# Robust Combinatorial Planning over Simple Boundary Interactions

Alexandra Q. Nilles
Department of Computer Science
University of Illinois at Urbana-Champaign
Email: nilles2@illinois.edu

Steven M. LaValle
Department of Computer Science
University of Illinois at Urbana-Champaign; and
Faculty of Information Technology and Electrical Engineering
University of Oulu, Oulu, Finland
Email: lavalle@illinois.edu

***Abstract*—Many mobile robots, such as vacuum bots, are now able to move reliably in straight lines and detect when they have encountered a physical or virtual obstacle. By planning explicitly over reorientation actions that the robot takes when encountering a boundary, we may avoid needing to compute high-fidelity state estimates and can also take advantage of the natural stabilizing dynamics of these behaviors. We model the robot motion as transitions between visible points on a polygonal environment and analyze the dynamical properties of the resulting 1D map. Using a geometric partitioning of the dynamical system, we can reason about plans with different levels of robustness and safety under nondeterminism in actuation. This approach also allows for reasoning about families of paths with similar topological and dynamical properties while planning, allowing us to choose robust members of those families, information which would be lost with a sampling-based approach. This poster will focus on recent work on development of task and motion primitives in this setup, working toward a usable high-level language with formal guarantees on the robustness of resulting low-level plans.***

## I. Introduction

For many years, roboticists have aimed to have their robots avoid obstacles, and for good reason! However, as robots become sturdier, we may begin to ask how interactions between a robot and its environment may be useful as a source of information and actuation. Specifically, we are interested in robots that can controllably "bounce" off boundaries, moving blindly forward in the interior of their environment between collisions. Given some knowledge of the environment and the initial position of the robot, can we create plans that just tell the robot what to do when it encounters a barrier? How robust are these plans to uncertainty in the bounce action?

This general motion model is applicable to many different physical systems. It is inspired by the motion of ground-based mobile robots, such as vacuum robots, and there is a developing line of work on how this motion strategy can enable minimalist algorithms for navigation and coverage [4] [1] [2]. Our approach could also be applied to planning planar motion of drones, to create reliable plans based on sensing sporadic boundary-crossing events, avoiding power-hungry systems such as GPS. This approach also applies to control of micro-organisms and small scale robots where environment interactions are common and engineerable, and the robots are not always fully controllable. The motion scheme described here can be implemented through mechanical design of self-propelling micro-robots. Some microorganisms have similar motion profiles and have inspired highly related dynamical systems research [6]. In general, this model is most useful for resource-constrained robots in environments with obstacles or other interesting environment geometry.

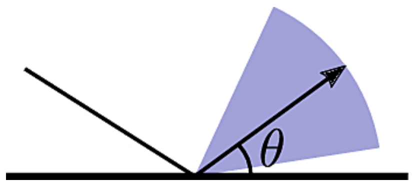

Fig. 1. Definition of an action $\theta$ taken when the robot encounters an environment boundary. The cone indicates the type of uncertainty that we allow in our model.

## II. Problem Statement

We assume our planner is given an exact, polygonal representation of the environment $P$, and a description of the start and goal positions ($S$ and $G$) as points in the boundary of the environment, $\partial P$. In general, we allow interior polygonal obstacles.

Our planner is an event-based planner, and outputs strategies in the form of sequences of *bounce rules* to be applied sequentially as the robot encounters boundaries. Bounce rules specify the desired heading of the robot when it next leaves the boundary, as shown in Figure 1. They specify the action to be taken at a particular stage, where actions are angles $\theta$ from an action set $U = [0, \pi]$.

This reorientation can be implemented in various ways; yaw control combined with sensors (ie: scanners) that can identify the boundary orientation relative to the robot, or even mechanically (imagine a robot which aligns a planar body part to the wall, and rotates the rest of its body to the desired heading). Our work focuses on the high-level plans, assuming this reorientation can be executed somewhat reliably.

## III. Combinatorial Reasoning

Since we assume the robots move in straight lines (with some possible uncertainty in heading), our planner first discretizes the polygon boundary using *visibility events*, points on the boundary where edges of the polygon pass in and out of

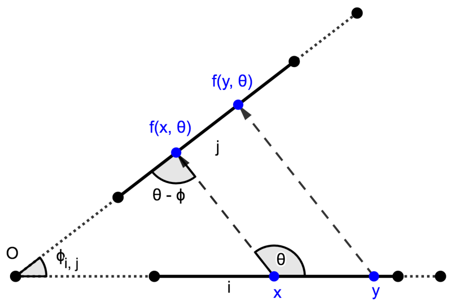

Fig. 2. The geometric set-up for calculating the contraction coefficients between two line segments (solid black segments). The robots start on the lower, horizontal segment, indexed by $i$. They are transitioning under angle $\theta$ to edge $j$. Edges $i$ and $j$ have an internal angle of $\phi_{i,j}$.

visibility as the robot slides along the boundary. This approach allows us to compute a roadmap of all the *safe* transitions (where any two points on a given segment along the boundary can transition to the same edge under a set of actions). For more details on this discretization, see [5]. This planner can be used to generate paths that are safe under uncertainty in actuation, and is able to report the action set required at each stage in the plan. A robot with some intrinsic error (such as shown in Figure 1) should then aim at the center of this set at each stage. However, by including dynamical information into this roadmap, we can improve the completeness and robustness of the planner.

## IV. Useful Dynamical Properties

We define the dynamical system $f : \partial P \times U \to \partial P$ which is the geometrically determined mapping between points on the boundary. In polygons, the resulting dynamical system is a piecewise function, with a different form for each pair of mutually visible edges. Sometimes, transitions between straight-line boundaries have the property of *contraction*: if the same action is taken from two different points, the distance between the next locations is smaller than the distance between the start points.

We define a *contraction coefficient* for each pair of mutually visible edges in the polygonal environment as $c(\theta, i, j) = \frac{|f(x,\theta)-f(y,\theta)|}{|x-y|}$, where $f$ is a contraction mapping if $c(\theta, i, j)$ is less than one. Using geometric reasoning, this contraction coefficient is calculated to be [1]

$$c(\theta, i, j) = \frac{\sin(\theta)}{\sin(\theta \pm \phi_{i,j})}.$$

See Figure 2 for an example of the geometric scenario.

It is also important to note that $f$ is linear in $x$, so when transitions are composed together, the contraction coefficient of the overall transition is the product of the individual transitions. This lets us construct trajectories which may have individual steps that allow the set of possible robot locations to grow, but that are overall robust and reduce uncertainty.

To plan while taking these properties into account, we can compute a further discretization of the polygon boundary. We are currently finely discretizing $\theta$ and storing the contraction coefficients along the environment boundary in the roadmap. However, this leads to a large branching factor in the search for plans. We are currently implementing a method where we compute values of $\theta$ that cause combinatorial changes in the resulting discretization (*critical angles*, as defined in [3]), and planning over actions that are not near critical angles, decreasing the sensitivity of resulting strategies to modelling errors.

## V. Task-Motion Primitives

Our understanding of the dynamics of these systems is developed to the point where we can easily compute reachable sets and long-term dynamical behaviors of different strategies. To make these systems more generally useful, we are developing some higher-level utilities for our planning tool. For example, we may wish for a robot to repeatedly visit a few different regions of the environment, or we may wish to generate strategies that only use a few different bounce actions (for mechanical designs in micro-scale settings). The first is a *spatial* task, and the second is a *behavioral constraint*, and these types of tasks and constraints can be combined.

Generally this is accomplished with different types of searches in the generated roadmap. For example, we can label sections of the roadmap with different "colors" and using a search algorithm to find safe plans which visit each region. Algorithms such as $A^*$ search can be used to find sequences of transitions which are optimally contracting (maximally reduce uncertainty in robot position and effect of nondeterminism). We can also search for strategies which use maximally large nondeterministic actions, which allow for strategies which are maximally tolerant of nondeterminism. The constraint of only using one or a few bounce angles is again a type of search, where the interval of bounce angles that will admit a path is updated at each transition and the search path is abandoned if the interval becomes empty.

## VI. Future Work

We are in active development of an interactive interface which allows user to specify or import polygonal maps and design different types of trajectories using a small API which defines spatial tasks and behavioral constraints [2].

Additionally, some interesting theoretical questions remain which may impact more general classes of problems. The contraction property described in this work holds not only in the two dimensional setting. In three dimensions, for example, similar conditions can be described for straight-line paths between two planar boundaries. We are developing the theory for more general cases and are interested in how our approach of making principled, geometric discretizations combined with

[1]The type of the operator in the denominator depends on if the segments would intersect on the right or the left of segment $i$ (in the case where the lines are parallel, $c(\theta, i, j) = 1$ always).

[2]see https://github.com/alexandroid000/bounce_viz for more documentation

dynamical classification of transitions can be applied in more general settings.

## Acknowledgments

We would like to acknowledge Israel Becerra and Samara Ren for their significant contributions to the project.

## References

[1] T. Alam, L. Bobadilla, and D. A. Shell. Minimalist robot navigation and coverage using a dynamical system approach. In *IEEE IRC*, 2017.

[2] Tauhidul Alam, Leonardo Bobadilla, and Dylan A Shell. Space-efficient filters for mobile robot localization from discrete limit cycles. *IEEE Robotics and Automation Letters*, 2018.

[3] L. H. Erickson and S. M. LaValle. Toward the design and analysis of blind, bouncing robots. In *IEEE ICRA*, 2013.

[4] Jeremy S. Lewis and Jason M. O'Kane. Planning for provably reliable navigation using an unreliable, nearly sensorless robot. *International Journal of Robotics Research*, September 2013.

[5] Alexandra Q Nilles, Yingying Ren, Israel Becerra, and Steven M LaValle. A visibility-based approach to computing nondeterministic bouncing strategies. In *The Workshop on Algorithmic Foundations of Robotics*, 2018.

[6] S. E. Spagnolie, C. Wahl, J. Lukasik, and J. L. Thiffeault. Microorganism billiards. *Physica D: Nonlinear Phenomena*, 2017.