A Visibility-Based Approach to Computing Nondeterministic Bouncing Strategies

Alexandra Q. Nilles[1], Yingying Ren[1], Israel Becerra[1], and Steven M. LaValle[2,1]

[1] Department of Computer Science
University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA
{nilles2, yren17, israelb4, lavalle}@illinois.edu
[2] Faculty of Information Technology and Electrical Engineering
University of Oulu, Oulu, Finland

Abstract. Inspired by motion patterns of some commercially available mobile robots, we investigate the power of robots that move forward in straight lines until colliding with an environment boundary, at which point they can rotate in place and move forward again; we visualize this as the robot "bouncing" off boundaries. Different boundary interaction rules can be defined for such robots, such as one that orients the robot relative to its heading prior to collision, or relative to the normal of the boundary. We introduce a new data structure, the *bounce visibility graph*, which is generated from a polygonal environment definition. The bounce visibility graph can be queried to determine the feasibility of path-based tasks such as navigation and patrolling under actuator uncertainty. If the task is feasible, then this approach synthesizes a strategy (a sequence of nondeterministic rotations). We also show how to compute stable cyclic trajectories and use these to limit uncertainty in the robot's position.

1 Introduction

Mobile robots have rolled smoothly into our everyday lives, and can now be spotted vacuuming our floors, cleaning our pools, mowing our grass, and moving goods in our warehouses. These tasks require that the robot's path achieve certain properties; for example, a vacuuming robot should cover the whole space while not visiting any particular area more frequently than others. A robot that is monitoring humidity or temperature in a warehouse may be required to follow a repeatable path so that data can be easily compared over time. In these examples and more, the strategies for controlling the robot's path may be decoupled from the specific tasks, and we envision building a library of useful high-level path controllers based on desired dynamical system properties.

Current algorithmic approaches to these tasks take two flavors: 1) maximizing the information available to the robot though high-fidelity sensors and map-generating algorithms such as SLAM, or 2) minimizing the information needed by the robot, such as the largely random navigation strategies of the early robot vacuum models. The first approach is very powerful and well-suited

to dynamic environments, but also resource-intensive (in terms of required energy, computation, and storage space). The second approach is more efficient but makes it more difficult to develop general-purpose strategies for navigation and loop-closure. We propose a dual approach. First, global geometric information about environment boundaries is provided to the system, either *a priori* or calculated online by an algorithm such as SLAM. This global geometry is then processed to produce a strategy providing strong formal guarantees and which can be executed with minimal processing power and only low bandwidth local sensors, such as bump sensors, instead of high bandwidth sensors such as cameras.

In this paper, we consider simple robots with "bouncing" behaviors: robots that travel in straight lines in the plane, until encountering an environment boundary, at which point they rotate in place and set off again. A growing line of work is considering the computational power of individual bouncing strategies and their applications to robotic tasks and applications in biological, microscale, and dynamical contexts. The environment interactions may be mechanical, with the robot actually making contact with a surface, or may be simulated with virtual boundaries such as the perimeter wire systems used by lawn mowing robots [27]. Physical implementations of the bouncing maneuver have been validated experimentally, for example in [2] and [14]. Often these lines of work consider only a small part of the strategy space, such as [10], which considers only robots that "bounce" at a single fixed angle to the normal. In this work, we present a tractable approach to reasoning over all possible bounce strategies.

This paper presents two ideas for simplifying the characterization and generation of paths of such mobile robots. First, that by using the geometric structure of a polygonal environment, we can create a combinatorial representation of the environment that lets us reason over a finite number of families of paths, instead of the infinite collection of all possible paths. Second, we can take advantage of the fact that linear functions mapping points between line segments are often *contracting*: two points under the function become closer together. We use this property to control uncertainty and synthesize strategies along with explicit descriptions of the amount of allowable actuator uncertainty for successful execution of the strategy.

2 Related Work

The motion model that we use, which relies on environment interactions, is a form of *compliant motion*, in which task constraints are used to guide task completion, even when the exact system state is not known. Our approach is related to the idea of funnelling: using the attraction regions of a dynamical system to guide states into a goal region. These ideas have been developed in the context of manipulation and fine motion control by Whitney [33], Mason [19], Erdmann [9], Goldberg [12], Lozano-Pérez, Mason, and Taylor [15], Lynch and Mason [17], and Burridge, Rizzi, and Koditschek [5], among many others.

A key aspect of our robot model is the intentional use of collisions with environment boundaries, enabled by the advent of more robust, lightweight mobile robots. Collisions as information sources for such mobile robots have also been recently explored for multi-robot systems [20]. Our specific motion model has been used in works such as [24] to determine minimal information processing requirements for mobile robot tasks. The first-class study of the motion model's dynamical properties was proposed in [10] and continued in [23]. Our work extends and improves these analysis tools, and presents new insights from the use of visibility to discretize the strategy space.

The most closely related work to ours is [14], which considered navigation for a robot that moves in a straight line until encountering an environment boundary, then rotates in place with bounded nonzero uncertainty, with the bound given as input to the task. They introduce the idea of "safe actions," which place the entire forward projected state of the robot within a single segment of the partitioned environment boundary. However, their boundary partition is computed with the robot's motion uncertainty as a fixed input. Our approach considers all possible amounts of uncertainty, and returns bounds on the required accuracy of movement as an output of the algorithm.

In [1] and [2], the authors describe navigation, coverage, and localization algorithms for a similar robot that rotates a fixed amount relative to the robot's prior heading. Their approach uses a grid discretization of the robot configuration space and searches the discretized dynamical system for paths and limit cycles. The algorithm is complete and correct up to discretization, though periodic trajectories may exist which require bounce angles not allowed by the discretization. By using a discretization induced by the environment geometry, we are able to find all possible limit cycles.

The questions we pose, and our methods of attacking the problem, are related to work on visibility in computational geometry [11]. Visibility has been considered extensively in robotics, but usually with the goal of avoiding obstacles [16, 28]. To plan over collisions, we use the edge visibility graph, analyzed in [25], and shown to be strictly more powerful than the vertex visibility graph. Our work is also related to problems in computational geometry that consider what parts of a polygon will be illuminated by a light source after multiple reflections (as if the edges of the polygon are mirrors) [3], or with diffuse reflections [26], which are related to our nondeterministic bounces.

The dynamical system defined by our robot model is related to *dynamical billiards* [32]. Along with specular reflection, modified dynamical systems have also attracted recent interest [7, 18, 32]. One very similar work was inspired by the dynamics of microorganisms; in [29], the authors show that *Chlamydomonas reinhardtii* "bounce" off boundaries at a specific angle determined by their body morphology. They characterize periodic and chaotic trajectories of such agents in regular polygons, planar curves, and other environments.

We are working toward a generalization and hierarchy of robot models, in a similar spirit to [4]. Their basic robot model, the simple combinatorial robot, is able to scan all visible vertices and detect the presence of an edge between

them. It is able to choose a destination vertex and move in a straight line to that vertex. They then augment this basic robot model with sensors and construct a hierarchy of these robot models. Our questions are related but different: if the robot is given a compact, purely combinatorial environment representation, what tasks can it accomplish with minimal sensing?

3 Model and Definitions

We consider the movement of a point robots in simple polygonal environments. All index arithmetic for polygons with n vertices is mod n throughout this paper. We do not require polygons in general position. We do not consider trajectories that intersect polygon vertices.. We define our robot to move forward in a straight line, until encountering an environment boundary (such as when a bump or range sensor is triggered). Once at a boundary, the robot is able to rotate in place. Of course, robots rarely move perfectly, so our analysis will assume the robot has some uncertainty in its motion execution, modelled nondeterministically.

More formally, the model is:

- The *configuration space* $X = P \times S^1$. P is a simple polygon with piece-wise linear boundary ∂P, and S^1 is the robot's orientation in the plane. For the most part, one can ignore configurations in the interior of the polygon and only consider transitions between points and intervals on the boundary. Let s refer to a point in ∂P without an associated robot orientation.
- The *action space* $U = [0, \pi]$, in which $u \in U$ is executed when the robot is in contact with an environment boundary; u is measured counterclockwise relative to the boundary. The action of departing the boundary while oriented at the boundary normal corresponds to $u = \pi/2$.
- The *state transition function* $f : X \times U \to X$, which describes how the state of the robot changes according to which action is chosen. We will often lift this function to act nondeterministically over sets of states and actions, propagating each state forward with each possible action, and unioning the resulting set of states.
- We model time as proceeding in stages; at each stage k, the robot is in contact with the environment boundary, executes an action u_k, and then moves forward until the next contact with the boundary at stage $k + 1$.

Definition 1. *Let $\alpha \in (0, \pi)$ be the robot's incoming heading relative to the environment boundary at the point of contact. Let $\theta \in (0, \pi)$ be a control parameter. A **bounce rule** b maps α and θ to an action u, and determines how the robot will reorient itself when it collides with a boundary. Bounce rules are defined in the frame in which the environment normal is aligned with the positive y axis and the robot's point of contact with the boundary is the origin.*

Definition 2. *A **nondeterministic bounce rule** is a bounce rule lifted to return a set of actions. In this paper, we will restrict nondeterministic bounce rules to return a convex set of actions (a single interval in $(0, \pi)$).*

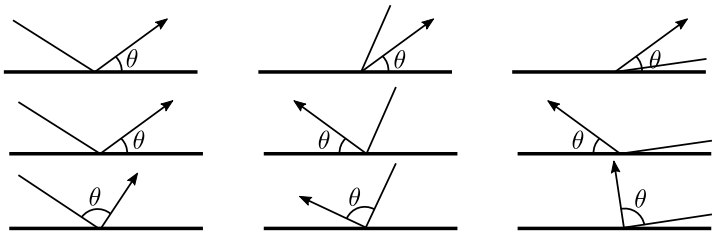

Fig. 1: Examples of different "bounce rules" that can be implemented on mobile robots. In the first row, $b(\alpha, \theta) = \theta$, which we refer to as a **fixed** bounce rule. In the second row, we have a **monotonic fixed** bounce rule, in which $b(\alpha, \theta) = \theta$ or $\pi - \theta$, depending on what is necessary to preserve monotonicity of travel direction along the x axis. In the third row, we have a **relative** bounce rule, $b(\alpha, \theta) = \alpha - \theta$, rotating α through θ in the clockwise direction. It is possible that this rotation will cause the heading of the robot to still be facing into the boundary, in which case the robot performs the rotation again until its heading points into the free space.

For example, a specular bounce (laser beams, billiard balls) has bounce rule $b(\alpha, \theta) = \pi - \alpha$. See Figure 1 for more examples of bounce rules.

Definition 3. *A **bounce strategy** is a sequence of nondeterministic bounce rules.*

Uncertainty is modelled by computing strategies which result in the robot achieving its goal as long as it executes an action in the specified *interval* at each stage.

4 Visibility-Based Boundary Partitioning

Given a polygonal environment (such as the floor plan of a warehouse or office space), we would like to synthesize bounce strategies that allow a robot to perform a given task (such as navigation or patrolling). To do so, we first discretize the continuous space of all possible transitions between points on ∂P. We find a visibility-based partition that encodes the idea of some points on ∂P having different available transitions.

Definition 4. *The **visibility polygon** of a point s in a polygon P is the polygon formed by all the points in P that can be connected to s with a line segment that lies entirely within P.*

Imagine a robot sliding along the boundary of a polygon, calculating the visibility polygon as it moves. In a convex polygon, nothing exciting happens, but in a nonconvex polygon, the reflex vertices (vertices with an internal angle greater than π) cause interesting behavior. As the robot slides, its visibility polygon may change continuously: edges shrink or grow, but the combinatorial structure of the polygon remains the same. However, when the robot aligns with a reflex vertex r and another vertex v (visible from r), a combinatorial change will occur

in the visibility polygon. Either v will become visible to the robot, adding an edge to the visibility polygon, or v will disappear behind r, removing an edge from the visibility polygon.

To compute all such points at which the visibility polygon changes structure, we must compute the **partial local sequence**, defined in [25]. The partial local sequence is a sequence of points on ∂P associated with a vertex v_0, and is constructed by shooting a ray through a reflex vertex v_0 from every visible vertex and keeping the resulting sequence of intersections with ∂P. See Figure 3 for an example of the vertices in the partial local sequence of v_0 (blue dots with accompanying rays). Each point in the partial local sequence marks the point at which a visible vertex appears or disappears behind a reflex vertex.

Thus, once all the partial local sequences have been inserted into the original polygon, the resulting points along the boundary define segments for which two points in the same interval can see the same edge set of the original polygon (though they may see different portions of those edges). Thus, the set of possible transitions from a segment becomes easier to characterize discretely. See Algorithm 1 for a pseudocode description of this partitioning process. Let P' be the polygon P with all partial local sequence points inserted.

Algorithm 1 PARTITIONPOLY(P)

Input: A polygon P as a list of vertices in counterclockwise order.

Output: A polygon P', which is P with all partial local sequence points added as new vertices.

```
v_new ← {}
reflex_verts ← GETREFLEXVERTS(P)
for v_r in reflex_verts do
    for v_vis in VISIBLEVERTS(P, v_r) do
        v_new ← v_new ∪ SHOOTRAY(v_vis, v_r)
    end for
end for
P' ← INSERTVERTS(v_new, P)
return P'
```

The SHOOTRAY function takes two visible vertices v_1 and v_2 and compute the first intersection of ΔP and ray v_1v_2. This operation will take $O(\log n)$ time after preprocessing the polygon P in $O(n)$ [31]. The VISIBLEVERTS function computes all visible vertices in the polygon given an input query vertex, and takes $O(n)$ [8]. So the total runtime of Algorithm 1 is $O(n^2 \log n)$.

Definition 5. *The **edge visibility graph** of a polygon P has a node for each edge of P, and has an arc between two nodes (e_i, e_j) if and only if there is a point s_i in the open edge e_i and a point s_j on the open edge e_j such that s_i and s_j can be connected with a line segment which is entirely within the interior of P.*

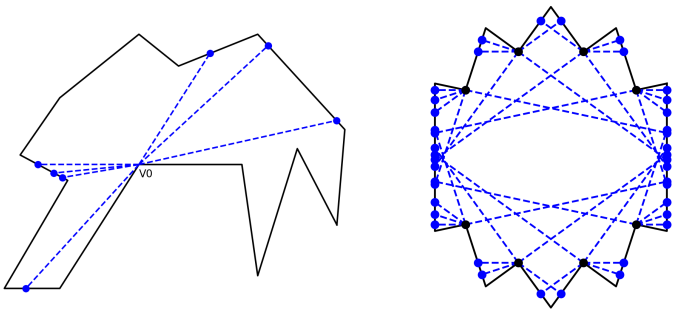

Fig. 3: On the left, the partial local sequence for v_0. On the right, an example polygon for which the bounce visibility graph has $O(n^4)$ edges.

Definition 6. *Let the* ***bounce visibility graph*** *be the directed edge visibility graph of* P'*.*

Although visibility is a symmetric property, we use directed edges in the bounce visibility graph so that we can model the geometric constraints on the visibility from one edge to another, which are not symmetric and govern what actions allow the robot to accomplish that transition. See Section 5 for further exploration of this idea.

We now define the **bounce visibility diagram**, a representation of the vertex-edge visibility structure of a partitioned polygon P'. If $|\partial P'|$ is the perimeter length of $\partial P'$, let the x axis of the bounce visibility diagram be the interval $[0, |\partial P'|)$, in which the endpoints of the interval are identified. Let the y axis of the diagram be a parameter θ, which is an angle between 0 and π. For a point $s \in \partial P'$, we can compute all the visible vertices and the angle at which they are visible. The graph of all angles at which vertices are visible from points in $\partial P'$ is the bounce visibility diagram.

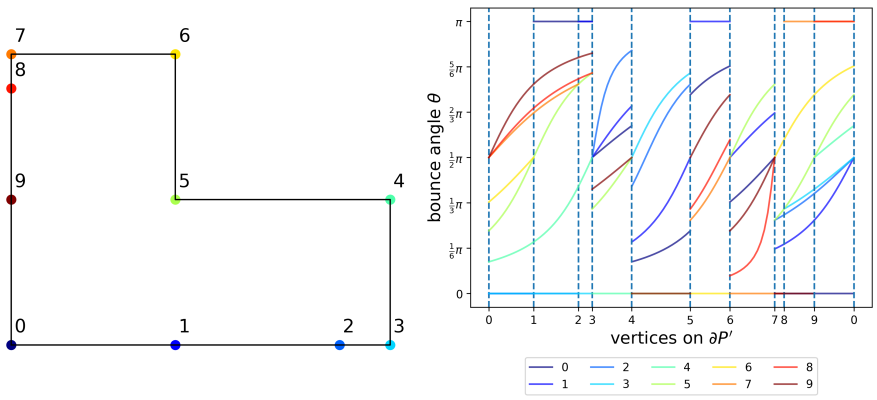

Fig. 5: A partitioned polygon and its corresponding bounce visibility diagram.

The bounce visibility diagram gives us some visual insight into the structure of the problem. First, it shows our motivation for the specific partitioning we have chosen: at each inserted vertex (1, 2, 8, and 9 in Figure 5), another vertex appears or disappears from view. Of course, such transitions also occur at the original vertices of P. Thus, the partition induced by the vertices of P and our inserted partial local sequences captures all combinatorial changes in the visible vertices.

Additionally, this diagram gives us some insight into types of possible segment-to-segment transitions under ranges of departure angles. For example, slicing the diagram vertically at a specific $s \in \partial P'$ produces a list of vertices visible from that point on the boundary (the combinatorial visibility vector) [30]. The vertical dotted lines, vertices of P', are the exact points at which the combinatorial visibility vector (cvv) changes. Within a segment (v_i, v_{i+1}), if two successive elements of the cvv are adjacent vertices, then *some* transition exists from (v_i, v_{i+1}) to a point on the edge between those vertices.

If for a segment (v_i, v_{i+1}), if there is an interval of θ such that no vertex lines are crossed from left to right, this corresponds to our notion of *safe actions*; the robot can transition from segment to segment under some amount of nondeterministic error in position and action. For example, in Figure 5, we see that a band of intervals exists at the top and bottom of the diagram with no "crossings" as you move from left to right (moving the robot around ∂P). These bands correspond to the safe actions to be formally described in Proposition 3.

Proposition 1. *The bounce visibility graph for a simple polygon with n vertices has $O(n^2)$ vertices and $O(n^4)$ edges.*

The proof of Proposition 1 is left to the Appendix.

5 Safe Actions

One major advantage of this approach is that it allows us to characterize properties of families of paths the robot may take. For example, the two paths in Figure 7 are produced by different sequences of bounce rules and visit different points of ∂P, but have the same sequence of edge collisions and have the same overall dynamical behavior (escape the room on the left, travel through hallway, then patrol the room on the right in a periodic orbit).

To characterize some of these families of paths, we will use the boundary partition technique defined in Section 4, then define *safe actions* between segments in the partition, in which safe actions are ones that are guaranteed to be successful from anywhere in a segment under an interval of bounce angles.

Definition 7. *Two edges e_i, e_j of a simple polygon are **entirely visible** to each other if and only if every pair of points $s_i \in e_i$ and $s_j \in e_j$ are visible (the shortest path between s_i and s_j lies entirely within P).*

Definition 8. *A **safe action** from edge e_i to edge e_j in a simple polygon is an action u such that $f(s, u) \in e_j$ for any $s \in e_i$ and u in some interval of actions $\tilde{\theta} \subseteq (0, \pi)$.*

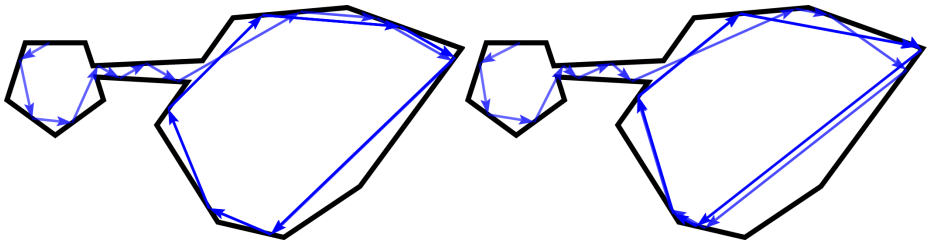

Fig. 7: Two paths demonstrating similar high-level properties with different collision points and bounce strategies.

Proposition 2. *Given two entirely visible line segments* $e_i = (v_i, v_{i+1})$ *and* $e_j = (v_j, v_{j+1})$ *in* $\partial P'$*, if a safe action exists from* e_i *to* e_j*, the maximum interval of safe actions is* $\tilde{\boldsymbol{\theta}}_{\boldsymbol{max}} = [\boldsymbol{\theta_r}, \boldsymbol{\theta_l}]$ *such that* $\theta_r = \pi - \angle v_j v_{i+1} v_i$ *and* $\theta_l = \angle v_{j+1} v_i v_{i+1}$.

For full proof, see the appendix. The sketch of the proof is that if θ_r is smaller than θ_l, every ray shot from e_i at $\theta \in [\theta_r, \theta_l]$ must intersect with e_j.

Note that this definition of a safe action is very similar to the definition of an interval of safe actions from [14]; the main differences in approach are the generation of boundary segments, methods of searching the resulting graph, and how we generate constraints on robot uncertainty instead of assuming uncertainty bounds are an input to the algorithm.

Lemma 1. *If two edges in* $\partial P'$ *are entirely visible to each other, then there will be at least one safe action between them.*

See Appendix for proof.

Corollary 1. *If two edges in* $\partial P'$ *share a vertex that is not reflex, and the two edges are not collinear, then there exist safe actions from one to the other in both directions.*

To see why Corollary 1 holds, observe that a line drawn between the non-overlapping endpoints of the edges defines a family of safe actions: any action parallel to this line or at a shallower angle is guaranteed to transition between the edges.

Definition 9. *Given two entirely visible segments* e_i *and* e_j*, rotate frame such that* e_i *is aligned with the* $x-$*axis with its normal pointing along the positive* $y-$*axis, such that segment* e_j *is above segment* e_i*. If the intersection of segment* e_i *and* e_j *would be on the left of segment* e_i*, then call the transition from* e_i *to* e_j *a* ***left transition****; if the intersection would be on the right of segment* e_i*, then call the transition a* ***right transition****.*

Proposition 3. *For every simple polygon* P *and the resulting partitioned polygon* P' *under Algorithm 1, each edge* $e \in P'$ *has at least two safe actions which allow transitions away from* e*.*

Proof. Let $e_i = (v_i, v_{i+1})$. Consider first only right transitions from e_i, where the safe action interval $\tilde{\theta}$ begins at $\theta_r = 0$ and ends at some nonzero θ_l. We will show that such a transition must exist.

By Corollary 1, if an edge e_i has an adjacent edge e_{i+1} which is not collinear or separated by a reflex angle, a safe transition exists between e_i and e_{i+1}.

If the adjacent edge is at a reflex angle, Algorithm 1 will insert a vertex in line with e_i on the closest visible edge. This vertex will be one endpoint of an edge e_k. e_k must be entirely visible from e_i, since Algorithm 1 will insert a vertex for any obstruction visible from v_i.

If the adjacent edge is collinear to e_i, the above reasoning can be extended to the next edge along ∂P which is not collinear. If e_i is an edge where both vertices were inserted by Algorithm 1, it is not immediately obvious that this will result in a safe transition, since v_i will not have transition points inserted by Algorithm 1.

All the same arguments extend symmetrically to left transitions, which will have safe actions of the form $\tilde{\theta} = [\theta_l, \pi)$.

Thus, each edge will have two guaranteed safe actions leading away from it, in some open interval of actions bordering the edge in each direction.

TODO: replace with figure proof?

□

6 Dynamical Properties of Paths

Our motion strategy allows us to disregard the robot's motion in the interior of P. Instead, the robot's state is an interval along the boundary ∂P, and we can formulate transitions as a discrete dynamical system under the transition function f. The properties of this dynamical system can be used to engineer paths corresponding to different robot task requirements.

One such generally useful property of mapping functions is *contraction*: when two points under the function always become closer together. We can use this property to control uncertainty in the robot's position, and to engineer stable limit cycles in Section 6.1.

Definition 10. *Let $\phi_{i,j}$ be the interior angle between two edges $e_i, e_j \in \partial P$.*

Definition 11. *A function g that maps a metric space M to itself is a **contraction mapping** if for all $x, y \in M$, $|g(x) - g(y)| \leq c|x - y|$, and $0 \leq c < 1$.*

In our case, we will always take M to be an interval in $\mathbb{R}$ and we will use the L1 norm.

Lemma 2. *If the transition from segment e_i to segment e_j is a left transition, then the transition function $f(x, \theta)$ between segments e_i and e_j is a contraction mapping if and only if $\theta > \frac{\pi}{2} + \frac{\phi_{i,j}}{2}$; if a right transition, the transition function $f(x, \theta)$ is a contraction mapping if and only if $\theta < \frac{\pi}{2} - \frac{\phi_{i,j}}{2}$.*

The proof is a straightforward application of the law of sines, and is included in the appendix.

Corollary 2. *For all pairs of adjacent segments not separated by a reflex angle, there exists a range of actions for which f is a contraction mapping.*

Definition 12. *The **contraction coefficient** of a transition between two line segments is the ratio of the absolute distance between points before and after. For a transition from e_i to e_j in ∂P, let the contraction coefficient be denoted as $C(\theta, \phi_{i,j})$. For a left transition, $C(\theta, \phi_{i,j}) = |\frac{\sin(\theta)}{\sin(\theta - \phi_{i,j})}|$; for a right transition, $C(\theta, \phi_{i,j}) = |\frac{\sin(\theta)}{\sin(\theta + \phi_{i,j})}|$.*

See the proof of Lemma 2 in the appendix for derivation of C. For a sequence of transitions $f_0, \ldots, f_k$, we can construct the overall mapping from the domain of f_0 to the range of f_k through function composition. Since f is a linear mapping, the contraction coefficient of a composition of multiple bounces can be determined by multiplying the contraction coefficients of each bounce.

Definition 13. *Given a sequence of m feasible transitions $F = \{f_0, f_1, \ldots, f_{m-1}\}$, at stage k the robot will be located on edge $e(k)$ and will depart the edge with action θ_k; the contraction coefficient of the overall robot trajectory $f_{m-1} \circ \ldots \circ f_0$ is $C(F) = \prod_{k=0}^{m-1} C(\theta_k, \phi_{e(k),e(k+1)})$.*

If $C(F) < 1$, the composition of F is a contraction mapping. This is true even if some transition along the way is not a contraction mapping, since $C(F)$ is simply the ratio of distances between points before and after the mapping is applied. Furthermore, the value of $C(F)$ tells us the exact ratio by which the size of the set of possible robot states has grown or shrank after F is executed.

6.1 Cycles

A very common use of contraction mappings is to prove the existence of fixed points. If a contraction mapping takes an interval back to itself, the mapping has a unique fixed point, by the Banach fixed point theorem [13]. Since the transition function of a path is the composition of the individual transition functions, we can create a self-mapping by constructing a path which returns to its originating edge. A fixed point of this mapping corresponds to a stable limit cycle. Such trajectories in regular polygons were characterized in [23]. Here, we present a more general proof for the existence of limit cycles in all convex polygons.

Definition 14. *$\phi_{P,min}$ is the smallest interior angle in a polygon P.*

Theorem 1. *For all convex polygons with n edges, there exist constant fixed-angle bouncing strategies which result in a period n limit cycle regardless of the robot's start position.*

Proof. See Figure 12 in the Appendix for the geometric setup.

Without loss of generality, assume $\theta \in (0, \frac{\pi}{2}]$. The robot will always bounce to the next adjacent edge if and only if $\theta \in (0, \min_i(\angle v_{i+2}v_iv_{i+1}))$.

Suppose we have two start positions s_1 and s_2 on edge e_0 and a constant fixed bounce r ule$b(\alpha, \theta) = \theta$. At stage k, s_1 and s_2 will be located at $f^k(s_1, \theta)$ and $f^k(s_2, \theta)$.

By Definition 13, the distance between s_1 and s_2 changes after one orbit of the polygon by the ratio $\frac{|f^n(s_1,\theta)-f^n(s_2,\theta)|}{|s_1-s_2|} = \prod_{i=0}^{n-1} C(\theta, \phi_{i,i+1})$.

If this ratio is less than one, then $f^n(s, \theta)$ is a contraction mapping from e_0 back to itself, and has a unique fixed point by the Banach fixed-point theorem [13]. By Lemma 2, this constraint is satisfied if $\theta < \frac{\pi}{2} - \frac{\phi_{i,i+1}}{2}$ for all i. We can guarantee that this condition holds for the orbit by requiring

$$\theta \in (0, \min(\min_{i=0,1,\ldots,n-1}(\angle v_{i+2}v_iv_{i+1}), \min_{i=0,1,\ldots,n-1}(\frac{\pi}{2} - \frac{\phi_{i-1,i}}{2}))),$$

in which case the fixed-angle bouncing strategy with $b(\alpha, \theta) = \theta$ leads to a convex cycle which visits each edge of P sequentially, regardless of the robot's start position. The analysis follows similarly for orbits in the opposite direction. □

Proposition 4. *For all points s on the boundary of all simple polygons, a constant fixed-angle controller exists which will cause the robot's trajectory to enter a stable limit cycle.*

Proof. First we observe that by Proposition 3, for every segment $e \in \partial P'$, safe actions always exist for two action intervals. These intervals are the ones bordering the segment itself: by staying close enough to the boundary, the robot may guarantee a safe transition. By Lemma 2, these safe actions will also admit contraction mappings. Thus, we may choose a constant fixed-angle controller such that it results in safe, contracting transitions from all segments in P'. Since there are a finite number of segments in P', this controller must result in limit cycle from every point in P. □

7 Applications

7.1 Navigation

Here, we define the *navigation* task, which in our case will assume that our robot has unavoidable uncertainty in its actuation. Thus, we wish to compute a strategy of *nondeterministic bounce rules* guaranteed to take a robot from any point in a starting set to a point in a goal set, as long as some action in the bounce angle interval is successfully executed at each stage. We require that the start and goal sets are convex (single intervals $[s_1, s_2] \subset \partial P$).

Definition 15. ***Navigation**: Given a polygonal environment P, a convex starting set $S \subset \partial P$ of possible robot positions along the environment boundary, and*

a convex goal set $G \subset \partial P$, determine a strategy π which will be guaranteed to take the robot from any point in S to a point in G, or determine that no such strategy exists.

Using the formalisms built up so far, we can perform the navigation task as a search over a discrete bounce visibility graph. Shortest paths in the graph will correspond to paths with the fewest number of bounces. Many different strategies for searching this graph could be defined; here, we will show how to search for a constant fixed-angle bounce controller, such as the one analyzed in [23] for regular polygons.

We define a few helper functions:

- MKBVG: Uses the visibility information generated in Algorithm 1 to generate the bounce visibility graph in $O(n^4)$ time.
- MKSAFEBVG: Using Definition 8 and Proposition 2, we can create a *safe roadmap*, G_{safe}, out of the bounce visibility graph by traversing all edges and removing edges with an empty $\tilde{\theta}_{max}$, and labelling the remaining edges with the interval of safe actions.

Algorithm 2 SAFECONSTANTFIXEDNAVIGATE(P, S, G, k)

Input: A polygon P, intervals S and G in ∂P, and an integer bound k.

Output: A strategy (list of bounce rules), or a special value NONE if no strategy can be found.

1: $P' \leftarrow$ PARTITIONPOLY(P)
2: $BVG \leftarrow$ MKBVG(P')
3: $BVG_{safe} \leftarrow$ MKSAFEBVG(BVG)
4: **return** SEARCHCONSTANTFIXEDSTRATS(BVG_{safe}, S, G, k)

Proposition 5. *There exist simple polygons such that under Algorithm 1, there exist edges in the partitioned polygon P' that are **not** reachable by a safe action from any other edge in P'.*

Proof. See Figure 9 for an example in which the segment between vertices 10 and 11 (node 10 in G_{safe} is not reachable via a safe action from any other segments.

The only such segments will be segments for which both endpoints are vertices inserted by Algorithm 1 or reflex vertices, since by Corollary 1, segments adjacent to non-reflex vertices of P will be reachable by a safe action. Thus, naive navigation using paths in G_{safe} is not complete.

Leveraging Contraction Mappings: Say no safe path exists between two subsets of ∂P. We may use limit cycles to reduce the uncertainty in the robot's position enough to create a safe transition.

By Proposition 4, limit cycles are reachable from any point $s \in \partial P$ under a constant fixed-angle controller (and we know the range of angles from which

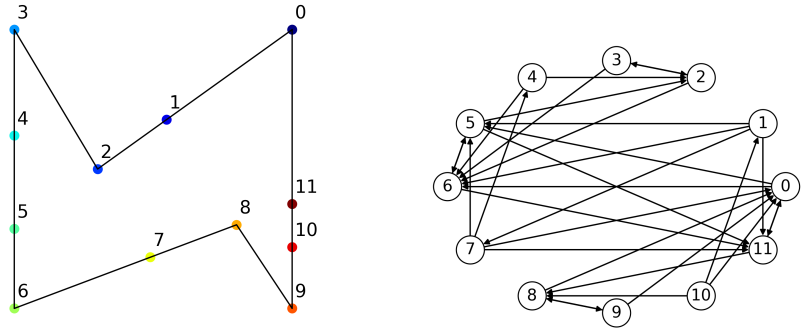

Fig. 9: A polygon after Algorithm 1 and the corresponding graph G_{safe}.

this constant controller may be drawn). Say the contraction coefficient of a given limit cycle is C, and the length of edge e_i is ℓ_i. After k iterations of the cycle, the distance between the robot's position and the fixed point of the cycle's transition function is less than $C^k \ell_i$. Thus, we may reduce the uncertainty in the robot's position to less than ϵ after $\lceil \log(\epsilon/\ell_i)/\log(C) \rceil$ iterations of the cycle. If instead of a deterministic fixed-angle controller, we have a nondeterministic controller with a range of output angles, we need to compute the resulting range of C coefficients and interval of possible fixed points, and choose k to be high enough to shrink the robot's location uncertainty enough to create new safe actions. The best way to choose such cycles to maximize allowable uncertainty, or minimize k, is still an open question.

Making Concise Strategies: The edge information along the paths can be used to create concise strategies which require only a few bounce rules. The most concise strategy is a constant controller $b(\alpha, \theta) = \theta$. To determine if such a strategy is possible for a given path, intersect all safe action intervals along the path and check if the resulting interval is nonempty. In general, deciding whether a given path may be executed with k constant fixed bounce rule controllers is the interval hitting-set problem, which is NP-hard, though in practice we are mostly interested in small k. Perhaps some geometric heuristics can be found to solve the problem more efficiently in certain cases.

7.2 Patrolling

Note that all cycles in the bounce visibility graph for which the contraction coefficient of the cycle is less than 1 in magnitude will induce limit cycles. There are exponentially many possible limit cycles, since a cycle may contain transitions which are not contraction mappings. We may search this space of cycles to solve various robotic tasks. We define the *patrolling* task as:

Definition 16. *Given an environment P, a set of possible starting states S, and a sequence of edges of the environment $E = \{e_1, \ldots, e_k\}$, determine a strategy*

which causes the robot to visit each edge of the sequence in order from any point in S.

This task is related to the Aquarium Keeper's Problem in computational geometry [6]. It may be solved by coloring nodes in the bounce visibility graph by which edge of P they belong to, and then searching for safe cycles which visit edges in the correct order. If a cycle is contracting, it will result in a converging stable trajectory. We may also define another patrolling task:

Definition 17. *Given an environment P, a set of possible starting states S, find a strategy π such that the robot's trajectory converges to a stable limit cycle reachable from all points in S.*

To solve this patrolling task, we leverage Proposition 4 which states that a limit cycle is reachable under some constant fixed-angle bounceru le from all points in ∂P. To determine if a single strategy will hold for all $s \in S$, if S spans multiple partitions in $\partial P'$, we can intersect the angle constraints for paths to cycles from S, and check for a non-empty intersection.

8 Open Questions and Future Work

We have presented a visibility-based approach to reasoning about families of paths and strategies for a class of mobile robots. This approach gives insights into the feasibility of nondeterministic control of such robots; however, many open questions remain!

Comparing bounce rules. Our approach can be used to compare different families of bounce strategies in a given polygon, by comparing the reachability of the transition systems induced by each strategy family. This would require either reducing the full bounce visibility graph given constraints on possible bounces, or constructing a more appropriate transition system. It is not clear the best way to analyze relative bounce rules, in which the outgoing angle of a robot after a collision is a function of the incoming angle. One method would be to choose a subset of configuration space to be the initial conditions of such a robot, and then propagate this set forward.

Optimal Strategies. Say we wish to find the strategy taking the robot from region S to region G with the maximum amount of allowed uncertainty in the bounce rules (the sum of the interval sizes of each action set along the path). This problem can be framed as an optimization problem over the space of all transition function compositions.

Localization. A localization strategy is a nondeterministic strategy that produces paths which reduce uncertainty in the robot's position to below some desired threshold, from arbitrarily large starting sets. The use of limit cycles to produce localizing strategies has been explored in [2], and it would be interesting to take a similar approach with our environment discretization.

Ergodic Trajectories. We are very interested in using the dynamical properties of this robot model to create controllers that produce ergodic motion, in

which the time-averaged behavior of the system approaches the space-averaged behavior, causing the robot to not spend "too much" time in any given part of the state space. Measures of ergodicity have recently been used in exploration tasks [21]. Chaotic dynamical systems have also been used directly as controllers for mobile robots [22].

Obstacles. There is no obvious reason that Algorithm 1 and the bounce visibility graph cannot be extended to polygonal regions with obstacles.

Acknowledgement This work was supported by NSF grants 1035345 and 1328018, and CONACyT post-doctoral fellowship 277028.

References

1. Alam, T., Bobadilla, L., Shell, D.A.: Minimalist robot navigation and coverage using a dynamical system approach. In: IEEE IRC (2017)
2. Alam, T., Bobadilla, L., Shell, D.A.: Space-efficient filters for mobile robot localization from discrete limit cycles. IEEE Robotics and Automation Letters (2018)
3. Aronov, B., Davis, A.R., Dey, T.K., Pal, S.P., Prasad, D.C.: Visibility with multiple reflections, pp. 284–295. Springer Berlin Heidelberg, Berlin, Heidelberg (1996). DOI 10.1007/3-540-61422-2_139
4. Brunner, J., Mihalák, M., Suri, S., Vicari, E., Widmayer, P.: Simple robots in polygonal environments: A hierarchy. In: International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics (2008)
5. Burridge, R.R., Rizzi, A.A., Koditschek, D.E.: Sequential composition of dynamically dexterous robot behaviors. International Journal of Robotics Research (1999)
6. Czyzowicz, J., Egyed, P., Everett, H., Rappaport, D., Shermer, T., Souvaine, D., Toussaint, G., Urrutia, J.: The aquarium keeper's problem. In: Proceedings of the second annual ACM-SIAM symposium on Discrete algorithms, pp. 459–464. Society for Industrial and Applied Mathematics (1991)
7. Del Magno, G., Lopes Dias, J., Duarte, P., Gaivão, J.P., Pinheiro, D.: SRB measures for polygonal billiards with contracting reflection laws. Communications in Mathematical Physics **329**(2), 687–723 (2014)
8. ElGindy, H.A., Avis, D.: A linear algorithm for computing the visibility polygon from a point. J. Algorithms **2**, 186–197 (1981)
9. Erdmann, M.A.: Using backprojections for fine motion planning with uncertainty. International Journal of Robotics Research **5**(1), 19–45 (1986)
10. Erickson, L.H., LaValle, S.M.: Toward the design and analysis of blind, bouncing robots. In: IEEE ICRA (2013)
11. Ghosh, S.K.: Visibility algorithms in the plane. Cambridge university press (2007)
12. Goldberg, K.Y.: Orienting polygonal parts without sensors. Algorithmica (1993)
13. Granas, A., Dugundji, J.: Elementary Fixed Point Theorems, pp. 9–84. Springer New York, New York, NY (2003)
14. Lewis, J.S., O'Kane, J.M.: Planning for provably reliable navigation using an unreliable, nearly sensorless robot. International Journal of Robotics Research **32**(11), 1339–1354 (2013)
15. Lozano-Pérez, T., Mason, M.T., Taylor, R.H.: Automatic synthesis of fine-motion strategies for robots. International Journal of Robotics Research **3**(1), 3–24 (1984)

16. Lozano-Pérez, T., Wesley, M.A.: An algorithm for planning collision-free paths among polyhedral obstacles. Communications of the ACM **22**(10), 560–570 (1979)
17. Lynch, K.M., Mason, M.T.: Pulling by pushing, slip with infinite friction, and perfectly rough surfaces. International Journal of Robotics Research (1995)
18. Markarian, R., Pujals, E., Sambarino, M.: Pinball billiards with dominated splitting. Ergodic Theory and Dynamical Systems **30**, 1757–1786 (2010)
19. Mason, M.T.: The mechanics of manipulation. In: Proceedings IEEE International Conference on Robotics & Automation, pp. 544–548 (1985)
20. Mayya, S., Pierpaoli, P., Nair, G., Egerstedt, M.: Collisions as information sources in densely packed multi-robot systems under mean-field approximations. In: Proceedings of Robotics: Science and Systems Conference (2017)
21. Miller, L.M., Silverman, Y., MacIver, M.A., Murphey, T.D.: Ergodic exploration of distributed information. IEEE Transactions on Robotics **32**(1), 36–52 (2016)
22. Nakamura, Y., Sekiguchi, A.: The chaotic mobile robot. IEEE Transactions on Robotics and Automation **17**(6), 898–904 (2001)
23. Nilles, A., Becerra, I., LaValle, S.M.: Periodic trajectories of mobile robots. IROS (2017)
24. O'Kane, J.M., LaValle, S.M.: Localization with limited sensing. IEEE Transactions on Robotics **23**(4), 704–716 (2007)
25. O'Rourke, J., Streinu, I.: The vertex-edge visibility graph of a polygon. Computational Geometry: Theory and Applications **10**(2), 105–120 (1998)
26. Prasad, D.C., Pal, S.P., Dey, T.K.: Visibility with multiple diffuse reflections. Computational Geometry **10**(3), 187–196 (1998)
27. Sahin, H., Guvenc, L.: Household robotics: autonomous devices for vacuuming and lawn mowing [applications of control]. IEEE Control Systems **27**(2), 20–96 (2007)
28. Siméon, T., Laumond, J.P., Nissoux, C.: Visibility based probabilistic roadmaps for motion planning. Advanced Robotics **14**(6) (2000)
29. Spagnolie, S.E., Wahl, C., Lukasik, J., Thiffeault, J.L.: Microorganism billiards. Physica D: Nonlinear Phenomena (2017)
30. Suri, S., Vicari, E., Widmayer, P.: Simple robots with minimal sensing: From local visibility to global geometry. The International Journal of Robotics Research (2008)
31. Szirmay-Kalos, L., Márton, G.: Worst-case versus average case complexity of ray-shooting. Computing **61**(2), 103–131 (1998). DOI 10.1007/BF02684409. URL http://dx.doi.org/10.1007/BF02684409
32. Tabachnikov, S.: Geometry and Billiards. American Mathematical Society (2005)
33. Whitney, D.: Force feedback control of manipulator fine motions. Transactions of the ASME, Journal of Dynamical Systems, Measurement, & Control **99**, 91–97 (1977)

9 Appendix

A Proof of Proposition 1

Proposition 1. *The bounce visibility graph for a simple polygon with n vertices has $O(n^2)$ vertices and $O(n^4)$ edges.*

Proof. Consider a polygon P with n vertices, r of which are reflex vertices. To construct the bounce visibility graph, we insert the vertices of the partial local sequence for each vertex in P. For a convex vertex, its partial local sequence will not add any new vertices to P. However, a reflex vertex can add $O(n)$ new vertices.

Up to half of the vertices in the polygon can be reflex, so the complexity of r is $O(n)$. Therefore, the number of vertices in P', returned by Algorithm 1 is $O(n^2)$. Each vertex indexes an edge in P', and thus a node in the edge visibility graph of P'. At most, a given vertex in P' may see all other vertices, so in the worst case, the bounce visibility graph will have $O(n^4)$ edges. See Figure 3 for an example of such a polygon. □

A.1 Worst Case Example for Algorithm 1

We might hope that if r is large, then not all of the reflex vertices will produce a large number of new vertices, and we may bound the size of the edge set in the visibility graph. Unfortunately, the number of reflex vertices, the new vertices produced in their partial local sequence, and the new vertices' visibility can be large at the same time. We will present a family of input polygons with bounce visibility graph edge-set size of $O(n^4)$.

Let $n = 4t+2$, in which t is a positive integer. We design a polygon with $r = 2t$ reflex vertices. The polygon is symmetric with respect to its medium horizontal line. In the top half, the reflex vertices are uniformly located on a circle and thus they are visible to each other; the convex vertices are chosen so that they are outside the circle and the line through an edge will not intersect other edges. Each reflex vertex will have at least $t-1$ new vertices in its partial local sequence. There will be $2t(t-1)+n$ vertices in the polygon after we insert all new vertices in the partial local sequence for all reflex vertices. Each of them can see at least $t(t-1)+n/2$ other vertices. Thus the number of edges in the transition graph for the polygon with inserted vertices is $O((2t(t-1)+n)(t(t-1)+n/2)) = O(t^4) = O(n^4)$. Fig 3 shows the polygon for $t = 4$ with all the vertices in the partial local sequences.

B Proof of Proposition 2

Proposition 2. *Given two entirely visible line segments $e_i = (v_i, v_{i+1})$ and $e_j = (v_j, v_{j+1})$, if a safe action exists from e_i to e_j, the maximum range of safe actions is $\tilde{\boldsymbol{\theta}}_{\boldsymbol{max}} = [\boldsymbol{\theta_r}, \boldsymbol{\theta_l}]$ such that $\theta_r = \pi - \angle v_j v_{i+1} v_i$ and $\theta_l = \angle v_{j+1} v_i v_{i+1}$.*

Proof. Let edge $e_i = (v_i, v_{i+1})$ be aligned with the positive x axis with the clockwise endpoint at the origin, without loss of generality. Due to the edges being entirely visible, $e_j = (v_j, v_{j+1})$ must be in the top half of the plane, above e_i.

Take the quadrilateral formed by the convex hull of the edge endpoints. Let the edges between e_i and e_j be $e_l = (v_i, v_{j+1})$ and the right-hand edge $e_r = (v_{i+1}, v_j)$. Let θ_l be the angle between e_l and the positive x axis ($0 < \theta_l < \pi$); similarly for e_r and θ_r. See Figure 10 for an illustration of the setup.

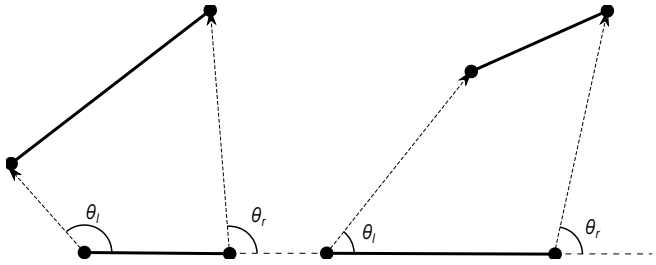

Fig. 10: Angle range such that a transition exists for all points on originating edge (left: such a range exists, right: such a range does not exist)

There are three cases to consider: if e_l and e_r are extended to infinity, they cross either above or below edge e_i, or they are parallel.

Case 1: e_l and e_r meet below edge e_i. In this case, $\theta_l > \theta_r$ and if a ray is cast from any point on e_i at angle $\theta \in [\theta_r, \theta_l]$, the ray is guaranteed to intersect e_j in its interior.

Case 2: e_l and e_r meet above edge e_i. In this case, $\theta_l < \theta_r$, and there is no angle θ such that a ray shot from *any* point on e_i will intersect e_j. To see this, imagine sliding a ray at angle θ_l across the quadrilateral - at some point before reaching v_{i+1}, the ray must stop intersecting e_j, else we would have $\theta_l > \theta_r$.

Case 3: e_l and e_r are parallel. This implies that $\theta_l = \theta_r$, which is the only angle for which a transition from any point on e_i is guaranteed to intersect e_j, and $\tilde{\theta}_{max}$ is a singleton set.

Thus, for each case, we can either compute the maximum angle range or determine that no such angle range exists. □

C Proof of Lemma 1

Lemma 1. *If two segments are entirely visible to each other, there will be at least one safe action between them.*

Proof. From the proof of Proposition 2, we can see that if case one holds in one direction, case two will hold in the other direction, so a safe action must exist from one edge to the other in one direction. If case three holds, there is a safe action both directions but $\tilde{\theta}_{max}$ is a singleton set.

D Proof of Lemma 2

Lemma 2. *If the transition from segment e_i to segment e_j is a left transition, then the transition function $f(x, \theta)$ between segments e_i and e_j is a contraction mapping if and only if $\theta > \frac{\pi}{2} + \frac{\phi_{i,j}}{2}$; otherwise, the transition function $f(x, \theta)$ is a contraction mapping if and only if $\theta < \frac{\pi}{2} - \frac{\phi_{i,j}}{2}$.*

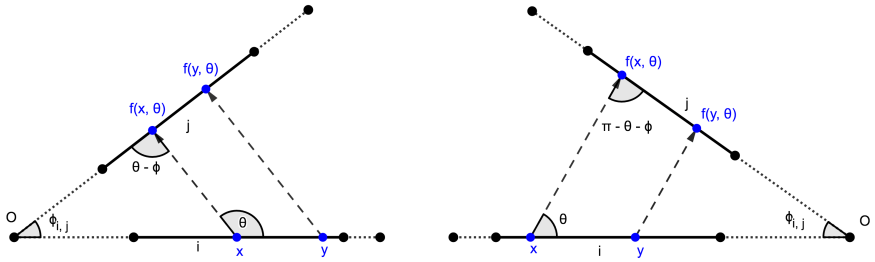

Fig. 11: The two cases for computing contraction mapping conditions.

Proof. We consider the two cases of transition separately:

1. For the transition shown in the left hand side of Figure 11, $\overline{xf(x,\theta)} \parallel \overline{yf(y,\theta)} \Rightarrow \frac{|f(x,\theta)-f(y,\theta)|}{|x-y|} = \frac{|f(x,\theta)|}{|x|} = \frac{\sin(\pi-\theta)}{\sin(\theta-\phi_{i,j})} = \frac{\sin(\theta)}{\sin(\theta-\phi_{i,j})}$. The transition will be contraction if and only if $\frac{|f(x,\theta)-f(y,\theta)|}{|x-y|} < 1 \iff \sin(\theta) < \sin(\theta - \phi_{i,j})$. If $\theta < \frac{\pi}{2}$, then $\sin(\theta) > \sin(\theta - \phi_{i,j})$. Thus we need $\theta > \frac{\pi}{2}$. If $\theta - \phi_{i,j} > \frac{\pi}{2}$, then $\sin(\theta) < \sin(\theta - \phi_{i,j})$ and we are done; otherwise we need $\pi - \theta < \theta - \phi_{i,j} \Rightarrow \theta - \frac{\phi_{i,j}}{2} > \frac{\pi}{2}$. Combining all conditions, we have the transition will be contraction if and only if $\theta > \frac{\pi}{2} + \frac{\phi_{i,j}}{2}$.
2. Similarly, for a right transiton shown in the right diagram of figure 11, $\overline{xf(x,\theta)} \parallel \overline{yf(y,\theta)} \Rightarrow \frac{|f(x,\theta)-f(y,\theta)|}{|x-y|} = \frac{|f(x,\theta)|}{|x|} = \frac{\sin(\pi-\theta)}{\sin(\pi-\theta-\phi_{i,j})} = \frac{\sin(\theta)}{\sin(\theta+\phi_{i,j})}$. The transition will be contraction if and only if $\frac{|f(x,\theta)-f(y,\theta)|}{|x-y|} < 1 \iff \sin(\theta) < \sin(\theta + \phi_{i,j})$. If $\theta > \frac{\pi}{2}$, then $\sin(\theta) > \sin(\theta + \phi_{i,j})$. Thus we need $\theta < \frac{\pi}{2}$. If $\theta + \phi_{i,j} < \frac{\pi}{2}$, then $\sin(\theta) < \sin(\theta + \phi_{i,j})$ and we are done; otherwise we need $\theta < \pi - \theta - \phi_{i,j} \Rightarrow \theta < \frac{\pi}{2} - \frac{\phi_{i,j}}{2}$. Combining all conditions, we have the transition will be contraction if and only if $\theta < \frac{\pi}{2} - \frac{\phi_{i,j}}{2}$.

□

E Supplementary Figure for Theorem 1

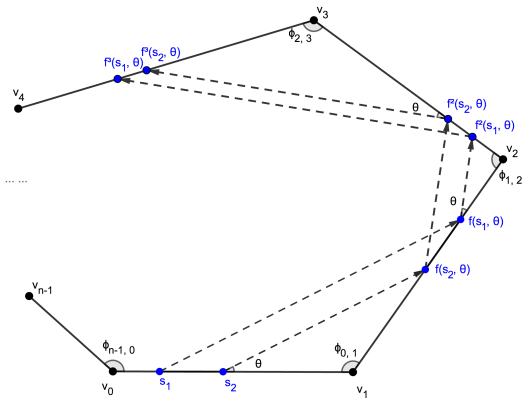

Fig. 12: The notation setup for the proof of contracting cycle in a convex polygon.