An Efficient Algorithm for One-Step Planar Compliant Motion Planning With Uncertainty*

Amy J. Briggs†

Department of Computer Science
Upson Hall
Cornell University
Ithaca, New York 14853

May 1990

Abstract

Uncertainty in the execution of robot motion plans must be accounted for in the geometric computations from which plans are obtained, especially in the case where position sensing is inaccurate. We give an $O(n^2 \log n)$ algorithm to find a single commanded motion direction that will guarantee a successful motion in the plane from a specified start to a specified goal whenever such a one-step motion is possible. The plans account for uncertainty in the start position and in robot control, and anticipate that the robot may stick on or slide along obstacle surfaces with which it comes in contact. This bound improves on the best previous bound by a quadratic factor, and is achieved in part by a new analysis of the geometric complexity of the backprojection of the goal as a function of commanded motion direction.

*A preliminary version of this paper appears in *Proc. ACM Symposium on Computational Geometry, Saarbrücken, West Germany,* June 1989.

†This paper describes research done in the Computer Science Robotics Laboratory at Cornell University. Support for our robotics research is provided in part by the National Science Foundation under Grant IRI-8802390 and by a Presidential Young Investigator award, and in part by the Mathematical Sciences Institute.

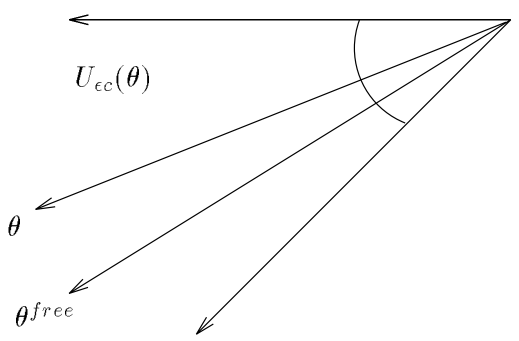

Figure 1: Uncertainty cone $U_{\epsilon c}(\theta)$. The actual direction θ^{free}, chosen by the control system, may be any direction in the cone.

1 Introduction

Motion plans for real robots must account for errors during execution. Consequently, given bounds on errors in sensing and control, we would like to plan motions that are guaranteed to succeed even in the worst case. *Uncertainty* as to control fundamentally changes the complexity of motion planning and the techniques employed. The introduction of uncertainty leads naturally and necessarily to allowing the robot to contact and comply with obstacle surfaces, because doing so greatly enriches the set of problems that can be solved.

We address the concrete and basic problem of finding a single commanded motion direction to maneuver a point robot from an uncertain start position in the plane to a specified goal where the robot is guaranteed to stop. Motions are strictly translational; thus the problem, for example, of modeling the geometric interactions of a peg and a hole can be reduced to navigating a point in configuration space [Loz83]. As in the classical motion planning problem, we assume the problem is posed in an environment of planar polygonal obstacles that is known and can be modeled exactly. The realization of a command, however, is subject to uncertainty since robots have imprecise sensing and imperfect control and therefore can only execute commands to within a given accuracy. Given a bound ϵ_c on the control error, we model this error as a cone in configuration space about a vector in the direction of the commanded motion θ and call it the *uncertainty cone* $U_{\epsilon c}(\theta)$ (see Figure 1). While executing a motion plan, the robot complies with the environment and may choose any direction consistent with the commanded direction and the control uncertainty. The direction chosen may vary over the execution of the plan. In what follows, n denotes the number of

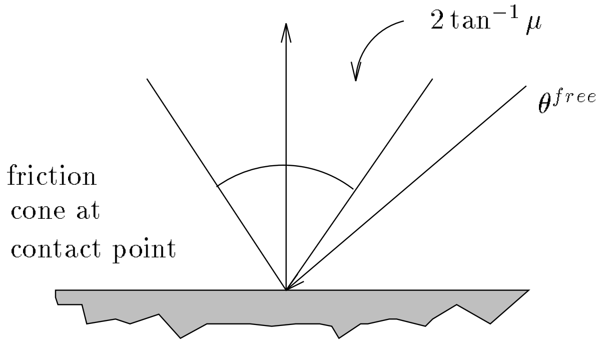

Figure 2: Sliding occurs at motion direction θ^{free}.

vertices in the environment and E denotes the number of edges in the visibility graph, where we assume $E = \Omega(n)$.

To deal with uncertainty in control, we allow *compliance* on obstacle surfaces, where a *compliant* motion is one during which the robot may slide or stick on obstacle surfaces. We model this effect by assuming generalized damper dynamics [Whi77, Mas82, Don89] and Coulomb friction at point contacts, where the coefficient of friction μ is known and remains fixed for the environment. To determine if an obstacle surface is a sticking surface at motion direction θ, we check whether a vector at direction θ lies inside the negative friction cone at the point of contact. If the vector lies outside the negative friction cone, then sliding will occur, otherwise sticking may occur (see Figure 2). To determine if sticking can occur on a vertex, we assume that the vertex can produce reaction forces that are linear combinations of the reaction forces that the adjacent edges can produce [Erd84]. In this model, we take a worst case approach; that is, we assume that if sticking is possible at a point, then the motion plan must prevent the point from being reached unless the point is in the goal.

Our method involves the construction of a concise representation for a structure called the *non-directional backprojection* of the goal [Erd84, Don88]. By analyzing the changes to the backprojection as the motion direction varies, we give a complete characterization of the non-directional backprojection and its complexity, and an efficient representation. To achieve this result, we develop an amortization strategy that is powerful enough to bound the number of changes to the boundary of the backprojection by $O(n^2)$. Hence, we can tighten the previous bound [Don88, Don90] on its combinatorial complexity from $O(n^3)$ to $O(n^2)$, and improve the algorithm for computing it from $O(n^4 \log n)$ to $O(n^2 \log n)$. As our examples show, the non-directional backprojection is a complex object that can undergo very large global changes at a single event while remaining locally monotonic. We obtain our result by characterizing this local monotonicity.

We state the problem as follows:

Definition 1 *Given a planar polygonal environment $\mathcal{P}$ with start region R and goal G, both having a constant number of vertices, the* one-step planar compliant motion planning problem *is to find a commanded motion direction θ such that any trajectory from R consistent with the control uncertainty ϵ_c is guaranteed to reach G. The path should avoid obstacles or comply with the environment.*

1.1 Related Work

Canny and Reif [CR87] have shown that, in three dimensions, the problem of one-step motion planning is $\mathcal{NP}$-hard and the problem of multi-step motion planning is $\mathcal{NEXPTIME}$-hard. Previously, Natarajan had shown the multi-step problem to be $\mathcal{PSPACE}$-hard [Nat86]. Canny has also given doubly-exponential upper bounds for the 3-D translational multi-step problem [Can89]. Erdmann introduced the concept of a backprojection and showed how to compute a backprojection in the plane [Erd84]. Our algorithm extends and improves the $O(n^4 \log n)$ algorithm by Donald [Don88, Don90] for finding a translational motion direction θ in the plane. We presented an earlier version of this work in [Bri89]. Friedman, Hershberger, and Snoeyink have considered the problem of planning compliant motion within a simple polygon [FHS89]. For other related work on motion planning with uncertainty, see [Bro89] and [Lat88].

1.2 Preliminaries

Given goal G, and commanded direction θ, the *backprojection* $B_\theta(G)$ is the set of all initial positions such that any trajectory consistent with the control uncertainty is guaranteed to reach the goal. Donald and Erdmann show that for constant size G, $B_\theta(G)$ can be computed in $O(n \log n)$ time using plane sweep techniques. Erdmann's algorithm is as follows [Don89, Erd86]:

1. For each non-goal vertex, determine whether the inverted control uncertainty cone $U_{\epsilon c}(\theta)$ intersects the friction cone at that vertex. If so, call this vertex a *sticking* vertex

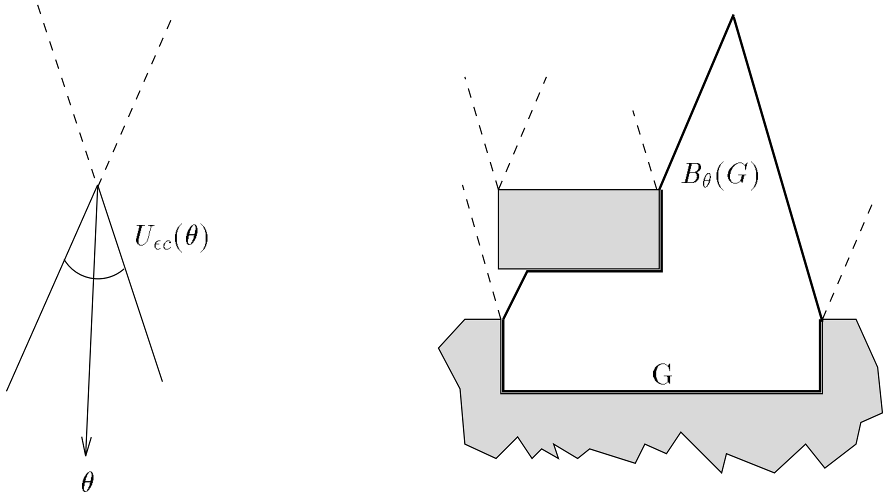

Figure 3: Backprojection construction.

under commanded motion θ.

2. On each sticking vertex, erect two constraint rays parallel to the edges of the inverted control uncertainty cone.

3. Compute the arrangement of the environment with these $O(n)$ additional constraint rays.

4. Starting at a point in the goal, trace out the backprojection region (see Figure 3).

We call the maximal initial segment in free space of a constraint ray erected on a sticking vertex a *free space edge*. The sticking vertex (called the anchor vertex) on which a free space edge e is erected remains fixed while the angle of e changes with θ. Recall that constraint rays lie parallel to one of the edges of the control uncertainty cone, which lie at angle $\theta \pm \epsilon_c$. A *left edge* anchored at p, denoted $\ell(p, \theta)$, is a free space edge lying at angle $\theta + \pi - \epsilon_c$ for motion direction θ and control uncertainty ϵ_c. A *right edge* anchored at q, denoted $r(q, \theta)$, is a free space edge lying at angle $\theta + \pi + \epsilon_c$ (see Figure 4). If free space edge ℓ anchored at p and free space edge r anchored at $q \neq p$ intersect, we call the point of their intersection a *free space vertex*. Free space vertices are uniquely determined by their generating edges. A free space edge is *vgraph critical* when it lies coincident with an edge of the visibility graph and therefore joins two obstacle vertices in free space. We employ the convention that as θ

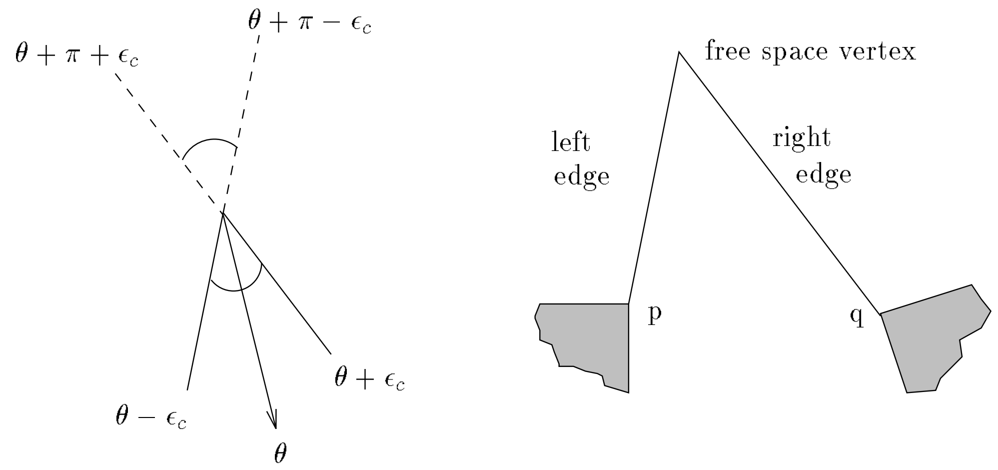

Figure 4: Free space vertex generated by the intersection of two free space edges. The left edge lies at angle $\theta + \pi - \epsilon_c$ and the right edge lies at angle $\theta + \pi + \epsilon_c$.

increases monotonically over the range $[0, 2\pi)$, the corresponding control uncertainty cone $U_{\epsilon c}(\theta)$ rotates in an anticlockwise direction, which has the effect of locally "rotating" the backprojection in an anticlockwise direction.

Lemma 1 *For any direction θ and goal G of constant size, the backprojection $B_\theta(G)$ has size $O(n)$.*

Proof: The environment $\mathcal{P}$ has n obstacle edges and n vertices, that in turn contribute $O(n)$ constraint rays when cones are erected on all the sticking vertices. The backprojection is built from obstacle edges and free space edges, where a free space edge is a constraint ray anchored at an obstacle vertex and intersecting an obstacle edge or another constraint ray. Since at most two free space edges are anchored at each obstacle vertex, the backprojection has size at most $O(n)$. $\square$

In order to find a commanded motion direction θ for which all trajectories from R are guaranteed to reach G, we evaluate the predicate $R \subset B_\theta(G)$ for selected values of θ. As we shall see, it suffices to consider those values of θ, called *critical values*, at which the topology of the backprojection changes. To this end, it is convenient to index the backprojection $B_\theta(G)$ with the critical motion direction θ at which it arises. This leads to the definition of the *non-directional backprojection* $B(G)$ as the set in $\Re^2 \times S^1$

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

Given that the topology of the backprojection does not change between critical events, Donald's approach [Don88, Don90] is to build $B(G)$ by computing a backprojection slice at each critical θ, and then test $B(G)$ for intersection with the cylinder $R \times S^1$ over the start region R. His algorithm builds a representation of size $O(n^3)$ for the non-directional backprojection in time $O(n^4 \log n)$. To obtain his result, Donald shows that we can bound the number of motion directions at which the topology of the backprojection changes, and thereby obtain a polynomial-sized representation for the non-directional backprojection. Since our proof allows this bound to be improved from $O(n^2)$ to $O(E)$, we give our own rendering.

Lemma 2 *There are $O(E)$ motion directions at which the topology of the backprojection changes.*

Proof: Note that the topology of the backprojection may change when any of the following events occurs:

Sliding critical event The determination of sliding vs. sticking on an obstacle edge changes due to a change in the motion direction θ.

Vgraph critical event A constraint ray becomes coincident with a visibility edge.

Vertex critical event A free space vertex of the backprojection coincides with an obstacle edge.

Since the topology of the backprojection can change only when an edge is inserted or deleted, these are the only events that can cause a topological change.

Sliding critical events contribute $O(n)$ critical values of θ since the determination of sliding vs. sticking on an edge can change at most 4 times. The visibility graph has $O(E)$ edges, and since all constraint rays are parallel to one or the other of the two edges of the control uncertainty cone, there are $O(E)$ vgraph critical values of θ.

Vertex critical events contribute $O(E)$ critical values since we can charge each vertex critical event to a unique visibility edge or sliding event as follows. Suppose the free space vertex determined by left edge $\ell(p, \theta_0)$ and right edge $r(q, \theta_0)$ lies on obstacle edge e. Let x

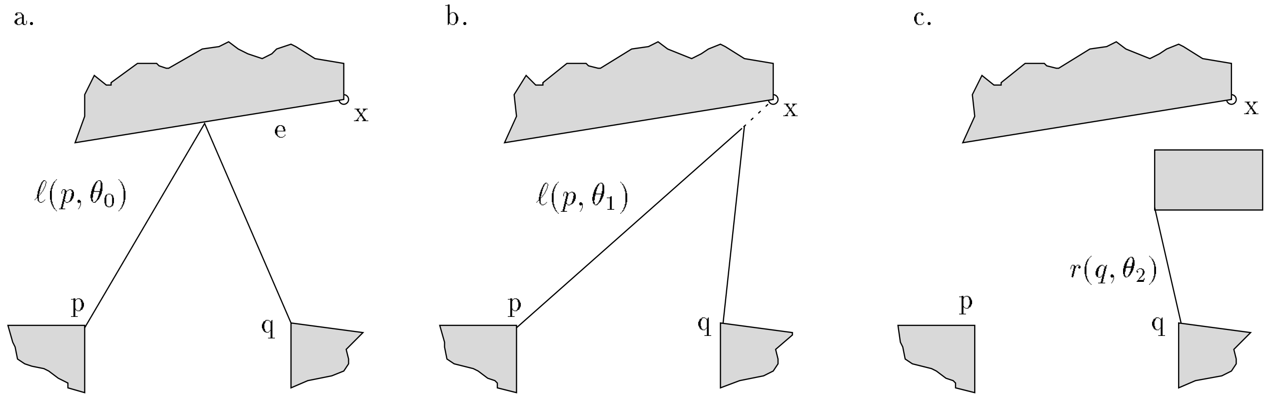

Figure 5: Vertex critical event is charged to a visibility edge. Figure *a* shows the vertex event. Figures *b* and *c* show the previous vgraph events to which the vertex event may be charged.

be the endpoint of e such that $\ell(p,\theta_0)$ has passed over x. Then one of the following must have occurred (see Figure 5):

1. Edge $\ell(p,\theta_1)$, $\theta_1 < \theta_0$, was coincident with the visibility edge between p and x.

2. x is not visible from p and edge $r(q,\theta_2)$, $\theta_1 < \theta_2 < \theta_0$, was coincident with a visibility edge between q and some vertex on an obstacle between p and x.

3. No such vgraph event as in 1. or 2. occurred, in which case at least one of p and q has become a sticking vertex at θ_3, $\theta_1 < \theta_3 < \theta_0$.

We assume that the environment is in general position, that is, at most one critical event will occur at any θ. Under this assumption, one of these events will occur first. In other words, we can choose θ_i, $1 \leq i \leq 3$, for which $\theta_0 - \theta_i$ is minimized. Thus we can charge the vertex event to a vgraph or sliding event at θ_i. At most one other vertex event will be charged to the critical event at θ_i since, without any intervening sliding or vgraph events, a free space vertex travels in a piecewise circular arc and can coincide with an obstacle edge at most twice. We have shown above that the number of sliding and vgraph events is $O(E)$, so it follows that the number of vertex critical events is $O(E)$. $\square$

2 Main Result

As θ changes, the topology of the backprojection changes when edges are inserted or deleted at critical events. Using amortization techniques, we show that over the entire range of θ, the number of topological changes to the boundary of the backprojection is bounded by $O(n^2)$. Thus we can compute a representation for the non-directional backprojection incrementally instead of computing a slice for each critical θ. Namely, we fix $\theta_0 = 0$ and compute the ordered set

$$\{B_{\theta_0}(G), \Delta B_{\theta_1}(G), \ldots, \Delta B_{\theta_\omega}(G)\}$$

where $\theta_i \in \{\theta | \theta \text{ is critical}\}$, $\theta_0 < \theta_1 < \cdots < \theta_\omega$, and $\Delta B_{\theta_i}(G)$ encodes the net change to the data structure representing backprojection slice $B_{\theta_{i-1}}(G)$ to obtain the data structure representing backprojection slice $B_{\theta_i}(G)$ by adding and deleting vertices. We show below that our representation for $B(G)$ has size $O(n^2)$ and is computed in time $O(n^2 \log n)$.

2.1 The Non-Directional Backprojection

In this section we bound the size of the non-directional backprojection by bounding the total number of changes to the backprojection over all θ.

Recall that under the assumption of general position, at most one critical event will occur at motion direction θ. Under this assumption, however, it is not true that a critical event causes only one topological change or only constant topological change to the backprojection. Indeed, we can construct examples where $O(n)$ changes occur at some θ, or where $O(n^{3/2})$ critical events occur, each causing $O(\sqrt{n})$ changes to the backprojection (see Figures 6 and 7). But in the following discussion, we show that the total number of changes to the backprojection over the full range of θ is $O(n^2)$.

In what follows, we make use of an object called a *conforming ray*. A conforming ray is specified by an anchor vertex q and is defined for the interval $[\theta_1, \theta_2]$ in which q is a sticking vertex. Crucial to this definition is the assumption that every polygonal obstacle in the environment contains at least one sticking vertex. Consider an obstacle in the configuration space. Even in the case of no friction, a motion in direction θ could stick on a vertex if θ points into the span of the normal vectors on the edges adjacent to the vertex. The sum of the angles spanning the normal vectors over all vertices is 2π, so any θ will point into the cone spanning the normal vectors at some vertex. Any vertex at which this happens could

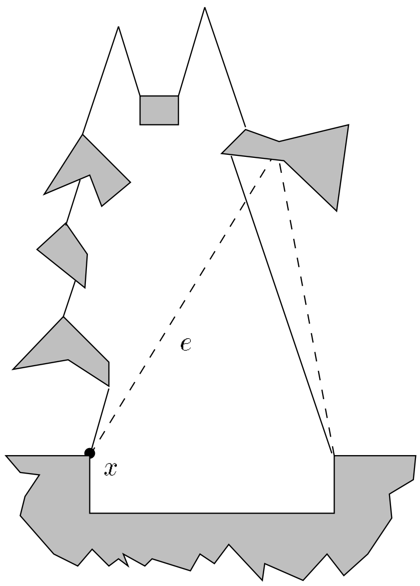

Figure 6: Combinatorially large change in backprojection caused by vgraph event at vertex x.

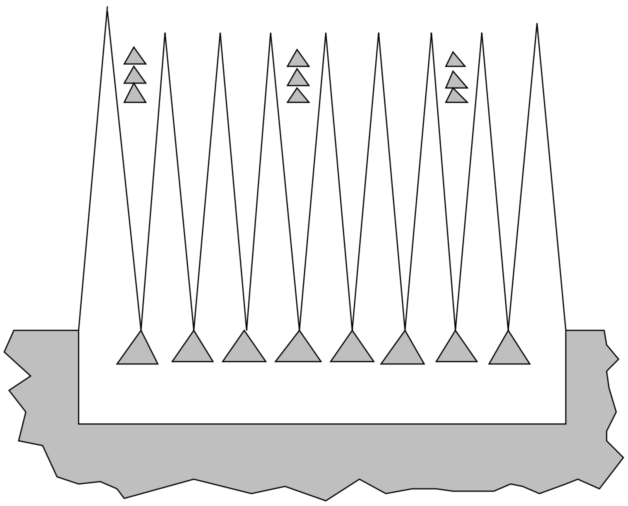

Figure 7: Each of the $O(n)$ spikes of the backprojection will be vgraph critical with each of the $O(\sqrt{n})$ clusters of obstacles. Each cluster is of size $O(\sqrt{n})$, so each of $O(n^{3/2})$ vgraph critical events causes $O(\sqrt{n})$ changes to the topology of the backprojection.

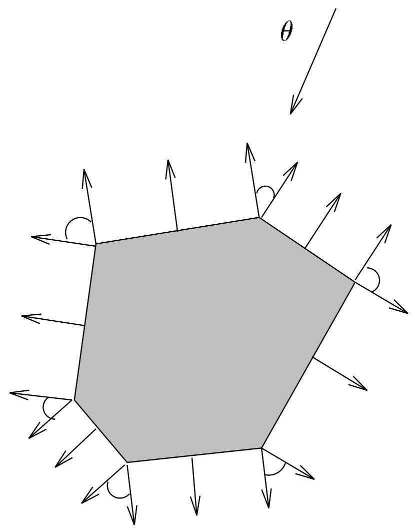

Figure 8: At least one vertex on the polygon must be a sticking vertex.

cause sticking, and so will be called a sticking vertex. (See Figure 8.) For each sticking vertex, we construct a *leading* conforming ray and a *trailing* conforming ray.

Definition 2 *A* leading conforming ray *on sticking vertex q for commanded motion θ is formed by extending $\ell(q, \theta)$ at angle $\theta + \pi - \epsilon_c$ until it intersects an obstacle. We then follow the boundary of the obstacle anticlockwise until reaching the first vertex q' that is a sticking vertex for direction θ. Treat this vertex as q and continue. A* trailing conforming ray *on sticking vertex p for commanded motion θ is formed by extending $r(p, \theta)$ at angle $\theta + \pi + \epsilon_c$ until it intersects an obstacle. We then follow the boundary of the obstacle clockwise until reaching the first vertex p' that is a sticking vertex for direction θ. Again, treat this vertex as p and continue (see Figure 9).*

For purposes of discussion, we say that a point in configuration space is *on* a conforming ray or *contained* in it if the conforming ray passes through the point. A point x *leaves* a conforming ray if the ray no longer passes through x. We imagine that conforming rays sweep out area continuously between critical events. If obstacle vertex x enters the boundary of the backprojection as some conforming ray sweeps over x, we say that the conforming ray *brings x into the boundary of the backprojection.* We are interested in whether a vertex is part of the *boundary* of the backprojection since the backprojection itself is neither open nor closed.

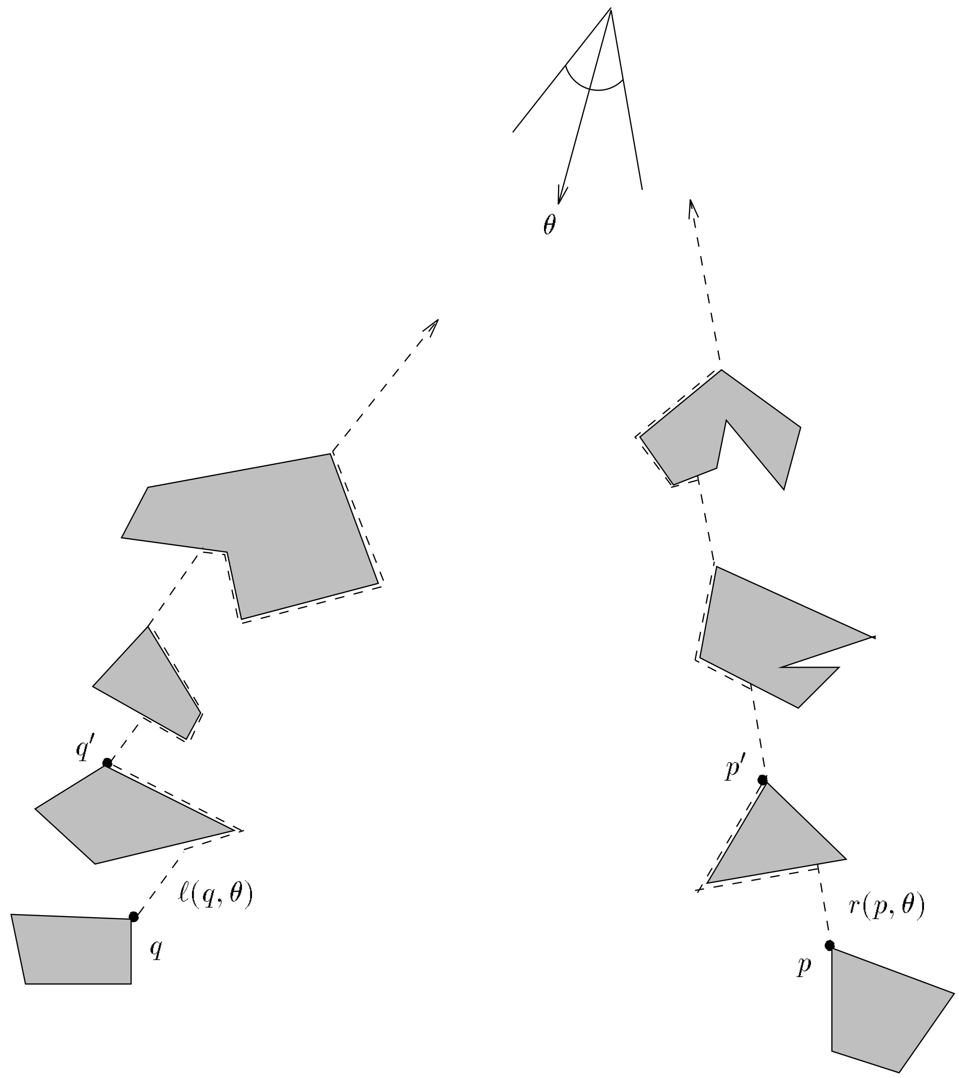

Figure 9: Leading conforming ray on q and trailing conforming ray on p.

For example, in the third backprojection shown in Figure 10, vertex z is contained in the backprojection, and vertex x is not, while both are on the boundary of the backprojection.

Note that the conforming rays bounding the backprojection move monotonically with θ in the absence of sliding events. By this we mean that between sliding events, conforming rays sweep out area in one direction as θ increases. In particular, once a point on a conforming ray leaves the conforming ray, it will not become part of that conforming ray again. This is because each conforming ray is composed simply of obstacle edges and free space edges, which are parallel to the bounding edges of the control uncertainty cone and change monotonically. As critical events occur, the backprojection changes and vertices are added to or deleted from the backprojection boundary. Since a vertex is deleted only as many times as it is added, to establish a bound on the number of topological changes due to vgraph events, we need only consider insertions of vertices to the boundary of the backprojection. When x enters the boundary of the backprojection due to a vgraph event, at least one conforming ray will contain x. We can then say that x enters the backprojection boundary as this conforming ray sweeps over x, and that the entry of x can be charged to this conforming ray. Since conforming rays are monotonic between sliding events, this conforming ray will bring x into the backprojection boundary at most once. Note that even if a conforming ray could back up due to a sliding event, we could charge any resulting topological changes to the sliding event since there are only $O(n)$ sliding events. With $O(n)$ vertices in the environment and $O(n)$ conforming rays, each capable of bringing in each vertex at most once, we have $O(n^2)$ vertices entering the boundary of the backprojection due to vgraph events, i.e., we have $O(n^2)$ topological changes attributable to vgraph events.

We are now ready to prove the following:

Theorem 1 *There are $O(n^2)$ changes to the topology of the backprojection over all values of θ.*

Proof: With the above discussion we have shown that each vertex brought into the boundary of the backprojection due to a vgraph event can be charged to a conforming ray on which it lies (see Figure 10). There are $O(n)$ conforming rays, and n obstacle vertices, so this gives $O(n^2)$ changes to the topology of the backprojection over all vgraph critical events.

Sliding events contribute only $O(n)$ critical values of θ, so the total number of changes to the topology of the backprojection due to sliding events is $O(n^2)$. A vertex critical event

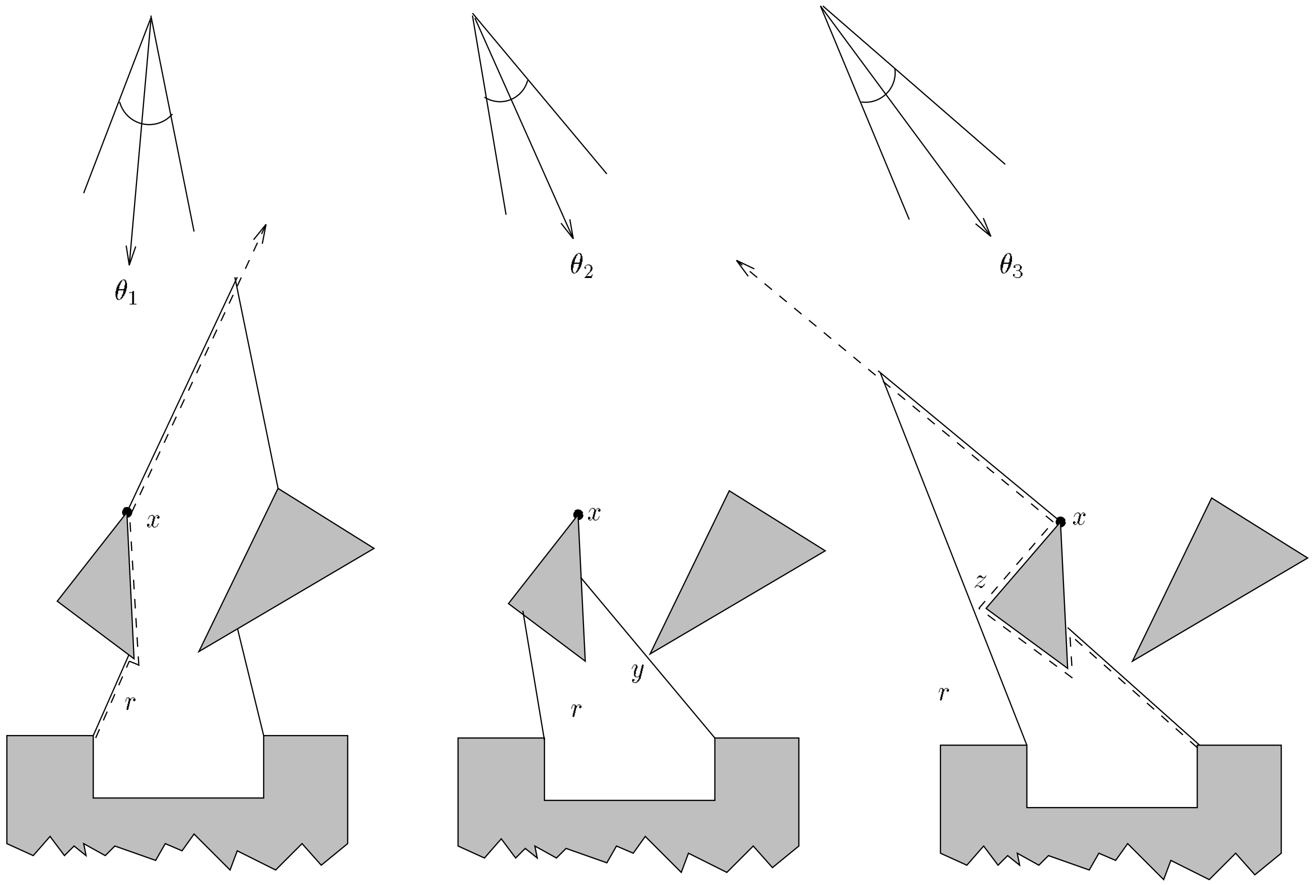

Figure 10: Two entries of obstacle vertex x into the boundary of the backprojection are charged to different conforming rays.

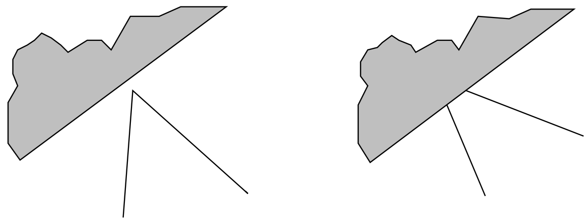

Figure 11: Change to boundary of backprojection due to a vertex critical event.

occurs when a free space vertex coincides with an obstacle edge. Since the environment is assumed to be in general position, no new constraint rays arise and therefore the only change to the boundary of the backprojection is that an obstacle edge segment is inserted or deleted (see Figure 11). Thus the $O(E)$ vertex critical events cause only local changes to the boundary of the backprojection, so they contribute only $O(E)$ topological changes.

Thus the total number of changes to the topology of the backprojection is $O(n^2)$. □

3 The Algorithm

In this section we prove the following:

Theorem 2 *Given a goal G of constant size and an arrangement of input polygons $\mathcal{P}$ of size $O(n)$, a representation of size $O(n^2)$ for the non-directional backprojection $B(G)$ can be computed in time $O(n^2 \log n)$.*

The original arrangement of obstacles has size n and is taken as input. The visibility edges are then computed in time $O(n^2)$ [AAG+86] where n is the number of vertices in the environment. The visibility graph and arrangement remain fixed throughout the computation, which then begins by fixing θ at an initial value, say $\theta_0 = 0$, and computing a backprojection $B_{\theta_0}(G)$. To keep the backprojection updated as θ changes, a priority queue for each type of critical value is maintained. These queues are initialized as follows:

- Sliding critical values: For each obstacle edge e, find each θ at which the determination of sliding vs. sticking on e will change. Insert the pairs (θ, e) in the sliding critical queue, ordered by θ.

- Vgraph critical values: For each constraint ray e, find each θ at which e will become coincident with a visibility edge. Insert the pairs (θ, e) in the vgraph critical queue, ordered by θ. The visibility graph, which has size $O(E)$, can be found and the critical values sorted in time $O(n^2 \log n)$.

- Vertex critical values: For purposes of this discussion, we say that free space backprojection edges e_1 and e_2 are *consecutive* if e_1 is a left edge, e_2 is a right edge, and e_2 is the next free space edge along the boundary of the backprojection. Then for each pair of consecutive free space edges on the boundary of $B_{\theta_0}(G)$, find the first θ at which their intersection point p will lie on an obstacle edge. Insert the pair (θ, p) in the vertex critical queue, ordered by θ.

Since each queue contains $O(n^2)$ elements, each requires time $O(n^2 \log n)$ to initialize and $O(\log n)$ to update.

Note that the sliding and vgraph critical values can be computed ahead of time, while the vertex critical values cannot since we do not know where the free space vertices will be. To solve this problem, we introduce a data structure to help keep track of potential vertex events. Recall that a free space edge is the maximal initial segment of a semi-infinite constraint ray. As the computation proceeds, for each constraint ray e, we keep an updated priority queue of obstacles that the ray intersects. If ray e, anchored at p, lies coincident with vertex v of obstacle j, then the edges of j incident to v are added to or deleted from the priority queue for e depending on whether e will intersect them as θ increases. Since there are $O(n)$ constraint rays and $O(n)$ obstacle edges, the overall time to maintain these queues is $O(n^2 \log n)$. Then at each vgraph or sliding event, we find the new free space vertices and calculate their impending intersections with obstacles, based on the priority queues for their generating edges. This can be done by computing the circular arc in which the free space vertex travels and intersecting it with the common edges in the two queues. These potential vertex events are added to the vertex critical event queue, and the computation continues. As we showed earlier, there is only constant amortized change to the boundary of the backprojection for vgraph events, so there is only a constant amortized number of new free space vertices due to a single vgraph event.

Suppose we have some slice of the non-directional backprojection computed and wish to compute the next backprojection slice. The next critical value of θ can be found by comparing the first elements of the three priority queues, and removing the minimum valid event. Note that an event is valid if it arises from a vertex or edge that is part of the backprojection; not all queued events will be valid. If the next event arises from a sliding criticality, then a new backprojection slice can be computed using the plane sweep algorithm, since there are only $O(n)$ sliding critical values. In the case of a vertex critical value, the only change to the backprojection will be that a free space vertex either enters or leaves the boundary of an obstacle. The backprojection boundary can then be updated locally to reflect the change. On the other hand, a vgraph criticality can cause $O(n)$ changes to the backprojection (see Figure 6). Since there are potentially $O(n^2)$ vgraph critical values, we would like to update the backprojection incrementally when one arises. This can be accomplished by deleting the critical edge e from the backprojection and then tracing out the new backprojection region starting at one endpoint of e until some vertex on $B_\theta(G)$ is reached (see Figure 6). The following lemma shows that this can be accomplished in time linear in the size of the change.

Lemma 3 *Given polygonal environment $\mathcal{P}$, goal G, commanded motion direction θ, control uncertainty ϵ_c, and backprojection vertex u, vertex v adjacent to u on the backprojection $B_\theta(G)$ can be found in $O(1)$ time, making use of a data structure that costs $O(n^2 \log n)$ to maintain over the entire computation.*

Proof: For each sticking vertex v, we keep a priority queue of constraint rays that intersect the left ray of v, ordered by the distance from v to the ray intersection. Each time a new constraint ray is created or deleted at a sliding critical event, we update the priority queues. Since there are $O(n)$ rays, and each is inserted into $O(n)$ queues at a cost of $O(\log n)$ per insert, the overall time to maintain the data structure is $O(n^2 \log n)$. $\square$

3.1 Algorithm One-Step

As the non-directional backprojection $B(G)$ is computed, we can check incrementally for containment of the start region. Some or all of the vertices of $R \subset \mathcal{P}$ may be in free space, so we say that a *pseudo-critical* event occurs when an edge of R intersects $B_\theta(G)$. If a *pseudo-critical* event occurs at θ_j, then we test the backprojection at $B_{\theta_j}(G)$ for containment of R. To do this, we maintain an updated backprojection slice $B_{\theta_i}(G)$ for $\theta_i \in \{\theta | \theta \text{ is critical}\}$ as the computation proceeds. The plane sweep algorithm [Don89] is used to decide $R \subset B_{\theta_j}(G)$, for $\theta_j \in \{\theta | \theta$ is pseudo-critical$\}$, which requires time $O(n \log n)$ for R of constant size. If θ is found such that $R \subset B_\theta(G)$, then motion direction θ from R is guaranteed to reach the goal. Otherwise, no one-step motion is guaranteed to reach G. We have thus established the following:

Theorem 3 *The one-step planar compliant motion planning problem with uncertainty can be solved in time $O(n^2 \log n)$.*

4 Conclusion

In this work, we give an $O(n^2 \log n)$ algorithm for finding a commanded motion direction θ that will guarantee a trajectory from a specified start region to a specified goal region amidst planar polygonal obstacles where control is subject to uncertainty. This result represents a quadratic improvement over the best previous bounds. It is achieved by a new analysis of the geometric complexity of the non-directional backprojection, which yields an efficient algorithm for computing its representation.

We expect that an implementation of this algorithm would perform well in practice. Our robotics lab at Cornell has implemented an approximate generalized damper on a PUMA 560 robot and found through experiments that the control error cones are small enough that implemented backprojection algorithms perform well [JDC89].

We are currently working on the problem of computing a forward projection in the configuration space $\Re^2 \times S^1$. We hope to use such an algorithm for solving the planar compliant motion planning problem with rotation.

4.1 Acknowledgements

I am grateful to my advisor Bruce Donald and to Donald Johnson for their enthusiasm and help in so many ways. I also thank John Canny and Joe Mitchell for helpful discussions and Randy Brost and Samir Khuller for their comments on a draft of the paper.

References

[AAG+86] T. Asano, T. Asano, L. Guibas, J. Hershberger, and H. Imai. Visibility of disjoint polygons. *Algorithmica*, 1:49–63, 1986.

[Bri89] A. J. Briggs. An efficient algorithm for one-step planar compliant motion planning with uncertainty. In *Proc. ACM Symp. on Computational Geometry, Saarbrücken, Germany*, pages 187–196, June 1989.

[Bro89] R. C. Brost. Computing metric and topological properties of configuration-space obstacles. In *Proc. IEEE International Conference on Robotics and Automation, Phoenix, Az.*, May 1989.

[Can89] J. F. Canny. On computability of fine motion plans. In *Proc. IEEE International Conference on Robotics and Automation, Phoenix, Az.*, May 1989.

[CR87] J. F. Canny and J. Reif. New lower bound techniques for robot motion planning problems. In *Proc. IEEE Symp. on Foundations of Computer Science*, 1987.

[Don88] B. R. Donald. The complexity of planar compliant motion planning under uncertainty. In *Proc. ACM Symp. on Computational Geometry, Urbana*, June 1988.

[Don89] B. R. Donald. *Error detection and recovery in Robotics*, volume 336 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1989.

[Don90] B. R. Donald. The complexity of planar compliant motion planning under uncertainty. *Algorithmica*, 5(3), 1990.

[Erd84] M. A. Erdmann. On motion planning with uncertainty. Technical Report MIT–AI–TR 810, MIT Artificial Intelligence Laboratory, 1984.

[Erd86] M. A. Erdmann. Using backprojections for fine motion planning with uncertainty. *IJRR*, 5(1), 1986.

[FHS89] J. Friedman, J. Hershberger, and J. Snoeyink. Compliant motion in a simple polygon. In *Proc. ACM Symp. on Computational Geometry, Saarbrücken, Germany*, June 1989.

[JDC89] J. Jennings, B. R. Donald, and D. Campbell. Towards experimental verification of an automated compliant motion planner based on a geometric theory of error detection and recovery. In *Proc. IEEE International Conference on Robotics and Automation, Phoenix, Az.*, May 1989.

[Lat88] J. C. Latombe. Motion planning with uncertainty: The preimage backchaining approach. Technical Report STAN-CS-88-1196, Department of Computer Science, Stanford University, March 1988.

[LMT84] T. Lozano-Pérez, M. T. Mason, and R. H. Taylor. Automatic synthesis of fine-motion strategies for robots. *Int. J. of Robotics Research*, 3(1), 1984.

[Loz83] T. Lozano-Pérez. Spatial planning: A configuration space approach. *IEEE Trans. on Computers (C–32)*, pages 108–120, 1983.

[Mas82] M. T. Mason. Manipulator grasping and pushing operations. Technical Report MIT–AI–TR–690, MIT Artificial Intelligence Laboratory, 1982.

[Mas84] M. T. Mason. Automatic planning of fine motions: Correctness and completeness. In *IEEE International Conference on Robotics, Atlanta Ga.*, 1984.

[Nat86] B. K. Natarajan. *On Moving and Orienting Objects*. PhD thesis, Department of Computer Science, Cornell University, Ithaca, N.Y., 1986.

[NP82] J. Neivergelt and F. P. Preparata. Plane-sweep algorithms for intersecting geometric figures. *CACM*, 25(10), 1982.

[Whi77] D. Whitney. Force feedback control of manipulator fine motions. *Journal of Dynamic Systems, Measurement, and Control*, pages 91–97, June 1977.