Exploiting Visual Constraints in the Synthesis of Uncertainty-Tolerant Motion Plans II: The Nondirectional 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

The directional backprojection is one of the primary tools used by fine-motion planning systems. Informally, a directional backprojection is the set of configurations from which a commanded motion is guaranteed to reach a goal configuration. It has been shown that, for force controlled motions, the topology of the directional backprojection only changes at a finite set of critical velocity orientations.

In a related paper, we have shown how visual constraints can be exploited in the construction of directional backprojections. Here, we show how the introduction of visual constraints into the backprojection formalism affects the computation and structure of the nondirectional backprojection. Specifically, by examining the behavior of visual constraints as a function of the direction of the commanded velocity, we are able to determine the new criteria for critical velocity orientations (i.e. velocity orientations at which the topology of the directional backprojection, including visual constraint rays, might change).

1 Introduction

Real robotic systems must be able to cope effectively with uncertainties. To this end, Lozano-Pérez, Mason and Taylor [11] have introduced the *preimage* formalism for the automatic synthesis of fine-motion strategies. Informally, a directional 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 [6] has introduced *backprojections* as a means of usefully approximating preimages by separating goal reachability from goal recognizability. Thus, fine-motion planning can be reduced to recursively computing a sequence of directional backprojections. In [7], we introduced a method for exploiting visual constraints in the construction of directional backprojections.

Since directional backprojections are computed with respect to a specific commanded velocity, one difficulty that confronts a backprojection planner is the selection of an appropriate commanded velocity. The *nondirectional backprojection* is a representation of all directional backprojections together with their respective commanded velocities. As such, it provides an effective tool for selecting commanded velocities. For the two-dimensional case, Donald has shown [4] that the topology of the directional backprojection changes only at certain critical velocity orientations; in non-critical intervals between critical orientations, the topology of the directional backprojection does not change. Thus, a finite set of directional backprojections (one for each critical orientation, and one for each non-critical interval) provides a sufficient representation of the nondirectional backprojection. In this paper, we describe how incorporating visual constraints into the planning process affects the structure and computation of the nondirectional backprojection.

The remainder of the paper is organized as follows. In Section 2 we briefly review the computation of directional backprojections. Following this, in Section 3 we review the effect of visual constraints on the structure and computation of directional backprojections. In Section 4 we review nondirectional backprojections, and the criteria that are used to determine critical velocity orientations. In Section 5 we describe the effect of visual constraints on the nondirectional backprojection. This includes a discussion of new criteria for critical velocity orientations, and a discussion of the time complexity for computing nondirectional backprojections that include visual constraints. In Section 6, we discuss the extension of our algorithms to the three-dimensional case. Finally, in Section 7 we summarize the contributions of our work.

2 Directional Backprojections

The formal definition of a *directional preimage* $P_\theta(G)$ is as follows. Let G be a goal region in $\mathcal{C}_{valid}$ (where $\mathcal{C}_{valid}$ is the set of valid configurations in the configuration space $\mathcal{C}$). A motion command $M = (\vec{v}_\theta, TC)$, consists of a commanded velocity $\vec{v}_\theta$ (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, $R \subseteq \mathcal{C}_{valid}$, such that if M commences from any point in R, TC will eventually return True and the motion will terminate in G.

1050-4729/93 $3.00 © 1993 IEEE

A major difficulty of computing preimages is that, due primarily to uncertainty in sensing and control, there are circumstances under which a real termination predicate may not be able to reliably detect entry into the goal region. For this reason, Erdmann introduces backprojections [6] as a means of approximating preimages by separating the reachability and recognizability issues. Essentially, a backprojection is a preimage without a termination predicate; that is, the set of all points from which a commanded velocity is guaranteed to enter the goal, regardless of whether entry into the goal is recognized. The backprojection approach to motion planning can be characterized as recursively finding some subset of the goal region that is guaranteed to be recognized (see for example [10, 6]), and constructing the backprojection for this region, until the initial configuration is contained within a computed backprojection.

Donald and Canny [4] present a plane-sweep algorithm [12] that computes a directional backprojection, $B_\theta(G)$, in $O(n \log n)$ time, where n is the number of vertices of the C-space obstacle region. 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.

3 The Effect of Visual Constraints on the Directional Backprojection

In this section we briefly review the impact of visual constraints on the computation and structure of the directional backprojection. Details can be found in [7]. We assume here that the C-space representation of the visual constraint rays has already been constructed (C-space VC rays will be referred to as CVC rays).

When visual constraint rays are included in the directional backprojection, there are three new event types that must be considered by the plane-sweep algorithm that constructs the directional backprojection: (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 ray of an inverted velocity uncertainty cone, (3) the intersection of two CVC rays.

The decision criteria for an event corresponding to the intersection of a CVC ray and a C-obstacle vertex (or edge) is illustrated in Figure 1. Denote the intersection point by 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 inverted uncertainty cone erected at b. We assume e_{i-1} has already been added to the backprojection. 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 1.

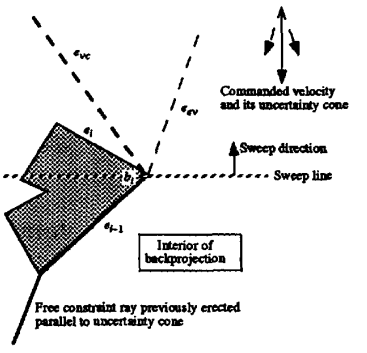

Figure 1: Deciding how to continue the backprojection from a C-obstacle vertex

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.

For the event corresponding to the intersection of a CVC ray and a free edge of the velocity uncertainty cone, the CVC ray should be used to continue the backprojection only when the termination point of the visual constraint ray on a C-obstacle edge is known to be in the backprojection.

For the event corresponding to the intersection of two CVC rays the CVC ray that results in the larger backprojection should be used to continue the backprojection.

As we have shown in [7], the asymptotic time complexity of computing the directional backprojection with visual constraints is $O((n+c) \log n)$, where c is the number of intersections of CVC rays.

4 Critical Orientations and the Nondirectional Backprojection

The backprojection algorithm described in Section 3 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:

$$B(G) = \bigcup_{\theta} (B_\theta(G) \times \{\theta\}). \qquad (1)$$

Donald has shown that the topology of the directional backprojection changes only at a finite set of critical velocity orientations, $\theta \in S^1$ [4]. Therefore, the nondirectional backprojection can be represented by a finite set of directional backprojections; one for each critical orientation, and one for each non-critical 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).

4.1 Critical Orientations

Critical orientations occur under the following three conditions [4]:

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 *vgraph-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.

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 [5] 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.

4.2 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 constraint rays, and therefore $O(n^2)$ possible intersections of free constraint rays. 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 is $O(n^3)$. 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 the ones 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, despite the fact that at most one vertex or edge changes across critical orientations. Recently, Briggs [1] presented an algorithm based on this observation that achieves an $O(n^2 \log n)$ bound for computing the nondirectional backprojection. Among other improvements, it uses a dynamic data structure to keep track of the rotating free-space vertices rather than computing all possible free-space vertices in advance.

5 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 3, the decision of whether to continue the backprojection along a CVC ray from a given vertex event depends, among other things, on whether the incident C-obstacle edge e_i is a sliding or a sticking edge. Sliding *vs.* sticking behavior changes only at sliding-critical orientations [4], 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.

5.1 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

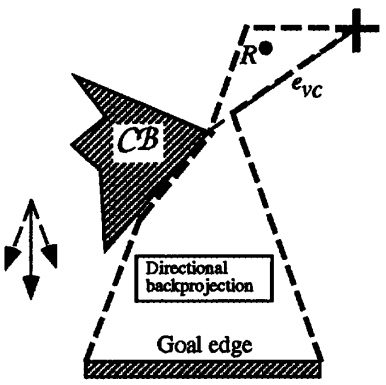

Figure 2: A free-edge-critical orientation

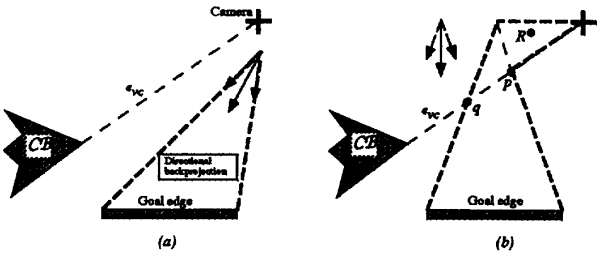

Figure 3: The backprojection does not always change at a free-edge-critical orientation

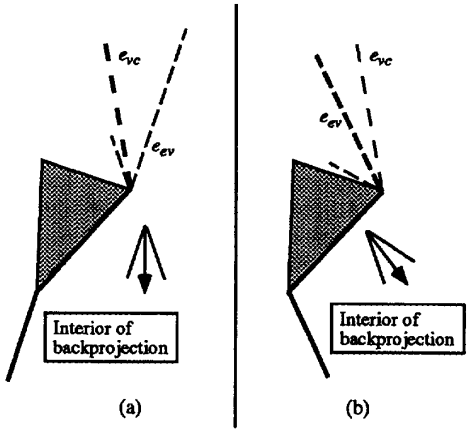

Figure 4: How the backprojection changes across a VC-critical orientation

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 Figure 2. By analogy to Donald's vgraph-critical orientations, we express this new critical orientation as follows:

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

Proposition 1 *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. Figure 3 illustrates an example in which the topology of the backprojection does not change across a free-edge-critical orientation. The bold dashed lines bound the directional backprojection, and the CVC rays are indicated by e_{VC}. Note that in Figure 3(b), the backprojection enclosed by the bold dashed lines is incorrect, as was discussed in Section 3.

5.2 VC-Critical Orientations

Before describing the second critical orientation criteria 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):

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.*

Figure 4 shows how the backprojection changes across such a critical orientation.

Proposition 2 *There are $O(n)$ VC-critical orientations.*

Proof: As was shown in Section 5.1, 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 described above, given by the two free edges of the velocity uncertainty cone. Hence there are $O(n)$ VC-critical orientations. □

5.3 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 3 *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 intersections of CVC rays.*

Proof: Donald's critical-slice method [4] 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 $O(n^2)$ additional VC-critical orientations, but this does not asymptotically increase the total number of critical orientations since there are already $O(n^2)$ vgraph-critical orientations [4]. 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 $O(n^3(n+c)\log n)$. □

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: Briggs's algorithm [1] computes the nondirectional backprojection in $O(n^2 \log n)$ time when there are $O(n^2)$ critical orientations. Therefore it should be possible to extend Briggs's algorithm to compute $B_{vc}(G)$ in time $O(n^2 \log n)$. It remains to provide a constructive proof of this conjecture. □

6 Backprojections for $\mathcal{C} = \mathbf{R}^3$

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

6.1 Critical cell decomposition

As discussed in Section 4, Donald exploits the polygonal structure of the two dimensional 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 $J = \phi \times \theta = [0, \pi) \times [0, 2\pi)$. 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 three dimensional backprojection volume is in general *not* closed by the intersection of two planes, so it is not obvious exactly how the topology changes across adjacent critical cells. If rotation is prohibited, we speculate that there exists an algebraic representation of the critical orientation criteria, in which case a doubly-exponential exact cell decomposition [3] could be used to determine the critical cell boundaries.

6.2 Velocity space discretization

There is also the possibility of discretizing velocity space (approximate cell decomposition), as originally suggested by Erdmann for two dimensional nondirectional backprojections. However, in addition to the usual resolution problems, the sheer number of cells would result in having to compute so many slices that this alternative is not attractive. In addition, consider that Donald's critical-slice algorithm was improved by Briggs by exploiting the fact that only one edge or vertex is added to or removed from the directional backprojection at each critical orientation. This is not necessarily the case when approximate cell decomposition is used to partition velocity space, so we cannot expect that geometric analysis will yield a method of incrementally constructing each backprojection slice from the previous one, as Briggs was able to do.

6.3 Computational complexity

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

Conjecture 2 *Considering visual constraint surfaces in the three dimensional directional backprojection does not increase the asymptotic time complexity of computing it.*

Rationale: The operations necessary to support visual constraint rays are also necessary for supporting the basic algorithm. Visual constraint rays 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. In particular, if we

consider polyhedral obstacles only, intersecting planar visual constraint rays with obstacle faces and existing free constraint surfaces is no harder than intersecting circular uncertainty cones with those same surfaces. □

7 Conclusions

By examining the behavior 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, and examined how the modifications to the algorithm affect its asymptotic time complexity. We have also addressed the problem of extending the backprojection algorithm to the three-dimensional case.

References

[1] A. J. Briggs. An efficient algorithm for one-step planar compliant motion planning with uncertainty. In *ACM Annual Symposium on Computational Geometry*, Saarbruchen, West Germany, June 1989.

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

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

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

[7] A. Fox. Exploiting visual geometric constraints in robot motion planning with uncertainty. Master's thesis, University of Illinois at Urbana-Champaign, Dept. of Electrical and Computer Engineering, 1992.

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

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

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

[11] 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.

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