Exploiting Visual Constraints in the Synthesis of Uncertainty-Tolerant Motion Plans I: The Directional Backprojection

Armando Fox Seth Hutchinson
The Beckman Institute for Advanced Science and Technology
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
Urbana, IL 61801

Abstract

In our earlier work, we have shown that 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). Here we 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. We describe an implemented backprojection planner for $C = \mathbf{R}^2$ that is based on Donald and Canny's plane-sweep algorithm for computing the directional backprojection, and discuss the effects of visual constraints on the asymptotic time complexity of the modified algorithm.

1 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).

To account for control and sensing uncertainty, Lozano-Pèrez, Mason and Taylor [13] introduced the theoretical framework of *preimage planning*. Informally, a preimage of a goal is the set of points from which a commanded motion is guaranteed to reach and terminate in the goal. Erdmann [7] derived *backprojections* as a means of usefully approximating preimages by separating goal reachability from goal recognizability. Donald's work on automatic error detection and recovery [5, 6] considers *model uncertainty* in addition to position and velocity uncertainty, and presents an extension to the preimage strategy that computes motion plans guaranteed to succeed in the presence of all three kinds of uncertainty. To date, the only sensing modalities that have been incorporated into a preimage or backprojection planner have been force and position sensing.

One limitation of force control is that it can only be used to constrain motion along directions normal to constraint surfaces (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.

One way to cope with this limitation 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. Recently, a number of researchers have begun to investigate visual servo control [1, 3, 8, 14]. 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.

The remainder of the paper is organized as follows. In Section 2 we briefly review *visual constraint surfaces*, which are generated by projecting workspace features onto the image plane of a fixed camera [11]. 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. An implemented hybrid position/vision servo control system that executes visually constrained motions is described in [3, 2]. In Section 3 we show how visual constraint surfaces in the workspace are mapped to configuration space constraint surfaces. In Section 4 we review preimages and backprojections. In Section 5 we show how the directional backprojection (i.e. the backprojection with respect to a specified velocity) can be extended to exploit visual constraint surfaces. Here we discuss our implemented backprojection algorithm that extends Donald and Canny's plane-sweep algorithm for computing backprojections [4], and evaluate the added computational complexity of considering visual constraint surfaces.

2 A Geometric Specification for Visual Constraints

Consider a workspace containing a number of solid objects and a fixed camera. If the imaging process is modelled by perspective projection [10], 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 ob-

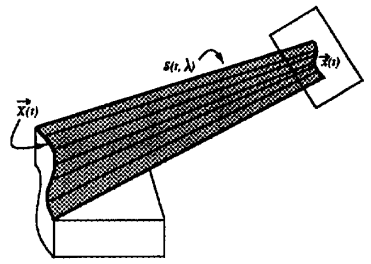

Figure 1: Ruled visual constraint surface generated by a curved contour

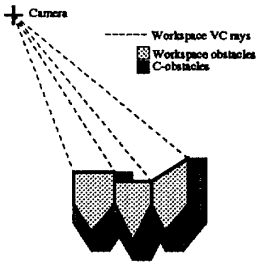

Figure 2: Construction of workspace VC rays from unoccluded obstacle vertices

ject 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.

A *visual constraint (VC) surface* is a ruled surface bounded by a 3D object edge, its corresponding image edge, and the rays joining their respective endpoints, as in Figure 1. In the special case of polyhedral obstacles, all image features will be straight line segments, so that the VC surfaces will be polygons. The relevant equations are given in [11]. Note that a visual constraint surface does not intersect any obstacles, since a necessary condition for an edge to have a projection on the camera image plane is that the edge not be occluded.

Visual compliant motion. During force-controlled compliant motion, a physical surface is used to constrain the motion of a robot along one or more degrees of freedom. 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 the VC surface. Visual compliance can be achieved by means of a closed-loop visual servo system, as described by Castaño [2].

Visual guarded motion. We define *visual guarded* motion analogously to guarded motion using physical surfaces. In the latter, the robot moves until force feedback indicates that it has contacted a physical surface. 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.

3 Computing Visual Constraint Rays for a Two Dimensional Workspace

In the case of a two dimensional 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

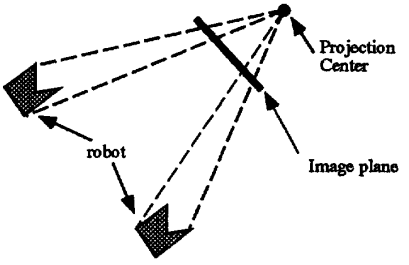

Figure 3: Different positions of the polygonal robot give different CM vertices

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 in Figure 2.

Since the robot is polygonal, 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 purposes 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 (although 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. Figure 3 shows the same robot in two different positions for which the CM vertices are different. In certain non-general 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.

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

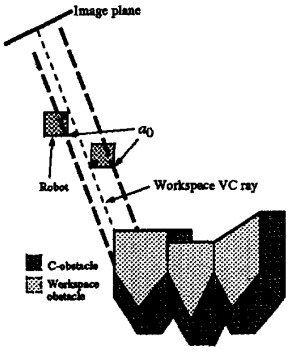

Figure 4: Construction of two C-space VC rays from a single workspace VC ray

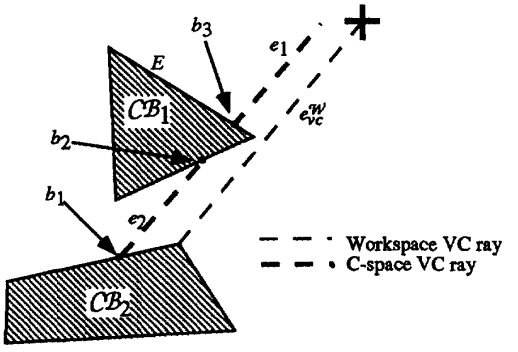

Figure 5: A CVC ray may intersect the interior of $\mathcal{CB}$.

the workspace VC ray. Suppose that a particular vertex a_0 of the robot is taken as the origin of the robot frame to compute a representation of the C-space $\mathcal{C}$. Since trajectories in $\mathcal{C}$ 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 $\mathcal{C} = \mathbf{R}^2$, 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 $\vec{a_j}(\vec{q}) = \vec{a_0}(\vec{q}) + (\vec{a_j}(0) - \vec{a_0}(0))$.

Let $e_{vc}^{\mathcal{W}}$ 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 $e_{vc}^{\mathcal{W}}$, vertex a_0 will move along a straight line trajectory parallel to $e_{vc}^{\mathcal{W}}$ but displaced from it by $\vec{a_0}(0) - \vec{a_j}(0)$. We construct a CVC ray e_{vc} in $\mathcal{C}$, whose endpoints are the endpoints of $e_{vc}^{\mathcal{W}}$ 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 $e_{vc}^{\mathcal{W}}$. Similarly, visual guarded motion of a_0 terminating on e_{vc} corresponds to visual guarded motion of a_j terminating on $e_{vc}^{\mathcal{W}}$, which is a visually observable event. The construction of CVC rays using this technique is illustrated in Figure 4. In the figure, the light dashed line represents the workspace VC ray, and the two bold dashed lines represent the corresponding CVC rays.

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 $\mathcal{CB}$, retaining only those segments of the CVC ray that lie outside of $\mathcal{CB}$. In Figure 5, the CVC ray constructed from the workspace VC ray $e_{vc}^{\mathcal{W}}$ intersects the interior of C-obstacle $\mathcal{CB}_1$ (we will use the notation $\mathcal{CB}_i$ to indicate a particular C-obstacle, and $\mathcal{CB}$ to represent the union of all C-obstacles). That part of the CVC ray that lies outside of $\mathcal{CB}$ 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.

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. Figure 6 shows one example of intersecting CVC rays. Since the CVC rays corresponding to a single workspace VC ray are parallel, the intersecting CVC rays cannot 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 $e_{vc_k}^{\mathcal{W}}$ to compliant motion of vertex $a_j, j \neq i$ along workspace VC ray $e_{vc_l}^{\mathcal{W}}, k \neq l$. 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 $e_{vc_1}^{\mathcal{W}}$, to compliant motion of its bottom left vertex along $e_{vc_2}^{\mathcal{W}}$.

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 above. The details of this process are described in [9].

4 Preimages and Backprojections

4.1 Preimage Planning

Lozano-Pérez, Mason and Taylor [13] present a formalism for the automatic synthesis of fine-motion strategies using *preimages*. The main advantage to the preimage formalism is that it allows the fine-motion planner to explicitly consider uncertainties in position and control.

In [13], position uncertainty is modeled by an error ball, $B_{\epsilon p}(p)$, 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

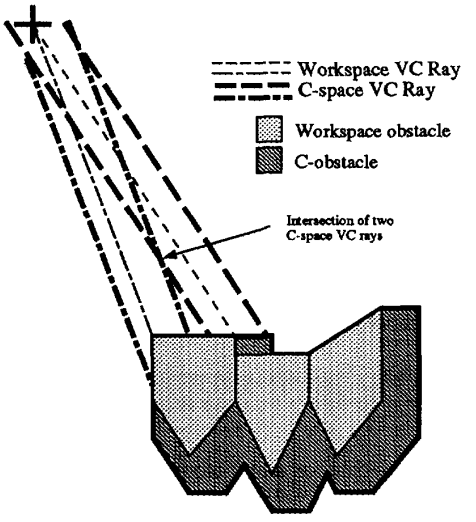

Figure 6: Intersection of two CVC rays

similar definition holds for measured vs. 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* of the velocity uncertainty cone. A free edge is an edge that is parallel to an edge of the inverted velocity cone erected at some C-obstacle vertex.

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 C). A motion command $M = (\vec{v}_\theta, TC)$, consists of a commanded velocity $\vec{v}_\theta$ (which is a 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, $R \subseteq C_{valid}$, such that if M commences from any point in R, TC will eventually return True and the motion will terminate in G. A *maximal directional preimage* is the largest possible preimage relative to a given motion direction and goal region.

4.2 Backprojections

A major difficulty of 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 [7] as a means of approximating preimages. Essentially, a backprojection is a preimage without a termination predicate; that 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 straightforward to compute, and they can usefully approximate preimages for certain types of planning problems. The backprojection approach to motion planning can be characterized as recursively finding some subset of the goal region that is guaranteed to be recognized [12, 7], and constructing the backprojection for this region.

4.3 Computing Backprojections in $C = \mathbf{R}^2$

Donald and Canny [4] present an algorithm for computing backprojections of polygonal goal regions for the case $C = \mathbf{R}^2$. The algorithm works by sweeping a line across the plane in the direction opposite that of the commanded velocity. The 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.

Donald shows [4] 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.)

5 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 types of event that must be considered by the plane-sweep algorithm. Following this, the formal decision criteria for determining whether to include a VC ray in the backprojection boundary are presented. We then discuss the time complexity of the modified directional backprojection algorithm. Finally, we present examples of VC-enlarged backprojections for $C = \mathbf{R}^2$ computed by our implementation of the modified algorithm.

5.1 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

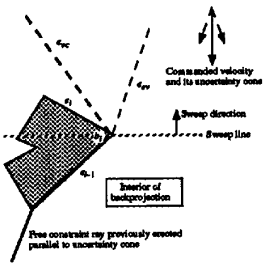

Figure 7: Deciding how to continue the backprojection from a C-obstacle vertex

will be considered in the remainder of the paper (since artificial vertices are introduced when the CVC ray intersects a C-obstacle edge).

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, $O(n^2)$ intersections of pairs 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 $e^{\mathcal{W}}_{vci}$ and $e^{\mathcal{W}}_{vcj}$. Thus, such an intersection point can be created by positioning one CM vertex of the robot on $e^{\mathcal{W}}_{vci}$, and then moving the robot compliantly along this ray until the remaining CM vertex contacts $e^{\mathcal{W}}_{vcj}$.

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.

5.2 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 above 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 Figure 7. 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 Figure 7.

The decision criteria are:

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.

5.3 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. Such a sticking edge will never be included in the backprojection, therefore neither should e_{vc}.
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, e_j should *not* be included in the backprojection since 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, e_i should be included in the backprojection, and 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.* All goal edges are in the backprojection, and 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 exactly when e_{vc} terminates on an edge already known to be in the backprojection.

Formally, the decision criteria at a vertex event at which a free edge of the velocity uncertainty cone e_{ev} and a CVC ray e_{vc} intersect is as follows. As in Section 5.2, the vector $\vec{y}$ points along the sweep line toward the interior of the backprojection.

1. 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}.
2. Otherwise, continue the backprojection along e_{ev}.

5.4 Intersection of 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}.

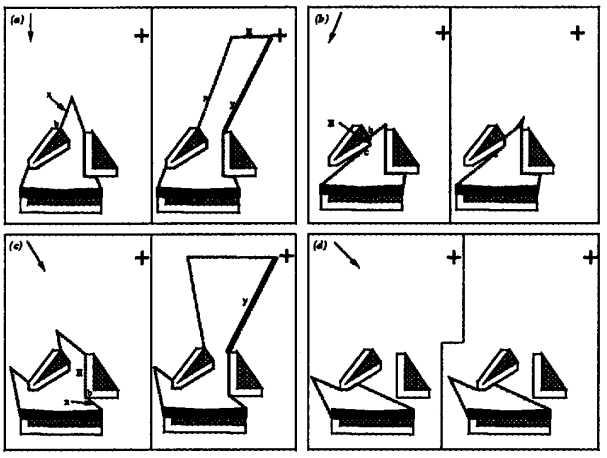

Figure 8: Effect of considering VC rays in computing the directional backprojection

5.5 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 $O(n \log n)$. During the plane-sweep, each vertex is processed in constant time. Therefore we make the following proposition, whose proof is given in [9].

Proposition 1 *The time to compute the directional backprojection with visual constraint rays, $B_{vc_\theta}(G)$, is $O((n+c)\log n)$, where c is the number of intersections of pairs of CVC rays.*

6 Results

Figures 8(a)–(d) compare the traditional directional backprojection to the VC-enlarged directional backprojection for a variety 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. The CVC rays never make the backprojection smaller, and frequently make it larger. In the figure we use the following conventions. The directional backprojection is enclosed by a dashed line, with edges contributed by visual constraints highlighted in bold. Workspace obstacles are shaded; C-obstacles are outlined. Solid arrows denote the commanded velocity direction. The camera projection center (workspace coordinates) is indicated by a cross. The goal polygon is shaded black. The commanded velocity direction $\theta = 0$ corresponds to movement straight down the page.

7 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.

References

[1] P. Allen, B. Yoshimi, and A. Timcenko. Real-time visual servoing. In *IEEE International Conference on Robotics and Automation*, pages 851–856, April 1991.

[2] A. Castaño. Resolved-rate hybrid vision/position servo control of a robotic manipulator. Master's thesis, University of Illinois at Urbana-Champaign, 1992.

[3] A. Castaño and S. A. Hutchinson. Hybrid vision/position servo control of a robotic manipulator. In *IEEE International Conference on Robotics and Automation*, pages 1264–1269, Nice, France, May 1992.

[4] B. R. Donald. *Error Detection and Recovery for Robot Motion Planning with Uncertainty*. PhD thesis, Massachusetts Institute of Technology, 1987.

[5] B. R. Donald. A geometric approach to error detection and recovery for robot motion planning with uncertainty. *Artificial Intelligence*, 37(1-3):223–271, December 1988.

[6] B. R. Donald. Planning multi-step error detection and recovery strategies. *International Journal of Robotics Research*, 9(1):3–60, February 1990.

[7] M. Erdmann. On motion planning with uncertainty. Master's thesis, Massachusetts Institute of Technology, 1986.

[8] J. T. Feddema and O. R. Mitchell. Vision-guided servoing with feature-based trajectory generation. *IEEE Transactions on Robotics and Automation*, 5(5):691–700, October 1989.

[9] A. Fox and S. Hutchinson. Exploiting visual constraints in the synthesis of uncertainty-tolerant motion plans. Technical Report UIUC-BI-AI-RCV-92-05, The Beckman Institute, University of Illinois, 1992.

[10] Berthold Klaus Paul Horn. *Robot Vision*. MIT Press, Cambridge, 1986.

[11] S. A. Hutchinson. Exploiting visual constraints in robot motion planning. In *IEEE International Conference on Robotics and Automation*, Sacramento, CA, April 1991.

[12] J.C. Latombe, A. Lazanas, and S. Shekhar. Robot motion planning with uncertainty in control and sensing. *Artificial Intelligence*, 52:1–47, 1991.

[13] T. Lozano-Pérez, M. T. Mason, and R. H. Taylor. Automatic synthesis of fine-motion strategies for robots. *International Journal of Robotics Research*, 3(1):3–24, Spring 1984.

[14] N. Papanikolopoulos, P. K. Khosla, and T. Kanade. Vision and control techniques for robotic visual tracking. In *IEEE International Conference on Robotics and Automation*, pages 857–864, April 1991.