Toward the Design and Analysis of Blind, Bouncing Robots

Lawrence H. Erickson and Steven M. LaValle
Department of Computer Science
University of Illinois at Urbana-Champaign
{lericks4, lavalle}@uiuc.edu

***Abstract*— What kind of tasks are robots with extremely simple control laws capable of performing? Consider a point robot that navigates by aligning itself to a certain fixed angle relative to the environment boundary, then driving in a straight line. Even without knowing the robot's exact location, basic, noisy odometry and counting sensors can be used to narrow down the robot's possible locations over time. This paper describes methods of determining the possible locations of the robot after the robot has moved a sufficient distance or has impacted the boundary a sufficient number of times.**

I. Introduction

Consider a robot whose entire navigation strategy consists of driving in a straight line until it contacts the environment boundary, then bouncing off of it at an angle determined by a function of the vector normal to the boundary point that was struck and the angle relative to the normal at which the robot contacted the boundary. What kind of behaviors can be expected from such a robot? Can it do anything useful? Can it explore the entire environment? Can the robot be easily localized? Can it navigate reliably?

Since the robot's next destination is entirely determined by its current location, these bouncing strategies define a *dynamical system*. Our goal is to model this dynamical system, then design control laws to take advantage of it.

A closely related dynamical system is *mathematical billiards*. In mathematical billiards, an environment contains an agent that travels in a straight line until it contacts the environment boundary, at which point the agent makes a specular bounce and travels in a straight line again. A bounce is *specular* if the incident path and the post-collision path make the same angle relative to the normal of the boundary (see Figure 1). Mathematical billiards are of particular interest in physics for the modeling of ideal gases [11]. There are also applications in the study of elastic collisions and optics [12]. The path of the agent is known to be chaotic in numerous environments [5]. The ergodicity properties of such agents in polygons are also heavily studied [8].

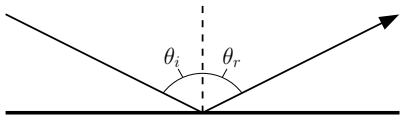

Fig. 1. A bounce is specular if the incident angle relative to the surface normal θ_i is equal to the reflected angle θ_r.

When the agent is deflected towards the normal instead of making a purely specular bounce, the problem is referred to as *pinball billiards* [10]. The special case where the bounce is normal to the boundary contacted by the robot (with the incident angle being irrelevant) is referred to as *slap billiards*. The authors of [10] were primarily interested in the properties of the system's attractors.

In this paper, we have chosen to examine a family of bouncing strategies that have the robot bounce at a constant angle relative to the normal of the edge that was impacted (see Figure 2) with the incident angle being irrelevant. This is a generalization of the slap billiards model. We are primarily interested in localization and navigation questions. Both tasks rely on making the movement of the robot "predictable" in some fashion. We attempt to accomplish part of this task by determining, for a given bouncing strategy, which portions of the polygon will be unreachable to the robot after it has bounced a certain number of times or travelled more than a certain distance. The remainder of the polygon can be considered a "trap" region that the robot is localized to after a sufficient amount of travel.

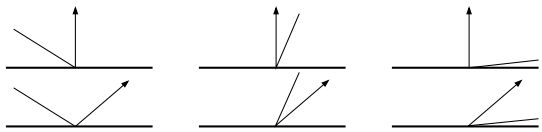

Fig. 2. [top] The paths created when the robot is instructed to bounce in the direction perpendicular to the edge that was contacted. [bottom] The paths created when the robot is instructed to bounce at an angle 50 degrees clockwise of perpendicular.

Strategies for localization in robots equipped with contact sensors include [7] and [9]. However, in these papers, the angle at which the robot leaves the wall is a function of the number of prior bounces. Even if there is a small amount of uncertainty in the robot's orientation, it is generally possible to force the robot into a relatively small area. This is generally going to be impossible under the strategies examined in this paper. Also related are sensorless manipulation and localization strategies using tilted platforms [6], conveyor systems [1], vibrating surfaces [3], and microactuators [4].

While the wild, sensorless "weaselball" robots of [2] provided some inspiration for the system studied in this paper, they have little to do with each other. The system studied in this paper is entirely deterministic, and "weaselball" robots do not bounce off of environment boundaries at some fixed angle relative to the normal.

The remainder of this paper is organized as follows. Section II formally describes the robot movement model and

provides definitions used throughout the paper. Section III describes some common traits that the paths produced by these robots possess. Section IV describes an algorithm that incrementally classifies the boundary into "trap" and "non-trap" regions. Section V discusses the results and directions for future research.

II. Model and Definitions

A point robot R moves in a closed polygonal region P with boundary ∂P. The robot's motion strategy is dependent on a single parameter θ, where $-\pi/2 < \theta < \pi/2$. The robot drives in a straight line until contacting ∂P. After contacting ∂P, the robot rotates until its heading is θ radians clockwise of the inward-facing normal of the edge with which the robot is in contact. It then moves in a straight line until contacting ∂P again, and the process is repeated indefinitely (see Figure 3).

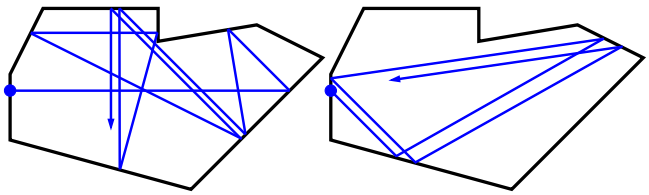

Fig. 3. Paths of the robot starting from the blue circle. [left] The robot's path when $\theta = 0$. [right] The robot's path when $\theta = \pi/4$.

Since the incident angle of the robot when it strikes a wall has no effect on the direction of the robot, the movements of the robot can be described by a map $B_\theta : \partial P \to \partial P$, where the robot impacting the boundary at $p \in \partial P$ will next impact the boundary at $B_\theta(p)$. A superscripted B_θ^k will denote k applications of the map to the input. We will call this map the robot's *bouncing strategy*. A sequence of points $[p_0, \ldots, p_k]$ is a *flow* of B_θ if $p_i = B_\theta(p_{i-1})$ for $1 \leq i \leq k$. We note that the bouncing strategy is not well-defined for vertices of ∂P. However, since the number of points of ∂P that would send the robot to a vertex is finite (each ordered pair of source edge and destination vertex can produce at most one such point), we will ignore flows that would have the robot impact a vertex.

Let q be a point in ∂P. Given a bouncing strategy B_θ, a flow $[p_0, \ldots, p_k]$ is a *trail* of q if $p_k = q$. A flow is a *trajectory* of q if $p_0 = q$. Note that for all natural n, the n-step trajectory of q is unique, while an n-step trail of q may not be unique. In fact, it is possible for an n-step trail of q not to exist (for example, if there exists no $p \in \partial P$ such that $B_\theta(p) = q$). Let $T(q, B_\theta)$ denote the set of trails of q under bouncing strategy B_θ. Let $d(p, q)$ be the distance between points p and q. The *distance* of a flow $[p_0, \ldots, p_k]$ is equal to $\sum_{i=0}^{k-1} d(p_i, p_{i+1})$. We will say that $T(q, B_\theta)$ is *distance unbounded* if for any positive real a, there exists a trail $t \in T(q, B_\theta)$ in which the length of t is at least a. The *link distance* of a flow $[p_0, \ldots, p_k]$ is equal to k. We will say that $T(q, B_\theta)$ is *link unbounded* if for any positive integer j there exists a trail $t \in T(q, B_\theta)$ such that t has link distance at least j.

We choose to focus on these notions of distance bounded and link bounded due to the ease at which they can be used by a robot with simple sensors. If the robot has a linear odometer and a map of the polygon (even a noisy odometer that gives a lower bound on the distance travelled up to some constant factor of the truth), then as it travels, it can gradually rule out distance bounded regions as possible locations. Similarly, if the robot has a counter that keeps track of how many times it has bounced, it can gradually rule out link bounded regions as possible locations.

III. Behaviors

This section describes some common behaviors observed in the trajectories produced by these bouncing strategies, for the purpose of providing the reader with some intuition about how these robots move about.

Regardless of the precise value of θ, the bouncing strategy B_θ always exhibits two notable behaviors. First, parallel lines cause the robot to get "stuck" in a 2-cycle. This is a generalization of an observation in [10].

Observation 1: Let e_1 and e_2 be parallel edges of P. Let $p_1 \in e_1$ and $p_2 \in e_2$ be points on the boundary of P. If $B_\theta(p_1) = p_2$, then $B_\theta(p_2) = p_1$.

Proof: If $B_\theta(p_1) = p_2$, then there are no edges of P intersecting $\overline{p_1 p_2}$. Since e_1 and e_2 are parallel and $B_\theta(p_1) = p_2$, the inward-facing normal of e_1 is the opposite direction as the inward-facing normal of e_2. Therefore, θ radians clockwise of the inward-facing normal of e_1 is the opposite direction as θ radians clockwise of the inward-facing normal of e_2. Since no edges of P intersect $\overline{p_1 p_2}$, we get that $B_\theta(p_2) = p_1$. ■

Note that small adjustments to the robot's bouncing strategy or to the robot's position will not cause this cycle to disappear.

Secondly, the robot will tend to "escape" corners (see Figure 4). Regardless of the choice of θ, a sequence of bounces between two non-parallel edges will cause the robot to move further away from the point where the edges intersect (or where they would intersect if they were extended). Lemma 2 describes exactly how this process works.

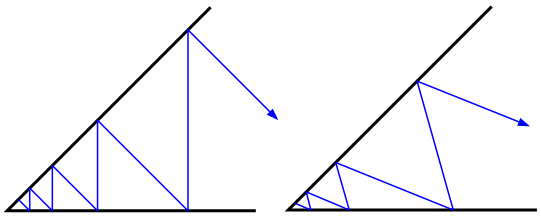

Fig. 4. The robot escaping a corner for two different values of θ.

Though the robot tends to escape corners, certain structures will cause the robot to become "trapped". The simplest example is when $\theta = 0$ and P is an obtuse triangle. The robot will eventually become confined to a region of P centered around the obtuse vertex (see Figure 5). In fact, for a fixed value of θ, polygons generally have some portion of the boundary that is unreachable after the robot has contacted the wall some minimum number of times, and some portion of

the boundary that is unreachable after the robot has travelled some minimum distance. The remainder of the boundary can be considered a "trap" for that particular choice of θ.

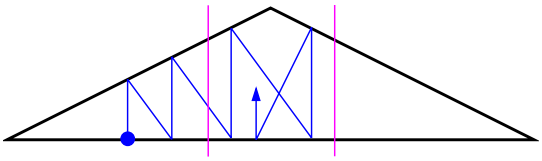

Fig. 5. A robot bouncing in an obtuse triangle with $\theta = 0$. Regardless of the robot's starting point, it will eventually be confined to the portion of the triangle between the purple lines. A sample path is shown.

IV. Classification of the Boundary

In this section, the goal is to determine what portions of the boundary correspond to distance unbounded regions, distance bounded link unbounded regions, and link bounded regions for a fixed value of θ.

This section is divided into two subsections. Subsection IV-A describes in detail the behavior of the robot when it is escaping corners, as that behavior is what produces the distance bounded link unbounded regions of the boundary. Subsection IV-B describes an algorithm that classifies the boundary, and discusses its limitations.

A. Codependent Segments

If the interior angle of a vertex of ∂P is less than $|\pi/2-\theta|$, then the vertex is a *codependent vertex*. If the robot bounces between two edges incident to a codependent vertex, then it moves away from the codependent vertex, at a rate described in the following lemma.

Lemma 2: Let e_1 and e_2 be non-parallel edges of P. Let q be the point where the infinite extensions of e_1 and e_2 intersect. Let ϕ be the interior angle of the extensions of e_1 and e_2. Let $p_1 \in e_1$ be a point, and suppose that $B_\theta(p_1) \in e_2$. If $B_\theta^2(p_1) \in e_1$, then there exists some c, where $c > 1$, dependent only on θ and ϕ such that $cd(p_1, q) = d(B_\theta^2(p_1), q)$.

Proof: The law of sines establishes that

$$\frac{d(p_1,q)}{\sin(\frac{\pi}{2}+\theta-\phi)} = \frac{d(B_\theta(p_1),q)}{\sin(\frac{\pi}{2}-\theta)} \tag{1}$$

and

$$\frac{d(B_\theta^2(p_1),q)}{\sin(\frac{\pi}{2}+\theta)} = \frac{d(B_\theta(p_1),q)}{\sin(\frac{\pi}{2}-\theta-\phi)}. \tag{2}$$

Combining these equations and simplifying the terms inside the sines yields

$$d(B_\theta^2(p_1),q) = \left(\frac{\cos 2\theta + 1}{\cos(2\theta)+\cos(2\phi)}\right) d(p_1,q) = cd(p_1,q). \tag{3}$$

Since $\cos(2\phi) < 1$ (as $0 < \phi < \pi/2$), we get that $c > 1$. ∎

We will describe a pair of line segments ℓ_1, ℓ_2 that meet at a codependent vertex $v \in \partial P$ as *codependent segments* if for each point $p_1 \in \ell_1$, there exists a point $p_2 \in \ell_2$ such that $B_\theta(p_2) = p_1$ and B_θ does not map any distance unbounded point of $\partial P \setminus \ell_2$ to p_1. Additionally, for each $q_1 \in \ell_2$, there exists a point $q_2 \in \ell_1$ such that $B_\theta(q_2) = q_1$ and B_θ does not map any distance unbounded point of $\partial P \setminus \ell_1$ to q_1 (see Figure 6).

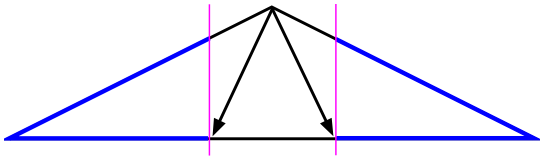

Fig. 6. The area in between the purple lines is distance unbounded when $\theta = 0$. The pair of blue segments on the left side of the triangle are codependent. Note that B_θ does not map any point on the top right edge to the blue area on the left. The pair of blue segments on the right are also codependent.

Corollary 3: Let $\ell_1 \subseteq e_1$ and $\ell_2 \subseteq e_2$ be codependent segments that meet at a convex vertex v. For each point $p \in \ell_1 \cup \ell_2$, there exists some constant d such that the distance of each $t \in T(p, B_\theta)$ is less than d.

Proof: Because e_1 and e_2 are codependent segments and meet at a vertex, for each k, p has only one trail of length k, which we will denote as $[p_{k-1}, \ldots, p_2, p_1, p_0 = p]$. Lemma 2 implies that there exists some constant c such that $c^i|\overline{p_{i+1}p_i}| = |\overline{p_1p_0}|$. Since $c > 1$, we have, for all $k \geq 1$, that $\sum_{i=0}^{k-1} \frac{1}{c^i} < d$ for some constant d. ∎

Lemma 4: Let $S = \{t_0, t_1, t_2, \ldots\}$ be a subset of link unbounded $T(p_0, B)$ such that $t_0 = [p_0]$, $t_1 = [p_1, p_0]$, etc., and for each real number a, there exists a member of S that has length at least a. For each natural k, there exists three edges of P and a j such that for all $i > j$, the trail t_i contains at least k points on each of the three edges.

Proof: Suppose that this lemma were false, and there existed such a subset S that satisfied the condition of the lemma but contained trails that had arbitrarily high numbers of points on only two edges.

Without loss of generality, we may assume that t_i contains points from only two edges, because if only two edges are represented an arbitrarily high number of times, then there exists some k such that for all $j \geq k$, p_j is on one of two edges. The sum $\sum_{0 \leq i < k} |p_i p_{i+1}|$ is finite. We must then consider the set of trails $S' = \{t'_0, t'_1, t'_2, \ldots\}$, where $t'_0 = [p_k], t'_1 = [p_{k+1}, p_k]$, etc. Obviously, S contains a trail of length a for each possible a if and only if S' does as well, and the trails of S' contain points from only two edges.

If all t_i contains only two edges, then by Lemma 2, $c|\overline{p_ip_{i-1}}| = |\overline{p_{i-1}p_{i-2}}|$ for some constant c where $c > 1$. This implies that the distance of t_i as $i \to \infty$ converges to some constant. ∎

Note that each distance unbounded set of trails must contain a subset of the type described in Lemma 4.

B. Classification Algorithm

The classification algorithm (Algorithm 1) in this section operates by first determining which segments are obviously link bounded (segments that are not in the image of B_θ) and which segments are obviously distance bounded link unbounded (codependent segments). After this determination is made, the areas that are still potentially distance (link)

unbounded are checked to see if B_θ only maps to them from distance (link) bounded areas. This process repeats recursively. Algorithms 2 and 3 are subroutines of Algorithm 1.

The *θ-projection* of an interval w, denoted $\mathrm{Project}(\theta, w)$, is the set $\{p \in \partial P \mid \exists q \in w \text{ where } p = B_\theta(q)\}$.

Algorithm 1 DIVIDE(P) - Divide ∂P into distance unbounded, link unbounded distance bounded, and link bounded regions).

Label all of ∂P as link bounded.
for Each edge e_i **do**
  Label $\mathrm{Project}(\theta, e_i)$ as distance unbounded.
end for
Let S_0 be the set of intervals marked as distance unbounded.
Let $f = \cup_{s \in S_0} \mathrm{Project}(\theta, s)$.
for Each unordered pair of edges e_i,e_{i+1} that meet at vertex v_i **do**
  if $v_i \notin f$ **then**
    Determine the maximal subset of $e_i \setminus f$ containing v_i as an endpoint, and mark it as link unbounded distance bounded.
    Determine the maximal subset of $e_{i+i} \setminus f$ containing v_i as an endpoint, and mark it as link unbounded distance bounded.
  end if
end for
Mark all distance bounded intervals as "changed".
while Any interval is marked as "changed" **do**
  for Each vertex $v \in \partial P$ that is not incident to a pair of link unbounded distance bounded intervals **do**
    CHECK-VERTEX(v)
  end for
  for Each interval t marked as changed and edge e_j **do**
    UPDATE(t, e_j)
  end for
  Remove all "changed" labels.
  Change all "recently changed" labels to "changed" labels.
end while

Theorem 5: For each point $p \in \partial P$, there exists a k such that after k iterations of the "WHILE" loop of Algorithm 1, p is classified correctly.

Proof: Call a point $q \in \partial P$ *minimal* if it is not part of the projection from any edge. The minimal points can be easily found by exhaustively checking all projections.

Note that a point $p \in \partial P$ is distance unbounded if and only if there exists some $q \in \partial P$ such that q is distance unbounded and $B_\theta(q) = p$. Similarly, p is link unbounded if and only if there exists some link unbounded $q \in \partial P$ such that $B_\theta(q) = p$. The points in the trails of p can be arranged into a *trail tree* with p at the root. The children of a node representing a point q are the points $\{s \in \partial P \mid B_\theta(s) = q\}$. A path from the root terminates at a leaf when a minimal point or point whose only descendants would form an infinite path that alternates between two edges (see Figure 7). Note that p is distance bounded if and only if its tree is finite, and it is link bounded if and only if all of its leaves correspond to minimal points.

Algorithm 2 CHECK-VERTEX(v) - Check to see if there are segments incident to v that have become codependent.

Let e_i and e_{i+1} be the edges incident to v.
Let S be the set of distance unbounded intervals on edges other than e_i and e_{i+1}.
Let $g = \bigcup_{s \in S} \mathrm{Project}(\theta, s)$.
if $v \notin g$ **then**
  Let $h_i \subset e_i$ be the maximal interval with v as an endpoint such that $h_i \cap g = \emptyset$.
  Let $h_{i+1} \subset e_{i+1}$ be the maximal interval with v as an endpoint such that $h_{i+1} \cap g = \emptyset$.
  Mark h_i and h_{i+1} as link unbounded distance bounded, and mark both as "recently changed".
end if

Algorithm 3 UPDATE(t, e) - Check to see if any intervals have become link or distance bounded due to a change in t.

Let $f = \mathrm{Project}(\theta, t) \cap e$.
if t is link bounded **then**
  Let S be the set of link unbounded intervals.
  Let $g = \bigcup_{s \in S} \mathrm{Project}(\theta, s) \cap e$.
  Mark $f \setminus g$ as link bounded.
  Mark all intervals of $f \setminus g$ as "recently changed".
else if t is link unbounded distance bounded **then**
  Let S be the set of distance unbounded intervals.
  Let $g = \bigcup_{s \in S} \mathrm{Project}(\theta, s) \cap e$.
  let h be the portion of f that is link bounded.
  Mark $f \setminus (g \cup h)$ as link unbounded distance bounded.
  Mark $f \setminus (g \cup h)$ as "recently changed".
end if

Let the *tree depth* of a point $p \in \partial P$ be the maximum distance in p's trail tree between the root node and any leaf. Note that if p is distance unbounded, then p has infinite tree depth.

The algorithm initially labels every point in ∂P as distance unbounded unless it is a minimal point or a point in a codependent segment. Note that this means that at the start of execution, the points of ∂p with tree depth 0 are classified correctly.

Assume that after $k-1$ iterations of the "WHILE" loop, all distance bounded points with trail trees of depth $k-1$ or less are classified correctly. On the kth iteration of the loop, each point that is in the projection of an interval of points of depth less than k is checked to see if it is in the projection of an interval that has not yet been classified as distance bounded. If it is not, then the point is classified as distance bounded (link unbounded if it is in the projection of a link unbounded interval, link bounded otherwise).

However, the process described in the above paragraph

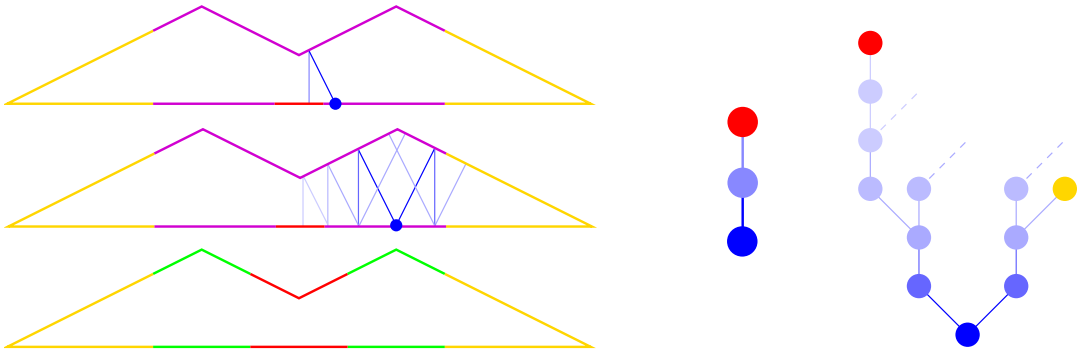

Fig. 7. [top left] The trail of a link bounded point of ∂P. Minimal link bounded portions of the polygon are in red. Minimal link unbounded distance bounded portions are in yellow. Non-minimal portions are in purple. [middle left] Part of the trail of a distance unbounded point of ∂P. The links of the trail become lighter as it goes further from the starting point. [bottom left] The full classification of ∂P, in which red is link bounded, yellow is link unbounded distance bounded, and green is distance unbounded. [middle] The trail tree of the top left subfigure. The red vertex indicates a link bounded minimal point. [right] A portion of the trail tree of the middle left subfigure. Dotted lines indicate infinitely deep subtrees that are not shown in the middle left subfigure. The red and yellow vertices indicate minimal points.

cannot find a vertex whose only possible descendants consist of a path whose vertices alternate between two edges. Therefore, if a codependent vertex v_i (incident to e_i and e_{i+1}) is in the projection of an interval of depth less than k, we must check if there is a pair of intervals $\overline{qv_i} \subseteq e_i$ and $\overline{rv_i} \subseteq e_{i+1}$ such that the only distance unbounded interval that projects onto $\overline{qv_i}$ is $\overline{rv_i}$ and vice versa. In that case, for any point $p \in \overline{qv_i} \cup \overline{rv_i}$, all potential descendants of p in a trail tree form a path that alternates between points on e_i and e_{i+1}. Therefore, p should be a leaf in the trail tree.

Only points in the projection of an interval of depth $k-1$ need to be tested, as a point with a trail tree depth of k must contain a point with a trail tree of depth $k-1$ in its tree. Therefore, after k iterations, all points of depth k are correctly classified. ■

While Algorithm 1 eventually classifies each point correctly, it is not guaranteed to terminate. For an example, consider an equilateral triangle with θ set to $\pi/2-\varepsilon$ for some small value of ε. There are only three distance unbounded points in ∂P, and they correspond to the vertices of an inscribed triangle (in clockwise order, let the vertices of this inscribed triangle be p, q, r). However, Algorithm 1 initially labels the entire triangle boundary as distance unbounded. As the WHILE loop of Algorithm 1 the portions of the boundary that are still labelled as potentially distance unbounded decrease, but they remain intervals, and never shrink down to individual points (see Figure 8).

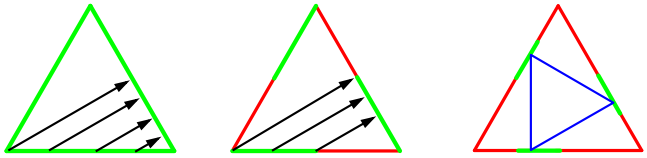

Fig. 8. Algorithm 1 in an equilateral triangle when $\theta = \pi/3$. [left] Initially, the entire boundary is marked as distance unbounded (green). [middle] Since portions of the boundary are not in the projection of any distance unbounded interval, some areas are marked as link bounded (red). [right] This process repeats, further shrinking the intervals marked as distance unbounded, but never reducing them to points. The endpoints of the blue inscribed triangle are the only actual distance unbounded points.

For an input polygon P and bouncing angle θ, let the *complexity* be the minimum number of intervals required to classify the boundary of ∂P, where each interval I is a connected subset of ∂P that lies on a single edge with all points in I being either distance unbounded, link unbounded distance bounded, or link bounded. Since the running time of any algorithm that classifies the boundary of a polygon P as a set of intervals must be at least as large as the number of intervals produced, it would be convenient if the complexity could be bounded by the number of edges. However, this is not the case.

Theorem 6: For each natural number k, there exists a polygon with 5 vertices with complexity at least k.

Proof: See Figure 9. The blue point in each polygon is actually an extremely small edge. The exact size of the edge can be made small enough that its projections can be considered points. The precise orientation of the edge causes some number of distance unbounded intervals to appear on the left side of the polygon. As the projection from the small blue edge moves closer to the left corner of the polygon, more distance unbounded regions appear. In order to prevent these regions from merging, the blue edge needs to shrink as the projection moves closer to the left corner. ■

While Algorithm 1 converges to the correct classification over time, there are no monotonicity guarantees with regard to the amount of the boundary reclassified in each step of the "WHILE" loop. In fact, it is possible to construct polygons where an arbitrarily large amount of the boundary is reclassified after an arbitrarily high number of iterations (see Figure 10).

V. Conclusion and Discussion

This paper has introduced a family of motion strategies for a robot based on a simple bouncing rule, and has provided an algorithm that determines which portions of the polygon the robot is able to be located in after using the strategy for a sufficiently long time (the distance unbounded regions).

In order to use the "traps" formed by the distance unbounded regions of the polygon, they must appear in some formation that makes control of the robot straightforward. Manipulation of the θ parameter by a small amount will

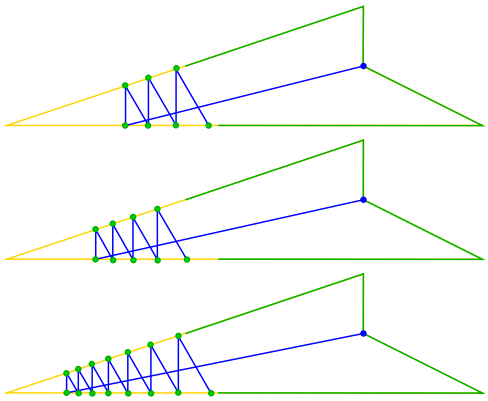

Fig. 9. A family of polygons, each with five edges, that can have arbitrarily high complexity. As the projection from the small blue edge on the right moves closer to the left corner, additional distance unbounded regions are created. The blue edge is small enough that its projection can be treated as a point. In this example, $\theta = \pi/2$, though the same technique will work with arbitrary θ.

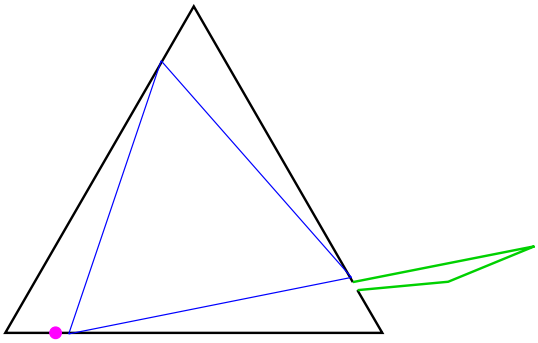

Fig. 10. When θ is $\pi/2-\varepsilon$, the only distance unbounded (green) intervals of the triangular region on the left are the endpoints of the blue inscribed triangle. The green triangular region on the right is initially labelled as distance unbounded by Algorithm 1. However, when the small region of ∂P under the purple dot is correctly labelled as link bounded (red), much of the boundary of the triangular region on the right will be reclassified as link bounded (red) or distance bounded link unbounded (yellow).

slightly move the distance unbounded regions along the boundary, but large alterations in θ may cause drastic changes in the distance unbounded regions. Each value of θ induces some signature on each polygon edge corresponding to the ordering of link bounded, link unbounded distance bounded, and distance unbounded intervals on that edge. A value of θ is *critical* if $\theta+\varepsilon$ and $\theta-\varepsilon$ produce different signatures for some edge for arbitrarily small values of ε. If all critical values could be identified for some input polygon, then Algorithm 1 (or some variant with better guarantees about termination) could be run once for each θ that lies halfway between two consecutive critical values, which would reveal the maximum degree of control that this family of bouncing strategies would provide in the input polygon. However, the authors have not been able to prove that the number of critical values is even finite.

Alternately, one could design environments for which control strategies are relatively simple. Figure 11 demonstrates an environment where a value of θ very close to $\pi/2$ or $-\pi/2$ forces the robot into the bottom triangle. Gradually altering θ eventually forces the robot into either the left or right triangle, depending on whether a value close to $\pi/2$ or $-\pi/2$ was used initially.

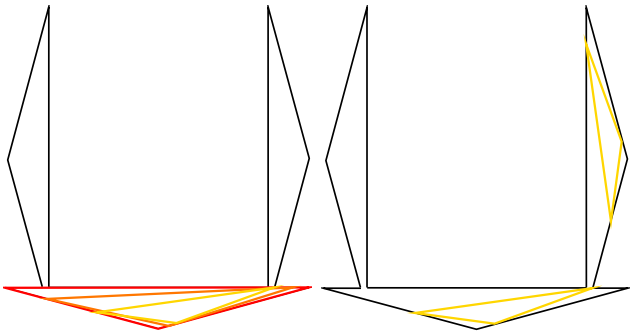

Fig. 11. [left] When θ is very close to $\pi/2$, the robot will be forced into the bottom triangle, and its path will converge to the red triangle. As θ is gradually lowered, the robot's path converges to an inscribed triangle in the bottom region (orange, then yellow triangles). [right] When the inscribed triangle's corner touches the opening into the right region, the robot will fall into the right region and its path will converge to the inscribed yellow triangle in the right region.

Acknowledgements

This work is supported in part by NSF grant 0904501 (IIS Robotics), NSF grant 1035345 (CNS Cyberphysical Systems), DARPA SToMP grant HR0011-05-1-0008, and MURI/ONR grant N00014-09-1-1052.

The authors would like to thank Vadim Zharnitsky and Nanxin Zhao for their assistance with this project.

References

[1] S. Akella, W. H. Huang, K. M. Lynch, and M. T. Mason, "Sensorless parts feeding with a one joint robot," in *Algorithms for Robotic Motion and Manipulation*, J.-P. Laumond and M. Overmars, Eds. Wellesley, MA: A. K. Peters, 1997, pp. 229–237.

[2] L. Bobadilla, O. Sanchez, J. Czarnowski, K. Gossman, and S. LaValle, "Controlling wild bodies using linear temporal logic," in *Proceedings of Robotics: Science and Systems*, 2011.

[3] K.-F. Bhringer, V. Bhatt, B. R. Donald, and K. Goldberg, "Algorithms for sensorless manipulation using a vibrating surface," *Algorithmica*, vol. 26, pp. 389–429, 2000.

[4] K.-F. Bhringer, B. Donald, R. Mihailovich, and N. MacDonald, "Sensorless manipulation using massively parallel microfabricated actuator arrays," in *Proc. IEEE International Conference on Robotics and Automation*, 1994, pp. 826–833.

[5] N. Chernov and R. Markarian, *Chaotic Billiards*. American Mathematical Society, 2006.

[6] M. A. Erdmann and M. T. Mason, "An exploration of sensorless manipulation," *IEEE Transations on Robotics and Automation*, vol. 4, no. 4, pp. 369–379, Aug. 1988.

[7] L. Erickson, J. Knuth, J. M. O'Kane, and S. M. LaValle, "Probabilistic localization with a blind robot," in *Proc. IEEE International Conference on Robotics and Automation*, 2008.

[8] S. Kerckhoff, H. Masur, and J. Smillie, "Ergodicity of billiard flows and quadratic differentials," *The Annals of Mathematics, Second Series*, vol. 124, no. 2, pp. 293–311.

[9] J. S. Lewis and J. M. O'Kane, "Reliable indoor navigation with an unreliable robot: Allowing temporary uncertainty for maximum mobility," in *Proc. IEEE International Conference on Robotics and Automation*, 2012.

[10] R. Markarian, E. J. Pujals, and M. Sambarino, "Pinball billiards with dominated splitting," *Ergodic Theory and Dynamical Systems*, vol. 30, pp. 1757–1786, 2010.

[11] Y. G. Sinai, "Dynamical systems with elastic reflections. ergodic properties of dispersing billiards," *Russian Mathematical Surveys*, vol. 25, no. 1, pp. 137–189, 1970.

[12] S. Tabachnikov, *Geometry and Billiards*. American Mathematical Society, 2005.