is that it can only be used to constrain motion along directions normal to the C-surfaces. Position control must be used to control motions in directions tangent to C-surfaces. Therefore, hybrid position/force control is not sufficient when the exact manipulator and goal positions are not known in the dimensions of position control. As an example, suppose that the robot's task is to insert a peg into a hole in a block, and that the position of the hole is not precisely known. The insertion cannot proceed until the peg is directly over the hole; but this cannot be achieved using either position control (since the exact goal position is not known), or force control (since force control can only be used to constrain motion normal to the block surface).

One way to cope with the limitations of hybrid force/position control is to add vision sensing to the control servo loop. If the geometry of the imaging process is known, then the task geometry can be used to constrain the remaining degrees-of-freedom by using visual servo control. In recent years, the integration of computer vision with robot

Manuscript received October 19, 1992; revised June 28, 1993. This work was presented in part at the IEEE International Conference on Robotics and Automation. This work was supported by the National Science Foundation under Grant IRI-9110270.

The authors are with The Beckman Institute for Advanced Science and Technology, the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA.

IEEE Log Number 9214697.

motion control has steadily progressed, from early look and move systems in which vision was used to recognize and locate an object prior to its manipulation [36], [37], to current systems in which visual feedback is incorporated directly into the control loop [2], [18], [30], [38], [40]. This recent ability has made sensor-based robotics useful for a number of tasks where sensorless manipulation had previously failed, for example, in welding [1], [10], [23]. To date, however, the corresponding *motion planning* problem has not been addressed. Thus, even though visual servo control systems are now available, there is no motion-planning system that is capable of exploiting such a control system.

In this paper, we present a geometric motion planner that exploits visual constraints in the synthesis of uncertainty-tolerant motion plans. Specifically, we extend the preimage formalism to exploit visual constraints. In Section II, we introduce the concept of *visual constraint surfaces*, which are generated by projecting workspace features onto the image plane of a fixed camera [21]. A visual constraint surface can be used to constrain visually controlled motions in the same way that physical surfaces can be used to constrain force controlled motions. In Section III, we show how visual constraint surfaces in the workspace are mapped to configuration space constraint surfaces for the special case of $C = R^2$.

In Section IV, we review preimages [27], backprojections [15], and algorithms for the construction of backprojections [12]. In Section V, we show how the directional backprojection, i.e., the backprojection with respect to a specified velocity, can be extended to exploit visual constraint surfaces. In particular, we discuss our implemented backprojection algorithm (which extends the plane-sweep algorithm presented in [12]), and evaluate the added computational complexity of considering visual constraint surfaces. By allowing visual constraint surfaces to be included as boundaries of the backprojection, we can often significantly increase the size of the backprojection, as illustrated in Fig. 13. Following this, in Sections VI and VII, we discuss the nondirectional backprojection, and show how it must be modified to exploit visual constraints. In Section VII, we discuss a number of related issues, including multistep versus single-step plans, the problem of optimal camera placement, and extending the backprojection algorithm to the 3D case.

II. A Geometric Specification for Visual Constraints

This section introduces *visual constraint surfaces*, *visual guarded motion*, and *visual compliant motion*. Although the planning algorithms subsequently discussed will be implemented in configuration space (C-space), this section will show how visual constraint surfaces are developed in the workspace. Their C-space representation is deferred to Section III.

A. Visual Constraint Surfaces

Consider a workspace containing a number of solid objects and a fixed camera. If the imaging process is modeled by perspective projection [20], projection rays from each point in the workspace converge on the camera focal center. In general, any one-dimensional object feature will project onto a planar curve on the camera image plane. We will refer to the projection of an object feature onto the camera image plane as an *image feature*. An object feature is said to be *unoccluded* if no projection ray emanating from that feature intersects the interior of any other object or the robot. Intuitively, this means that nothing is blocking the camera's view of the feature. In this paper, the only one-dimensional object features that will be considered are the 3D edges of objects. Note that for the special case of polyhedral objects, all image features are straight line segments.

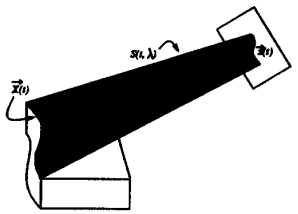

Fig. 1. Ruled VC surface generated by a curved 3D edge.

Consider a 3D edge defined by the parametric space curve $\vec{X}(\tau)$ and its corresponding image feature defined by the planar parametric curve $\vec{x}(\tau)$. The projection equations relating $\vec{X}(\tau)$ and $\vec{x}(\tau)$ depend only on the position and orientation of the camera and a set of camera parameters, e.g., the focal length of the lens. Assume that the image plane is defined with origin at $\boldsymbol{r}_0$ and local coordinate system specified by the orthonormal vectors $\boldsymbol{I}$, $\boldsymbol{J}$. The normal to the image plane is specified by $\boldsymbol{K} = \boldsymbol{I} \times \boldsymbol{J}$. Then, for the case of perspective projection, the projection equations relating $\vec{X}(\tau)$ and $\vec{x}(\tau)$ are given by:

$$\begin{cases} x_I(\tau) = \frac{-d_f(\vec{X}(\tau)-\boldsymbol{r}_0)\cdot\boldsymbol{I}}{(\vec{X}(\tau)-\boldsymbol{r}_0)\cdot\boldsymbol{K}-d_f} \\ x_J(\tau) = \frac{-d_f(\vec{X}(\tau)-\boldsymbol{r}_0)\cdot\boldsymbol{J}}{(\vec{X}(\tau)-\boldsymbol{r}_0)\cdot\boldsymbol{K}-d_f} \end{cases} \tag{1}$$

where D_f is the distance from the image plane to the center of projection of the camera, i.e., the focal length of the lens. These parameters can be obtained by calibration procedures described in [9]. Once they are derived, if the camera remains fixed (as we assume), the same projection equations can be used to compute the image-plane coordinates of any image feature, given the world coordinates of an object feature.

A *visual constraint (VC) surface* is a ruled surface $\boldsymbol{S}(\tau, \lambda)$ bounded by $\vec{X}(\tau)$, $\vec{x}(\tau)$, and the rays joining their respective endpoints, as in Fig. 1. In the special case of polyhedral obstacles, all image features are straight line segments, so that the VC surfaces are polygons. A VC surface is defined by

$$\boldsymbol{S}(\tau, \lambda) = \vec{X}(\tau) + \lambda(\vec{X}(\tau) - \vec{x}(\tau)). \tag{2}$$

Recall that a ruled surface is generated by a family of lines [17]. For $\boldsymbol{S}(\tau, \lambda)$, a particular generating line is obtained for each valid value of τ. We will refer to such a line as a generating line of $\boldsymbol{S}(\tau, \lambda)$, or simply a generating line. For perspective projection, each generating line is a projection ray through the center of projection and the image plane point $\vec{x}(\tau)$.

Note that a visual constraint surface does not intersect any obstacles. This is because a necessary condition for an edge to have a projection on the camera image plane is that the edge not be occluded. Similarly, if only a part of a particular 3D edge is visible (perhaps the rest of the edge is occluded by another object), only the visible part will have a projection on the image plane, so that the resulting surface will not intersect any obstacles.

1) Visual Compliant Motion: Compliant motion has been exploited in various motion planning and execution strategies [14], [16], [27], [28]. During compliant motion, a physical surface is used to constrain the motion of a robot along one or more degrees-of-freedom [28], [31], [35], [41]. For example, sliding motion along a surface might be achieved by ensuring that some constant force be maintained in the direction normal to the surface.

We define visual compliance as compliant motion along a (virtual) VC surface, such that the manipulator's motion is constrained to always remain in contact with a particular generating line of the VC surface. Visual compliance can be achieved by means of a closed-loop visual servo-system, as described in [8], [9].

2) Visual Guarded Motion: We define visual guarded motion analogously to guarded motion using physical surfaces, in which the robot moves until force feedback indicates contact with a physical surface [42]. We say that the force feedback provides a termination condition for the motion. VC surfaces can be used for visual guarded motion; that is, the manipulator can move along a trajectory that intersects a VC surface and be instructed to stop when this intersection occurs. This is possible because the intersection is a visually observable event.

Since VC surfaces are virtual rather than physical, the motion planner also has the option of ignoring them, in contrast to the force-based approach, which must explicitly consider all physical surfaces on which *sticking* may occur [15].

III. Visual Constraints in Two Dimensions

In this section, we describe the computation of visual constraint rays in the case of a 2D workspace populated by polygonal obstacles. We begin by discussing the construction of visual constraint rays in the workspace, which is a special case of the formalism developed in Section II. We then discuss the selection of robot features that will be used by the visual servo-system. Following this, we describe how to map visual constraint rays into the C-space $C = \boldsymbol{R}^2$. Finally, we discuss the time complexity of the algorithms presented in this section.

A. Workspace Visual Constraint Rays

In the case of a 2D workspace, the camera is a one-dimensional sensor positioned in the plane. Using perspective projection, all projection rays converge on the camera projection center. We assume that if an object vertex is unoccluded, i.e., a projection ray from that vertex to the camera focal point intersects the interior of no workspace obstacle, then the projection of that vertex in the camera image can be located by the vision system. Workspace VC rays can be computed by extending rays from unoccluded workspace obstacle vertices to the camera projection center, as shown in Fig. 2.

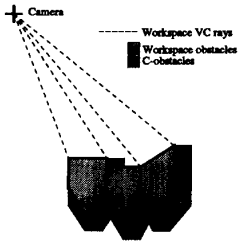

Fig. 2. Construction of workspace VC rays from unoccluded obstacle vertices.

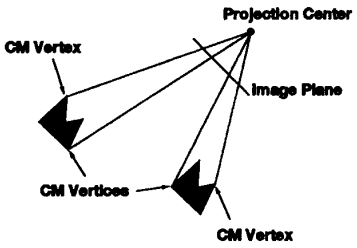

Fig. 3. Different positions of the polygonal robot give different CM vertices.

B. Selecting Robot Features for Visual Servo Control

In a 2D workspace, visual compliant motion is effected by moving a particular robot vertex so that it remains in contact with some VC ray emanating from a workspace object vertex. This raises the question of which vertices of the robot should be used in visual compliant motions.

If the robot is a simple polygon, its projection on the camera image plane is a line segment whose endpoints represent the two furthest-apart robot vertices simultaneously visible to the camera. Note that there may be other robot vertices that project to points on the line segment; however, for the purpose of visual compliant motion, we assume that the vision system can only robustly distinguish in real time the two vertices whose projections are the endpoints of the image plane line segment, i.e., the silhouette of the robot. This restriction could be lifted if the vision system were capable of robustly distinguishing other unoccluded vertices in real time.

We will refer to the two robot vertices that project to the endpoints of the line segment as *CM vertices*, to indicate that they are the only robot vertices suitable for effecting compliant motion along a VC ray. Note that the particular robot vertices that are CM vertices can change with the position of the robot. Fig. 3 shows the same robot in two different positions for which the CM vertices are different. In certain nongeneral configurations of the robot, two robot vertices may lie along the same projection ray. In such cases, we may arbitrarily select one of these as a CM vertex. Thus, for any specified position of the robot, we will obtain two CM vertices whose projections are the endpoints of the image plane line segment representing the robot. There cannot be only one CM vertex

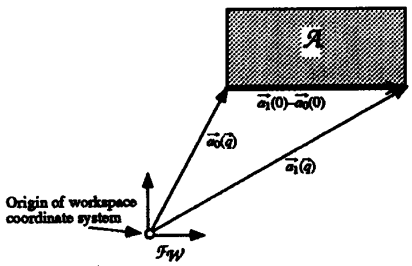

Fig. 4. The spatial relationship between the robot vertices.

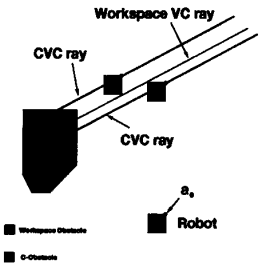

Fig. 5. Construction of two CVC rays from a single workspace VC ray.

unless the robot is itself a line segment and is collinear with a projection ray.

C. Configuration Space Representation of Visual Constraints

Visual constraint rays in the workspace give rise to C-space visual constraint rays (CVC rays). In mapping workspace VC rays to CVC rays, we must allow for either of the CM vertices to be moved compliantly along the workspace VC ray. Suppose that a particular vertex a_0 of the robot is taken as the origin of the robot's internal coordinate frame to compute a representation of the C-space, C. Since trajectories in C will specify the motion of vertex a_0 among the C-obstacles, we must find an appropriate representation for compliant motion of an arbitrary robot vertex a_j along a VC ray.Since we are considering the case where $C = \boldsymbol{R}^2$, the spatial relationship between a_0 and a_j is fixed. Specifically, if for some configuration $\vec{q}$, the world coordinates of a_0 are given by the vector $\vec{a}_0(\vec{q})$, then the world coordinates of a_j in the same configuration are given by $\vec{a}_j(\vec{q}) = \vec{a}_0(\vec{q}) + (\vec{a}_j(0) - \vec{a}_0(0))$. Fig. 4 illustrates this relationship for vertices a_0 and a_1.

Let e_{vc}^W be a VC ray emanating from a workspace obstacle vertex b_i, and let a_j be a CM vertex of the robot when the robot is positioned such that a_j coincides with b_i. Then, as the robot moves compliantly, maintaining contact between a_j and e_{vc}^W, vertex a_0 will move along a straight line trajectory parallel to e_{vc}^W but displaced from it by $\vec{a}_0(0) - \vec{a}_j(0)$. We construct a CVC ray e_{vc} in C, whose endpoints are the endpoints of e_{vc}^W displaced by $\vec{a}_0(0) - \vec{a}_j(0)$. Motion of a_0 (the reference vertex) along e_{vc} corresponds to visual compliant motion of vertex a_j along e_{vc}^W. Similarly, visual guarded motion of a_0 terminating on e_{vc} corresponds to visual guarded motion of a_j terminating on e_{vc}^W, which is a visually observable event. The construction of CVC rays using this technique is illustrated in Fig. 5.

1) Intersection of CVC Rays with C-Obstacles: When a CVC ray intersects a C-obstacle, visual compliant motion *cannot* be effected along the portion of the CVC ray that lies inside of the C-obstacle, since doing so would cause the robot to overlap a workspace obstacle. In this case, we must truncate the CVC ray at those points where it enters CB, retaining only those segments of the CVC ray that lie outside of B (we will use the notation CB_i to indicate a particular C-obstacle, and CB to represent the union of all C-obstacles). In Fig. 6, the CVC ray constructed from the workspace VC ray e_{vc}^W intersects the interior of C-obstacle CB_1. That part of the CVC ray that lies outside of CB includes two line segments: e_1 is the segment between the camera and artificial vertex b_3 on edge E, e_2 is the segment from artificial vertex b_2 to artificial vertex b_1. In this example, only the segments e_1 and e_2 are included in the set of CVC rays.

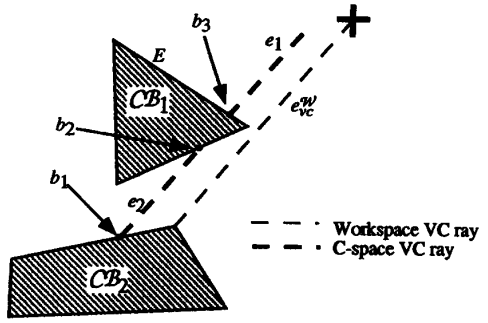

Fig. 6. A CVC ray may intersect the interior of CB.

2) Intersection of Multiple CVC Rays: Although workspace VC rays intersect only at the camera projection center, two C-space VC rays may intersect at a point other than the camera projection center. Fig. 7 shows one example of intersecting CVC rays. Since the CVC rays corresponding to a single workspace VC ray are parallel, the intersecting CVC rays can not have originated from the same workspace VC ray. The physical interpretation of the intersection of two CVC rays is a change from executing compliant motion of vertex a_i along workspace VC ray $e_{vc_k}^W$ to compliant motion of vertex a_j, $j \neq i$ along workspace VC ray $e_{vc_l}^W, k \neq l$. For example, such an intersection point might correspond to a change from compliant motion of the top right vertex of the square robot along VC ray $E_{vc_1}^W$, to compliant motion of its bottom-left vertex along $e_{vc_2}^W$.

3) Algorithmic and Complexity Issues: The C-space representation of the VC rays can be computed by using two successive plane-sweep algorithms. The first constructs all of the CVC rays, and the second is used to truncate these CVC rays, as described in Section III-C-1). Plane-sweep algorithms comprise a general class of algorithms that operate by stepping through a queue of geometrically interesting or critical events, and performing some local processing at each event in order to construct a global solution to some input problem. Conceptually, a line is "swept across" the plane, stopping at the critical events. At any time, the sweep-line divides the space into two half-spaces, such that the solution in one half-space has been computed and will not be affected by the computation of the solution in the other half-space. Further, between any two critical events, the solution that is

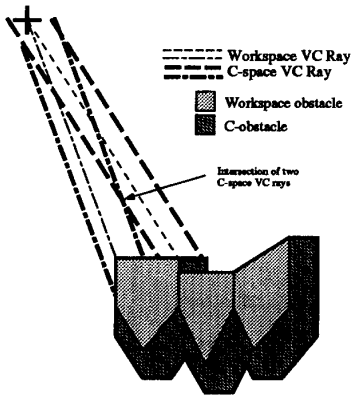

Fig. 7. Intersection of two CVC rays.

being constructed does not change in a qualitative way.

The input to a plane-sweep algorithm is an arrangement of geometric objects and the output is some desired operation on the objects. Plane-sweep algorithms can be used, for example, to compute the intersection points of an arrangement of line segments or the union of an arrangement of polygons [34]. The advantage of a plane-sweep algorithm over a naive algorithm is usually reduced-time complexity. As an example, a naive algorithm for computing the intersection points of n line segments runs in $O(n^2)$ time, but a plane-sweep algorithm can compute them in $O((n+c)\log n)$ time, where c is the number of intersection points. In the worst case $c = O(n^2)$, but in practice, it is usually small and the plane-sweep algorithm is more efficient than the naive algorithm.

The initial set of CVC rays can be constructed by a variant of the traditional plane-sweep algorithm in which the sweep-line is a half-line that is rotated about the projection center of the camera. Let P represent the projection center of the camera. Then anchor a half line at P, and sweep this half line from θ_1 to θ_2, where θ_1 And θ_2 are the angles at which the sweep-line anchored at P intersects the two extreme points of the line segment that defines the camera image plane. The obstacle vertices define the set of events at which the sweep stops. By properly maintaining the status of the set of intersections of the sweep-line with obstacle edges, it is possible to determine whether each visited vertex is unoccluded. Making this determination and performing the processing necessary to update the status of the sweep-line requires $O(\log n)$ operations at each vertex. Since there are $O(n)$ vertices, determining the set of unoccluded vertices requires $O(n \log n)$ operations. This is a slight variation of the algorithm for computing visibility graph edges described in [25].

In addition to determining the set of unoccluded vertices, we must also determine which vertices of the robot are CM vertices for each unoccluded obstacle vertex. In general, for a robot with m vertices this can be done by a naive algorithm using $O(m)$ operations for each obstacle vertex. However, we can interleave the process of determining the CM vertices with that of determining the set of unoccluded vertices by making the following observation. The set of CM vertices changes only when the rotational sweep-line becomes parallel to an edge in the convex hull of the robot. In other words, as the sweep-line rotates, the same two robot vertices will be the CM vertices until the sweep-line becomes parallel to an edge in the convex hull of the robot. Therefore, by modifying the rotational sweep so that it also stops at orientations parallel to edges in the convex hull of the robot, we can simultaneously compute the set of unoccluded obstacle vertices and the two CM vertices for each unoccluded vertex. The complexity of the resulting algorithm is $O(m \log m)$ to compute the convex hull of the robot, and $O((m+n)\log n)$ to perform the sweep. If we assume that $n > m$, the algorithm requires $O(n \log n)$ operations.

Truncating the CVC rays can also be accomplished by a plane-sweep algorithm. Here, a line is swept across the plane in the direction perpendicular to the camera image plane. The events are the vertices of C-obstacles and the intersections between C-obstacle edges and CVC rays. Such an algorithm requires $O((n+c)\log n)$ operations, where c is the number of intersections of CVC rays with C-obstacle edges.

IV. Preimages and Backprojections

In this section, we provide a review of preimages and backprojections. We begin by reviewing the preimage formalism of Lozano–Pérez, Mason, and Taylor [27]. Following this, we present a review of backprojections [15], including a discussion of issues related to goal recognizability. Finally, we describe the algorithm of Donald and Canny [12] for computing a directional backprojection in $O(n \log n)$ time (where n is the number of vertices of the C-space obstacle region). Readers that are familiar with this work may wish to skip this section.

A. Preimage Planning

Lozano–Pérez, Mason, and Taylor [27] present a formalism for the automatic synthesis of fine-motion strategies using *preimages*. Informally stated, a preimage for a specified goal region is a set of points from which a commanded motion is guaranteed to terminate recognizably in the goal region. The main advantage to the preimage formalism is that it allows the fine-motion planner to explicitly consider uncertainties in position and control.

In [27], position uncertainty is modeled by an error ball, $B_{ep}(p)$, in the C-space, centered on the actual position p. Velocity uncertainty is modeled by an *uncertainty cone*, whose vertex angle represents the maximum directional deviation between the commanded velocity and the actual velocity. If a position p_0 lies within the error ball centered on measured position p_0^*, then p_0^* is said to be *consistent with* p_0. Intuitively, this means that the sensor might "mistakenly" measure p_0 as p_0^*. A similar definition holds for measured versus actual velocity vectors.

The velocity uncertainty cone plays a key role in the computation of preimages (and, as will be seen below, in the computation of backprojections). Specifically, both preimages and backprojections may include in their boundaries *free edges* (also called *free rays*). A free edge is a line segment that is parallel to an edge of the inverted velocity cone, erected at

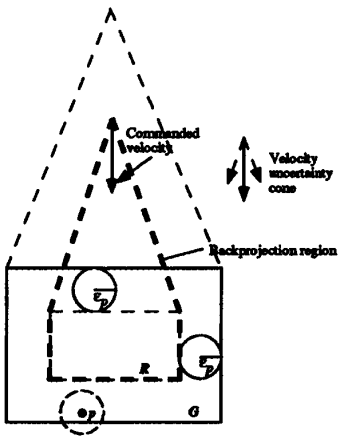

Fig. 8. Modifying the goal region to account for position uncertainty.

some C-obstacle vertex. For example, in Fig. 8, the top two boundary edges of the backprojection (outlined in light dashed lines) are free edges.

The formal definition of a *directional preimage* $P_\theta(G)$ is as follows. Let G be a goal region in C_{valid} (where C_{valid} is the set of valid configurations in the configuration space C). A motion command $M = (\vec{v}_\theta, TC)$, consists of a commanded velocity $\vec{v}_\theta$ (which is considered to be a unit vector with orientation θ), and a termination predicate TC, which is used to determine when the motion has achieved the goal. The preimage of G for motion M is defined as a subset of points, $R \subseteq C_{\text{valid}}$, such that if M commences from any point in R, TC will eventually return *true*, at which point the motion will terminate in G. A *maximal* directional preimage is the largest possible preimage relative to a given motion direction and goal region.[1]

A preimage planner works by backward-chaining from the goal region G to the C-space region I in which the initial configuration lies. If the backward-chaining process terminates successfully, the result is a sequence of directional preimages $P_1, P_2, \cdots, P_r$ such that: a) P_i is the directional preimage of P_{i-1} relative to the commanded velocity $\vec{v}_{\theta_i}$ and termination condition TC_i, b) P_1 is the directional preimage of G, and c) $I \subset P_r$. The reverse sequence of motion commands $M_r, M_{r-1}, \cdots, M_1$, with $M_i = (\vec{v}_{\theta_i}, TC_i)$, is the generated r-step motion strategy guaranteed to recognizably reach the goal configuration from the initial configuration.

Mason has shown that the LMT approach of preimage backchaining is bounded complete, i.e., if a solution with bounded number of motions exists, the LMT preimage backchaining method will find it, and that it suffices to consider directional preimages as subgoals in the recursive backchaining process [29]. These results, however, do not imply that preimages are computable. In fact, Erdmann has proven by a reduction from the halting problem, that, in arbitrary environments, preimages and backprojections are uncomputable [15]. That the recursively defined constraints in the proof do not generally occur in practice led Erdmann to conjecture without proof that in an environment with a known finite number of constraints, preimages should be computable. Canny has shown that this is indeed the case when the set of possible robot trajectories has a finite parameterization, and the set of feasible trajectories is a semialgebraic subset of the parameter space [7]. Canny's approach is to cast the fine motion planning problem as a decision problem in the theory of the real numbers, and to then use quantifier elimination algorithms (see, e.g., [11]) to derive parametric semialgebraic sets that are preimages. Backprojections from recognizable goal regions also constitute valid preimages [15], [26]. Such backprojections are the topic of the next section.

[1]This explanation paraphrases Latombe's discussion of preimages [25], which also presents the relevant equations and formal definitions of the termination predicate.

B. Backprojections

A major problem in computing preimages is that there are many circumstances under which a real termination predicate TC may not be able to reliably detect entry into the goal region. There are two primary reasons for this: uncertainty in sensing and limitations on the amount of information available to the termination predicate. Because of these difficulties, Erdmann introduces backprojections [15] as a means of approximating preimages. Essentially, a backprojection is a preimage without a termination predicate; that is, a backprojection is the set of all points from which an appropriate commanded velocity is guaranteed to enter the goal, regardless of whether entry into the goal is recognized. The lack of a termination predicate makes backprojections weaker than preimages, but backprojections are often easier to compute than preimages, and are appropriate for use in certain planning problems.

1) Uncertainty in Sensing: Fig. 8 depicts a rectangular goal region G. Its directional backprojection consists of the rectangular region G together with the region enclosed by light dashed lines. A trajectory with a commanded velocity straight down originating in the backprojection is guaranteed to enter G. Due to position sensing uncertainty, modeled by an error disk of radius ϵ_p, only points inside the rectangle R can be unambiguously sensed as being in the goal. This is because, for actual positions in G but not in R, there exists an interpretation of the measured position p^* that is not in G. Put another way, because of uncertainty, we can equivalently regard the robot as a disk of radius ϵ_p and assume perfect sensing, and require that the disk be entirely contained in G. This implies that the center point of the disk must be at least a distance ϵ_p from every edge of the goal region. However, if the reduced goal region R had been used as a base from which to backproject (resulting in the directional backprojection enclosed by bold dashed lines), all of the points in the goal region could be unambiguously determined to be in the goal region, since they are all at least ϵ_p distant from any possible nongoal interpretation.

Erdmann introduces a method for constraining goal sets so that they will be suitable bases for backprojection, but does not formalize all the requirements. Essentially, he begins by choosing a subset of the goal that is guaranteed to be recognizably in the goal, given position sensing uncertainty, and backprojects from this "foundation." The resulting backprojection is then

iteratively enlarged by adding those regions of the goal that are either recognizably in the goal, or from which any trajectory will recognizably enter the existing backprojection region. Although an informal example is given for the translating polygonal robot case, no algorithm is presented for this case or in general. Latombe, *et al.* [26] have formalized the idea of restricting the goal by constructing *goal kernels* (subsets of the goal for which recognizability is guaranteed), and by using sticking edges as goal regions. In both cases, since termination in the goal is guaranteed, such backprojections constitute valid preimages.

2) Limitations of the Termination Predicate: Erdmann discusses three possibilities for the termination predicate: no sense of history, no sense of time, and no sense of history or time. A termination predicate without history cannot rely on information regarding where the robot has been in the past to disambiguate a current position reading p^*. For example, suppose there are two goal regions, and the measured position p^* is consistent with being in either one. Knowing the set of past positions of the robot might be sufficient to determine which goal is indicated by the current position reading. That is, one of the two interpretations might be inconsistent with the sensor history despite being consistent with p^*. The termination predicate without history may not rely on past sensor readings in this way.

Similarly, the termination predicate without sense of time cannot determine which of several consistent interpretations of p^* is correct by considering the velocity of the robot and the elapsed time since leaving the initial position.

The motivation for considering a termination predicate with neither history or time is that for a given commanded motion, the starting position p_0^* and the elapsed time since starting the motion will not be known until runtime. Thus, it is impossible for the planner to develop a complete strategy based on these parameters. This is a byproduct of the decomposition of the robot motion problem into separate planning and execution stages. In this paper, we consider only termination predicates with no sense of history or of time.

C. Computing Backprojections in $C = \boldsymbol{R}^2$

Donald and Canny have implemented a plane-sweep algorithm for computing backprojections of polygonal regions for the case $C = \boldsymbol{R}^2$ [12]. The algorithm works by sweeping a line across the plane in the direction opposite that of the commanded velocity. The sweep-line stops at events that are: (1) vertices of C-obstacles, (2) vertices of the goal region, (3) the intersection of two free edges, (4) the intersection of a free edge with a boundary of the goal region, and (5) the intersection of a free edge with an edge of a C-obstacle. In each case, the backprojection is extended appropriately, using only local decision criteria.

Fig. 9 illustrates the operation of the algorithm on a simple example. The backprojection being built is enclosed by bold lines. The commanded velocity is straight down, with the illustrated uncertainty cone. In each frame, a new vertex is considered and appropriate edges indicated by arrows are added to the backprojection. When the sweep finishes, the directional backprojection has been computed. The vertices are numbered in the order in which they are encountered by the sweep. Vertices 1 and 2 appear to be on a horizontal line, so that the order in which they are encountered is ambiguous. In such cases, a small perturbation may be introduced so that vertices 1 and 2 are no longer on the same horizontal line, or the vertices may be sorted in planar-lexicographic (x, y)-order.

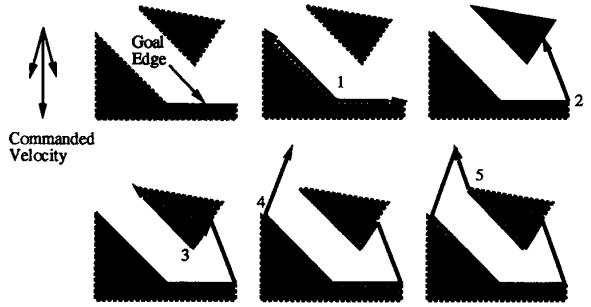

Fig. 9. Plane-sweep of a simple 2D polygonal environment.

Donald shows [12] that the algorithm is correct provided that the environment has a bounded number of vertices, and that the friction cone is larger than the velocity uncertainty cone. (This latter criterion is necessary because without it, the algorithm would not be able to rely only on local information to determine how to continue the backprojection.)

V. The Effect of Visual Constraints on the Directional Backprojection

In this section, we show how the backprojection algorithm of Donald and Canny can be modified to exploit visual constraints. We will refer to a backprojection that includes CVC rays as a VC-enlarged backprojection, and we will denote a VC-enlarged backprojection by $B_{vc_\theta}(G)$. We begin by describing the new event types that must be considered by the plane-sweep algorithm. We then present two examples of backprojection continuation at such events. Following the examples, the formal decision criteria for determining whether to include a CVC ray in the backprojection boundary are presented. We then discuss the time complexity of the modified directional backprojection algorithm. Finally, we present examples of $B_{vc_\theta}(G)$ for $C = \boldsymbol{R}^2$ computed by our implementation of the modified algorithm.

A. New Events for the Plane-Sweep Algorithm

The first step in modifying the Donald and Canny directional backprojection algorithm to exploit CVC rays is to determine the new events that must be considered during the plane sweep. When CVC rays are included, there are three new types of events that must be considered:

1) The intersection of a CVC ray with a C-obstacle edge (or a C-obstacle vertex);
2) The intersection of a CVC ray with a free edge of the inverted velocity uncertainty cone;
3) The intersection of two CVC rays.

When a CVC ray intersects a C-obstacle edge, we create an artificial vertex at the intersection point. If a particular C-

obstacle vertex has a CVC ray incident on it, that vertex is marked to indicate this fact, and the equation of the incident CVC ray is attached to it. Thus, only intersections of CVC rays with C-obstacle vertices will be considered in the remainder of the paper.

In the worst case, there will be $O(n)$ new artificial vertices for the CVC rays that intersect C-obstacle edges. There are $O(n^2)$ intersections of free edges with CVC rays, but during the construction of the backprojection, only the first intersection of a free edge with a CVC ray is considered. Therefore, the number of new events of this type that must be considered by the algorithm is $O(n)$.

Finally, there are, in the worst case, $O(n^2)$ pairwise intersections of CVC rays. To see this, consider that the intersection of two CVC rays occurs when the two CM vertices of the robot are simultaneously in contact with two distinct workspace VC rays, say e^W_{vci} and e^W_{vcj}. Such an intersection point can be created by positioning one CM vertex of the robot on e^W_{vci}, and then moving the robot compliantly along this ray until the remaining CM vertex contacts e^W_{vcj}.

Thus, the number of events considered by the modified plane-sweep algorithm is $O(n + c)$, where c is the number of intersections of pairs of CVC rays.

B. Example of Backprojection Continuation

Before presenting the formal decision criteria for the new events, we present the following two examples. These examples show how visual constraint surfaces can be used to bound the backprojection, which is made possibly by exploiting visual compliance (which is analogous to physical compliance).

Consider an obstacle vertex b_i with incident C-obstacle edges e_i and e_{i-1}, and incident CVC ray e_{vc}, as shown in Fig. 10.[2] The commanded velocity is straight down. Edge e_{i-1} has already been added to the backprojection, and it forms the left edge of the current backprojection boundary. The algorithm must decide whether to continue the backprojection along e_i, e_{vc}, or the free edge of the inverted velocity uncertainty cone e_{ev}.

Intuitively, the algorithm tries to make the backprojection as large as possible. Suppose e_i is a sliding edge. Then choosing it will always result in the maximal backprojection because if either e_{vc} or e_{ev} made the backprojection larger, it would intersect the interior of the C-obstacle bounded by e_i. Conversely, suppose e_i is a sticking edge. In this case, a simple comparison of the orientations of e_{vc} and e_{ev} suffices to determine which ray should be chosen, namely, the one that forms the larger angle with the direction perpendicular to the commanded velocity.

When a free edge intersects a CVC ray, it is not always the case that the CVC ray should be added to the boundary of the backprojeciton. Figs. 11 and 12 illustrate an example of such a case. Suppose that the motion begins at the point R with a commanded velocity straight down. When the robot

[2] Although CVC rays are not necessarily incident on C-obstacle vertices, we may assume this without loss of generality since artificial vertices are introduced where CVC rays intersect C-obstacle edges.

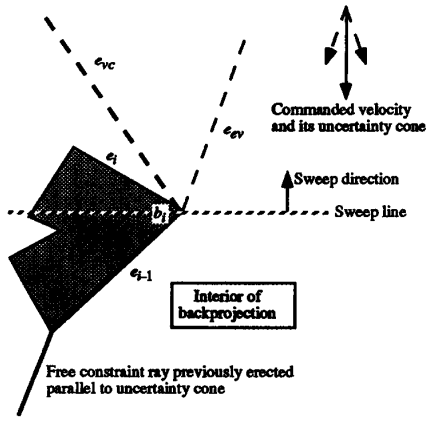

Fig. 10. Deciding how to continue the backprojection from a C-obstacle vertex.

intersects the CVC ray e_{vc}, the execution system equipped with visual feedback will begin visual compliant motion along the ray toward the goal. However, if visual compliant motion continues until the C-obstacle is contacted, the motion may not reach the goal. Instead, somewhere along e_{vc}, between points p and q, the execution system must resume motion in a downward direction. But the region of C-space in which this change of direction must occur is a free-space region, so that position sensing uncertainty becomes a problem. Specifically, since visual feedback cannot be used to determine where the robot is positioned along a projection ray, the robot will comply to e_{vc} and leave the backprojection region, continuing until contact is made with the C-obstacle upon which e_{vc} terminates. Therefore, we cannot guarantee that the single commanded motion would reach the goal, and we conclude that the CVC ray in this example should not be a backprojection boundary. We note that it would be possible to allow for the use of position sensing to detect when the robot has reached point p. However, determining that p has been reached is essentially equivalent to the goal recognizability problem, and therefore greatly complicates the computation of the backprojection (indeed, it was this difficulty that led Erdmann to separate goal recognizability and goal reachability in his original formulation of backprojections [15]).

C. Intersection of a CVC Ray and a C-Obstacle Edge

The decision criteria for an event corresponding to the intersection of a CVC ray and a (possibly artificial) C-obstacle vertex are as follows. As in the aforementioned example, the vertex event being processed is a (possibly artificial) C-obstacle vertex b, with incident obstacle edges e_i and e_{i-1} and incident CVC ray e_{vc}. We denote by e_{ev} the free edge of the velocity uncertainty cone erected at b. We assume e_{i-1} has already been added to the backprojection, as in Fig. 10. We denote the orientation of the sweep-line by $\vec{y}$, i.e., the direction of the sweep itself is perpendicular to $\vec{y}$. We assume that $\vec{y}$ points to the interior of the backprojection region that lies behind the sweep-line, so that it would point to the right in Fig. 10. The decision criteria are as follows:

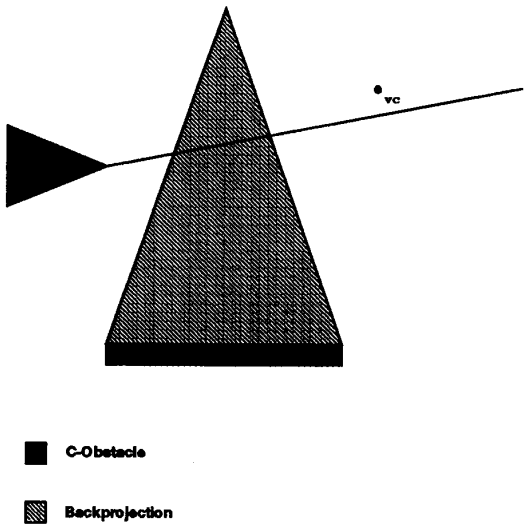

Fig. 11. The *correct* backprojection, when a CVC ray intersects a free edge, and the CVC ray does not terminate in the backprojection.

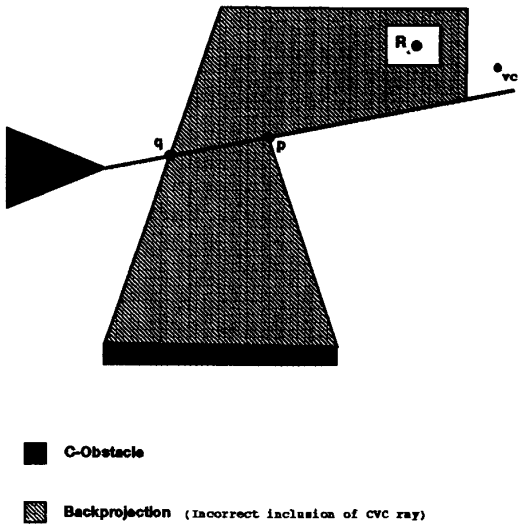

Fig. 12. An *incorrect* backprojection, when a CVC ray intersects a free edge, and the CVC ray does not terminate in the backprojection.

1) If e_i is a sliding edge, continue the backprojection along e_i;
2) Otherwise, if the angle between e_{vc} and $\vec{y}$ is greater than the angle between e_{ev} and $\vec{y}$, continue the backprojection along e_{vc}.
3) Otherwise, add e_{ev} to the backprojection.

D. Intersection of a CVC Ray and a Free Edge

The CVC ray should be used to continue the backprojection at the intersection of a CVC ray and a free edge of the velocity uncertainty cone only when the termination point of the CVC ray on a C-obstacle edge is known to be in the backprojection. This becomes evident by enumerating the possible types of C-edges on which a CVC ray e_{vc} may terminate.

1) CVC ray e_{vc} terminates on a nongoal sticking edge. In this case, compliant motion along e_{vc} would result in contact with a sticking edge from which motion to the goal is not possible. Therefore, e_{vc} should *not* be included in the backprojection.

2) CVC ray e_{vc} terminates on a nongoal sliding edge e_j along which sliding motion away from the goal occurs. In this case, a motion that brings the robot into contact with e_j will continue by sliding away from the goal. Therefore e_{vc} should *not* be included in the backprojection.

3) CVC ray e_{vc} terminates on a nongoal sliding edge e_i along which sliding motion toward the goal occurs. In this case, a motion that brings the robot into contact with e_i will continue by sliding towards the goal. Therefore e_{vc} can be included in the backprojection.

4) CVC ray e_{vc} terminates on a goal edge. Visual compliant motion along e_{vc} will bring the robot into contact with the goal, so e_{vc} can be included in the backprojection.

The cases enumerated above are exhaustive, and the cases in which e_{vc} should be included in the backprojection occur only when e_{vc} terminates on an edge already known to be in the backprojection.

The decision criteria at a vertex event v at which a free edge of the velocity uncertainty cone e_{ev} and a CVC ray e_{vc} intersect are as follows. As in Section V–C, the vector $\vec{y}$ points along the sweep-line toward the interior of the backprojection.

1) If the two edges incident on v are already in the backprojection, no new edge is added to the backprojection boundary (this case is illustrated in Fig. 15).
2) If e_{vc} is incident on a C-obstacle edge or vertex that is already included in the backprojection, *and* the angle between e_{vc} and $\vec{y}$ is greater than the angle between e_{ev} and $\vec{y}$, continue the backprojection along e_{vc}.
3) Otherwise, continue the backprojection along e_{ev}.

E. Intersection of a Two CVC Rays

Since the intersection of two CVC rays is a visually observable event, i.e., the two CM vertices simultaneously contact two workspace VC rays, at such an intersection point, the backprojection algorithm should be continued along the CVC ray that maximizes the size of the enclosed backprojection. Let $\vec{y}$ be as defined above, and let the two intersecting CVC rays be e_{vc1} and e_{vc2}. Then

1) If the angle between e_{vc1} and $\vec{y}$ is greater than the angle between e_{vc2} and $\vec{y}$, continue the backprojection along e_{vc1};
2) Otherwise, continue the backprojection along e_{vc2}.

F. Asymptotic Time Bounds

Before beginning the plane-sweep to compute a directional backprojection, $O(n)$ free edges can be erected at sticking vertices and a separate plane-sweep can be used to intersect them with each other and obstacle edges in time $O(n \log n)$. Therefore, we make the following proposition.

Proposition 1: The time to compute the directional backprojection with visual constraint rays $B_{vc_\theta}(G)$ is $O((n +$

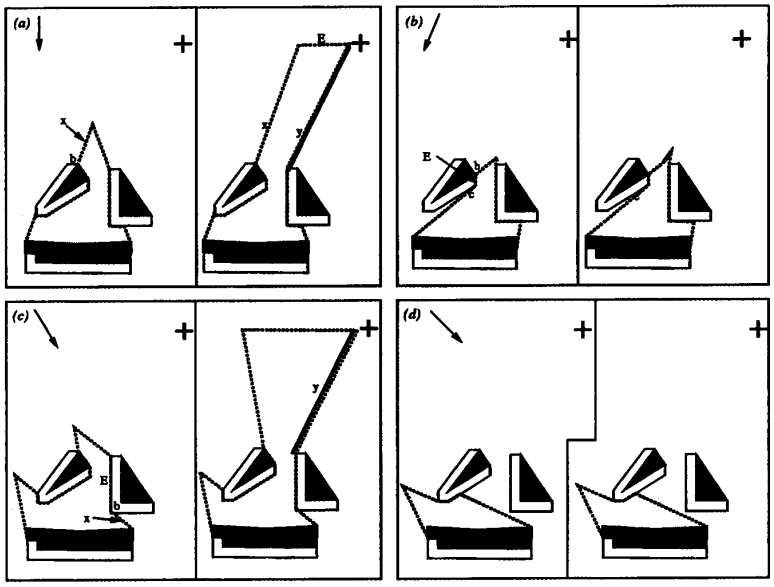

Fig. 13. Effect of considering CVC rays in computing the directional backprojection.

$c) \log n)$, where c is the number of pairwise intersections of CVC rays.

Proof: The total number of events to be examined is $O(n + c)$, since there are $O(n)$ additional artificial vertices introduced by the CVC rays, and c vertices that correspond to the intersection of pairs of CVC rays. At each event, a constant number of local comparisons is required: (a) a vertex incident on a C-obstacle edge requires one test to determine whether e_i is a sliding or a sticking edge, and if the latter, one test to determine whether e_{vc} or e_{ev} should be used to continue the backprojection; (b) a vertex that corresponds to the intersection of a free edge with a CVC ray requires one test to determine whether e_{vc} or e_{ev} should be used to continue the backprojection; (c) a vertex that corresponds to the intersection of a pair of CVC rays requires one test to determine which CVC ray should be used to continue the backprojection. Thus, the decision of how to continue the backprojection at any event is $O(1)$. Finally, as with all plane-sweep algorithms, at each event the algorithm must perform book-keeping operations that require time $O(\log n)$. Therefore, the asymptotic running time of our modified version of the Donald and Canny algorithm for computing a directional backprojection becomes $O((n + c) \log n)$. □

G. 2D Examples

We now present some examples contrasting backprojections that contain CVC rays to those obtained without considering CVC rays. The two sets of examples illustrate the effect of considering versus ignoring CVC rays. In each case, observe that the CVC rays never make the backprojection smaller, and frequently make it larger.

In all of the example figures, we use the following conventions.

1) The directional backprojection is enclosed by a dashed line, with edges contributed by visual constraints highlighted in bold.

2) Workspace obstacles are shaded; C-obstacles are outlined.

3) Solid arrows denote the commanded velocity direction.

4) The camera projection center (workspace coordinates) is indicated by a cross.

5) The goal polygon is shaded black.

6) The direction $\theta = 0$ corresponds to movement straight down the page.

Fig. 13(a)–(d) compare the traditional backprojection with the VC-enlarged directional backprojection for a range of commanded velocities. In each frame, the traditional directional backprojection, $B_\theta(G)$ is shown on the left, and the VC-enlarged backprojection $B_{vc_\theta}(G)$ is shown on the right.

Frame (a) corresponds to commanded velocity $\theta = 0$. In this case, the backprojection is significantly enlarged by the CVC ray y. Note that free edge x, given by the inverted velocity uncertainty cone erected at vertex b in both backprojections, terminates on a workspace obstacle vertex, but x does not close the backprojection in the right-hand figure since y is nearly parallel to x. (The top horizontal edge of the backprojection is given by the environment's bounding box.) That the backprojection continues from b along x rather than along a CVC ray indicates that x results in a larger backprojection than any CVC ray incident on b.

Frame (b) corresponds to $\theta = -\pi/6$. In this case, even though a small part of a CVC ray contributes to the backprojection, the backprojection is not significantly enlarged by considering it. This is because for the given position of the camera, there is no CVC ray incident on b that would enlarge the backprojection, since we have stipulated that visual compliant motion can only be robustly effected using

CM vertices. This example suggests that the enlargement of backprojections due to visual constraints is sensitive to camera placement; we will discuss this in Section VIII–B.

Frame (c) corresponds to $\theta = +\pi/6$. This is similar to the case $\theta = 0$; the backprojection is significantly enlarged by the CVC ray y. It is interesting to note that the backprojection seems to be enlarged the most when the orientations of CVC rays are most nearly perpendicular to the commanded velocity direction (and therefore the free edges of the velocity uncertainty cone). When this occurs, the free edges and CVC rays "fan out" from C-obstacle vertices to enlarge the backprojection into a funnel-like region.

The commanded velocity of frame (d) is nearly identical to that of frame (c). However, the VC-enlarged backprojection is significantly different for these two cases. The reason for this, as will be shown in the next two sections, is that the commanded velocity direction in these two cases is very near a critical orientation, at which the topology of the VC-enlarged backprojection changes in a qualitative way.

VI. The Nondirectional Backprojection

The backprojection algorithm presented in Section V computes a directional backprojection relative to a specific commanded velocity. A complete planner should consider all possible commanded velocities at each iteration of the backchaining algorithm. This can be achieved by considering the nondirectional backprojection $B(G)$, which is defined as the union of all directional backprojections together with their respective velocity directions:

$$B(G) = \bigcup_{\theta} (B_\theta(G) \times \{\theta\}). \tag{3}$$

Donald has shown that the topology of the directional backprojection changes only at a finite set of critical velocity orientations, $\theta \in S^1$ [12]. Therefore, the nondirectional backprojection can be represented by a finite set of directional backprojections; one for each critical orientation, and one for each noncritical interval. In this section, we review critical orientations, and the time complexity of computing the traditional nondirectional backprojection, i.e., the backprojection without visual constraints.

A. Critical Orientations

Critical orientations occur under the following three conditions [12].

1) A free edge becomes parallel to an edge in the obstacles' visibility graph. To see this, notice that a free edge erected at some obstacle vertex b_0 will rotate with θ and may eventually rotate to an angle θ_1 at which it intersects another obstacle vertex b_1. When the ray rotates beyond θ_1, it will be truncated by the obstacle edge incident on b_1, and part of that obstacle edge may be included in the backprojection. Given this argument, note that the critical angle θ_1 occurs exactly when the free edge coincides with the visibility-graph edge connecting b_0 to b_1. Hence, such orientations are called *v-graph critical.*

2) An obstacle edge changes from a sliding into a sticking edge or vice versa. This occurs when a free edge of the velocity uncertainty cone is parallel or antiparallel to an edge of the friction cone. These orientations are called *sliding-critical.*

3) The intersection point of two free edges of the backprojection intersects an obstacle edge. Since the free edges rotate with θ, so do the backprojection vertices formed by their intersections. When any such vertex intersects an obstacle edge, one of the free edges incident on that vertex disappears, to be replaced by the obstacle edge. These are called *vertex-critical* orientations.

Donald presents an algorithm for computing these critical orientations. He then shows that the nondirectional backprojection may be represented by a finite set of directional backprojections: one directional backprojection for each critical orientation, and one representative directional backprojection for each noncritical interval (where the value of θ at which the backprojection is computed may be chosen arbitrarily).

Since the representative directional backprojection inside a noncritical interval may be computed for an arbitrary value of θ in that interval, it is possible that the algorithm will fail to compute a directional backprojection that entirely contains the polygonal start region R. To avoid this problem, Donald [14] suggests adding R to the arrangement of polygons, thus adding the following critical orientation criterion.

4) An edge of R intersects a free edge of the backprojection. These orientations are called *R-critical.*

For an input of n C-obstacle vertices, R has a constant number of edges and there are $O(n)$ free edges bounding the backprojection. Therefore there are $O(n)$ R-critical orientations. If the directional backprojection for some R-critical orientation θ_i contains all the vertices of R, then a commanded motion from R with velocity $\vec{v}_{\theta_i}$ will reach the goal.

B. Time Complexity

Although Donald shows that there are $O(n^2)$ critical orientations of type 3), he proposes a naive $O(n^3)$ algorithm to compute them, as follows. There are $O(n)$ free edges, and therefore $O(n^2)$ possible intersections of free edges. These intersections are free-space vertices of the directional backprojection that trace out circles as the velocity orientation is changed. Each such circle may intersect $O(n)$ obstacle edges. Therefore, the number of intersections of circles with obstacle edges if $O(n^3)$. The $O(n^2)$ critical orientations are contained in this set of size $O(n^3)$.

The motivation for this algorithm is that, of all possible $O(n)$ free-space backprojection vertices, the subset of these that will contribute to the critical orientations is not known in advance; however, if *all* intersections of possible free-space vertices with obstacle edges are computed in advance, this set is guaranteed to contain all of those that will contribute to critical orientations.

Donald's critical-slice algorithm recomputes the backprojection from scratch at each critical orientation and inside each noncritical interval. Recently, Briggs has presented an algorithm that incrementally computes the nondirectional backprojection, achieving an $O(n^2 \log n)$ time complexity [4], [5]. The reduced complexity is due in part to an amortized analysis that shows there are at most $O(n^2)$ topological changes to the

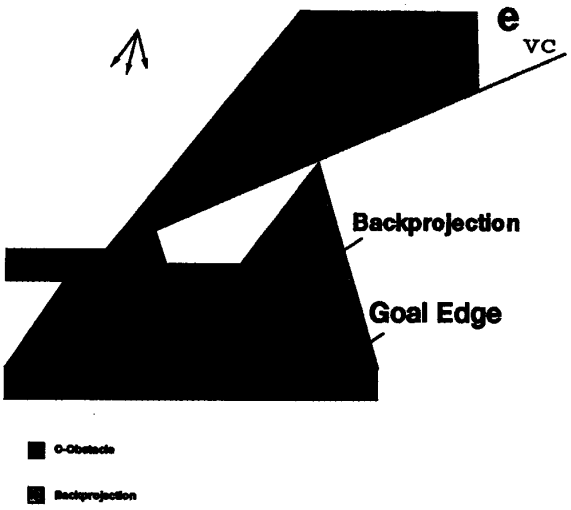

Fig. 14. A free-edge-critical orientation, just before an intersection of a free vertex with a CVC ray.

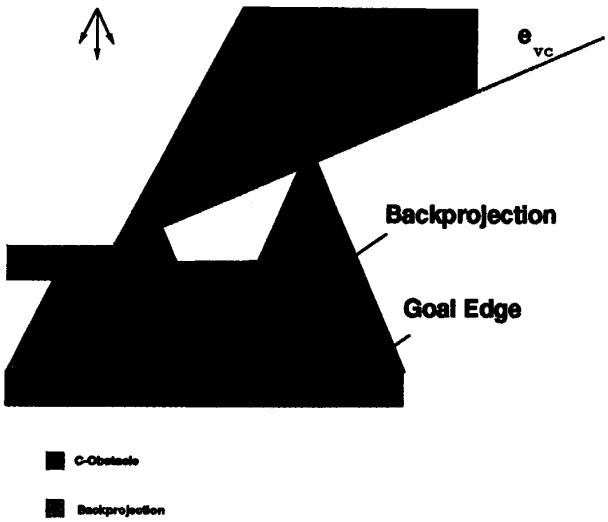

Fig. 15. A free-edge-critical orientation, just after the intersection of a free vertex with a CVC ray.

boundary of the backprojection over the entire range of θ. The algorithm also uses a dynamic data structure to keep track of the rotating free-space vertices, rather than computing all possible free-space vertices in advance.

VII. The Effect of Visual Constraints on the Nondirectional Backprojection

In this section, we describe how the introduction of visual constraints affects the computation of the nondirectional backprojection. In particular, we discuss the new critical orientations that result from the introduction of visual constraints, and the time complexity of a modified nondirectional backprojection algorithm.

According to the procedure outlined in Section V, the decision of whether to continue the backprojection along a CVC ray from a given vertex event depends, among other things, on whether incident C-obstacle edge e_i is a sliding or a sticking edge. Sliding versus sticking behavior changes only at sliding-critical orientations [12], so these orientations are also critical for VC-enlarged backprojections.

The introduction of visual constraints also adds two new criteria for critical orientations. The first is analogous to Donald's vertex-critical criterion, and the second to his vgraph-critical criterion.

A. Free-Edge-Critical Orientations

Suppose f_i is a vertex of the backprojection formed by the intersection of two rays of the inverted velocity uncertainty cone. As the commanded velocity direction θ varies, f_i moves along a circular arc. A critical orientation occurs when this circular arc intersects a CVC ray, since the decision of which of the free edges or CVC ray should be used to continue the backprojection may change. This is illustrated in Fig. 14 and 15. By analogy to Donald's v-graph critical orientations, we express this new critical orientation as follows.

5) A free-space vertex of the backprojction intersects a CVC ray. We call such orientations *free-edge-critical.*

Proposition 2: There are $O(n^2)$ free-edge-critical orientations.

Proof: We showed that the $O(n)$ workspace obstacle vertices give rise to $O(n)$ CVC rays. Donald shows that there are $O(n^2)$ vertex-critical orientations resulting from the intersection of free vertices with $O(n)$ obstacle edges. The same argument applies by treating the $O(n)$ CVC rays as obstacle edges. □

Of course, it is not always the case that the backprojection topology changes at free-edge-critical orientations, as illustrated in Figs. 11 and 12.

B. VC-Critical Orientations

Before describing the second critical orientation criterion added by CVC rays, we note the conditions from which it follows directly:

- The visibility of a vertex does not change with the commanded velocity direction θ, since the workspace obstacles and camera are fixed. Therefore the workspace VC rays do not change with θ.
- Consequently, the C-space representation of the VC rays does not change with θ, since CVC rays are constructed from workspace VC rays by considering only the vectors joining adjacent robot vertices. Since the robot cannot rotate, these vectors never change.
- With respect to the directional backprojection, CVC rays behave as if they were nonsticking obstacle edges terminating at the camera focal center, since they do not move and sticking can never occur on them.

With these statements in mind, we note the second new criterion for critical orientations added by CVC rays, constructed by analogy to Donald's condition 3):

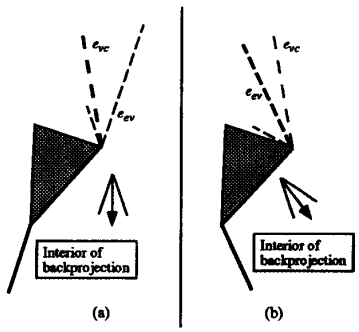

Fig. 16. How the backprojection changes across a VC-critical orientation.

6) A free edge becomes parallel to a CVC ray. At such an orientation, the decision of whether to add the free edge or the CVC ray to the backprojection may change. We will call such orientations *VC-critical.*

Fig. 16 shows how the backprojection changes across such a critical orientation.

Proposition 3: There are $O(n)$ VC-critical orientations.

Proof: As was shown in Section VII-A, there are $O(n)$ CVC rays in an environment that contains n C-obstacle vertices. Each of these introduces two critical orientations of the type previously described, given by the two free edges of the velocity uncertainty cone. Hence, there are $O(n)$ VC-critical orientations. □

C. New Asymptotic Time Bounds

If we denote by $B_{vc_\theta}(G)$ the directional backprojection with visual constraints, and by $B_{vc}(G)$ the nondirectional backprojection with visual constraints, we have

$$B_{vc}(G) = \bigcup_{\theta} (B_{vc_\theta}(G) \times \{\theta\})$$

Proposition 4: A representation of the nondirectional backprojection with visual constraints, $B_{vc}(G)$, can be computed in time $O(n^3(n+c)\log n)$, where c is the number of pairwise intersections of CVC rays.

Proof: Donald's critical-slice method [12] computes the nondirectional backprojection in time $O(n^4 \log n)$ when there are $O(n^3)$ critical orientations, by computing $O(n^3)$ slices each in time $O(n \log n)$. For the VC-enlarged backprojection, the complexity of computing a slice, $B_{vc_\theta}(G)$, is $O((n + c)\log n)$. There are $O(n)$ additional VC-critical orientations, and $O(n^2)$ additional free-edge critical orientations, but this does not asymptotically increase the total number of critical orientations since there are already $O(n^2)$ v-graph critical orientations [12]. The critical orientations can be found using Donald's proposed naive algorithm in time $O(n^3)$. Hence, the nondirectional backprojection with visual constraints can be computed in time $0(n^3(n + c)\log n)$. □

Conjecture 1: A representation of the nondirectional backprojection with visual constraints, $B_{vc}(G)$, can be computed in time $O(n^2 \log n)$.

Rationale: Using an amortization techniques, Briggs has shown that for the nondirectional backprojection (without visual constraints) there are at most $O(n^2)$ changes to the topology of the backprojection over all values of θ [4], [5]. This same analysis should apply to backprojections that include visual constraints, since there are only $O(n^2)$ additional critical orientations due to visual constraints. Brigg's algorithm [4], [5] computes the nondirectional backprojection in $O(n^2 \log n)$ time when there are $O(n^2)$ critical orientations. We believe that it should be possible to extend this algorithm to compute $B_{vc}(G)$ in time $O(n^2 \log n)$. It remains to provide a constructive proof of this conjecture. □

VIII. Discussion

In this section, we discuss a number of issues related to exploiting visual constraints in the computation of backprojections. We first address the impact of visual constraints on the number of steps required for a successful plan. Following this, we discuss the effects of camera placement on backprojections. Finally, we discuss extending our algorithms to the case of $C = \boldsymbol{R}^3$.

A. Single-Step Versus Multistep Plans

The directional backprojection with visual constraints $B_{vc_\theta}(G)$ is always at least as large as the directional backprojection $B_\theta(G)$ as defined by Erdmann [15], i.e., $B_\theta(G) \subseteq B_{vc_\theta}(G)$ (and in many cases the inclusion relation is proper). This is illustrated in the examples of Fig. 13. Let G be a goal region, and R be a start region in C_{valid}. A single-step plan is possible when R is contained entirely in $B_\theta(G)$ for some θ. Thus when $R \subseteq B_{vc_\theta}(G)$ and $R \nsubseteq B_\theta(G)$, single-step plans are possible when visual constraints are exploited, but not when they are ignored. This can be seen by noting the difference between the VC-enlarged and traditional directional backprojections in Figs. 13(a) and (c).

B. The Effects of Camera Placement

The examples in Fig. 17 illustrate the effect of moving the camera within the environment. In each set of figures, the commanded velocity is held constant while the camera assumes a range of positions from the upper right to the upper left of the environment.

The set of frames shown in Fig. 17(a) corresponds to commanded velocity direction $\theta = -\pi/6$. The square robot is shown in frame 1. Frame 1 is similar to Fig. 13(b). Frames 2 and 3 differ in the location of the camera.

The set of frames shown in Fig. 17(b) corresponds to $\theta = 0$. In frame 1, the region R is not included, because motions commencing in it, even if they visually comply to the CVC rays bounding R, may stick on the horizontal top edge of the obstacle. The remaining regions are bounded mostly by CVC rays. Note the inclusion of the vertex f. If visual compliant motion begins along the CVC ray incident on f, it may safely continue and intersect the (sliding) edge E. On the other hand, if motion commences from a point below f, uncertainty cone constraints allow the inclusion of sliding edge E. If the CVC ray incident on f did not intersect E, that CVC ray would not have been included in the backprojection, as was discussed in Section V.

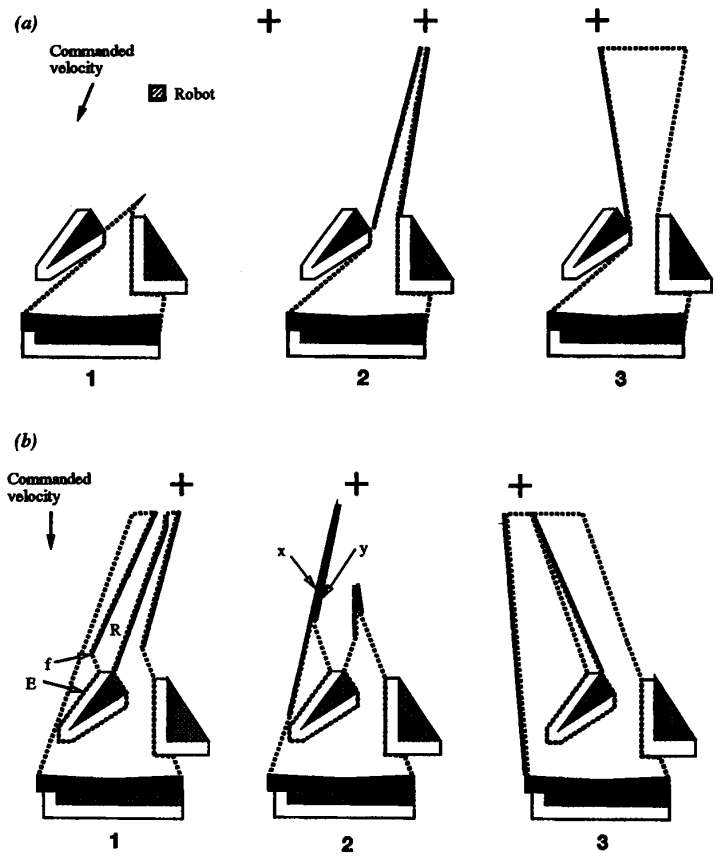

Fig. 17. Effect of moving the camera within the environment.

Combinations of CM-vertex constraints and close-together CVC rays arising from different workspace VC rays (x and y) give the backprojection of frame 2. Finally, frame 3 illustrates a situation similar to that of frame 1, except that, in this case, the excluded region of the backprojection is bounded by a CVC ray on the right and by a free edge of the velocity uncertainty cone on the left.

1) Camera Placement and Aspect Graphs: The examples of Fig. 17 suggest that the topology of the VC-enlarged backprojection is sensitive to camera position, since CM vertices are computed by considering the relative positions of the camera and the robot. The problem of where to place the camera to best exploit visual constraints appears to be closely related to the problem of computing aspect graphs for an arrangement of polyhedra. The notion of an aspect graph was originally introduced by Koenderink and van Doorn [24]. Constructing the 3D aspect graph involves decomposing the viewing space into cells such that moving the camera within a cell does not change the qualitative topological structure of the projected image of the polyhedron, and then characterizing the connectivity among these cells as the viewpoint is moved from one cell to another. The 2D aspect graph is analogous: it divides the plane into viewing cells inside each of which the set of vertices seen by the camera does not change, and characterizes the connectivity among these cells.

The set of CVC rays that can be used to enlarge the backprojection depends on the geometry of the robot as well as the camera position, since the spatial relationship between the robot and the camera determines which robot vertices are CM vertices. Although a full treatment of this topic is beyond the scope of this paper, we expect there to be *viewing-critical configurations* and *noncritical viewing regions*, such that moving the camera within a noncritical viewing region does not alter the qualitative structure of the VC-enlarged directional backprojection.

Conjecture 2: There exists a representation of size $O((mn)^2)$ of the noncritical viewing regions.

Rationale: It is known that in the 2D aspect graph of an arrangement of polygons, the noncritical regions are bounded by the supporting lines of the edges of the polygons. Since the m-vertex robot must be considered part of the arrangement of polygons for the purpose of computing CM vertices, and computation of CM vertices involves making each vertex of the robot coincident with each visible vertex of a polygon, there are effectively $O(mn)$ polygon vertices (and therefore edges) to consider. These define $O((mn)^2)$ intersections, which are the vertices of the noncritical regions. □

More thorough investigation of this issue is the subject of future research. Aspect graph computation is a classic problem in computer vision, and there is a large body of work on

computaton of aspect graphs not only for arbitrary polyhedra, but for several classes of curved solids as well; see, for example, [19] and [33]. We expect that future work in visual-constraint-based planning will benefit from the application of aspect graph techniques to the problem of optimum camera placement.

C. Backprojections for $C = \boldsymbol{R}^3$

Here, we briefly describe a number of the difficulties that are encountered when extending our algorithms to the 3D case. The reader should note these difficulties are inherent in the computation of 3D backprojections, and are not introduced by considering visual constraints. In fact, we will show informally that considering visual constraints does not make the 3D backprojection problem computationally harder.

1) 3D Velocity Uncertainty Cones: One of the constraints that makes a 2D implementation computationally attractive is the restriction that the robot and C-obstacles be polygonal, resulting in a backprojection bounded by straight line segments. In particular, the necessary intersections of line segments are easy to compute. Velocity uncertainty is simply bounded by a 2D uncertainty cone, which encloses all possible trajectories from a starting configuration.

However, in three dimensions, the velocity uncertainty cones are 3D cones with curved surfaces. Recall that in the absence of a sliding surface, the backprojection is bounded by the surfaces of uncertainty cones. Consider backprojection from a rectangular obstacle face with no other objects in the environment. According to Erdmann's algorithm [15], we erect the inverted velocity uncertainty cone along each sticking edge, and trace the backprojection bounded by the intersections of the cones. However, these intersections are no longer necessarily straight lines, nor are the backprojection boundaries planar. While the backprojection may still in theory be constructed this way, it is less computationally attractive because of the more difficult geometry.

2) Inscribed Ellipses: Another possible approach is to approximate the goal surface by its maximal inscribed ellipse. This is motivated by observing that under the uncertainty cone model, a motion may follow any trajectory inside a cone. The base of the cone is a circle in the plane perpendicular to the commanded motion direction. In general, it projects onto an arbitrary planar surface as an ellipse, the ratio of whose axes is determined by the angles at which the target plane intersects the uncertainty cone. Let us refer to such an ellipse as the *inscribed ellipse* of a goal surface. Following this approach, we might construct the backprojection of an inscribed ellipse, rather than the backprojection of the polygonal target surface. However, the resulting planner would be incomplete, as it would fail to include in the backprojection some points from which a commanded motion would reach a point on the target surface but not contained in the target surface's inscribed ellipse.

3) Critical Cell Decomposition: As discussed in Section VI, Donald exploits the polygonal structure of the 2D backprojection to derive a critical-slice method for decomposing the velocity direction space S^1 using a finite number of critical orientations and noncritical intervals. In three dimensions, two angles are necessary to specify a commanded motion direction, say ϕ and θ using the convention of spherical coordinates. Thus, the space of commanded motion directions is $J = [0, \pi) \times [0, 2\pi)$. To apply Donald's technique here, J must be decomposed into cells, inside each of which the topology of the backprojection does not change. The boundaries of the cells define critical orientations.

However, since the backprojection is no longer polyhedral, determining when changes in topology occur is considerably more difficult. Furthermore, although in two dimensions a backprojection polygon is closed by the intersection of two rays, a 3D backprojection volume is, in general, *not* closed by the intersection of planes, so it is not obvious exactly how the topology changes across adjacent critical cells. We speculate that there exists an algebraic representation of the critical orientation criteria, in which case, an algebraic cell decomposition [3], [11] could be used to determine the critical cell boundaries.

4) Computational Complexity: Without a complete computational complexity analysis, we make the following conjecture about the complexity of considering visual constraints in three dimensions.

Conjecture 3: Considering visual constraint surfaces in the 3D directional backprojection does not increase the asymptotic time complexity of computing it.

Rationale: The operations necessary to support VC surfaces are also necessary for supporting the basic algorithm. Ruled VC surfaces behave like frictionless obstacle surfaces in that they do not change with motion direction, and like uncertainty cone surfaces in that they are free constraint surfaces not supported by a physical object surface. □

IX. Conclusions

In this paper, we have introduced visual constraint surfaces as a mechanism to effectively exploit visual constraints in the synthesis of uncertainty-tolerant robot motion plans. Visual constraint surfaces can be used to effect visual guarded and visual compliant motions. By deriving a configuration space representation of visual constraint surfaces, we were able to include visual constraint surfaces as boundaries of the directional backprojection. We described an implemented backprojection planner for $C = \boldsymbol{R}^2$ based on Donald and Canny's algorithm [12].

By examining the effects of visual constraints as a function of the direction of the commanded velocity, we were able to determine new criteria for critical orientations, i.e., orientations at which the topology of the directional backprojection, including visual constraint surfaces, might change. We presented an algorithm to compute the nondirectional backprojection modified to include visual constraint surfaces.

Finally, we have discussed a number of issues that are related to the inclusion of visual constraint surfaces in backprojections, including multistep versus single-step plans, the problem of optimal camera placement, and extending the backprojection algorithm to the 3D case.

ACKNOWLEDGMENT

The authors are grateful to the anonymous reviewers, who provided many constructive comments that greatly improved the clarity of the presentation, and to Michael Barbehenn and Jean Ponce for their helpful comments on an earlier draft of this paper.

REFERENCES

[1] J. E. Agapakis, J. M. Katz, J. M. Friedman, and G. N. Epstein, "Vision-aided robotic welding: An approach and a flexible implementation." *Int. J. Robotics Res.*, vol. 9, no. 5, pp. 17–33, Oct. 1990.

[2] P. Allen, B. Yoshimi, and A. Timcenko, "Real-time visual servoing," in *Proc. IEEE Int. Conf. Robotics Automat.*, 1991, pp. 851–856.

[3] D. S. Arnon, "Geometric reasoning with logic and algebra," *Artificial Intell.*, vol. 37, nos. 1–3, pp. 37–60, Dec. 1988.

[4] A. J. Briggs, "An efficient algorithm for one-step planar compliant motion planning with uncertainty." in *Proc. ACM Annu. Symp. Computat. Geometry*, 1989.

[5] ——, "An efficient algorithm for one-step planar compliant motion planning with uncertainty," *Algorithmica*, vol. 8, pp. 195–208, 1992.

[6] R. A. Brooks, "Symbolic error analysis and robot planning," *Int. J. Robotics Res.*, vol. 1, no. 4, Winter 1982.

[7] J. F. Canny, "On computability of fine motion plans," in *Proc. IEEE Int. Conf. Robotics Automat.*, 1989, pp. 177–182.

[8] A. Castaño, "Resolved-rate hybrid vision/position servo control of a robotic manipulator," M.S. thesis, University of Illinois at Urbana-Champaign, 1992.

[9] A. Castaño and S. A. Hutchinson, "Visual compliance: Task-directed visual servo control," *IEEE Trans. Robotics Automat.*, vol. 10, no. 3, pp. 334–342, June 1994.

[10] W. F. Clocksin, J. S. E. Bromley, P. G. Davey, A. R. Vidler, and C. G. Morgan, "An implementation of model-based visual feedback for robot arc welding of thin sheet steel," *Int. J. Robotics Res.*, vol. 4, no. 1, pp. 13–26, Spring 1985.

[11] G. E. Collins, "Quantifier elimination for real closed fields by cylindrical algebraic decomposition," in *Lecture Notes in Computer Science, Vol. 33*. New York: Springer-Verlag, 1975, pp. 135–183.

[12] B. R. Donald, "Error detection and recovery for robot motion planning with uncertainty," Ph.D. dissertation, M.I.T., Cambridge, MA, 1987.

[13] ——, "A geometric approach to error detection and recovery for robot motion planning with uncertainty. *Artificial Intell.*, vol. 37, nos. 1–3, pp. 223–271, Dec. 1988.

[14] ——, "Planning multi-step error detection and recovery strategies," *Int. J. Robotics Res.*, vol. 9, no. 1, pp. 3–60, Feb. 1990.

[15] M. Erdmann, "On motion planning with uncertainty." M.S. thesis, M.I.T., 1986.

[16] ——, "Using backprojections for fine motion planning with uncertainty," *Int. J. Robotics Res.*, vol. 5, no. 1, pp. 19–45, Spring 1986.

[17] I. D. Faux and M. J. Pratt, *Computational Geometry for Design and Manufacture.* Chichester, England: Ellis Horwood, 1985.

[18] J. T. Feddema and O. R. Mitchell, "Vision-guided servoing with feature-based trajectory generation," *IEEE Trans. Robotics Automat.*, vol. 5, no. 5, pp. 691–700, Oct. 1989.

[19] Z. Gigus, J. Canny, and R. Seidel, "Efficiently computing and representing aspect graphs of polyhedral objects," *Int. J. Robotics Res.*, vol. 13, no. 6, pp. 542–551, June 1991.

[20] B. K. P. Horn, *Robot Vision.* Cambridge, MA: M.I.T. Press, 1986.

[21] S. A. Hutchinson, "Exploiting visual constraints in robot motion planning," in *Proc. IEEE Int. Conf. Robotics Automat.*, 1991.

[22] S. A. Hutchinson and A. C. Kak, "SPAR: A planner that satisfies operational and geometric goals in uncertain environments," *AI Mag.*, vol. 2, no. 1, pp. 30–61, Spring 1990.

[23] P. K. Khosla, C. P. Neuman, and F. B. Prinz, "An algorithm for seam tracking applications," *Int. J. Robotics Res.*, vol. 4, no. 1, pp. 27–41, Spring 1985.

[24] J. J. Koenderink and A. J. Van Doorn, "The internal representation of solid shape with respect to vision," *Biological Cybern.*, vol. 32, pp. 211–216, 1979.

[25] J. C. Latombe, *Robot Motion Planning.* Boston: Kluwer Academic, 1991.

[26] J. C. Latombe, A. Lazanas, and S. Shekhar, "Robot motion planning with uncertainty in control and sensing," *Artificial Intell.*, vol. 52, pp. 1–47, 1991.

[27] T. Loazno–Pérez, M. T. Mason, and R. H. Taylor, "Automatic synthesis of fine-motion strategies for robots," *Int. J. Robotics Res.*, vol. 3, no. 1, pp. 3–24, Spring 1984.

[28] M. T. Mason, "Compliance and force control for computer controlled manipulators," in *Robot Motion: Planning and Control,* B. Brady, J. M. Hollerbach, T. L. Johnson, T. Lozano-Pérez, and M. T. Mason, Eds. Cambridge, MA: M.I.T. Press, 1982, pp. 373–404.

[29] ——, "Automatic planning of fine motions: Correctness and completeness," in *Proc. IEEE Int. Conf. Robotics*, 1984, pp. 492–503.

[30] N. Papanikolopoulos, P. K. Khosla, and T. Kanade, "Vision and control techniques for robotic visual tracking," in *IEEE Int. Conf. Robotics Automat.*, pp. 857–864, 1991.

[31] R. P. Paul and B. Shimano, "Compliance and control," in *Proc. Joint Amer. Automat. Cont. Conf.*, 1976, pp. 694–1699.

[32] J. Pertin–Troccaz and P. Puget, "Dealing with uncertainties in robot planning using program proving techniques," in *Proc. Fourth Int. Symp. Robotic Res.*, 1987.

[33] J. Ponce and D. J. Kriegman, "Computing exact aspect graphs of curved objects: Parametric patches," in *Proc. Amer. Assoc. Artificial Intell.*, July 1990.

[34] F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction.* New York: Springer-Verlag, 1985.

[35] M. H. Raibert and J. J. Craig, "Hybrid position/force control of manipulators," *J. Dynamic Syst., Measure. Cont.*, vol. 102, pp. 126–133, June 1981.

[36] P. Saraga and B. M. Jones, "Simple assembly under visual control," in *Robot Vision,* Alan Pugh, Ed. U.K.: IFS, 1983, pp. 209–223.

[37] Y. Shirai and H. Inoue, "Guiding a robot by visual feedback in assembling tasks," *Patt. Recognition*, vol. 5, pp. 99–108, 1973.

[38] S. B. Skaar, W. H. Brockman, and R. Hanson, "Camera-space manipulation," *Int. J. Robotics Res.*, vol. 6, no. 4, pp. 20–32, Winter 1987.

[39] R. H. Taylor, 'The synthesis of manipulator control programs from task-level specifications," Report AIM-282, Stanford Artificial Intelligence Lab., 1976.

[40] L. E. Weiss, A. C. Sanderson, and C. P. Neuman, "Dynamic sensor-based control of robots with visual feedback," *IEEE J. Robotics Automat.*, vol. 3, no. 5, pp. 404–417, Oct. 1987.

[41] D. E. Whitney, "Force feedback control of manipulator fine motions," *J. Dynam. Syst., Measure. Cont.*, pp. 91–97, June 1977.

[42] P. M. Will and D. D. Grossman, "An experimental system for computer controlled mechanical assembly," *IEEE Trans. Comput.*, vol. C-24, no. 9, pp. 879–888, 1975.

Armando Fox was with the Vision/Robotics Group at the Beckman Institute, University of Illinois at Urbana-Champaign, during the course of the research reported here. He has had prior research experience with the MONSOON dataflow project at the M.I.T. Laboratory for Computer Science, and is currently working as a microprocessor architect with Intel Corp. His current research interests include high-performance and parallel computer architectures and programming environments, in which he plans to pursue a doctoral degree beginning in fall 1994.

Seth Hutchinson received the Ph.D. degree from Purdue University, West Lafayette, IN, in 1988.

He spent 1989 as a Visiting Assistant Professor of Electrical Engineering at Purdue University. In 1990, he joined the faculty at the University of Illinois in Urbana-Champaign, where he is currently an Assistant Professor in the Department of Electrical and Computer Engineering, and the Beckman Institute for Advanced Science and Technology. Dr. Hutchinson's current research interests include: integration of vision, force and position sensing for robot motion planning and control; dynamic planning of sensing strategies; constraint based reasoning; task planning for automated assembly; evidential reasoning applied to model based object recognition; and sensor integration.