UNIVERSITY OF CALIFORNIA
Santa Barbara

Geometric Pursuit Evasion

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

by

Kyle Thomas Klein

Committee in Charge:

Professor S. Suri, Chair

Professor J. Gilbert

Professor J. Hespanha

Professor V. Isler

June 2014

The Dissertation of
Kyle Thomas Klein is approved:

Professor J. Gilbert

Professor J. Hespanha

Professor V. Isler

Professor S. Suri, Committee Chairperson

March 2014

Geometric Pursuit Evasion

by

Kyle Thomas Klein

To my love Jen, who makes every day brighter.

Acknowledgements

First and foremost, I would like to thank my advisor Subhash Suri for his invaluable guidance and immense patience over the years. I am very grateful for all the time he spent working with me and thankful that I was able to find an advisor whom I was able to work so well with. Additionally, I would like to thank the members of my committee, John Gilbert, Joao Hespanha, and Volkan Isler for their feedback which has helped shape this dissertation.

I'd also like to again thank Volkan Isler and his former student Deepak Bhadauria for their collaboration with us. I am also grateful to the many anonymous reviewers who provided useful feedback on improving the papers that have been included in this dissertation.

Additionally I would also like to acknowledge my current and former lab mates Luca Foschini, Pegah Kamousi, Hakan Yıldız, Kevin Verbeek, Lucas Bang, and Jonathan Sun for their assistance and support over the years.

Finally I would like to thank my parents and sister for their continued support and encouragement throughout this entire process. Without them, such a long, and at times frustrating endeavor would have been far more difficult.

Curriculum Vitæ

Kyle Thomas Klein

Education

2010-2014 (Expected)	PhD in Computer Science, University of California, Santa Barbara. 4.0 GPA Advisor: Subhash Suri
2010-2013	Master of Science in Computer Science, University of California, Santa Barbara.
2007-2010	Bachelor of Science in Computer Science, University of California, Santa Barbara. 3.98 GPA, Highest Honors

Experience

2010-2014	Graduate Research Assistant, UC Santa Barbara
Summer 2012, 2013	Software Engineering Intern, Google, Mountain View
2011-2012	Teaching Assistant, UC Santa Barbara
Summer 2011	Software Engineering Intern, Google, Santa Monica
2009-2010	Undergraduate Researcher, UC Santa Barbara

Selected Publications

K. Klein and S. Suri. **Trackability with Imprecise Localization**. Under review.

K. Klein and S. Suri. **Pursuit Evasion on Polyhedral Surfaces**. Under review (journal).

K. Klein and S. Suri. **Capture Bounds for Visibility-Based Pursuit Evasion**. Under review (journal).

K. Klein and S. Suri. **Pursuit Evasion on Polyhedral Surfaces**. In *International Symposium on Algorithms and Computation (ISAAC 2013), pages 284–294, December 16-18, 2013*.

K. Klein and S. Suri. **Capture Bounds for Visibility-Based Pursuit Evasion**. In *Symposium on Computational Geometry (SoCG 2013)*, pages 329–338, June 17-20, 2013.

K. Klein and J. Neira. Nelder-Mead Simplex **Optimization Routine for Large Scale Problems: A Distributed Memory Implementation**. In *Computational Economics, 43(4), 2014*.

D. Bhadauria, K. Klein, V. Isler, and S. Suri. **Capturing an Evader in Polygonal Environments with Obstacles: The Full Visibility Case**. In *International Journal of Robotics Research (IJRR), 31(10), 2012*.

K. Klein and S. Suri. **Catch me if you can: Visibility-based Pursuit and Capture in Environments with Polygonal Obstacles.** In *Pro-*

ceedings of 26th Conference on Artificial Intelligence (AAAI), pages 2010–2016, July 22-26, 2012.

K. Klein and S. Suri. **Complete information pursuit evasion in polygonal environments.** In *Proceedings of 25th Conference on Artificial Intelligence (AAAI)*, pages 1120–1125, Aug 7-11, 2011.

K. Klein and S. Suri. **Multiagent pursuit evasion, or playing kabaddi.** In *Proceedings of 9th International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, pages 89–104, Dec 13-15, 2010.

K. Klein and S. Suri. **Robot kabaddi.** In *Proceedings of 22nd Canadian Conference on Computational Geometry (CCCG)*, pages 79–82, Aug 9-11, 2010.

Awards

NSF Graduate Research Fellow 2012

Best Paper Award

Complete Information Pursuit Evasion in Polygonal Environments, *UC Santa Barbara Graduate Workshop on Computing*, Oct 7, 2011

Outstanding TA Award

CS 56 - Advanced Applications Programming (Spring 2011)

CS 56 - Advanced Applications Programming (Winter 2011)

NSF Graduate Research Fellowship 2011 Honorable Mention

Abstract

Geometric Pursuit Evasion

Kyle Thomas Klein

In this dissertation we investigate pursuit evasion problems set in geometric environments. These games model a variety of adversarial situations in which a team of agents, called pursuers, attempts to catch a rogue agent, called the evader. In particular, we consider the following problem: how many pursuers, each with the same maximum speed as the evader, are needed to guarantee a successful capture? Our primary focus is to provide combinatorial bounds on the number of pursuers that are necessary and sufficient to guarantee capture.

The first problem we consider consists of an unpredictable evader that is free to move around a polygonal environment of arbitrary complexity. We assume that the pursuers have complete knowledge of the evader's location at all times, possibly obtained through a network of cameras placed in the environment. We show that regardless of the number of vertices and obstacles in the polygonal environment, three pursuers are always sufficient and sometimes necessary to capture the evader. We then consider several extensions of this problem to more complex environments. In particular, suppose the players move on the surface of a 3-dimensional polyhedral body; how many pursuers are required to capture the evader? We show that 4 pursuers always suffice (upper

bound), and that 3 are sometimes necessary (lower bound), for any polyhedral surface with genus zero. Generalizing this bound to surfaces of genus g, we prove the sufficiency of $(4g + 4)$ pursuers. Finally, we show that 4 pursuers also suffice under the "weighted region" constraints, where the movement costs through different regions of the (genus zero) surface have (different) multiplicative weights.

Next we consider a more general problem with a less restrictive sensing model. The pursuers' sensors are visibility based, only providing the location of the evader if it is in direct line of sight. We begin my making only the minimalist assumption that pursuers and the evader have the same maximum speed. When the environment is a simply-connected (hole-free) polygon of n vertices, we show that $\Theta(n^{1/2})$ pursuers are both necessary and sufficient in the worst-case. When the environment is a polygon with holes, we prove a lower bound of $\Omega(n^{2/3})$ and an upper bound of $O(n^{5/6})$ pursuers, where n includes the vertices of the hole boundaries. However, we show that with realistic constraints on the polygonal environment these bounds can be drastically improved. Namely, if the players' movement speed is small compared to the features of the environment, we give an algorithm with a worst case upper bound of $O(\log n)$ pursuers for simply-connected n-gons and $O(\sqrt{h} + \log n)$ for polygons with h holes.

The final problem we consider takes a small step toward addressing the fact that location sensing is noisy and imprecise in practice. Suppose a tracking agent wants to follow a moving target in the two-dimensional plane. We investigate what is the tracker's

best strategy to follow the target and at what rate does the distance between the tracker and target grow under worst-case localization noise. We adopt a simple but realistic model of relative error in sensing noise: the localization error is proportional to the true distance between the tracker and the target. Under this model we are able to give tight upper and lower bounds for the worst-case tracking performance, both with or without obstacles in the Euclidean plane.

Professor S. Suri
Dissertation Committee Chair

Contents

List of Figures

Introduction

Pursuit evasion problems ask a question of the following form: what strategy should a team of pursuers use to catch an adversarial evader? Due to the numerous variations of the capabilities of the pursuers and the evader, as well as definitions of what it means to "catch", an enormous number of challenging problems have been considered. One such problem is that of a team of cops chasing a fleeing robber. While the cops may have a high success rate by applying the intuitive strategy of simultaneously chasing and surrounding the robber, there is no guarantee that a robber, either through their own cunning or ignorance, will not escape its pursuers. Pursuit evasion games seek to not only formalize and prove the success of such an algorithm, but also its optimality under a variety of metrics such as duration, numbers of cops, etc. Given such a formal bound, we can then be certain that whenever possible the robber will be apprehended as efficiently as possible.

The research into these pursuit evasion problems has a rich history dating back as far as the 1930's when Rado posed the now classic lion and man problem (discussed

in Section 0.1). In the last eighty or so years, there has been significant work done in a variety of areas, motivated both by natural applications in surveillance, tracking, monitoring, military strategy, and search-and-rescue, and by mathematical richness and complexity that underlie these problems. Indeed, the literature on pursuit evasion problems spans a surprisingly broad list of areas from applied ones such as robotics, artificial intelligence, sensor networks and control systems, to theoretical ones such as mathematics, graph theory, game theory, algorithms and computational geometry. As a result an enormous and highly diverse literature has emerged dealing with many aspects of pursuit evasion. This dissertation focuses exclusively on *algorithmic* research, and will not discuss many other aspects dealing with strategies and counter-strategies (game theory), differential games (control theory), motion control, and feedback etc.

In particular we consider the problem of capture in geometric environments, where a team of pursuers is tasked with locating and capturing an adversarial evader. Our primary focus is on finding combinatorial bounds on the number of pursuers that are necessary and sufficient to capture the evader. Guibas et al. introduced a formal framework and analysis of visibility-based pursuit in complex polygonal environments [25], however, in order to make the problem tractable, they made one crucial simplifying assumption: *the evader loses if it is "seen" by any pursuer.* That is, the pursuers need to only detect the presence of the evader, and not physically catch it. Under this detection model, Guibas et al. managed to prove several interesting combinatorial bounds, namely, $\Theta(\log n)$

pursuers are always sufficient and sometimes necessary to locate an evader in a simply connected n-gon, and $\Theta(\sqrt{h} + \log n)$ in the presence of h obstacles. However, until our work there had been little progress on extending their detection bounds to the physical capture we consider.

Our work begins by attempting to disentangle two orthogonal issues inherent in pursuit evasion: localization, which is purely an informational problem, and capture, which is a problem of planning physical moves. In particular, we ask how complex is the capture problem if the evader localization is available for free? From here, our work follows a natural progression in regards to both the sensor capabilities and the complexity of the environments considered. As a result, we are able to build on our results to find combinatorial bounds on the number of pursuers required to catch an evader matching the detection bounds of Guibas et al. Additionally, we extend our results to more realistic real world scenarios by considering three dimensional environments as well as consider the sensor noise which will be present in any practical application.

0.1 Related Work

The history of pursuit evasion in mathematics can be traced back to the 1930s when Rado first posed the following puzzle: A lion and a man (each viewed as a single point) in a closed disc have equal maximum speeds; can the lion catch the man? The *apparent* answer to the problem is that the lion can win by the following strategy: move with

maximum speed in such a way that he always lies on the radius vector from the center to the man. However, this "conventional wisdom" was shown to be wrong in 1952 by Besicovitch, who showed a strategy for the man to survive forever [48]. Besicovitch's argument has the following form.

> Split time into a sequence of intervals, of lengths $t_1, t_2, t_3, \ldots$. At the ith step, the man runs for time t_i in a straight line that is perpendicular to his radius vector at the start of the step. He chooses to run into the half plane that does not contain the lion, so certainly the lion does not catch the man in this time step. The man then repeats this procedure for the next time step, and so on. Besicovitch shows that there exists a series of time intervals whose sum is infinite, such that the length of the radius vector remains finite. Thus, the man can run from the lion forever, while always remaining within the disc.

This simple argument highlights one of the aspects of pursuit evasion problems which makes them so attractive; while the questions posed are natural, their solutions are often surprising and non-trivial. As a result an enormous and highly diverse literature has emerged dealing with many aspects of pursuit evasion. Many different variations of the problem are studied depending on the nature of the environment (discrete vs. continuous, occluded vs. unoccluded), speed and movement constraints, capture rules, etc., and under colorful names such as Cops-and-Robbers [31], Hunter-and-Rabbit [27, 30], Homicidal Chauffeur [12], and Princess-and-Monster games [7].

Broadly speaking, the algorithmic formulations of pursuit evasion problems fall into two categories: discrete and continuous. We begin by discussing the former, which

primarily considers pursuit evasion on a graph and lays much of the groundwork for the continuous geometric spaces which is the domain of this work.

0.1.1 Pursuit Evasion in Graphs

Graph-based pursuit evasion has received significant interest for a variety of reasons. For one, graphs are a natural model for many possible applications, such as search in buildings or caves, or even inoculating a spreading virus in a computer network through strategic placement of antivirus software. In addition, their discrete nature can often make reasoning about their properties simpler than continuous models. While there are many pursuit evasion games set in graphs, the general idea is that pursuers and evaders traverse edges, the pursuers win if they occupy the same node or edge as the evader, and the evader wins if it can indefinitely avoid the pursuers. Though the literature covers a vast number of models, they primarily differ in the movement capabilities of the players, or the information available to either one or both of the pursuers and the evaders.

Graph Searching

Graph searching generally has the following setup. The evader can move along edges of the graph, and the pursuers can execute three moves: place a pursuer at a node; remove a pursuer from a node; and *clear an edge* by traversing it from one endpoint to the other. The pursuer's objective is then to have every edge in the graph simultaneously clear, however, an edge only remains clear as long as any path from a contaminated edge to a

clear edge is contains at least one pursuer. Edge search then asks, what is the minimum amount of pursuers required to clear all edges in the graph? Edge search was first studied in the 1970s by Parsons, who imagined the problem of attempting to find a lost spelunker in a network of caves [59]. A surprising result was shown by LaPaugh [45], namely, a search strategy using the optimal number of pursuers can always be converted to a search strategy that avoids recontamination, that is, no edge needs to be cleared twice. Unfortunately, even though LaPaugh's result shows that an optimal solution exists with a polynomial number of moves, it was later shown by Megiddo [50] that computing the minimum number of pursuers is NP-Complete for general graphs.

Since the original work by Parsons, many related models have been studied. Some examples of other graph searching games includes *node search* [36], where an edge is cleared when both of its incident nodes are occupied by pursuers, and *mixed search* [10], which allows clearing via either node or edge searching. Let $es(G)$, $ns(G)$, and $ms(G)$ denote the minimum number of pursuers required to search a graph G using edge, node, and mixed search respectively. Then the following inequalities are known due to [10, 36]:

$$ns(G) - 1 \leq es(G) \leq ns(G) + 1$$

$$ns(G) \leq ms(G) + 1$$

$$es(G) \leq ms(G) + 1$$

While all three problems are NP-Complete [36, 50], they are closely related to other well studied graph problems with known approximation algorithms. For example, it is known that the vertex separation number of a graph G, denoted $vs(G)$ obeys the equality $ns(G) = vs(G) + 1$ [36]. In addition, it was shown that the path width of a graph G, denoted $pw(G)$, is equal to its vertex separator, implying that $ns(G) = pw(G) + 1$. While path-width is NP-Complete, there is an $O((\log n)^{3/2})$ approximation algorithm, and for a fixed k, a solution can be computed linear time, though it is only practical for very small values of k.

Another well studied graph searching problem, sometimes called helicopter search, considers node search but with a unoccluded evader, that is, the pursuers always know the location of the evader [65]. Notice that instead of a question of de-contaminating the graph, this is a question of pinning the evader so that it has no escape route. This results in different bounds than that of Parson's original model, for example, it turns out that a tree can be searched with two pursuers, as opposed to the $\Omega(\log n)$ required for edge search on complete ternary trees [59]. Helicopter search is also closely related to the node search problem studied in [36], where the evader is lazy and only moves if it is about to lose. In fact, for both the visible and lazy evader [17, 65], it was shown that the search number of the graph is exactly one more than its tree width, which can be used in a reduction to show both problems are NP-Hard. However, similar to path width,

tree-width is fixed parameter tractable and an $O(\sqrt{\log n})$ approximation algorithm is known.

Cops and Robbers

The general graph-searching problem allows pursuers to "teleport" (jump from one vertex to any other in a single move), and considers arbitrarily fast moving evaders. In the Cops-and-Robbers game, the agents alternate turns and can only move to a neighboring node in a single move, thus giving the game a more natural physical interpretation. Typically, the game assumes the robber is unoccluded, and ends if a cop reaches the same node as the robber, or if the robber can indefinitely avoid the cops. Cops-and-Robbers was first studied independently in [57, 60], which showed that a single cop only wins if the graph is a member of a special class of graphs, namely dismantlable graphs. However, the question of the minimum number of cops needed to catch the robber for a graph G, often called the cop number and denoted $c(G)$, was not addressed. Since these original investigations, it has been shown that if the initial locations of the cop and robber are given, determining whether k cops can capture a robber is EXP-TIME Complete [20, 23].

An open question today is to understand the maximum cop number of general graphs. Aigner and Fromme [2] have shown that n node graphs with no cycles smaller than length 5 require at least as many cops as the minimum degree among all nodes of G. This combined with knowledge of dense graphs gives a lower bound of $\Omega(\sqrt{n})$, which

was conjectured to be tight by Meyniel. However, the best known upper bound is far from realizing this lower bound, indeed, recently the upper bound was decreased to $O(n/\log n)$ [16], and then again to $O(n \cdot 2^{-(1+o(1))\sqrt{\log n}})$ [64]. Other strategies of upper bounding relate to the genus of the graph, for example, Aigner and Fromme showed that a planar graph, which has genus zero, has cop number at most three. Schroeder extended this to graphs of arbitrary genus g, proving a cop number of at most $\lfloor \frac{3g}{2} \rfloor + 3$, however, this bound is not known to be tight. Additional work has studied special classes of graphs such as Cayley [21], chordal [31], random [11], and graph products [55], as well as altered models such as partial information [31].

0.1.2 Geometric Pursuit Evasion

Geometric pursuit evasion takes places in the continuous Euclidean space, as opposed to the discrete space of graph based games. The geometric environment is typically modeled as a polygon, possibly with holes (serving as polygonal obstacles), and the players can move anywhere in the free space (the obstacle free interior of the polygon). Pursuit evasion in polygonal environments has received considerable interest for nearly two decades. The geometric properties of polygons often results in intellectually deep problems, while also having the added benefit of being able to accurately model many real world physical structures. The work in this area can roughly be broken into two

rather broad categories, detection, where the pursuers need only find the evader, and capture, where the pursuers must physically reach the evader in order to apprehend it.

Detection

Suppose an arbitrarily fast evader is moving about some simple polygon P. The problem of detection then asks a group of pursuers to plan search paths such that no matter how the evader moves, within some finite time t it will be visible to one of the pursuers. In this case, the standard definition of visibility is used, specifically, the line segment connecting the pursuer to the evader must not intersect the exterior of the polygon. Unsurprisingly, via a reduction from edge search, it was shown that computing the minimum number of required pursuers for a given polygon is NP-Hard [25]. However, unlike graph searching, Guibas et al. constructed an n-gon in which no strategy using the optimal number of pursuers existed that avoided recontamination [25], in fact, in any successful search a region would necessarily be re-contaminated $\Omega(n)$ times.

Throughout the literature, the notion of what exactly a pursuer can see is a modeling decision that differentiates much of the work. In the standard model we consider a pursuer often called an ∞-searcher which has 360^o unlimited range vision of its surroundings [70]. However, other well studied models include the k-searcher, which can see only from k infinitely thin beams at a time, as well as searchers equipped with field of view sensors [22]. While a significant body of work exists on 2-searchers, it

tends to involve complex characterizations of searchable environments [58], and thus we shall focus on ∞-searchers.

A naive approach and trivial upper bound on the number of sufficient pursuers can be obtained from the well known art gallery problem, which guarantees complete visibility of an n-gon using at most $\lfloor n/3 \rfloor$ pursuers. However, in the classic seminal work by Guibas et al. [25], it was shown that for a simply connected (obstacle free) n-gon P, $\Theta(\log n)$ pursuers are both sufficient and sometimes necessary to detect an arbitrarily fast evader. Further, it was also shown that by adding h obstacles to the environment the bound increases to $\Theta(\log n + \sqrt{h})$. Additional work by Isler et al. studied detection, except pursuers had the ability to make randomized decisions which the evaders could not predict. These randomized algorithms allowed a single pursuer to detect the evader in simply connected polygons in expected polynomial time. Additionally, it was shown that $O(\sqrt{h})$ pursuers suffice in the presence of h obstacles [29].

In recent years there have been numerous works extending the original results of Guibas et al. For example, Lavalle and Hinrichsen studied the case of curved environments [46], and Tovar and Lavalle the case when the evader has bounded speed [73]. Additionally researchers have focused on problems such as finding a search strategy for a single pursuer that has optimal duration [69], as well as the case in which the environment is unknown to the pursuer [26, 62]. Further, in some cases the continuous environment is discretized, sacrificing formal bounds in hopes of using graph search-

ing techniques in order to obtain an algorithm that is simpler and performs well in practice [35].

Capture

In order to capture an evader a pursuer must reach the exact coordinate of the evader, in contrast to detection, which only requires the evader is seen with no distance requirement between the pursuer and the evader. As a result, capture is based on planning physical moves, unlike detection, which is purely an informational game. In this setting the notion of time is particularly important, that is, whether the players move simultaneously (continuous time) or alternate turns (discrete time). Indeed, as previously mentioned, using continuous time the lion can indefinitely avoid capture [48], although the lion can get arbitrarily close [6]. However, using the discrete time model the result shifts to a lion win, which is an easy corollary of a result by Sgall [66].

Since these initial investigations, there has been a large amount of work studying variations of the lion and man problem. Often this work considers partially unbounded environments, and establishes starting conditions under which pursuers can capture the evader [5, 42, 66], or classifies environments in which a single pursuer can win [4, 56]. As for combinatorial bounds on the number of pursuers required to capture an evader in more general environments, Isler et al. showed that in simply-connected polygons, two pursuers can capture an evader by using a randomized strategy [29]. However, extending these result to environments with obstacles has proved difficult. Indeed, seemingly the

only relevant result is a recent work of Karnad-and-Isler [34] that deals with a single circular obstacle.

The primary focus of this work is to solve the capture problem in more complex settings. While the first step in this scenario is to perform capture in the presence of obstacles, a longer term vision allows for three dimensional environments and varying sensing models. It is worth noting that these pursuit evasion games are also studied as a form of differential games and solved using the Hamilton-Jacobi-Isaacs equation. Unfortunately, the resulting system of differential equations is intractable for all but the simplest of the environments, and unsuited for the complex, multiply-connected environments we study in this dissertation.

0.2 Pursuit Evasion Model

While the model we consider differs slightly in each chapter, based on the environment we consider and the sensing abilities of the pursuers, the general form of the game is the same. The pursuit evasion game we consider uses the discrete time model: this avoids the intractable problem of computing players' moves and reactions *instantaneously*, and also allows approximation of the continuous time setting to an arbitrary level of accuracy by choosing an appropriately small time step $t > 0$. We assume a set of pursuers denoted $p_1, p_2, \ldots$ wish to capture an evader e, and are free to move about a continuous bounded

environment known to both sides. For the sake of notational brevity, we also use e to denote the current position of the evader, and p_i to denote the position of the ith pursuer.

All the players have the same maximum speed, which we assume is normalized to 1. In each move, a player can move to any position whose shortest path distance from its current position is at most one; that is, within *geodesic disk* of radius one. On the pursuers' move, all the pursuers can move simultaneously and independently. We say that pursuers win the game if a pursuer p_i is collocated with e, and evader wins the game if it can elude the pursuers indefinitely.

In accordance with the standard worst case model, we assume that the evader knows the location and future moves of the pursuers at all times. By proving our bounds against this adversarial model we guarantee that they hold regardless of the strategy the evader uses to avoid capture.

0.2.1 Summary of Results

In Chapter 1 we consider the complete information pursuit evasion problem set in polygonal environments. Suppose the pursuers have perfect knowledge of the evader's location, perhaps through access to a camera network, how are necessary and sufficient to guarantee a successful capture of the evader? We provide two separate algorithms to capture the evader with three pursuers. Additionally, we construct an example polygon

in which the evader can avoid capture indefinitely from two pursuers, establishing three as a tight bound.

In Chapter 2 we consider the complete information pursuit evasion game set on polyhedral surfaces. We show that 4 pursuers always suffice (upper bound), and that 3 are sometimes necessary (lower bound), for any polyhedral surface with genus zero. Generalizing this bound to surfaces of genus g, we prove the sufficiency of $(4g + 4)$ pursuers. Finally, we show that 4 pursuers also suffice under the "weighted region" constraints, where the movement costs through different regions of the (genus zero) surface have (different) multiplicative weights.

Next in Chapter 3 we study visibility-based pursuit evasion, where the pursuer's only know the location of the evader when it is in direct line of sight. We begin by making only the minimalist assumption that pursuers and the evader have the same maximum speed. When the environment is a simply-connected (hole-free) polygon of n vertices, we show that $\Theta(n^{1/2})$ pursuers are both necessary and sufficient in the worst-case. When the environment is a polygon with holes, we prove a lower bound of $\Omega(n^{2/3})$ and an upper bound of $O(n^{5/6})$ pursuers, where n includes the vertices of the hole boundaries.

We then show with additional assumptions these bounds can be drastically improved. Namely, if the players movement speed is small compared to the features of the environment, we give a deterministic algorithm with a worst case upper bound of $O(\log n)$ pursuers for simply-connected n-gons and $O(\sqrt{h} + \log n)$ for polygons with h holes.

Additionally, if the pursuers are allowed to randomize their strategy, regardless of the players movement speed, we show that with high probability $O(1)$ pursuers can capture the evader in a simply connected n-gon and $O(\sqrt{h})$ when there are h holes.

Finally, in Chapter 4 we further reduce the sensing capabilities of the pursuers by incorporating sensor noise. In particular, suppose a tracking agent wants to follow a moving target in the two-dimensional plane. However, the tracker only has a noisy estimate of the targets true location. We adopt a simple but realistic model of relative error in sensing noise: the localization error is proportional to the true distance between the tracker and the target. We investigate what is the tracker's best strategy to follow the target if they both can move with equal speed and at what rate does the distance between the tracker and target grow under worst-case localization noise. Additionally we investigate giving the tracker a speed advantage to compensate for the sensor noise, and the effect of obstacles on the tracking performance. Under a relative error model of noise, we are able to give upper and lower bounds for the worst-case tracking performance, both with or without obstacles in the Euclidean plane.

Chapter 1

Complete Information Pursuit Evasion in Polygons

1.1 Introduction

There are two fundamental issues inherent in pursuit evasion: *localization*, which is purely an informational problem, and *capture*, which is a problem of planning physical moves. In this chapter, we study the question: how complex is the capture problem *if the evader localization is available for free*? In other words, suppose the pursuers have complete information about the evader's current position, how much does it help them to capture the evader?

Besides being a theoretically interesting question, the problem is also a reasonable model for many practical settings. Given the rapidly dropping cost of electronic surveillance and camera networks, it is now both technologically and economically feasible

*Parts of this chapter appeared in a joint journal paper [9] which combined the results from two independently discovered algorithms, that appeared in the following publications [8, 37]

to have such monitoring capabilities. These technologies enable cheap and ubiquitous detection and localization, but in case of intrusion, a physical capture of the evader is still necessary. For instance, the scenario studied in [74] requires pursuers to capture an evader in an environment instrumented with a sensor network. The sensor network provides the location of the evader to the pursuers and facilitates communication among the pursuers. Our results immediately imply that three pursuers suffice regardless of the shape of the floor plan in their application.

Our main result is that under the full visibility setting, *three pursuers* are always sufficient to capture an equally fast evader in a polygonal environment with holes, using a *deterministic* strategy. Complementing this upper bound, we also show that there exist polygonal environments that require at least three pursuers to capture the evader even with full information.

We present two different algorithmic strategies for our main result, one called *Minimal Path Strategy* and the other *Shortest Path Strategy*. These were discovered independently by two teams, Bhadauria-Isler [8] and Klein-Suri [37] around the same time, and combined into a joint journal paper [9]. The former (Minimal Path Strategy) uses the visibility graph of the original polygon, and deploys pursuers along the first, second and third shortest paths in this graph to trap the evader in progressively smaller sub-polygons (Section 1.3). The latter (Shortest Path Strategy) operates in the continuous domain, and guards a carefully chosen shortest path so as to trap the evader in a smaller

polygonal region (Section 1.5). Despite their high-level similarity, the two algorithms differ significantly in details, and offer independent insights into the problem, motivating the inclusion of both in this chapter.

The bound on capture time, which is asymptotically the same for both strategies, is independent of the number of the holes of the polygon, although the capture time depends on both n and the diameter of the polygon.

Our work bears some resemblance to, and is inspired by, the result of Aigner and Fromme [2] on planar graphs, showing that graph searching on planar graph requires 3 cops. In that work, the graph is unweighted, does not deal with Euclidean distances, and require players to move to only neighboring nodes. Unlike the graph model, our search occurs in continuous Euclidean plane, and players can move to any position within distance one. Thus, while our bounds are similar, the proof techniques and technical details are quite different.

1.2 The Problem Formulation

We assume that an evader and pursuers are free to move in a two-dimensional closed polygon P, which has n vertices and h holes using the standard model of Section 0.2 (the pursuers and evader move within the free-space of P). The bounds in our algorithm

depend on the number of vertices n and the diameter of the polygon, $\mathrm{diam}(P)$, which is the maximum distance between any two vertices of P under the shortest path metric.[1]

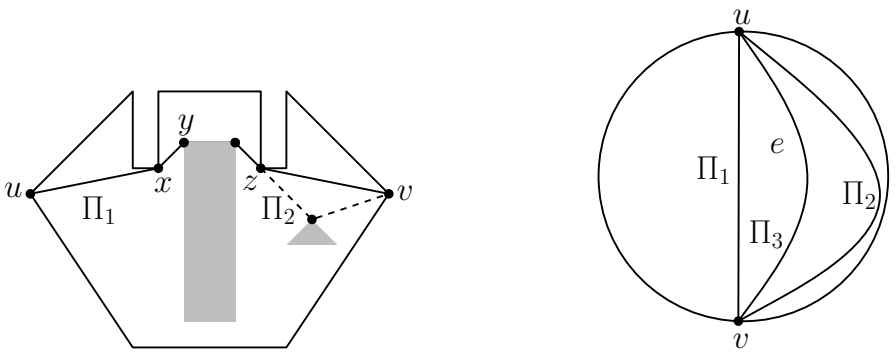
In order to focus on the complexity of the capture, we assume a complete information (full visibility) setup: each pursuer knows the location of the evader at all times. We also endow the evader the same information, so e also knows the locations of all the pursuers.

We begin with a high level description of the minimal path strategy, followed by its technical details and proof of correctness in the next section.

1.3 The Minimal Path Strategy

We show that three pursuers, denoted p_1, p_2, p_3, can always capture an evader using a deterministic strategy, regardless of the evader's strategy and the geometry of the environment. The minimal path strategy is to progressively trap the evader in an ever-shrinking region of the polygon P. The pursuit begins by first choosing a path Π_1 that divides the polygon into sub-polygons (see Figure 1.1(a))—we will use the notation P_e to denote the sub-polygon containing the evader. We show that, after an initialization period, the pursuer p_1 can successfully guard the path Π_1, meaning that e cannot move across it without being captured.

[1] We assume that the *area* quantity of the polygon is at least as large as the *diameter* of the polygon, which can be always ensured through an appropriate scaling, if needed. We give a more precise argument later in the chapter. This assumption helps us frame the bounds using the diameter alone.



(a) (b)

Figure 1.1: (a) A polygonal environment with two holes (a rectangle and a triangle). xy is a visibility edge of $G(P)$, while xz is not. Π_1 and Π_2 are the first and the second shortest paths between anchors u and v. The figure (b) illustrates the main strategy of trapping the evader through three paths.

Figure 1.1(b) illustrates the overall strategy: in a general step, the sub-polygon P_e containing the evader is bounded by two paths Π_1 and Π_2, satisfying a geometric property called *minimality*, each being guarded by a pursuer. We then choose a third path Π_3 splitting the region P_e into two non-empty subsets. If both regions have holes, then we argue that the pursuer p_3 can guard Π_3, thereby trapping e either between Π_1 and Π_3 (Figure 1.1(b)), or between Π_2 and Π_3, in which case the pursuit iterates in a smaller region. If Π_3 is not guardable within one of the regions, then we show that the pursuer p_3 can *evict* the evader from this region, forcing it into a smaller region (as measured by the number of vertices) where the search resumes.

1.3.1 Visibility Graphs and Path Guarding

In order for this strategy to work, the paths Π_i need to be carefully chosen and must satisfy certain geometric conditions, which we briefly explain. First, although the pursuit occurs in continuous space, our paths will be computed from a *discrete* space, namely, the *visibility graph* of the polygon. The visibility graph $G(P)$ of a polygon P is defined as follows: the nodes are the vertices of the polygonal environment (including the holes), and two nodes are joined by an edge if the line segment joining them lies entirely in the (closed) interior of the polygon. (In other words, the two vertices joined by an edge must have line of sight visibility.) This *undirected* graph has n vertices and at most $O(n^2)$ edges. We assign each edge a *weight* equal to the Euclidean distance between its two endpoints. See Figure 1.1(a) for an example.

One can easily see that, given two vertices u and v of P, the *shortest path* from u to v in $G(P)$ is also the shortest Euclidean path constrained to lie inside P. (The shortest Euclidean path has corners only at vertices of $G(P)$.) However, we cannot make such a claim for the *second*, or in general the kth, shortest path—one can create an infinitesimal "bend" in the shortest path Π_1 to create another path that is arbitrarily close to the first shortest path but does not belong to $G(P)$. Therefore, we will only consider paths that belong to $G(P)$ and are "combinatorially distinct" from Π_1—that is, they differ in at least one visibility edge. However, even then the kth shortest path between two nodes

can exhibit counter-intuitive behavior. For instance, while in graphs with non-negative weights the first shortest path is always loop-free, the *second*, or more generally kth, shortest path can have loops—this may happen if repeatedly looping around a small-weight cycle (to make the path distinct from others) is cheaper than taking a different but expensive edge [28]. Therefore, we will consider only shortest loop-free paths. One of our technical lemmas proves that these paths are also *geometrically* non-self-intersecting. (This is obvious for the shortest path Π_1 but not for subsequent paths.) In addition, we argue that these paths also satisfy a key geometric property, called *minimality*, which allows a pursuer to guard them against an evader.

1.4 Proof of Sufficiency of 3 Pursuers

We begin with the discussion of how a single pursuer can guard a path in P, trapping the evader on one side. We then discuss the technically more challenging case of guarding the second and the third paths. In order to guarantee that a path in P can be guarded, it must satisfy certain geometric properties. We begin by introducing two key ideas: a *minimal path* and the *projection* of an evader on a path. In the following, we use the notation $d(x, y)$ to denote the shortest path distance between points x and y. When we require that distance to be measured within a subset, such as restricted to a path Π, we write $d_\Pi(x, y)$. That is, $d_\Pi(x, y)$ is the length of path Π between its points x and y. Occasionally, we also use the notation $\Pi(x, y)$ to denote subpath of Π between

points x, y. We use the notation $x \prec y$ to emphasize that the point x precedes y on the path Π: that is, if Π is the path from node u to node v, then $x \prec y$ means that $d_\Pi(u, x) < d_\Pi(u, y)$. The following property is important for patrolling of paths.

Definition 1. (**Minimal Path:**) Suppose Π is a path in P dividing it into two sub-polygons, and P_e is the sub-polygon containing the evader e. We say that Π is *minimal* with respect to P_e if, for all points $x, z \in \Pi$ and $y \in (P_e \setminus \Pi)$, the following holds:

$$d_\Pi(x, z) \leq d(x, y) + d(y, z)$$

Intuitively, a minimal path cannot be shortcut: that is, for any two points on the path, it is never shorter to take a detour through an interior point of P_e. (This is a weak form of triangle inequality, which excludes detours only through points contained in P_e.) The next definition introduces the projection of the evader on to a path, which is an important concept in our algorithm.

Definition 2. (**Projection:**) Suppose Π is a path in P dividing it into two sub-polygons, and P_e is the sub-polygon containing the evader e. Then, the *projection* of e on Π, denoted e_π, is a point on Π such that, for all $x \in \Pi$, e is no closer to x than is e_π.

Thus, if a pursuer is able to position itself at the projection of e at all times, then it guarantees that the evader cannot cross the path without being captured. With these definitions in place, we now discuss how to guard the first path Π_1.

1.4.1 Guarding the First Path

We choose two vertices u and v on the outer boundary of P, and call them *anchors*. We let Π_1 be the shortest path from u to v in $G(P)$; this is also the shortest Euclidean path between u and v constrained to lie inside the environment. Our first observation is that this path Π_1 is always minimal.

Lemma 1. *The path Π_1 between u and v is minimal.*

Proof. For the sake of contradiction, suppose there are two points $x, z \in \Pi_1$ that violate the minimality. Let the point $y \notin \Pi_1$ be the witness of this violation, namely, $d(x, y) + d(y, z) < d_{\Pi_1}(x, z)$. But then Π_1 can be shortened with the subpath $\Pi_1(x, z)$, contradicting the fact that Π_1 is the shortest u, v path. □

The following lemma shows that the projection of e is always exists for a minimal path.

Lemma 2. *Suppose Π is a minimal path between the anchor nodes u and v. Then, for every position of the evader e in P_e, a projection e_π exists.*

Proof. Let us first consider the more interesting case where $d_\Pi(u, v) \geq d(u, e)$. In this case, we claim that the point z at distance $d(e, u)$ along Π is a projection of e. Indeed, for

any point $x \in \Pi$ such that $z \prec x$, the condition $d_\Pi(z, x) > d(e, x)$ leads to a violation of the minimality of Π, as follows:

$$d_\Pi(u, x) \;=\; d_\Pi(u, z) + d_\Pi(z, x) \;=\; d(u, e) + d_\Pi(z, x) \;>\; d(u, e) + d(e, x)$$

Similarly, for any point x that $\prec z$, the condition $d(x, e) < d_\Pi(x, z)$ also leads to a violation:

$$d(u, e) \;\leq\; d_\Pi(u, x) + d(x, e) \;<\; d_\Pi(u, x) + d_\Pi(x, z) \;=\; d_\Pi(u, z)$$

which is a contradiction because $d(u, e) = d_\Pi(u, z)$.

On the other hand, if $d_\Pi(u, v) < d(u, e)$, then we choose v as the projection. In this case, the argument is identical to the second case above: $\forall x \prec v$, $d(x, e) \geq d_\Pi(x, v)$, and thus v is a projection.

□

The next lemma shows how a pursuer can guard a minimal path. Whenever we refer to the projection, we mean the unique point chosen by Lemma 2, that is, the point on Π at $d(u, e)$ from u, or v, whichever is closer.

Lemma 3. *Suppose* Π *is a minimal path between the anchors* u, v *in* P*, and a pursuer* p *is located at the current projection of* e*. Suppose on its turn the evader moves from* e *to* e'*. Then, the pursuer* p *can either capture the evader or relocate to the new projection* e'_π *in one move.*

Proof. First, suppose that the new position e' is on different side of the path Π than e, namely, the evader crosses the path, say, at a point z. Because the evader can move at most distance one, we have the inequality $d(e, z) + d(z, e') \leq 1$. On the other hand, since p is located at the projection of e before the move, $d_\Pi(p, z) \leq d(e, z)$. Therefore, the new position of the evader e' is within distance one of p, and the pursuer can capture the evader on its move.

If the evader does not cross Π, and moves to a position e' on the same side of the path, let e'_π be the projection of e', as defined in Lemma 2. Because the evader moves distance at most one further from u or at most one closer to u, it must satisfy $d(e_\pi, e'_\pi) \leq 1$, and so p can relocate from e_π to e'_π in one move. □

Before proceeding further, we make a minor technical digression, to establish that any path guarded by pursuers can be bounded by the *area* of the polygon. The strategy of progressively trapping the evader within smaller sub-polygons brings out a somewhat counterintuitive property of polygon divisions: *a sub-polygon can have a larger diameter than the original polygon.* Figure 1.2 shows an example where the diameter of the shaded sub-polygon P' is larger than the original environment. This complicates the time complexity analysis of our pursuit strategy because it depends on the length of paths that are guarded. We resolve this dilemma by arguing these path lengths cannot exceed the area of the original environment, which in turn is bounded by $\mathrm{diam}(P)^2$. Of course, diameter is a one-dimensional quantity, while area is a two-dimensional quantity,

but we only care about their numerical magnitudes. We show the required inequality by choosing an appropriate *scale* (units) for the environment, as shown in the following lemma.

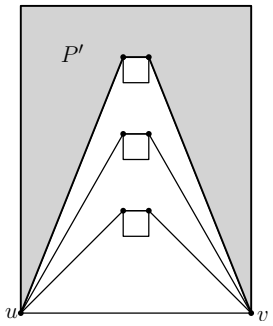


Figure 1.2: Example depicting a shaded sub-polygons P' with diameter larger than $\mathrm{diam}(P)$.

Lemma 4. *Suppose* Π *is a* u, v *path in sub-polygon* P' *of* P*. Then, by applying a suitable rescaling of units we can always guarantee* $d_\Pi(u, v) \leq diam(P)^2$.

Proof. If $d_\Pi(u, v) \leq area(P')$, then the lemma holds trivially, because $area(P') < area(P) \leq \mathrm{diam}(P)^2$. Therefore, assume that $d_\Pi(u, v) > area(P')$. By a simple *rescaling of the units*, we can get the desired reverse inequality, as follows. Suppose we rescale the unit of measurement from 1 to $1 + \alpha$. This increases the area of a triangle by a factor of $(1 + \alpha)^2$, while a segment only increases in length by a factor of $1 + \alpha$. Therefore, a suitably large choice of α will always ensure that the polygon's area exceeds the length of Π, because the former grows by a factor of $(1 + \alpha)^2$ while

the latter grows linearly. In particular, if $(1+\alpha)^2 \cdot area(P') \geq (1+\alpha) \cdot d_\Pi(u,v)$, we obtain $\alpha \leq \frac{d_\Pi(u,v)}{area(P')} - 1$, and therefore any choice of $\alpha > \frac{d_\Pi(u,v)}{area(P')} - 1$ will suffice. □

With this technical lemma, we can assume throughout the rest of the chapter that $d_\Pi(u,v) \leq diam(P)^2$ always holds. The following lemma shows that within $O(\mathrm{diam}(P)^2)$ a pursuer p can either reach the current projection of e or capture it.

Lemma 5. *Suppose Π is a minimal path between anchors u, v in P, and a pursuer p is located at u. Then in $O(\mathrm{diam}(P)^2)$ moves, p can move to e's projection.*

Proof. By Lemma 3, the projection of e can only shift by distance at most one along the path Π. Thus, p's strategy is simply to move along the path from one end to the other until it coincides with the current projection of e, or captures it. Meanwhile, if the projection ever "crosses over" the current position of p, the pursuer immediately can move to the new projection because at that moment p must be within distance one of the target location. Since p moves a distance of 1 in each turn, and Lemma 4 guarantees we can scale P such that all paths encountered have length at most $\mathrm{diam}(P)^2$, the entire initialization phase takes at most $O(diam(P)^2)$ moves. □

1.4.2 Geometric Structure of Pursuer Paths

We now come to the main part of our pursuit strategy. The key idea is to progressively trap the evader in a region bounded by two minimal paths, which are guarded by two pursuers, and to use the third pursuer to further divide the current region. When the third

pursuer subdivides the current region containing e, two possibilities emerge: either the third path is minimal with respect to both regions and thus guardable by the third pursuer, limiting the evader to a smaller region than before; or it is only minimal with respect to one of the regions and the other is hole-free, in which case the third pursuer uses the capture strategy for a simply-connected polygon to evict the pursuer from this region (or capture it). In order to formalize our strategy, we first show a key geometric property of the second and third shortest paths between the anchors in the visibility graph, namely, that they are non-self-intersecting, and therefore lead to well-defined closed regions.

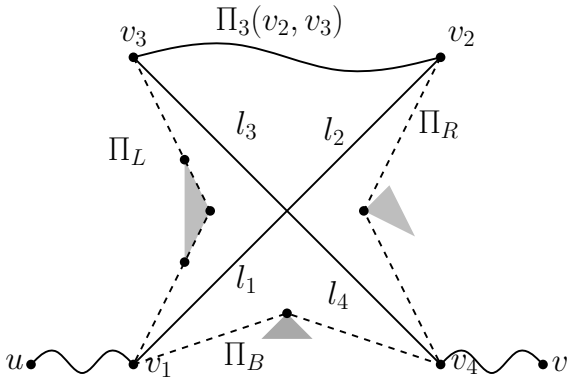


Figure 1.3: Non-self-crossing of shortest paths Π_1, Π_2, Π_3.

Lemma 6. *Let Π_1 be the shortest path between two anchor points u and v on P's boundary, and focus on the sub-polygon P_e that lies on one side of Π_1. Let Π_2 and Π_3, respectively, be the second and the third simple (loop-free) shortest paths in the visibility graph $G(P_e)$ between u and v. Then, Π_2 and Π_3 are non-self-crossing.*

Proof. Without loss of generality, suppose the path Π_3 violates the lemma, and that two of its edges (v_1, v_2) and (v_3, v_4) intersect. See Figure 1.3. We first note that the intersection point cannot be a vertex of the visibility graph because otherwise the path has a cycle, and we assumed that Π_3 is loop-free. As shown in the figure, we break the segment (v_1, v_2) into l_1 and l_2, and (v_3, v_4) into l_3 and l_4. By the triangle inequality of the Euclidean metric, it is easy to see that the shortest v_1, v_3 path homotopic to the segments l_1 and l_3, denote it Π_L, will have length strictly less than $l_1 + l_3$. Similarly, define Π_R and Π_B, as paths between v_2, v_4 and v_1, v_4, respectively. Now consider the following three paths between v_1 and v_4, each contained in $G(P_e)$: $\Pi_L \cdot \Pi_3(v_3, v_2) \cdot \Pi_R$, Π_B, and the shorter of $\Pi_L \cdot (v_3, v_4)$ and $(v_1, v_2) \cdot \Pi_R$. They are all shorter than Π_3, each has one less intersection than Π_3, and at least one of them must be distinct from both Π_1 and Π_2, thus contradicting the choice of Π_3. If further intersections exist, the argument can be applied again, until all such intersections are removed. □

1.4.3 Shrinking, Guarding and Evicting

In a general step of the algorithm, assume that the evader lies in a region P_e of the polygon bounded by two minimal paths Π_1 and Π_2 between two anchor vertices u and v. (Strictly speaking, the region P_e is initially bounded by Π_1, which is minimal, and portion of P's boundary, which is not technically a minimal path. However, the evader cannot cross the polygon boundary, and so we treat this as a special case of the minimal

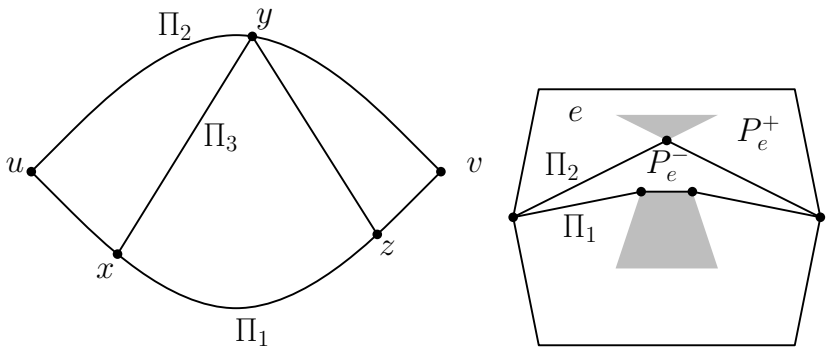
path to avoid duplicating our proof argument.) We also assume that Π_1 and Π_2 only share vertices u and v; if they share a common prefix or suffix subpath, we can delete those and advance the anchor nodes to the last common prefix vertex and the first common suffix vertex. This ensures that the region P_e is non-degenerate.

The key idea of our proof is to show that, in the visibility graph $G(P_e)$, if we compute a *shortest path* from u to v that is distinct from both Π_1 and Π_2, then it divides P_e into *only* two regions, and that the evader is trapped in one of those regions. We will call this new path the *third* shortest path Π_3. Specifically, Π_3 is the simple (loop-free) shortest path from u to v in $G(P_e)$ distinct from Π_1 and Π_2. (One can compute such a path using any of the algorithms for computing k loop-free shortest paths in a weighted undirected graph [28, 54, 75].)

Lemma 7. *The shortest path Π_3 between the anchor nodes u and v divides the current evader region P_e into two regions.*

Proof. If the path is disjoint from Π_1 and Π_2 except at endpoints, then P_e is clearly subdivided into two (possibly disconnected) regions. If Π_3 shares vertices only with Π_1 or only with Π_2, *but in multiple disjoint subpaths creating multiple regions*, then each subpath shares its first and last vertices with either Π_1 or Π_2, and thus we can replace all but one with subpaths of Π_1 or Π_2 and obtain a path no longer than Π_3. Therefore, let us suppose that Π_3 shares vertices with both the paths, and so "hops" between Π_1 and Π_2, sharing common subpaths with them, and creates three or more regions. In that

case, Π_3 must leave and rejoin Π_1 and Π_2 at least once, as shown by points x, y, z in Figure 1.4(a). We observe that $d_{\Pi_2}(y, v)$ is no longer than $d(y, z) + d_{\Pi_1}(z, v)$, otherwise Π_2 is not the second shortest u, v path, which is a contradiction. Thus the third region can be removed by altering Π_3 to use the subpath $\Pi_2(y, v)$. (A symmetric case arises when the roles of Π_1 and Π_2 are swapped.) Thus, we conclude that Π_3 can create only two subregions. □



(a) (b)

Figure 1.4: The left figure illustrates the proof of Lemma 7; the right figure illustrates the two subregions created by a path, Π_2 in this case.

Clearly, if P_e contains one or more holes, then at least one of the regions created by the third shortest path Π_3 also contains a hole. The following lemma argues that Π_3 is minimal with respect to such a region. (The next lemma then addresses the case when the region is hole-free.)

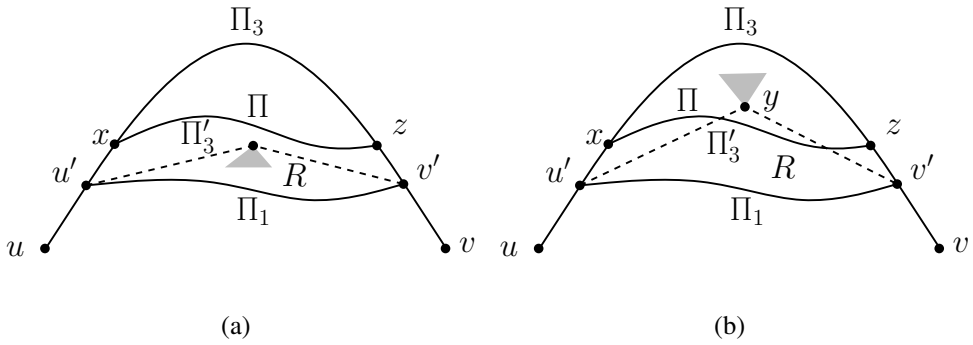


Figure 1.5: Illustrates the proof of Lemma 8.

Lemma 8. *Suppose* Π_3 *divides the region* P_e *into two subregions* P_e^+ *and* P_e^-*, and assume that* P_e^+ *contains at least one hole. Then,* Π_3 *is a minimal path within the region* P_e^+.

Proof. Assume, for the sake of contradiction, that the minimality of Π_3 is violated for two points $x, z \in \Pi_3$. Let u' be the vertex immediately preceding the point x, possibly $x = u'$, and v' is the vertex immediately following z, possibly $z = v'$, on Π_3. Consider the shortest path in $G(P_e)$ from u' to v'. This path must be distinct from $\Pi_3(u', v')$, as a shortest path is necessarily minimal, while by assumption $\Pi_3(u', v')$ is not. Thus, if this path is *not* a subpath of either Π_1 or Π_2, we can immediately improve the length of Π_3 by using this subpath, thereby contradicting the choice of Π_3. Therefore, assume without loss of generality that the shortest path from u' to v' is a subpath of Π_1. Further, let Π denote the shortest path from point x to point z in P_e^+, and consider the region R bounded by $\Pi_1(u', v')$, Π and the segments (z, v') and (x, u'). If there are any holes in

R then there is a distinct path Π_3' shorter than Π_3 obtained by tightening Π around those holes as shown in Figure 1.5(a). Thus the hole in P_e^+ must be outside R, however pick the closest vertex on a hole in P_e^+ to Π, call it y. Then a path Π_3' shorter than Π_3 can be obtained using y as shown in Figure 1.5(b). Thus in all cases, if P_e^+ contains a hole, Π_3 can be shortened, which contradicts its optimality. Thus Π_3's minimality cannot be violated, and the proof is complete. □

Since Π_1 and Π_2 are the two shortest paths between u and v, the region between them necessarily contains a hole: otherwise, all vertices except u and v must be reflex (within the region), which is a contradiction since every simply polygon must have at least three convex vertices. Thus, at least one of the regions created by Π_3 has a hole, and so Π_3 is minimal for that region. The region without holes must have a very special and simple structure, as shown by the following lemma, and it can be cleared using the search strategy for simply-connected polygons.

Lemma 9. *Suppose* Π_3 *divides the region* P_e *into two subregions* P_e^+ *and* P_e^-*. If* Π_3 *fails to be minimal with respect to* P_e^+*, then* Π_3 *has the following simple structure: two edges plus a subpath of either* Π_1 *or* Π_2*.*

Proof. Suppose Π_3 fails to be minimal in P_e^+. Then, by Lemma 8, P_e^+ is hole-free. Non-minimality means that the path can be shortcut, and so all vertices of Π_3 cannot be reflex. Let y be a vertex of Π_3 that is convex in P_e^+, and let x and z, respectively, be the

predecessor and successor vertices of y. We claim that x and z are either both vertices of Π_1 or both vertices of Π_2. Suppose not. Then, the shortest path from x to z in P_e, call it Π, is shorter than $\Pi_3(x, z)$. By assumption, at least one of x and z is not in Π_1, and similarly for Π_2, thus Π cannot be a subpath of Π_1 or Π_2.

But, then the path $\Pi_3(u, x) \cup \Pi \cup \Pi_3(z, v)$ is shorter than Π_3 and distinct from Π_1 and Π_2, contradicting the choice of Π_3. Thus, x and z both belong to either Π_1 or Π_2, and assume, without loss of generality, that they belong to Π_1. Then P_e^+ is bounded by $\Pi_1(x, z)$ and the edges (x, y) and (y, z), and the proof is finished. □

Now, if both regions created by Π_3 have holes, then the minimality of Π_3 allows a third pursuer to guard this path, and the pursuit continues in one of the smaller regions. However, if one region is hole-free and Π_3 is not minimal within it, a different strategy is required. Lemma 11 shows how to either capture the evader in such a region, or to force the evader out of (evict) this region, while guarding Π_3 so the evader *cannot reenter* this region.

This is accomplished by fixing an origin O in the region (say, some vertex in P), and then letting the pursuer move along the shortest path between O and the current evader position. It can be shown that the pursuer makes sufficient progress towards the evader by invoking a result of Isler et al. on the visibility-based version of the cops-and-robbers game in simply-connected polygons. In their model, a cop can see the robber only if the line segment connecting the two players does not intersect the boundary of the

polygon. They showed that a single cop can locate the robber, and two cops can capture the robber in any simply-connected polygon. In the two-cop strategy, one cop starts from an arbitrary point O and moves so that it stays on the shortest path between the robber's current location and O. Further, whenever the cop moves, its squared distance from O increases by at least $1/n$. Since the cop can not see the robber when it is occluded from his field of view, the second cop is used to determine the motion direction when the robber is not visible. They also bound the number of searches necessary. Since in our model the players know each other's locations at all times, the second cop is not necessary, giving us the following result:

Lemma 10 (Capture in a simply connected polygon [29])**.** *A single pursuer can capture the evader in any simply-connected polygon P in $O(n \cdot diam(P)^2)$ moves.*

In the following Lemma we use this result to force the evader out of (evict) this region.

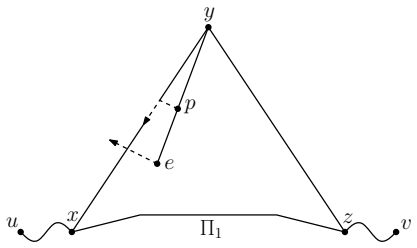


Figure 1.6: An illustration of the pursuer's eviction strategy. Dashed lines denote moves where e moved first.

Lemma 11. *Suppose the evader lies in hole-free region of k vertices that is bounded by Π_3 and another minimal path. If Π_3 is not minimal with respect to this region, then, in $O(k \cdot \mathrm{diam}(P)^2)$ moves, a single pursuer p can either capture the evader or force it out of the region and place itself on e's projection on the path Π_3.*

Proof. Assume, without loss of generality, that our hole-free region is bounded by a minimal path Π_1 and the path Π_3, which by Lemma 9 must consist of two edges, say, (x, y) and (y, z). The pursuer p's strategy is to move to y, and at each turn move to the point closest to e that is distance one from p and lies on the shortest y, e path, with one modification. Namely, if p's move takes it outside the region, then it moves along Π_3 toward e_π (which must exist as Π_3 is minimal with respect to the other region) until e reenters, at which point its resumes the pursuit, as depicted in Figure 1.6.

As the shortest path between any two vertices consists of at most two edges, this region can have diameter no larger than $2 \cdot \mathrm{diam}(P)$. Thus if e never leaves the region, then by the known result of Lemma 10, a successful capture occurs in $O(k \cdot \mathrm{diam}(P)^2)$ moves. Therefore, assume that e leaves the region at some point. Since Π_1 is minimal, the evader cannot leave the region through that path, and so assume without loss of generality that the evader crosses the segment (x, y) of Π_3. Because p always stays on the shortest path between e and y, in an unmodified pursuit p's move would cross (x, y) as well. In the modified pursuit, p stops at the point where it crosses (x, y) and advances toward the projection of e.

We note that the projection of e is within distance one of where e crossed (x, y). As a result, because p crossed (x, y) at a point closer to y than e, if e_π lies on the subpath $\Pi_3(p, v)$, then p can reach e_π in one move, and Π_3 is guarded and we are done. Otherwise, p need simply advance forward along Π_3 toward e_π. If e never re-enters the hole free region, then by Lemma 5 p will reach the projection within $O(\mathrm{diam}(P)^2)$ moves.

In the case e re-enters the hole-free region, we note that it must do so by crossing the segment (x, p), and that for each turn e was outside the hole-free region p moved distance one along the shortest path from y to e. Thus on its next turn p can resume its pursuit, while having increased its squared distance from y by at least $1/k$, which will guarantee a successful capture occurs in $O(k \cdot \mathrm{diam}(P)^2)$ moves should e remain within the hole-free region. Thus e may continually move back and forth between the hole-free region, but within $O(k \cdot \mathrm{diam}(P)^2)$ moves e will either be captured, or the pursuer will successfully guard Π_3 by reaching the projection. □

We can now summarize the main result of this chapter.

Theorem 1. *By following the Minimal Path Strategy, three pursuers can capture an evader in $O(n \cdot \mathrm{diam}(P)^2)$ moves in a polygon with n vertices and any number of holes.*

Proof. Whenever a new path is introduced which is minimal with respect to both regions, the size (number of vertices) of the region P_e containing e shrinks by at least one. Thus,

the number of such paths guarded during the course of the pursuit before e is captured is at most n, and the total cost of guarding them is at most $O(n \cdot \mathrm{diam}(P)^2)$. If Π_3 is only minimal with respect to one region R, then in $O(k \cdot \mathrm{diam}(P)^2)$ moves the evader will either be forced into R and a pursuer will guard Π_3 or the evader will be captured. In such a case, the vertices on the two bounding edges of Π_3 were not removed, thus only $k-3$ of the k vertices were removed from P_e. When $k > 3$ the cost of removals sums to at most $O(n \cdot \mathrm{diam}(P)^2)$. When $k = 3$, the evader is being evicted from a triangle, bounded by two edges of Π_3 which meet at a vertex y, and an edge of either Π_1 or Π_2. We bound the number of such removals by showing each vertex can only be chosen as y twice. Either y is an interior vertex of P_e, and will not be chosen again as an interior vertex (as it is now on a bounding path), or y is already on a bounding path, and y will become an anchor, and never be chosen again. Thus, there are at most $2n$ removals where $k = 3$, and their total cost is at most $O(n \cdot \mathrm{diam}(P)^2)$.

Finally, the sub-polygon containing the evader will be reduced to a triangle. Notice this must occur, as otherwise a path Π_3 exists which would split P_e. This region clearly has diameter no larger than $\mathrm{diam}(P)$, and thus the evader can be captured by the third pursuer in $O(\mathrm{diam}(P)^2)$ moves with the known result of Lemma 10, for a total of $O(n \cdot \mathrm{diam}(P)^2)$ moves over the entire pursuit. □

1.5 The Shortest Path Strategy

In this section we present an alternative strategy to capture the evader. In contrast to the Minimal Path Strategy which chooses the first, second and the third shortest paths in the visibility graph to trap the evader, the *Shortest Path Strategy* directly picks a shortest path in the evader's region to trap the evader in a smaller region with fewer vertices. See Figure 1.7.

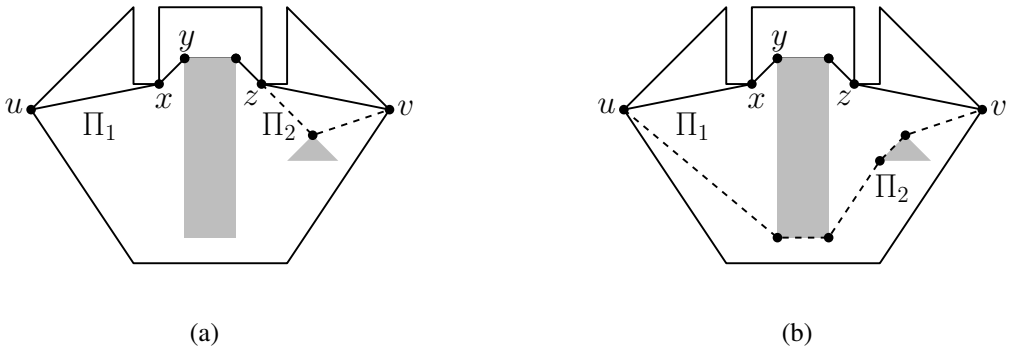


Figure 1.7: In (a), the next path (Π_2) chosen by the Minimal Path Strategy (Section 1.3). In (b), the Shortest Path Strategy using the obstacle move (Section 1.5.1).

A shortest path is guarded in two phases. In the initialization phase, a pursuer moves onto the evader's projection. Afterward, the pursuer stays on the projection as described in Lemma 3. Note that a shortest path in a polygon is minimal with respect to any subset of the polygon (see also Lemma 1). Hence, it can be guarded regardless of P_e.

We will divide the pursuers' strategy into rounds. In each round, the pursuers will coordinate their moves and restrict the evader to a smaller polygon by choosing two points and guarding the shortest path between them.

Before presenting the full strategy, we describe two types of moves. In each round pursuers will perform either a *slicing move* and/or an *obstacle move*. Each of the two moves is a sequence of steps taken by a single pursuer. Before presenting the details, we introduce the notation we will use for the rest of the paper.

We will use P_i to denote the the evader's region P_e at round i. We denote the boundary of P_i by δP_i. Let $n(P_i)$ be the total number vertices in P_i (including the obstacle vertices). The boundary δP_i will consist of at most two *shortest paths*, π_1 and π_2, each guarded by a dedicated pursuer. The rest of the boundary will either consist of a portion of δP, the original polygon's boundary, or the boundaries of the obstacles. Hence if the evader tries to escape from P_i it has to cross either π_1 or π_2 which will result in capture by Lemma 3. We label the vertices of π_1 and π_2 in the order they are encountered while traversing δP_i in clockwise direction. Without loss of generality, let $\pi_1 = u_1, \ldots, u_k$ and let $\pi_2 = u_l, \ldots, u_m$ (See Figure 1.8).

At the end of each round, the strategy will maintain the following invariants:

1. $n(P_i) > n(P_{i+1})$, the number of vertices in P_{i+1} are strictly smaller than the number of vertices in P_i.

2. $P_{i+1} \subset P_i$, i.e., the new polygon is a subset of the previous one.

3. the paths guarded by the pursuers forming the boundary of P_{i+1} are both the shortest paths in P_{i+1}.

We are now ready to present the two types of moves and analyze their properties.

1.5.1 Obstacle Move

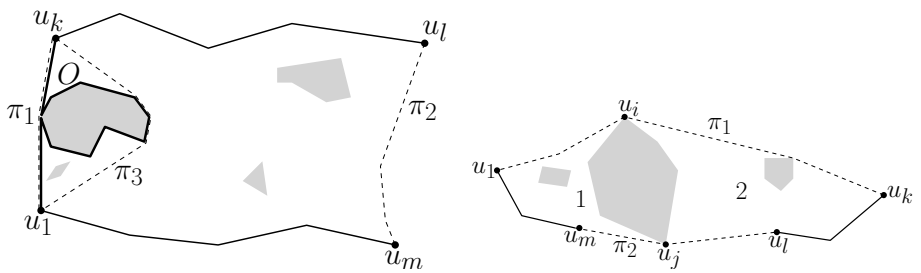


Figure 1.8: Two possible obstacle moves. In (a), to compute π_3, we extend the boundary ∂P_i to include ∂O (shown as the bold path). We then compute the shortest path from u_1 to u_k. In (b), an obstacle move where new paths to be guarded are portions of the old paths.

This move is performed when an obstacle O is touching either π_1 or π_2. First consider the case where there is an obstacle touching exactly one of π_1 or π_2. Suppose there is an obstacle touching π_1 but not π_2 as shown in Figure 1.8(a). In this case, the obstacle move is performed by finding a shortest path from u_1 to u_k in the interior of P_i excluding the points on π_1 that touch O. To compute this path, we treat obstacles touching π_1 as part of the boundary and compute a shortest $u_1 - u_k$ path as shown in

Figure 1.8(a). More precisely, let G be the visibility graph of P_i. We remove every edge of G which contains a point in $(\pi_1 \cap O)$. Then, we compute the shortest path from u_1 to u_k in this reduced visibility graph.

Let this shortest path be π_3. The third pursuer starts guarding π_3. Since the evader can be either between π_3 and π_1 or between π_3 and π_2, one of the pursuers from π_1 or π_2 will be free and the evader will be restricted to a smaller region.

In the remaining case, there is an obstacle which is touching the boundary of P_i in multiple points resulting in multiple connected components (see Figure 1.8(b)). This means that the interior of P_i is composed of multiple connected components. In this case the evader is already restricted to the connected component it lies in. The obstacle move is to simply switch to guarding the portion of π_1 and π_2 which are part of the boundary of this region. For example, on the right side of the Figure 1.8, if the evader is in region 2 then the new π_1 (resp. π_2) is the path from u_i to u_k (resp. u_l to u_j).

Lemma 12. *After an obstacle move, all the invariants mentioned above are maintained.*

Proof. We verify that each invariant is maintained.

1. In each obstacle move, we remove an obstacle from P_i and at least one vertex of this obstacle is not included in P_{i+1}.

2. An obstacle move divides P_i into at least two regions, and we pick one. Therefore, $P_{i+1} \subset P_i$.

3. π_3 is a shortest path in P_{i+1}. So are π_1 and π_2. Hence, the two guarded paths in P_{i+1} are both shortest paths.

□

1.5.2 Slicing Move

The slicing move is used to restrict the evader to a smaller polygon when no obstacle touches the guarded paths. In a slicing move two points u_a and u_b are picked from δP_i such that u_a (respectively u_b) lies on the boundary portion between u_k and u_l (respectively u_1 and u_m). We compute a shortest path between u_a and u_b and use the third pursuer to guard this path as shown in Figure 1.9. Note that if there is no path between u_a and u_b in P_i, this means that u_a and u_b are in two different components (i.e. P_i is disconnected). This can happen only when there is an obstacle whose boundary is touching δP_i at multiple locations making it disconnected. In this case we can use the obstacle move presented in the previous section (Figure 1.8(b)).

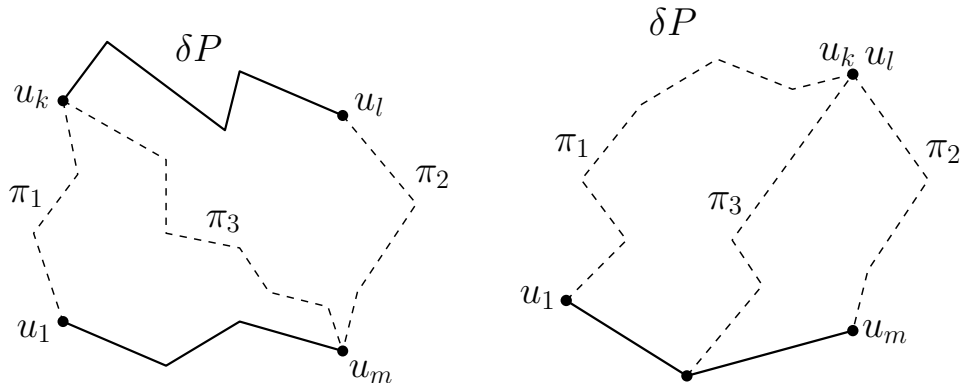
We now describe how u_a and u_b are chosen.

First, we observe that π_1 and π_2 can not have common endpoints at both ends. Since π_1 and π_2 are both shortest paths, it must be that $\pi_1 = \pi_2$ and the evader has already

been captured, otherwise we get a contradiction with the fact that neither π_1 nor π_2 is touching an obstacle.

Second, if π_1 and π_2 intersect at a vertex which is not an end-point, then P_i is disconnected and the evader can be trapped in a smaller polygon simply by discarding the components which do not contain the evader.

Hence, we are left with three possibilities which yield three variants of the slicing move based on the number of boundary vertices between the endpoints of π_1 and π_2 (Figures 1.9 and 1.10).



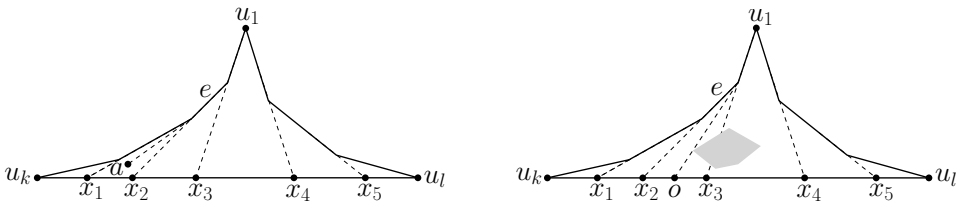
(a) Case 1: The endpoints of π_1 and π_2 are different

(b) Case 2: The paths share one endpoint. The other endpoints are not adjacent

Figure 1.9: The first two instances of the slicing move.

Case 1: If π_1 and π_2 share no common endpoints, π_3 is chosen as the shortest path connecting u_k and u_m (i.e. we pick u_k as u_a and u_m as u_b). This case is illustrated in Figure 1.9(a).

Case 2: In the second case, π_1 and π_2 share a common endpoint (say u_k), and there is at least one vertex on the boundary between the other endpoints (u_m and u_1). In this case π_3 is chosen as the shortest path connecting $u_k = u_l$ and an arbitrary vertex between the other two endpoints. This case is illustrated in Figure 1.9(b).



(a) A funnel without obstacles is partitioned by extending the edges on π_1 and π_2.

(b) A funnel with an obstacle inside. There exists a point o on the boundary such that the shortest path from u_k to o touches the obstacle.

Figure 1.10: Case 3. π_1 (resp. π_2) are the shortest paths from u_1 to u_k (resp. u_l). They share one endpoint (u_1) and the other endpoints are adjacent. i.e. (u_k, u_l) is an edge on the polygon boundary.

Case 3: In the third case, π_1 and π_2 have exactly one common endpoint and the other endpoints are adjacent (See Figure 1.10). Since an obstacle move is not possible, π_1 and π_2 are not touching any obstacles. In this case, π_1 and π_2 along with the boundary edge (u_k, u_l) form a structure called a *funnel* [24]. The common end-point (u_1 in Figure 1.10) is the apex of the funnel. Both π_1 and π_2 are inwardly convex: when walking from the apex to u_k, one would always turn locally right. This is because π_1 is a shortest path and no obstacle is touching it from the inside. Therefore, if there was a left turn, one

could find a shorter path from u_1 to u_k than π_1 which is a contradiction. A symmetric argument holds for π_2.

We now show that when the evader's current region P_i is a funnel formed by π_1, π_2 and the polygon boundary, the pursuers can trap the evader inside a triangle in such a way that at least one side of the triangle is a subset of the polygon boundary and the remaining sides are guarded by the pursuers. We start with the case when there are no obstacles inside the funnel. Even though the pursuers can readily win the game in this case by using the third pursuer and the strategy for simply connected polygons, reducing the game to a triangle yields improved capture time.

No obstacles: When there are no obstacles inside the funnel, the inward convex structure of π_1 and π_2 yields a simple partition of the funnel which can be used for computing shortest paths easily. The partition is obtained by extending each edge of π_1 and π_2 toward the edge (u_k, u_l) as shown in Figure 1.10(a). Suppose edge e on π_1 was extended to form the boundary of a partition cell. The shortest path from u_1 to point a in this partition cell continues along π_1 until it leaves e, followed by a line segment from the last vertex of e to a. We refer the last vertex on the boundary as the *corner* vertex of a point.

The pursuers scan the funnel from left to right until they reduce it to a triangle as follows: Extend all edges on π_1 and π_2 and let $x_1, \ldots, x_m$ be the intersection of the extensions with the boundary edge (u_k, u_l) as shown in Figure 1.10(a). We define

$x_0 = u_k$. Pursuer 3 guards the shortest path from u_1 to x_1. If the evader is to the left of π_3, we get a triangle. If the evader is to the right, we iterate by releasing the pursuer guarding the path from u_1 to u_k and use him to guard the shortest path from u_1 to x_2. The pursuers continue guarding the paths from u_1 to $x_2, x_3, \ldots, x_m$ until a triangle is reached.

Note that every time the funnel is shrunk by guarding x_i, the number of vertices is reduced by one: when guarding x_i, we introduce a vertex at x_i but remove two vertices: x_{i-1} and the corner of x_i. Hence the invariant $n(P_{i+1}) < n(P_i)$ is maintained.

Obstacles inside the funnel: In this case, we show that there exists a point on the edge (u_l, u_k) whose shortest path from u_1 touches an obstacle: Remove all the obstacles from the funnel and compute the partition described above. We start from the leftmost partition and move toward right. For each partition, we order all the obstacle vertices in that partition in anti-clockwise direction with respect to their corner vertex. We extend the line segment from the corner vertex to the first obstacle vertex in this ordering until it hits edge (u_l, u_k). In Figure 1.10(b), for partition tx_2x_3 we extend the line segment from t to the first vertex in the ordering until it hits (u_l, u_k) at o. Therefore the shortest path π_3 from u_1 to o touches the obstacle. The third pursuer guards this path. We now consider the part of the funnel the evader is restricted to. If the part contains no obstacles, we continue as in the previous case and reduce it to a triangle. Otherwise, π_3 is touching an obstacle. We perform an obstacle move and consider this a part of the move.

Observe that in forming π_3 we introduced a new vertex in P_i (at o in Figure 1.10(b)). However, in computing P_{i+1} we removed at least two vertices: if the evader and the obstacle are on opposite sides of π_3, either u_k or u_l as well as all vertices on the obstacle touching π_3 are removed. If they are on the same side either u_k or u_l in addition to at least one of the obstacle vertices π_3) are removed. Hence the invariant $n(P_{i+1}) < n(P_i)$ is maintained.

We now show that a slicing move maintains all invariants.

Lemma 13. *After a slicing move, all the invariants are maintained.*

Proof. For case 3, we have already shown that $n(P_{i+1}) < n(P_i)$. In all other cases, similar to the proof of Lemma 12, it can be easily verified that the slicing move maintains all invariants. □

1.5.3 Complete Strategy and Analysis

We are now ready to describe the full strategy. At the beginning of the game, two pursuers pick two separate edges on the boundary and guard them as π_1 and π_2. Afterward, the pursuers continue with performing either an obstacle move or a slicing move until the evader region becomes a triangle as follows: If an obstacle is touching π_1 or π_2, they perform an obstacle move. If an obstacle move is not possible and P_i is not a funnel, they perform one of the slicing moves given in case 1 or case 2 until they reach a funnel. Once a funnel is reached, the pursuers reduce it to a triangle as described in case 3. When

a triangle is reached, they use the third pursuer and the strategy for simply-connected polygons to capture the evader.

We now present our main result which shows that the sequence of moves described above result in capture in finite number of steps.

Theorem 2. *By following the Shortest Path Strategy, three pursuers can capture an evader in $O(n \cdot \mathrm{diam}(P)^2)$ moves in a polygon with n vertices and any number of holes.*

Proof. Suppose the step size of the pursuers and the evader is one. Let P be the initial polygon and n be the number of vertices of P. In order to guard a shortest path $\Pi \in P_i$, a pursuer must reach Π and move along it toward the evader's projection. Since the length of Π is bounded by $\mathrm{diam}(P)^2$ by Lemma 4, it can be guarded in $O(\mathrm{diam}(P)^2)$ steps.

At each round, at most two paths are guarded (Case 3 of a slicing move may contain an obstacle move) and at least one vertex is removed. Hence the total number of steps until the evader is trapped in a triangle is bounded $O(n \cdot \mathrm{diam}(P)^2)$. Once the evader is trapped in a triangle P_i, by Lemma 10, it can be captured in $O(3 \cdot diam(P_i)^2)$ steps. Since P_i is a triangle, the shortest paths inside P_i are the same as shortest paths inside P, hence its diameter is no greater than $diam(P)$. Therefore, the number of steps to capture the evader inside a triangle is $O(diam(P)^2)$.

To sum up, the strategy takes at most n rounds and the length of each round is $O(\mathrm{diam}(P)^2)$. Therefore the total number of steps is $O(n \cdot diam(P)^2)$. □

1.6 Necessity of 3 Pursuers

In this section, we complement the sufficiency of three pursuers with a lower bound. We show that any *deterministic* strategy requires at least 3 pursuers in the worst-case, and thus the upper bound of the previous section is tight.

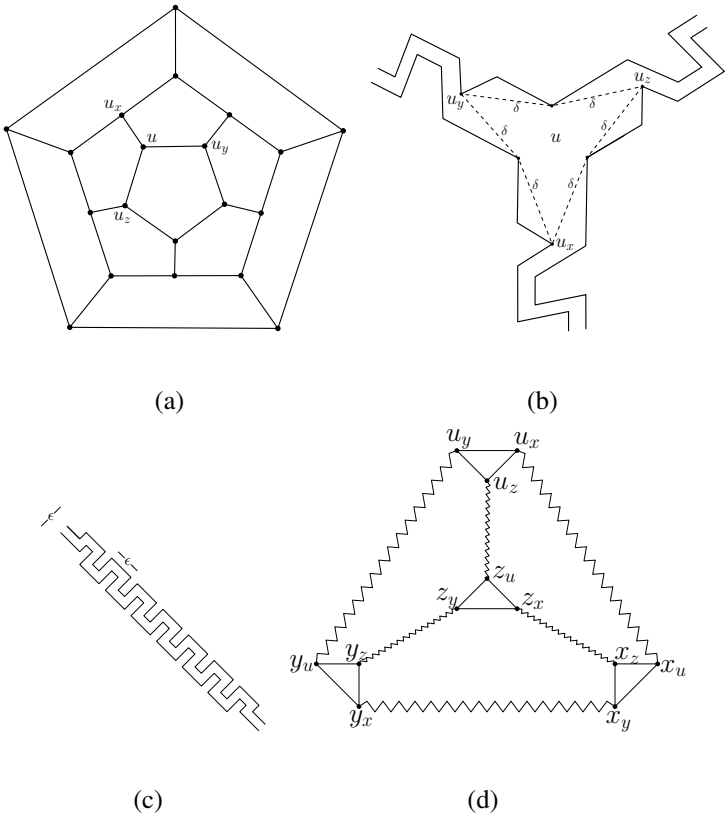


Figure 1.11: A planar graph with min-degree 3 and no three or four cycles (a), example constructed intersection (b), example edge construction (c), and example of corridors connecting intersections for the complete graph on four vertices (d), where jagged edges denote length $1 - 2\delta$ and straight edges 2δ.

Theorem 3. *There exists an infinite family of polygons with holes that require at least three pursuers to capture an evader even with complete information about the evader's location.*

Proof. The proof is based on a reduction from searching in *planar graphs*. In particular, consider a planar graph G, with vertices of degree 3, and no cycles of length three or four (see Figure 1.11(a)). Aigner and Fromme [2] proved the correctness of a simple strategy to avoid capture on such a graph, which involves moving only when a pursuer is capable of capturing it. Consider a vertex u of G with neighbors u_x, u_y and u_z. Then it is easy to see that no other vertex in the graph has more than one neighbor in the set $\{u_x, u_y, u_z\}$. Therefore, if there are only two pursuers, at least one of u's neighbors is *not adjacent to any pursuer*, and the evader can move to that neighbor without being captured on the pursuer's next turn. This argument repeats ad infinitum, showing that two pursuers cannot capture the evader in this graph. We now describe how to construct a polygon from G where the evader can mimic this reactive strategy and avoid capture forever against two pursuers.

Using Fary's Theorem, embed G so that each edge maps to a straight line segment. We now transform this straight-line embedding into a polygon with holes. First replace each node of G with an *intersection* shown in Figure 1.11(b). An intersection replacing a node u of G with neighbors x, y, z has three points labeled u_x, u_y and u_z, which we call intersection points or i-points for short. The intersection is constructed such that

the shortest path between any pair of i-points (within a single intersection) has length exactly 2δ, and a shortest path through a given intersection will visit two i-points. To finish the construction, we then connect each of these intersections with corridors, such that a corridor replacing an edge from u to x will contain the i-points u_x and x_u, and by introducing artificial bends (as seen in Figure 1.11(c)) we can guarantee the shortest u_x, x_u path in each corridor has length $1 - 2\delta$. The resulting connections between intersections for the complete graph on 4 vertices are depicted in Figure 1.11(d). It is easy to see that such a construction can ensure that all the corridors are non-overlapping, and by proper scaling of the environment we can meet all corridor length conditions. With this transformation, the outer face of the graph becomes the boundary of the polygon P, while each face of the plane graph becomes a hole.

We now argue that in the constructed polygon P, the evader can indefinitely avoid capture from two pursuers. To do so, the evader will move between the i-points of P, and guarantee that after each move the following invariant holds: both p_1 and p_2 are at least distance $1 + 2\delta$ from all i-points of e's current intersection. The game begins by each pursuer choosing a location in P, and it is easy to see that the evader can then choose some i-point such that the invariant initially holds. We then must show, that at each turn if this invariant is violated, e can move to re-establish it. By doing so we guarantee neither pursuer is ever closer than 2δ to e, and thus e can indefinitely avoid capture.

Suppose e is located at an i-point of an intersection u such that the invariant is satisfied, and the following move by the pursuers violates the invariant. Let the i-points of u be u_x, u_y, and u_z. We claim a pursuer can be within distance $1 + 2\delta$ of an i-point of at most one of x, y, and z, and break our analysis into two cases, either p lies within distance $1 - 2\delta$ of an i-point of u, or not.

In the first case, suppose without loss of generality p is within distance $1 - 2\delta$ of u_x, meaning it lies in the corridor from u to x. Then, as the invariant held before p moved necessarily $d(p, u_x) \geq 2\delta$. Further, as the i-points of u are 2δ apart, it is easy to see that $d(p, u_y) \geq 4\delta$, and $d(p, u_z) \geq 4\delta$. Thus, as $d(u_y, y_u) = 1 - 2\delta$ and $d(u_z, z_u) = 1 - 2\delta$ it follows that p is at least distance $1 + 2\delta$ from the i-points of y and z.

Consider the second case where p is further than $1 - 2\delta$ from the i-points of u and within $1+2\delta$ of i-points of two intersections in the set $\{x, y, z\}$. Without loss of generality suppose they are y and z. Then there exists i-points y_v and z_w such that $d(p, y_v) < 1+2\delta$ and $d(p, z_w) < 1 + 2\delta$. Consider the following cycle, $p, y_v, y_u, u_y, u_z, z_u, z_w, p$, which has length at most $(1 + 2\delta) + 2\delta + (1 - 2\delta) + 2\delta + (1 - 2\delta) + 2\delta + (1 + 2\delta) = 4 + 6\delta$. This cycle then has length less than 5, as we can always construct P with an arbitrarily small δ. Further, as p is at least $1 - 2\delta$ from the i-points of u, the shortest paths from p to y_v and z_w to p cannot pass through a corridor adjacent to u without being longer than $1 + 2\delta$, thus this cycle surrounds one or more holes of P. However, G has no cycles of

length three or four, thus the cycle in P then must have length five or more, and this is a contradiction.

Thus each pursuer is within distance $1 + 2\delta$ of an i-point of at most one intersection in the set $\{x, y, z\}$. Thus one of x_u, y_u, and z_u will satisfy the invariant and as they are all within distance one of the i-points of u, e can move to the one which satisfies the invariant. Thus, at each turn e can re-establish the invariant and indefinitely avoid capture.

□

Chapter 2

Complete Information Pursuit Evasion on Polyhedral Surfaces

2.1 Introduction

In the previous Chapter, we showed that three pursuers are always sufficient to capture the evader in a polygon with obstacles if the evader's location is already known to the pursuers. However, many robotics applications involve searching or tracking on "terrain-like" surfaces. We thus investigate a pursuit-evasion game played on the (closed) surface of a 3-dimensional polyhedron. Suppose multiple pursuers attempt to capture an adversarial evader, with all players constrained to remain on the polyhedral surface, and all able to move equally fast, how many pursuers are needed to capture the evader in finite time?

In addition to the problem's practical motivations, it is also well-motivated from a theoretical perspective; the surface acts as an "intrinsic" obstacle, introducing non-linearity

*Parts of this chapter appeared in [40].

in the behavior of shortest paths. For instance, *although the genus zero polyhedral surface is topologically equivalent to a disk, the game has a distinctly different character and outcome than its planar counterpart (circular arena).* In particular, it is known that a single pursuer can always win the discrete-time man-and-the-lion game in the plane (an easy corollary of [66]). Therefore, one may hope that an appropriate topological extension of the "follow the shortest path towards the evader" strategy will also succeed on the polyhedral surface. However, we show that this is not possible, and provide a constructive lower bound that *at least 3 pursuers* are needed in the worst-case for successful capture on a polyhedral surface. Intuitively, the problem is caused by the discontinuity in mapping "straight line" shortest paths in the unobstructed planar arena to geodesics on the polyhedral surface; in the unobstructed plane, a small move by the evader only causes a small (local) change in the straight line connecting pursuer and the evader, but on the polyhedral surface, the geodesic can jump discontinuously.

Complementing our lower bound, we show that 4 pursuers always suffice on any polyhedral surface of genus zero. Specifically, we present a strategy for the pursuers that always leads to capture of the evader in $O(\mathrm{diam}(S)(n^2 \log n + \log \mathrm{diam}(S)))$ time steps, where n is the number of vertices of the polyhedral surface S and $\mathrm{diam}(S)$ is its diameter (the maximum shortest path distance between any two points). We then generalize our result to surfaces of non-zero genus and prove that $(4 + 4g)$ pursuers can always capture an evader on the surface of any genus g polyhedron. Our technique for

analyzing pursuit evasion on polyhedral surfaces appears to be quite general, and likely to find application in other settings. As one example, we consider pursuit evasion under the "weighted region" model of shortest paths, where non-negative weights dictate the per-unit cost of travel through different regions of the surface.

2.1.1 Related Work

There exists an extensive literature on pursuit-evasion in 3-dimensional environments and surfaces, but no result appears to be known on the number of pursuers necessary for capture. Instead, the prior research has focussed on heuristics approaches for capture [41], classification of environments where capture is achievable [5], or on game-theoretical questions [43, 51].

The most relevant work to our research is the cops-and-robbers games in graph theory, where Aigner and Fromme have shown that 3 cops always suffice against a robber in any planar graph [2], and $\lfloor 3g/2 + 3 \rfloor$ cops are necessary for graphs of genus g [63]. However, the continuous-space of polyhedral surfaces requires very different set of techniques from those used for graphs.

2.2 Preliminaries and the Lower Bound

We assume that an evader and pursuers are free to move on the (closed) surface of a 3-dimensional polyhedron S using the standard model of Section 0.2. We assume that

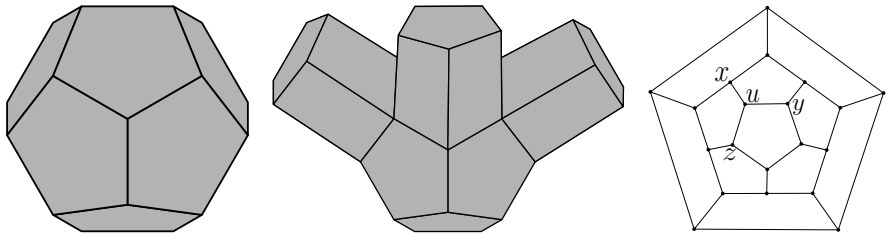
S has n vertices, and therefore $O(n)$ faces and edges. Without loss of generality, we assume that each face is a triangle, which is easily achieved by triangulating the faces with four or more sides.

We use the notation $\Pi(a, b)$ for a shortest path between two points a and b on the surface S, and $d(a, b)$ for the length of this path. (In general, the path $\Pi(a, b)$ is not unique, but its length is.) The path $\Pi(a, b)$ is piece-wise linear and its vertices lie on the edges or vertices of the surfaces. Throughout, we will use the terms *vertices* and *edges* to refer to the graph of the polyhedral surface, and *points* and *arcs* to refer to the geometric objects embedded on the surface such as a path. We explain specific properties of these shortest paths that are used in our analysis in Section 2.3.3. The following theorem establishes the lower bound for our pursuit game.

Theorem 4. *In the worst-case at least three pursuers are required to capture an evader on the surface of a polyhedron.*

Proof. We start with a dodecahedron D, all of whose edges have length 1, as shown in Fig. 2.1(a). Our polyhedron S is constructed by extending each face of D *orthogonally* (to the face) into a "tower" of height $\mathrm{diam}D + 1$, where $\mathrm{diam}D$ is geodesic diameter of the dodecahedron; see Fig. 2.1(b). The polyhedron S has 12 such towers, one for each of the 12 pentagonal faces of D. The "walls" of these towers meet along the edges of D, forming the skeleton graph, which we denote $G(D)$, as shown in Fig. 2.1(c). We now

argue that an evader can indefinitely avoid capture from two pursuers on the surface of this polyhedron.



(a) (b) (c)

Figure 2.1: A dodecahedron (a); partial construction with three faces orthogonally extended (b); and the skeleton graph (c).

Suppose there are only two pursuers, p_1 and p_2. Initially, they choose their locations on S, and then the evader picks its initial position at a vertex of $G(D)$ to satisfy $d(p_i, e) > 1$, for $i = 1, 2$. (It is easy to see that this is possible.) Our proof shows that regardless of the pursuers' strategies, the evader can indefinitely maintain this distance condition (after its move) by always moving among the vertices of $G(D)$, and thus evade capture forever.

The key observation is that the evader's choice to remain on the skeleton graph $G(D)$ means that pursuers gain no advantage from positions not on $G(D)$. In particular, any pursuer located on the top face of a tower is not an immediate threat to the evader, and thus can be safely ignored by the evader. (Such a pursuer is more than 2 moves away from threatening the evader.) Similarly, for any pursuer p positioned on a wall, map

its position to the nearest point p_s on the skeleton, called the *shadow of* p. (Thus, p_s is the foot of the perpendicular from p to the edge of $G(D)$ that is closest to p.) Since $d(p_s, e) \leq d(p, e)$, the evader only needs to ensure its distance to p_s remains more than 1. Thus, we only need to ensure that the evader maintains its distance condition with respect to two pursuers (or their shadows) that are constrained to move along the skeleton graph. *However, the pursuers are not constrained to the vertices of the skeleton graph; they can situate in the interior of skeleton graph edges.*

The evader's strategy is *reactive*: it remains at a vertex until some pursuer is within distance 1. When one or both pursuers are within distance 1 of the evader, we show that the evader can move to a safe neighboring vertex and restore its distance condition. In particular, suppose evader's current location is vertex u in $G(D)$, and let x, y, z be the three neighboring vertices of u. Then it is easy to see that no point other than u among (the line segments forming) the edges of $G(D)$ is within distance one of more than one neighbor in the set $\{x, y, z\}$. This follows because the minimum length path joining any two points of the skeleton graph lies entirely in the skeleton graph (i.e., it does not use the walls or top faces of the surface S). Thus, at least one neighbor of u among $\{x, y, z\}$ is more than 1 away from both the pursuers, and this is the vertex to which e moves.

□

2.3 Catching the Evader with 4 Pursuers

We begin with a high level description of the pursuers' strategy, and then develop the necessary technical machinery to prove its correctness.

2.3.1 Surround-and-Contract Pursuit Strategy

The pursuers' overall strategy is conceptually quite simple: repeatedly shrink the region containing the evader while making sure that it cannot escape from this region, which can be intuitively thought of as a *surround-and-contract* strategy. More specifically, at any time, the evader is constrained within a connected portion S_i of the surface S, which is bounded by at most three paths, each guarded by a pursuer. The fourth pursuer is used to divide S_i into two non-empty regions (contraction), trapping the evader within one of them. This division is done in such a way that that at least one of the 3 pursuers bounding S_i becomes free, thus allowing the process to continue until the target region reduces to a single triangle, and the capture can be completed.

The paths used by the pursuers are shortest paths on the polyhedral surface, *restricted to the current region.* The computation of shortest paths on a polyhedral surface is a well-known problem in computational geometry, and we rely on the following result of [14, 53]: given a source point x on the surface of a polyhedron S of n vertices, one can compute a shortest path map encoding the shortest paths from x to all other points

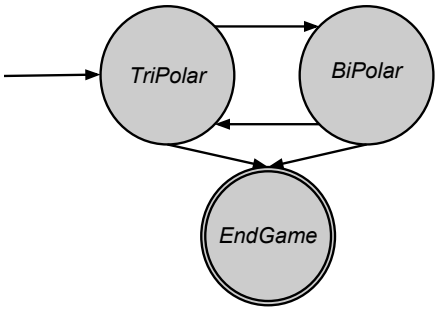


Figure 2.2: A finite state machine representing the possible states of the pursuit and transitions between them.

on S, in $O(n^2)$ time using $O(n \log n)$ space. With this map, one can find the shortest path from x to any other point y in time $O(\log n + k)$ when the path consists of k arcs.

We use *phases* to monitor the progress of the algorithm: in phase i, the region containing the evader is denoted S_i where $S_i \subseteq S_{i-1}$, for all i. Each time the pursuers guard a new path dividing S_i, the phase transitions, with S_{i+1} as the region containing the evader. In addition, each region S_i has a rather special form: it is bounded by either two or three shortest paths. The finite automaton of Figure 2.2 shows the simple state diagram of the pursuit: the pursuit transitions between regions bounded by 2 and 3 paths until it reaches a special terminal state marked ENDGAME. For ease of reference, we name the first two states BIPOLAR and TRIPOLAR to emphasize that the regions corresponding to these states are bounded by shortest paths between 2 or 3 points (poles). The region in the terminal state ENDGAME is also bounded by 3 paths but contains no vertices in the interior (only the points of the boundary paths), which simplifies the

search leading to capture. In particular, the three possible states throughout the pursuit are the following:

BIPOLAR: S_i is bounded by two shortest paths $\Pi(a, b)$ and $\Pi(a, b)'$ between two points (poles) a and b.

TRIPOLAR: S_i contains at least one interior vertex, and is bounded by three shortest paths $\Pi(a, b)$, $\Pi(b, c)$, and $\Pi(a, c)$.

ENDGAME: S_i has no interior vertices and is bounded by three shortest paths $\Pi(a, b)$, $\Pi(b, c)$, and $\Pi(a, c)$.

We initialize the pursuit by choosing a triangular face (a, b, c) of the surface, and assigning one pursuer to each of the three (single-arc) shortest paths $\Pi(a, b)$, $\Pi(b, c)$, and $\Pi(a, c)$. If the evader lies inside the triangle face, we enter the terminal state ENDGAME; otherwise, we are in state TRIPOLAR. The fourth pursuer shrinks the region S_i, resulting in a smaller TRIPOLAR region, or forces a transition to a BIPOLAR region. In each state BIPOLAR, at least one interior vertex is eliminated from S_i. Further, each state consists of a finite number of phases, which guarantees that the algorithm terminates in the region ENDGAME.

In the following, we use $\nu(S_i)$ to denote the *number of interior* vertices of S_i; that is, the number of vertices in S_i that are not on the boundary paths. Throughout the pursuit, the following invariant is maintained.

PURSUIT INVARIANT. During the ith phase of the pursuit, (1) $S_i \subseteq S_{i-1}$, (2) $\nu(S_i) \leq \nu(S_{i-1})$, and if phase $i-1$ is in state BIPOLAR, then $\nu(S_i) < \nu(S_{i-1})$, and (3) at most 4 paths are guarded, each by a single pursuer at any time.

The first condition ensures that the region containing the evader only shrinks; the second ensures that at least one interior vertex is removed in state BIPOLAR; and the third ensures that 4 pursuers succeed in capturing the evader.

2.3.2 Guarding Shortest Paths

Our algorithm employs one pursuer to guard a shortest path, ensuring that any attempt by the evader to cross the shortest path leads to capture. In Chapter 1 Section 1.3, we show that a single pursuer can accomplish this for minimal paths in polygons. However, a minimal path is a merely a more general form of a shortest path, and thus any shortest path is minimal and thus also guardable by a single pursuer. Further, the proofs in Section 1.3 rely only on the minimality of the paths and not the two-dimensional problem, thus the result easily extends to this setting, giving the following result.

Lemma 14. *Consider a shortest path $\Pi(a, b)$ on the polyhedral surface S, and suppose a pursuer p is located at the endpoint a of this path. Then, after at most $L + 1$ moves, p can locate itself at the canonical projection of the evader, where L is the (Euclidean) length of the $\Pi(a, b)$.*

2.3.3 Pursuit Strategy for the TRIPOLAR State

In TRIPOLAR state, the current region S_i is bounded by three shortest paths, $\Pi(a, b)$, $\Pi(a, c)$, and $\Pi(b, c)$, between the three poles a, b, c. The pursuers' strategy is to force the game either into BIPOLAR or ENDGAME state while preserving the Pursuit Invariant. Towards that goal, we need to introduce some properties of shortest paths on polyhedral surfaces.

It is well-known that a shortest path is a sequence of line segments (arcs), whose endpoints lie on the edges of the surface, and that the path crosses any edge of the surface at most once. Thus, the sequence of edges crossed by a path, called the *edge sequence*, consists of at most n edges. Given a source point a and an edge (b, c), it is also known that (b, c) is partitioned into $O(n)$ closed *intervals of optimality* [53], where the shortest path from a to any point d in an interval follows the same edge sequence. Let us suppose that an edge (b, c) is partitioned into k intervals of optimality, $[d_0, d_1], [d_1, d_2], \cdots, [d_{k-1}, d_k]$, where the edge sequence for the interval $[d_{i-1}, d_i]$ is denoted as σ_i. Since two adjacent intervals, say $[d_{j-1}, d_j]$ and $[d_j, d_{j+1}]$, share a common endpoint d_j, there are two equal length shortest paths from a to d_j, following edge sequences σ_j and σ_{j+1}. Because our algorithm may guard one or both of these shortest paths, we use a superscript to identify the associated edge sequence. In particular, the shortest path from x to y under the edge sequence σ_j is denoted $\Pi(x, y)^j$.

The following lemma shows that if the shortest paths $\Pi(a, b)$ and $\Pi(a, c)$ have the same edge sequence, and $\Pi(b, c)$ is a single arc, then the interior of the region bounded by these 3 paths has no vertex of the surface, which implies that the pursuit region has entered the terminal state ENDGAME.

Lemma 15. *Suppose the current region S_i is bounded by pairwise shortest paths between the three points a, b, c, and that $\Pi(b, c)$ consists of a single arc. Then, the paths $\Pi(a, b)$ and $\Pi(a, c)$ follow the same edge sequence if and only if S_i contains no interior vertices.*

Proof. Clearly, if $\Pi(a, b)$ and $\Pi(a, c)$ have the same edge sequence, then there cannot be an interior vertex in S_i because $\Pi(b, c)$ is a single arc. For the converse, if S_i has no interior vertices and $\Pi(b, c)$ is a single arc, then S_i can only contain edges that intersect both $\Pi(a, b)$ and $\Pi(a, c)$. These edges do not cross each other, and therefore they must be crossed by $\Pi(a, b)$ and $\Pi(a, c)$ in the same order. □

By the preceding lemma, if $\Pi(a, b)$ and $\Pi(a, c)$ follow the same edge sequence and $\Pi(b, c)$ consists of a single arc, then we are in the terminal state ENDGAME. Therefore, assume that either the edge sequences of $\Pi(a, b)$ and $\Pi(a, c)$ are unequal or $\Pi(b, c)$ consists of multiple arcs. In both cases, the following lemma shows how to either reduce $\Pi(b, c)$ to a single point, which changes the state to BIPOLAR, or replace $\Pi(a, b)$ and $\Pi(a, c)$ with shortest paths with the same edge sequence, and $\Pi(b, c)$ with a single arc, which changes the state to ENDGAME.

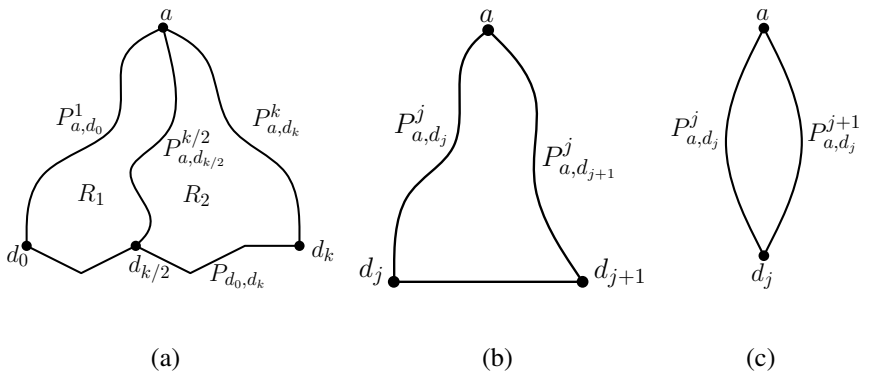


Figure 2.3: Illustration for the proof of Lemma 16.

Lemma 16. *Suppose S_i is in state* TRIPOLAR*, then we can force a transition either to state* BIPOLAR *or state* ENDGAME*.*

Proof. Consider the shortest path map with source a, and suppose it partitions $\Pi(b, c)$ into k intervals of optimality (across all of $\Pi(b, c)$'s arcs), $[d_0, d_1]$, $[d_1, d_2] \cdots$, $[d_{k-1}, d_k]$ with corresponding edge sequences $\sigma_1, \sigma_2, \cdots, \sigma_k$, where $d_o = b$ and $d_k = c$. Relabel $\Pi(a, b)$ as $\Pi(a, d_0)^1$, and $\Pi(a, c)$ as $\Pi(a, d_k)^k$, and order the paths by their endpoints on $\Pi(b, c)$ as follows:

$$\Pi(a, d_0)^1, \Pi(a, d_1)^1, \Pi(a, d_1)^2, \Pi(a, d_2)^2, \ldots, \Pi(a, d_{k-1})^k, \Pi(a, d_k)^k$$

We leave two pursuers to guard (maintain canonical projections on) the paths $\Pi(a, b)$ and $\Pi(a, c)$, and deploy a guard on the center path $\Pi(a, d_{k/2})^{k/2}$ (constrained to lie within the current region); see Figure 2.3(a). This path splits the original region S_i into two non-empty regions, each containing half the intervals of optimality, and we

recurse the process on the side with the evader, namely, the region S_{i+1}. The first two conditions of the invariant are trivially satisfied, since the evader region can only shrink, and the third condition holds because the pursuer associated with either the path $\Pi(a, b)$ or $\Pi(a, c)$ is freed up, keeping the total pursuer count at four.

The recursion terminates when the evader is confined between two successive paths in the original ordering. In particular, if the evader is trapped between paths $\Pi(a, d_j)^j$ and $\Pi(a, d_{j+1})^j$, then we have state ENDGAME as shown shown in Fig. 2.3(b). On the other hand, if the evader is trapped between two paths $\Pi(a, d_j)^{j-1}$ and $\Pi(a, d_j)^j$, we have successfully transitioned to state BIPOLAR, as shown in Fig. 2.3(c). It is clear that throughout this search, the evader remains confined to a subsurface of S_i and cannot escape without being captured, and that the pursuit invariant is maintained. Because the path $\Pi(b, c)$ has at most n arcs, with n intervals of optimality each, we have $k \leq n^2$. Thus, in $O(\log n)$ phases, we can force a change of state to either BIPOLAR or ENDGAME. □

2.3.4 Pursuit Strategy for the BIPOLAR State

We now describe how to make progress when the search region is BIPOLAR. Without loss of generality, assume that the current region S_i is bounded by two shortest paths between points a and b, each guarded by a pursuer. The algorithm shrinks the region by removing at least one vertex from the interior of S_i. In particular, let c be a vertex

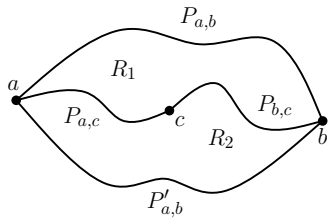


Figure 2.4: An abstract illustration of the two paths that may be guarded during state BIPOLAR.

of the surface that lies in the interior, and consider the two shortest paths (constrained to remain inside S_i) from c to a and b. The concatenation of these two paths splits S_i into two subregions, say R_1 and R_2, both bounded by three paths. (These paths can share a common prefix, starting at c, but they do not cross each other.) Only one of these regions contains the evader, and so by guarding $\Pi(a, b)$ an $\Pi(a, c)$ the state of the search transitions to either TRIPOLAR or ENDGAME depending on whether or not this region, which becomes S_{i+1}, contains an interior vertex. See Figure 2.4 for illustration. During this transition the pursuit invariant holds because (1) $R_1, R_2 \subseteq S_i$, (2) both R_1 and R_2 contain at least one fewer interior vertex, namely, c, and (3) at most 4 pursers are used. Thus, we have established the following lemma, completing the discussion of the state BIPOLAR.

Lemma 17. *If the evader lies in a* BIPOLAR *region S_i, then we can force a transition to a* TRIPOLAR *or* ENDGAME *region with at least one fewer interior vertex, and no more than 4 pursuers are used during the pursuit.*

2.3.5 Pursuit Strategy for the ENDGAME State

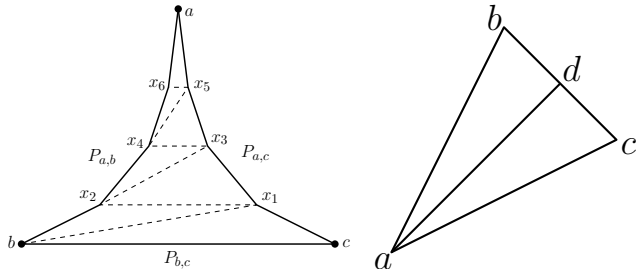
We now describe how the pursuers capture the evader when the search region is ENDGAME. First, by Lemma 16, the path $\Pi(b, c)$ can be reduced to a single arc. Next, by Lemma 15, since S_i has no interior vertices, $\Pi(a, b)$ and $\Pi(a, c)$ follow the same edge sequence. Thus, S_i consists of a chain of faces, each a triangle or a quadrilateral. For ease of presentation, we assume that all faces are triangles, which is easily achieved by adding a diagonal to each quadrilateral. The pursuers perform a sweep of S_i, by repeatedly replacing $\Pi(b, c)$ with the previous edge in the edge sequence of $\Pi(a, b)$ and $\Pi(a, c)$, until the evader is trapped in a triangle each of whose sides are guarded by a pursuer. For example, in Figure 2.5(a), the fourth pursuer guards the edge (b, x_1), which either confines the evader to the triangle b, c, x_1 or frees the evader guarding $\Pi(b, c)$.

Lemma 18. *Once the evader enters the* ENDGAME *state, the 4 pursuers can shrink the confinement region to a single triangle of S_i in $O(n)$ phases.*

Finally, the following lemma completes the capture inside the triangle.

Lemma 19. *If S_i consists of a single triangle, then in $O(\mathrm{diam}(S) \log \mathrm{diam}(S))$ moves the evader can be captured.*

Proof. The pursuers progressively "shrink" the triangle containing the evader, leading to eventual capture, as follows. Pick the midpoint of the arc (b, c), say d, and deploy



(a) (b)

Figure 2.5: Illustrating the algorithm used for capture in state ENDGAME.

a guard on the arc (a, d); see Figure 2.5(b). This path splits the original triangle into two non-empty triangles, and we recurse the process on the triangle containing the evader. Notice that the pursuer associated with either the path $\Pi(a, c)$ or $\Pi(a, b)$ is freed up, keeping the total pursuer count at four. After $\log \mathrm{diam}(S)$ applications (b, c) will be replaced with an arc of length at most one, at which point a pursuer can capture the evader by sweeping the triangle once. At most $O(\log \mathrm{diam}(S))$ paths of length $O(\mathrm{diam}(S))$ are guarded, and so this process takes at most $O(\mathrm{diam}(S) \log \mathrm{diam}(S))$ moves. □

We analyze the total number of moves before the evader is captured. The total number of pursuer moves over all the BIPOLAR and TRIPOLAR moves is bounded by the number of paths guarded times the number of steps to guard those paths. By Lemma 14, the time to guard a path (reach the canonical projection) is proportional to its length, and the following result shows an upper bound on this length.

Lemma 20. *At a given phase i,* $\mathrm{diam}S_i$ *is* $O(n \cdot \mathrm{diam}(S))$.

Proof. Consider the longest shortest path Π in S_i. Because it crosses any edge of the surface at most once, it has $O(n)$ arcs, each of length at most $\mathrm{diam}(S)$, which yields the desired bound. □

We can now state our main result.

Theorem 5. *On a n-vertex genus 0 polyhedral surface S, 4 pursuers can always capture the evader in $O(\mathrm{diam}(S)(n^2 \log n + \log \mathrm{diam}(S)))$ moves.*

Proof. There are at most n phases in state BIPOLAR because each occurrence removes at least one interior vertex from S_i. Only two paths are guarded per phase, so BIPOLAR takes $O(n^2 \cdot \mathrm{diam}(S))$ moves. There are at most $O(\log n)$ phases during TRIPOLAR to force the state transition, each requiring a single path to be guarded. Further, there at most n transitions from state BIPOLAR to TRIPOLAR, and thus there are at most $O(n \log(n))$ phases in state TRIPOLAR, requiring $O(n^2 \log(n) \cdot \mathrm{diam}(S))$ moves. In state ENDGAME, there are $O(\log n)$ phases where a path is guarded to reduce $\Pi(b, c)$ to a single arc, an additional $O(n)$ phases where an arc is guarded to confine the evader to a face, and finally $O(\log(\mathrm{diam}(S)) \cdot \mathrm{diam}(S))$ moves are needed to capture the evader in a triangle. Adding them up, in the worst case, the total number of moves is $O(n^2 \log(n) \cdot \mathrm{diam}(S) + \log(\mathrm{diam}(S)) \cdot \mathrm{diam}(S))$. □

2.4 Catching the Evader on Genus g Surface

In this section, we show that $(4 + 4g)$ pursuers are sufficient to catch the evader on a polyhedral surface of genus $g > 0$. The main idea is to cut the surface along $2g$ cycles, reducing it to a genus 0 surface. By assigning 2 pursuers to each cycle, we can ensure that each cycle is guarded, and that the evader cannot cross a cycle without being captured. We then use the 4 remaining pursuers to search the genus zero surface and capture the evader.

The existence of these $2g$ cycles that split a genus g surface into genus 0 subsurfaces follows from a result of Erickson and Whittlesey [19]. The intuition behind their algorithm is simple: compute the cut locus of S, which is the closure of the set of points with at least two shortest paths from a base-point x. Then greedily choose the shortest cycle that does not disconnect the surface of S; remove this cycle; and choose the next shortest cycle that does not disconnect S with the first cycle removed, and so on. After choosing $2g$ such cycles, all remaining cycles disconnect the surface, and thus the resulting surface has genus zero. Erickson and Whittlesey show that this greedy strategy results in a set of $2g$ loops with the minimum possible sum of lengths, for the given base-point. We need only a weaker property that each cycle is geodesic (composed of two shortest paths), and thus cannot be shortcut. In particular, we need the following result from [19].

Theorem 6. *[19] Given any piecewise-linear manifold M in $\mathbb{R}^3$ and any base-point $x \in M$, we can compute the shortest system of loops for M based at x in $O(n^2)$ time.*

Using a total of $4g$ pursuers, the $2g$ cycles found by Erickson and Whittlesey's algorithm can be guarded, confining the evader to a gn vertex subsurface of genus zero, whereupon the evader can be captured by 4 additional pursuers.

Theorem 7. *On a n-vertex genus g polyhedral surface S, $4g + 4$ pursuers can always capture the evader in $O(((gn)^2 \log(gn) + \log \operatorname{diam}(S)) \cdot \operatorname{diam}(S))$ moves.*

2.5 Pursuit Evasion with Weighted Regions

Our surround-and-contract technique appears to be quite general, and may be applicable to many other settings where shortest paths are well-behaved and where the frequency of state transitions between BIPOLAR and TRIPOLAR can be combinatorially bounded. As one illustrative example, we consider the pursuit evasion problem on a polyhedral surface under a *region weighted* definition of shortest paths, and deduce the same result that 4 pursuers suffice in this setting as well.

In the weighted region model, each face f (triangle) of the polyhedral surface S is associated with a non-negative real weight w_f, and traveling along a line segment of length ℓ on this face incurs a *cost* of $w_f \cdot \ell$. (These weights can be used to model the non-homogeneity of movement speed on a terrain, such as paved roads, dirt roads, marshlands, sand, water etc.) Each edge e of the polyhedron also has a weight $w_e \in (0, \infty]$ subject

to the condition $w_e \leq min\{w_f, w'_f\}$, where f, f' are the two faces adjacent to e. This condition is imposed to disallow the unnatural phenomena of paths that travel arbitrarily close to an edge, but not on it.[1]

A *minimum cost path* between a source x and a destination y under the weighted metric is a non-self-intersecting piecewise linear curve, which only makes turns at edges and vertices of the surface. These shortest paths, however, have some characteristics that are distinctly different from Euclidean length paths: e.g., the straight line segment xy is not necessarily the minimum cost path for two points x and y in the interior of a convex face f. Nevertheless, we show below that the basic structure and proof of surround-and-contract holds in this generalized path setting.

We first dispense with a computational issue. Computing a weighted region path is computationally non-trivial, and all the polynomial-time algorithm only compute a $1 + \epsilon$ approximation [3, 49, 52]. (Algorithms that compute an exact minimum cost path take time doubly exponential in n [52].) Our primary goal, however, is to show the correctness of the strategy and sufficiency of 4 pursuers, and we do not concern ourselves with the computational aspects.

A more serious issue is the *combinatorial complexity* of the paths. While Euclidean shortest paths cross a single edge of the surface at most once, a weighted minimum cost

[1] edges of infinite weight and, for such an edge, it is permissible to travel along it at the cost of its neighboring face, but it is not permissible to cross it. This allows for the modeling of an impenetrable obstacle.

path can cross an edge $\Theta(n)$ times! Without a more careful analysis, this can lead to an exponential blowup in the complexity of the surface regions: a region bounded by 3 shortest paths can have boundary complexity $\Omega(n^2)$, causing a shortest path in it to have complexity $\Omega(n^4)$, and so on. In the following, we offer a more refined analysis of the weighted shortest paths and prove that such an explosion does not occur.

2.5.1 Path Complexity under the Weighted Metric

The number of moves required to transition between states BIPOLAR and TRIPOLAR is controlled by the number of arcs in a shortest path and the intervals of optimality into which an arcs is subdivided. Deriving a non-trivial upper bound on this complexity requires delving into the proof of the weighted region shortest path algorithm of Mitchell et al. [52], which is quite complicated. Instead, we offer below a substantially simpler and direct proof for bounding the number of intersections between an edge and a shortest path, which we are able to generalize to our more involved setting.

Lemma 21. *Let S be an n-vertex polyhedron with weighted regions, then any shortest path in S has at most $O(n^2)$ arcs.*

Proof. Fix a shortest path P in S_i, order the edges of S_i in the increasing order of weight, and let $e_1, e_2, \ldots, e_n$ be this order. We claim that the edge e_i intersects P only $O(i)$ times.

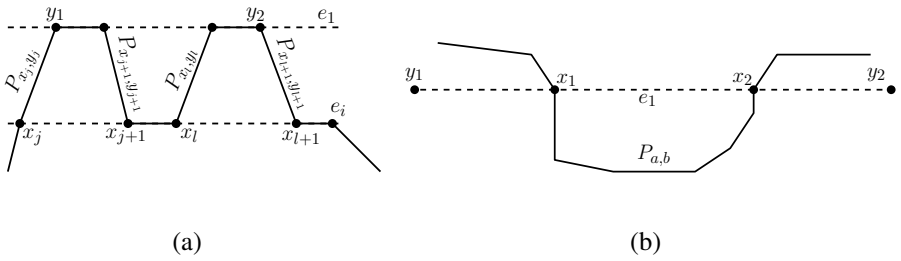


Figure 2.6: Illustration of proof of Lemma 21.

Consider the edge e_i, and suppose it is intersected by Π at $x_1, x_2, x_3, \ldots x_k$ in that order (meaning x_3 could precede x_2 on e_i, but Π visits x_2 before x_3). Then we claim that Π can always be chosen such that only one subpath $\Pi(x_j, x_{j+1})$ intersects e_1. For the sake of contradiction, suppose that $\Pi(x_l, x_{l+1})$ also intersects e_1, and without loss of generality suppose that $j + 1 \leq l$. Further, let the first and last points $\Pi(x_j, x_{l+1})$ intersects e_1 be y_1 and y_2 respectively (see Figure 2.6(a)). Then, we can construct a path Π' by replacing the subpath $\Pi(y_1, y_2)$ of Π with the arc (y_1, y_2). As $\Pi(y_1, y_2)$ has Euclidean length at least as long as (y_1, y_2), and is weighted at least as much as e_1 everywhere, Π' can be no longer than Π. This process can be repeatedly applied until there is only one subpath $\Pi(x_{j'}, x_{j'+1})$ which intersects e_1.

Now suppose there are two subpaths $\Pi(x_j, x_{j+1})$, and $\Pi(x_l, x_{l+1})$ which intersect e_2 (excluding the single subpath which intersects e_1) first at y_1 and last at y_2. Since we know neither subpath can intersect e_1, neither subpath can traverse an edge of weight smaller than e_2. Further, neither subpath can traverse a face f of smaller weight, as to do so would require crossing one of f's bounding edges, which we know have weight

less than or equal to the faces they bound. Thus, $\Pi(y_1, y_2)$ has no arc with weight less than that of e_2, and thus the arc (y_1, y_2) has weighted length no more than $\Pi(y_1, y_2)$ and can replace it. Note that (y_1, y_2) cannot intersect Π, as to do so would contradict the choice of Π as the shortest path. Therefore, Π can be chosen such that only one subpath $\Pi(x_{j'}, x_{j'+1})$ intersects e_2 (in addition to the single subpath which intersects e_1). This process can be continually applied for each edge e_j where $j < i$ until we have accounted for all $i - 1$ edges with weight less than or equal to e_i.

The resulting scheme accounts for at most $i-1$ subpaths to intersect e_i. Suppose then that were an i-th subpath, $\Pi(x_j, x_{j+1})$. This subpath could necessarily only intersect e_l such that $l > i$. Thus, we can construct a path Π' no longer than Π by replacing the subpath $\Pi(x_j, x_{j+1})$ of Π with the arc (x_j, x_{j+1}). Thus in the worst case $k = 2(i-1) + 2 = 2i$, where there are $i - 1$ subpaths intersecting e_i, and an arc preceding and following x_2 and x_{k-1} respectively. Thus in the worst case Π consists of $O(n^2)$ arcs. □

The following lemmas generalize the previous result to bound the growth in path complexity during our algorithm.

Lemma 22. *Suppose S_i is a subregion of S whose boundary paths have m edges in total. Then a shortest path Π inside S_i has at most $m + O(n^2)$ arcs.*

Proof. Using a nearly identical proof to that of Lemma 21, it can be shown that the i-th shortest edge of S will still have $O(i)$ subpaths intersecting it, excluding those of the boundary. In Figure 2.6(a) we replaced $\Pi(y_1, y_2)$ with the arc (y_1, y_2). However, the situation in Figure 2.6(b) may arise, that is, e_1 may be intersected by the boundary of S_i. Because the boundary consists of only shortest paths, replacing (y_1, y_2) with the concatenation of $(y_1, x_1), \Pi(x_1, x_2), (x_2, y_2)$ will guarantee that there is only one *new* subpath intersecting e_i that intersects e_1, but, the boundary portion of that subpath may intersect e_i several times. Thus, a shortest path in S_i has at most $O(n^2)$ arcs from the $O(i)$ intersections per edge, and at most m additional arcs from the boundary of S_i. □

Lemma 23. *Suppose after j state transitions the evader is confined to S_i. Then, a shortest path in S_i consists of at most $O(j \cdot n^2)$ arcs.*

Proof. We now show that the boundary paths can only grow in complexity during state transitions (and not between two phases without a state transition). Thus a boundary path would gain at most $O(n^2)$ arcs per state transition, for a total of $O(j \cdot n^2)$ after j state transitions.

First observe that if S_i is in state BIPOLAR, then the boundary of S_{i+1} will have at most $O(n^2)$ more arcs than S_i. This is because the paths found splitting S_i can only have $O(n^2)$ arcs not on the boundary, and each edge of S_i's boundary can only be present in one of the two paths (if we clip common prefixes following the poles). Next, note that while in state TRIPOLAR, there can be many stages, in each of which a single path is

guarded, however, all the paths are from the *same shortest path map*. Thus, when we transition from state TRIPOLAR to BIPOLAR we have one path consisting of a single arc, and two paths which may traverse the boundary and have gained at most $O(n^2)$ new arcs. Again, no edge of the boundary will be present twice, and thus boundary can only have gained $O(n^2)$ new arcs. Finally, in state ENDGAME we need not worry about growth in path complexity, as once $\Pi(b, c)$ is reduced to a single edge Lemma 24 guarantees a path crosses each edge at most once. Thus after j state transitions, the boundary of S_i has at most $O(j \cdot n^2)$ arcs, and thus any shortest path in S_i has at most $O(j \cdot n^2)$ arcs. □

Next, we note that in the weighted case an arc on an n vertex surface may have up to $O(n^3)$ intervals of optimality [52]. Note that during the n stage transitions it takes to reach state ENDGAME the boundary will at most $O(n^3)$ arcs. Thus a surface S_i may effectively have $O(n^3)$ vertices, and thus a single arc may have $O(n^9)$ intervals of optimality. However, Lemma 16 will still force a transition out of TRIPOLAR in $O(\log n)$ phases. Thus in $O(n \cdot \log n)$ phases the pursuit will reach state ENDGAME.

2.5.2 Modifications to State ENDGAME

Unlike in the unweighted case, the straight arc between two points no longer is necessarily a shortest path. Thus, we cannot simply "walk" the arc (b, c) up S_i as before.

However, the minimum weight internal edge (an edge not in the boundary) in S_i is a shortest path and can be guarded, thus we will exploit this to reduce S_i to a single face. [2]

Recall that state ENDGAME consists of a single chain of triangles (after reducing (b, c) to a single arc). Let (d_1, d_2) be the internal edge with minimum weight in S_i. Without loss of generality, suppose that d_1 is on $\Pi(a, b)$, then we deploy the fourth pursuer to guard $\Pi(d_1, c)$. The following Lemma shows that $\Pi(d_1, c)$ crosses each edge at most once.

Lemma 24. $\Pi(d_1, c)$ *crosses each edge at most once.*

Proof. Suppose that the edge (x, y) is intersected twice at points x_1 and x_2, and let (x, y) be the first such edge in the edge sequence followed by $\Pi(a, b)$ and $\Pi(a, c)$ starting from a. Then notice, there must be a second edge after (x, y) in the edge sequence that is also intersected twice. This is true as $\Pi(d_1, c)$ only turns at edges and vertices, and thus must intersect at least one edge between the first and second intersection of (x, y), and then this edge would be intersected again by $\Pi(d_1, c)$ before reaching c. Let the edge with minimum weight that is intersected twice (besides (x, y)) be (u, v), and suppose it is intersected at u_1 and u_2. See Figure 2.7(a).

Suppose that (x, y) has weight ω_1 and (u, v) has weight ω_2. If $\omega_1 \leq \omega_2$, construct a path $\Pi(d_1, c)'$ by replacing $\Pi(x_1, x_2)$ with the arc (x_1, x_2). Similarly, if $\omega_2 \leq \omega_1$,

[2] It is possible that the minimum weight internal edge is not a shortest path. However, any shorter path necessarily includes part of the guarded boundary, and the evader cannot move along such a path without being captured.

construct $\Pi(d_1, c)'$ by replacing $\Pi(u_1, u_2)$ with the arc (u_1, u_2). In both cases, $\Pi(d_1, c)'$ is no longer than $\Pi(d_1, c)$, and each edge has one less intersection. This can be repeatedly applied until no edge is intersected more than once. □

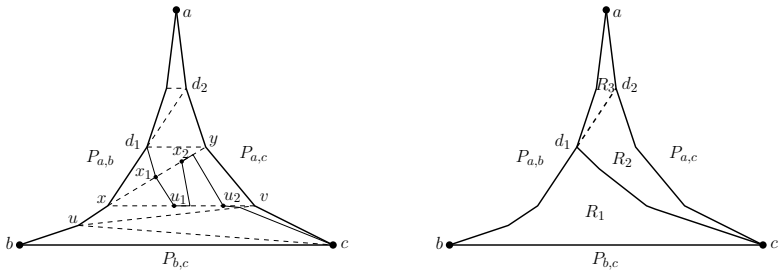


Figure 2.7: In (a) an illustration for the proof of Lemma 24, and in (b) an illustration for the proof of Lemma 25.

Thus $\Pi(d_1, c)$ can be guarded without introducing any additional internal edges. The following Lemma shows how to remove all internal edges from S_i by guarding such a path.

Lemma 25. *Suppose S_i is a weighted subsurface in state* ENDGAME*, and $\Pi(b, c)$ consist of a single arc, then S_i can be reduced to a single face.*

Proof. We describe a procedure to remove an internal edge in at most two phases and maintain the property that one of the three bounding paths is an arc. This procedure can then be applied recursively to remove the remaining internal edges. Without loss of generality, suppose (d_1, d_2) is the minimum weight internal edge, and d_1 is on $\Pi(a, b)$,

see Figure 2.7(b). Deploy the fourth pursuer to guard $\Pi(d_1, c)$. If the evader is in R_1, then we are done, as (d_1, d_2) is no longer an interior edge, (b, c) is a single arc, and the pursuer guarding $\Pi(a, c)$ is no longer necessary.

Otherwise, (b, c) no longer needs to be guarded, and the pursuer is reused to guard (d_1, d_2), which is a shortest path. Then, regardless of whether the evader is in R_2 or R_3, (d_1, d_2) is no longer an internal edge and has become part of the boundary, and one of the bounding paths no longer needs to be guarded. Thus this process can be applied recursively to R_2 or R_3 to remove further internal edges. Further, in both stages the Pursuit Invariant is trivially maintained.

By Lemma 23 and the fact there are $O(n)$ state transitions, each boundary path can cross at most $O(n^3)$ edges. Thus, after $O(n^3)$ phases all internal edges are removed and the evader is confined to a single face. □

2.5.3 Weighted Time to Capture

With the preceding sections we have covered all three states, and thus conclude that our algorithm will result in the capture of the evader. However, we must still address the time to capture. As in the unweighted case, the diameter of the environment can grow as we confine the evader to smaller a smaller subsurfaces of S, but unlike before, it can grow larger. This is because the original diameter may be small due to passing through regions with small weights, which are subsequently removed via path guarding. Let the

minimum (non-zero) weight assigned to any face or edge be ω_{min} and similarly let ω_{max} be the maximum weight. The following Lemma bounds the path growth in terms of the input parameters.

Lemma 26. *In a given phase i, $\mathrm{diam}S_i$ is at most $O(\frac{\omega_{max}}{\omega_{min}} \cdot n^3 \cdot \mathrm{diam}(S))$*

Proof. Consider an arbitrary shortest path Π in S_i. By Lemma 23 we know that any shortest path in S_i consists of at most $O(n^3)$ arcs. Let x, y be an arc of Π with Euclidean length ℓ. Then, the shortest x, y path in S has weighted length at most $\omega_{min} \cdot \ell \leq \mathrm{diam}(S)$. In S_i, the arc x, y has length at most $\omega_{max} \cdot \ell$, and thus the arc x, y has weighted length at most $\frac{\omega_{max}}{\omega_{min}} \cdot \mathrm{diam}(S)$. Thus, as there are at most $O(n^3)$ arcs, the maximum length of any shortest path in S_i is at most $O(\frac{\omega_{max}}{\omega_{min}} \cdot n^3 \cdot \mathrm{diam}(S))$. □

Theorem 8. *Given a polyhedron S with n vertices, and weighted regions with min weight ω_{min} and max weight ω_{max}, 4 pursuers can capture the evader in $O(\frac{\omega_{max}}{\omega_{min}} \cdot n^6 \cdot \mathrm{diam}(S) + \log((\frac{\omega_{max}}{\omega_{min}}) \cdot \mathrm{diam}(S)) \cdot \frac{\omega_{max}}{\omega_{min}} \cdot \mathrm{diam}(S))$ moves.*

Proof. First notice, that with the exception of ENDGAME, the only increase in the time bound from the non-weighted problem is the increased worst case path length. Thus, the worst case number of moves in states BIPOLAR and TRIPOLAR is $O(\frac{\omega_{max}}{\omega_{min}} \cdot n^4 \log(n) \cdot \mathrm{diam}(S))$. Then, in state ENDGAME there may be up to $O(n^3)$ phases in which a path is guarded taking a worst case $O(\frac{\omega_{max}}{\omega_{min}} \cdot n^6 \cdot \mathrm{diam}(S))$ moves. Finally, when the evader is captured on the final face, it can have diameter at most $O(\frac{\omega_{max}}{\omega_{min}} \cdot \mathrm{diam}(S))$, and thus

by Lemma 19 the evader can be captured in at most $O(\log((\frac{\omega_{max}}{\omega_{min}}) \cdot \mathrm{diam}(S)) \cdot \frac{\omega_{max}}{\omega_{min}} \cdot \mathrm{diam}(S))$ moves. Therefore, the worst case number of moves to capture the evader is $O(\frac{\omega_{max}}{\omega_{min}} \cdot n^6 \cdot \mathrm{diam}(S) + \log((\frac{\omega_{max}}{\omega_{min}}) \cdot \mathrm{diam}(S)) \cdot \frac{\omega_{max}}{\omega_{min}} \cdot \mathrm{diam}(S))$. □

Chapter 3

Visibility Based Pursuit Evasion

3.1 Introduction

In visibility-based pursuit-evasion the pursuers are equipped with cameras, able to maintain omni-directional line-of-sight visibility, and only know the location of the evader when it is visible, that is, in direct line of sight. Thus, not only must the pursuers plan and coordinate their moves until some pursuer can reach the same location as the evader, they must also contend with the fact that the location of the evader is often unknown. The problem is motivated by applications in robotics, and has drawn a significant interest since it was introduced by Suzuki and Yamashita [70], although much of the prior work has focused on the simpler problem of *evader detection*, where the pursuers win as soon as the evader is "seen" by some pursuer [22, 25, 29, 58, 72].

We begin by making only the minimally necessary assumption that all players (pursuers and evader) have equal maximum speed, *which is normalized to one* by

Parts of this chapter appeared in the following publications: [38, 39]

appropriate scaling of the environment. On its turn, each player can move to any position within distance one of its current location, where the distance is measured using the shortest path (geodesic) distance avoiding the obstacles in the environment. Our first result gives a tight bound of $\Theta(n^{1/2})$ for the number of pursuers needed to capture the evader when the environment is a simply-connected (hole-free) polygon of n vertices. Generalizing this result, we show that at least $\Omega(n^{2/3})$ pursuers are needed for capture in polygons with holes. Complementing this lower bound, we prove an upper bound of $O(n^{1/2}h^{1/4})$, for $h \leq n^{2/3}$, and $O(n^{1/3}h^{1/2})$ otherwise, where h is the number of holes in the polygon. More simply, the upper bound is $O(n^{5/6})$.

We then show with additional assumptions these bounds can be drastically improved. Namely, if the players' movement speed is small compared to the "feature size" of the environment, we give a deterministic algorithm with a worst case upper bound of $O(\log n)$ pursuers for simply-connected n-gons and $O(\sqrt{h} + \log n)$ for multiply-connected polygons with h holes. Further, if the pursuers are allowed to randomize their strategy, regardless of the players' movement speed, we show that $O(1)$ pursuers can capture the evader in a simply connected n-gon and $O(\sqrt{h})$ when there are h holes with high probability.

These results may be considered a theoretical bridge between two incomparable results. On one hand, Guibas et al. [25] prove that successful pursuit-evasion requires $O(\log n)$ pursuers in a simple polygon, and $O(h^{1/2} + \log n)$ pursuers in a polygon with

h holes. Their strategy works even against an arbitrarily fast evader, but it only ensures *a line of sight detection* of the evader, not physical capture. On the other hand, we can match these bounds for capture, but require that the maximum step size (speed) of the evader be less than the minimum feature of the environment. With no such restriction, then many more pursuers are needed for the capture *even if pursuers also move as fast as the evader.* However, when the pursuers are allowed to randomize their movements the minimal condition of equal speeds is enough for a randomized capture algorithm to to match the bounds for randomized localization of Isler et al. [29].

Additionally, recall that in Chapter 1 we showed that *if the location of the evader is always known to the pursuers*, e.g., using an ubiquitous camera network, then 3 pursuers are enough to win the game. In a sense, this suggests that "localization" of the evader is the more difficult part of the pursuit evasion, and the evader's power comes from its ability to "disappear" from the collective sights of all the pursuers.

3.2 Capture in Simple Polygons

In this section, we establish the tight bound of $\Theta(n^{1/2})$ for the number of pursuers needed to capture the evader. We use the standard model for geometric pursuit evasion given in Section 0.2, where the environment is an n vertex polygon P. The players' sensing model is *visibility-based*: two players see each other only when they are in line

of sight. We assume a global communication model so that if the evader is visible to one pursuer, then all pursuers know the location of the evader.

Our first theorem shows that, in the worst-case, at least $\Omega(n^{1/2})$ pursuers are needed to capture the evader in n-vertex polygons.

3.2.1 The Lower Bound Construction

Theorem 9. *In an n-vertex simple polygon, at least $\Omega(n^{1/2})$ pursuers are needed in the worst-case to capture an equally fast evader.*

Proof. We give a construction of a polygon and the evader's strategy that forces $\Omega(n^{1/2})$ pursuers for a win. The polygon consists of a long corridor acting as a "base," of length B, with $n-1$ equally spaced "notches," and n long "channel corridors," interleaved with the notches. See Figure 3.2.1. Each channel corridor also has a notch at one end, and the length of each such corridor is C, chosen so that $C > B$. The players' maximum speed is set to $2C + B$, ensuring that players can move between the notches of two arbitrary channels in a single move, but pursuers cannot search more than two channels in one move: searching three or more channels requires speed of at least $4C$, which is strictly larger than $2C + B$. (The channel lengths take into account the notches, and we can scale the polygon as necessary to normalize the speed to unit speed.)

We now argue that capturing the evader in this polygon requires at least $\frac{1}{2}n^{1/2}$ pursuers. Given any placement of fewer than $\frac{1}{2}n^{1/2}$ pursuers, there exists a consecutive

block of $n^{1/2}$ channel corridors and the section of base between them that does not have any pursuers. If the evader moves to a channel in this block, pursuers cannot determine the identity of the channel because the notches in the base block their visibility. Since each pursuer can search at most two corridors on its move, collectively these fewer than $\frac{1}{2}n^{1/2}$ pursuers cannot search all the $n^{1/2}$ corridors, leaving at least one safe corridor for the evader to hide. The evader can, therefore, continue to elude the pursuers indefinitely by repeatedly moving into such a "safe" corridor on its turn. This completes the proof. □

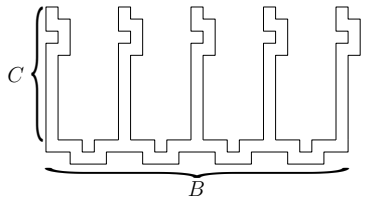


Figure 3.1: Construction for the proof of Theorem 9.

The rest of this section presents a matching upper bound, by giving an algorithm that guarantees a win for $O(n^{1/2})$ pursuers in all simply-connected polygons of n vertices. The next few subsections describe the geometric preliminaries and constructions that form the basis for our proof.

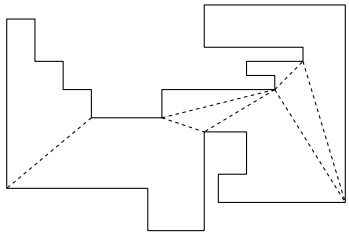


Figure 3.2: A block partition of a polygon ($k = 8$).

3.2.2 A k-block Partition

We begin by describing a partition of the polygon P into subpolygons, called k-blocks, that plays an important role in our pursuit strategy. Each *k-block* is just a connected subpolygon of at most k vertices, and the partition satisfies the following properties:

1. the number of k-blocks in the partition is $O(n/k)$,
2. the edges common to adjacent k-blocks are polygon diagonals, and
3. the adjacency graph of the partition, called the *block graph*, is a binary tree.

Specifically, our k-block partition is an "unrefinement" of a triangulation of P: a triangulation partitions P into $(n-2)$ triangles, using $(n-3)$ diagonals; our partition retains $O(n/k)$ carefully chosen diagonals so that the resulting subdivision has the k-block partition properties. See Figure 3.2.2 for an example. In fact, the degree bound of the adjacency graph is the only non-trivial property—a naive partition can easily lead

to unbounded fanout. We call the diagonals separating the k-blocks *cut edges*. The following lemma shows constructively that a k-block partition always exists.

Lemma 27. *Every simply-connected polygon on n vertices admits a k-block partition for any $3 \leq k \leq n$.*

Proof. Let T be a triangulation of P. The dual graph of the triangulation is a binary tree; the nodes of the graph are the triangles and its edges connect adjacent triangles. We describe a recursive algorithm to identify the *cut* edges that define the desired k-block partition. Since a subtree of size k corresponds to a subpolygon with $k+2$ vertices, we choose cut edges to break the tree into components of at most $k-2$ nodes, which form k-blocks.

We inductively assume that the tree is rooted at a degree one node r, which can initially be an arbitrary leaf node. For any node u in the tree, let $s(u)$ denote the size of the subtree rooted at u, including the node u itself. If $s(r) \leq k-2$, then we are done. Otherwise, we choose any node u such that $s(r) - s(u) \leq k-2$ but $s(r) - s(x) > k-2$ for any child x of u, and cut the edge between u and its parent. Next, if u has only one child x, then we simply recurse on the subtree T_u, but if u has both its children x and y, then we also cut the edges $(u,x), (u,y)$ and then recurse on the subtrees $T_u \setminus (u,y)$ and $T_u \setminus (u,x)$. (During the recursive call, the size fields of the root nodes are recomputed for the new subtrees.)

For the correctness of the algorithm, we first note that the partition clearly creates valid k-blocks. Second, by construction, each block is bounded by at most three cut edges, thus ensuring that the block graph is a binary tree. Finally, to show that the total number of cuts made is $O(n/k)$, observe that in the block graph, any node of degree 1 or 2 is adjacent to a block so that the union of the neighboring blocks contains more than k vertices—otherwise, our algorithm will not have made the cut between the blocks. Since the number of degree 3 nodes is at most the number of leaves, the graph has size $O(n/k)$, and the proof is finished. □

After an initial search to localize the evader to a block, a placement of one pursuer per cut edge is sufficient to maintain the identity of the current block containing the evader. In particular, let $B(e)$ denote the current block containing the evader. Then the following lemma is straightforward.

Lemma 28. *Suppose a pursuer is placed on each cut edge of a k-block partition of the polygon. Then, after any move of the evader that crosses a block boundary, the pursuers know the identity of $B(e)$, the block containing the evader.*

The initial search can be performed using the following result of Guibas et al. [25]. We use the notation $\mathrm{diam}(P)$ for the *diameter* of the polygon under the shortest path metric.

Lemma 29 ([25]). *Given a simply connected n vertex polygon P, $O(\log n)$ pursuers can locate the evader in $O(n \cdot \mathrm{diam}(P))$ moves.*

3.2.3 Critical Moves

We begin with a simple sufficient condition to trigger the end-game: an immediate capture of the evader. Specifically, we say that an evader's move from its current position e to the new position e' is *critical with respect to a pursuer* p if there exists a point e_c on the evader's path that is *both* visible to p and closer to p's current position than to the evader's start position. Mathematically, a move from e to e' is critical for pursuer p at point e_c if (1) e_c lies on the path from e to e', (2) e_c is visible to p, and (3) $d(p, e_c) \leq d(e, e_c)$. If the evader's move is critical with respect to k pursuers, we call it a k-*critical* move. Figure 3.3(a) shows a k-critical event for $k = 4$, with f serving as the critical point. The following lemma shows the important connection between a k-block partition and a k-critical move.

Lemma 30. *Suppose each cut edge of the k-block partition is guarded by a pursuer, and a group of pursuers $p_1, p_2, \ldots, p_k$ are so positioned that an evader's move becomes k-critical with respect to these k pursuers. Then, one of the pursuers p_i can capture the evader on its next move.*

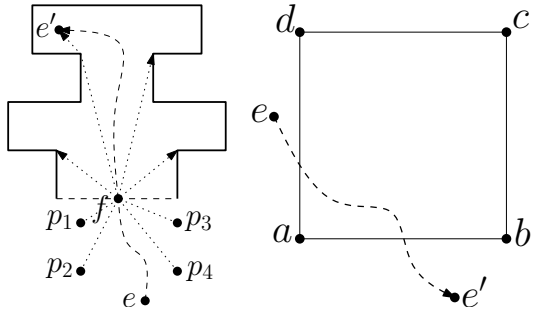
Proof. By definition of a k-critical move, for each pursuer p_i, there is a critical point, say, e_{c_i}, on the evader's path, closer to p_i than to the evader's start position. That is,

$d(p_i, e_{c_i}) \leq d(e, e_{c_i})$, for all i. (The critical points for different pursuers need not be the same.) By triangle inequality, we also have that

$$\begin{aligned} d(p_i, e') &\leq d(p_i, e_{c_i}) + d(e_{c_i}, e') \leq d(e, e_{c_i}) + d(e_{c_i}, e') \\ &\leq d(e, e') \leq 1 \end{aligned}$$

where the second inequality follows from the definition of a critical move and the last inequality from the unit maximum speed assumption. Thus, if the terminal position e' of the evader is *visible* from any of the critical points e_{c_i}, then the corresponding pursuer can capture the evader by first moving to its (visible) critical point and then to e'. Therefore, assume that none of the critical points are visible from the evader's position e'. In this case, we first move all the pursuers to a carefully chosen *waypoint* f, defined as follows. The waypoint f is the *last location at which the evader is seen by any pursuer* during its move from e to e'. After moving to the waypoint, each pursuer p_i still has $1 - d(p_i, f) \geq d(f, e')$ amount of remainder distance in its current move because $d(p_i, f) \leq d(e, f)$. If the evader's terminal position e' is visible from f, the evader can be captured by any of the pursuers.

Thus, assume that the evader's location e' is invisible from the waypoint also. Suppose $B(e')$ is the block containing the evader, which the pursuer know by Lemma 28, and by construction $B(e')$ has at most k vertices. We first observe that the waypoint f must lie in $B(e')$: the evader's entry into $B(e')$ was seen by a cut edge pursuer. Consider



(a) (b)

Figure 3.3: In (a) illustration of proof of Lemma 30. In (b) an example crossing sequence.

the shortest path in $B(e')$ from f to the evader's final position e'—necessarily, this path cannot be longer than the portion of the evader's path between f and e', and because the two end positions are mutually invisible, the shortest path must contain at least one polygon vertex. Without loss of generality, let z be the *last* (closest to e') vertex on the path from f to e'. The vertex z is necessarily in $B(e')$, and is visible from e'. Since there are at most k choices for the vertex z, each of our k pursuers follows a shortest path from the waypoint f to one of these vertices (see Figure 3.3(a) for illustration), and the one reaching z can successfully capture the evader. This completes the proof. □

3.2.4 Forcing a Critical Move

The main problem now is to devise a pursuer strategy that forces a k-critical move in a finite number of steps. Unfortunately, the cut edges can be arbitrarily longer than the

normalized speed of the players, and thus even if we position k pursuers on an edge of length $L \gg k$, the evader can cross the edge without triggering a critical event. We, therefore, resort to a more complex structure and search strategy, which is motivated by the following simple observation: *If the evader crosses a square-shaped region of the environment with pursuers at its corners, then it is a critical move.*

In order to make this idea more precise, we first define a *crossing sequence*. Let R be a square contained entirely within the polygon P. A *crossing sequence* for R is a sequence of moves in which the evader enters and exits the square through *distinct boundary edges*. Figure 3.3(b) shows an example. (We note that an evader path is *not* a crossing sequence if it enters and exits the square through the same edge.)

Lemma 31. *Consider a square R fully contained in the polygon P, and let p_a and p_b, respectively, be two pursuers located at the corners a and b of R. Then, any crossing sequence in which the evader exits R through the edge (a, b) forces a critical event with respect to p_a or p_b.*

Proof. First, consider the simpler case when the crossing sequence consists of a single evader move: that is, the evader, originally outside the square, crosses it in a single move, exiting through the edge (a, b), say, at a point x. In this case, elementary geometry shows that $\min\{d(p_a, x), d(p_b, x)\} \leq d(e, x)$, ensuring a critical event.

The case when the crossing sequence consists of multiple moves requires more tedious, but still elementary, argument. See Figure 3.4 for an illustration. We first

introduce the idea of a projection. Given the evader's position e inside the square, its *projection on an edge* (a, b), denoted $\pi_e(a, b)$, is the point on (a, b) that is closest to e. (In other words, the projection is the foot of the perpendicular from e to (a, b).) The key observation is that a pursuer located at the projection $\pi_e(a, b)$ is *closer* to any point of (a, b) than the evader, and so any evader move crossing the edge (a, b) is critical for that pursuer. If the evader enters R through an edge adjacent to (a, b), namely, (a, d) or (b, c), then p_a or p_b can easily maintain their position on the evader's projection on (a, b): because the "horizontal projection" of the evader's position can change by at most one in a move, the pursuers p_a or p_b can reposition themselves at the evader's projection on the edge (a, b).

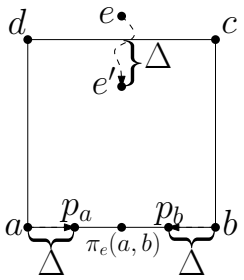


Figure 3.4: Illustrating the proof of Lemma 31

Thus, it remains only to consider the evader's entrance through the edge (c, d), which is the opposite side of the square from (a, b). In this case, clearly, both p_a and p_b can be arbitrarily far from the projection $\pi_e(a, b)$—the side length of R can be much larger than 1, the players' speed, and the evader may enter in the middle of the edge (c, d).

However, in this case, the evader is also far away from the edge (a, b), and we ensure that a crossing sequence has a critical event for at least one of the pursuers. In order to track the pursuers' progress, let us introduce $\Delta = d(e, \pi_e(c, d))$, the distance between the evader's current position and its projection on the entrance edge (c, d).

The pursuer p_a now aims for the point that is $\min\{\Delta, d(a, \pi_e(a, b))\}$ away from a along the edge (a, b), while p_b aims for an analogous point measured from b. (See Figure 3.4.) Each move of the evader can increase Δ by at most one, and so p_a and p_b can both reach their targets each turn. Once p_a or p_b reaches the projection, the claim of the lemma is clearly satisfied. Otherwise, both the pursuers are as close to the projection as is the evader, and together they guarantee that any move of the evader that causes it to exit through (a, b) is critical for both the pursuers. In particular, for any position of p_a in the interval between a and $\pi_e(a, b)$, the pursuer p_a guarantees a critical event, while p_b guarantees a critical event for any position in the interval from $\pi_e(a, b)$ to b. This completes the proof. □

An easy consequence of Lemma 31 is that by placing two pursuers at each vertex of R, for a total of eight, we can guarantee that any crossing sequence through the square R is critical move. (We could, in fact, reduce the number of pursuers to just four, one per corner but for convenience and a future simplicity, we choose to keep all eight pursuers.) One subtle point, however, is that the evader can exit through the same side it entered, therefore, *without completing a crossing sequence*, but in the process force pursuers

to move off their desired corner positions! Our next lemma shows that the pursuers can immediately recover their initial positions, following any such "fake" move by the evader.

Lemma 32. *If the evader enters and exits the square R through the same side, without completing a crossing sequence, then all eight corner pursuers can recover their initial positions on the next move after the evader's exit.*

Proof. Without loss of generality, assume that the evader enters and exits R through the side (c, d), and that the exit move occurs at time t. By the projection invariant maintained by the pursuers, each of them is within distance 1 of its initial corner at time t, and therefore can recover its initial state on its next move. It is worth pointing out that if the evader exits through (c, d) but immediately reenters R through a different edge *in the same move*, then a crossing sequence is completed, immediately leading to capture—this follows from the projection invariant maintained by the pursuers. □

The idea of crossing sequence through a square extends easily to squares "truncated" by intersection with the polygonal environment. The intersection $R \cap P$ between the square R and the polygon P consists of possibly many (connected) cells. Consider one such cell F that does not contain any *vertex of* P, and call it *empty*. It is easy to see that F has a constant number of boundary edges—at worst, each of its corners can be lopped off by a polygon edge, resulting in an 8-sided cell. Thus, there are at most four

sides of F inherited from R, each possibly truncated by a polygon edge, and at most four sides defined by polygon edges. Since the polygon edges are impenetrable by either the evader or the pursuers, it is easy to see that the critical move claim of the preceding lemma holds also for such a truncated cell F of the square R. In particular, we have the following easy corollary of Lemma 31.

Corollary 1. *Given a square $R = (a, b, c, d)$, let F be an* empty *cell of the common intersection $R \cap P$, let (a', b') be a non-polygon edge of F, and let $p_{a'}, p_{b'}$ be two pursuers placed at a' and b'. Then, any crossing sequence by the evader exiting F through the edge (a', b') is a critical move for one of these pursuers.*

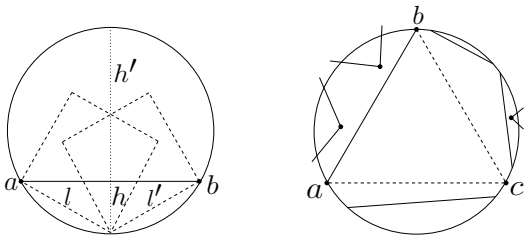
3.2.5 Edge Covers and the Constrained Delaunay Triangulation

We mentioned earlier that no bounded number of pursuers on a cut edge can prevent the evader from crossing it. Instead we build a geometric "cover" around each cut edge in such a way that the evader cannot cross the cover without being captured. We begin with the following technical lemma that forms the basis of such a cover.

Lemma 33. *Consider a circle C and a chord (a, b) in it. Then, there always exist two squares R_1, R_2 contained in C so that (a, b) lies in the union $R_1 \cup R_2$.*

Proof. Let r be the radius of the circle C. Let h and h' be the lengths of two segments into which (a, b) divides the circle's diametric chord that is perpendicular to (a, b). Without loss of generality, assume that $h \leq h'$, which implies that $h \leq r$. It is now easy

to see that the two squares, each with side length $\ell = ((|ab|/2)^2 + |h|^2)^{1/2} \leq r\sqrt{2}$, satisfy the conditions of our lemma, as illustrated in Figure 3.5(a). Since every circle of radius r admits a contained square of side length $r\sqrt{2}$, these squares lie within C. □



(a) (b)

Figure 3.5: (a) illustrates the proof of Lemma 33 and (b) shows an example triangle from a CDT.

Our idea is to cover each cut edge with the union of two squares as in Lemma 33, but use a particular kind of underlying triangulation to achieve the necessary empty-cell condition (cf. Corollary 1). Specifically, we use the *Constrained Delaunay Triangulation* of P [15, 68] as the basis for our partition. The constrained Delaunay triangulation has the following properties: (1) each edge of the polygon appears as an edge of the triangulation, and (2) each triangle's circumcircle encloses no vertex that is visible from the interior of the triangle. Figure 3.5(b) shows an example.

Consider a cut edge (a, b) of the constrained Delaunay triangulation, which by definition has a circumcircle C empty of any visible vertices of P. By Lemma 33, we can find two squares R_1, R_2 that "cover" (a, b) and lie entirely within C. These

squares may intersect the boundary of the polygon P but, by definition of CDT, the cells containing the edge (a, b) are empty of any visible vertices. We define the $cover(a, b)$ as the union of these two "truncated squares." These edge covers are utilized in the following way in our pursuit strategy.

Let B be a k-block partition, and B_i a k-block of B. Define $B_i^- = B_i \backslash \{\cup_j cover(a_j, b_j)\}$ as the *contracted block* corresponding to B_i, where (a_j, b_j) are the (at most three) cut edges bounding B_i. Similarly, we define $B_i^+ = B_i \cup_j \{cover(a_j, b_j)\}$ as the *extended block* corresponding to B_i. Since each edge cover has a constant number of vertices, all contracted or extended blocks clearly have size $O(k)$. We call two blocks (contracted or extended) neighbors if their original blocks share a common cut edge. We have the following lemma.

Lemma 34. *Any evader move between two neighboring contracted blocks is a crossing sequence.*

Proof. Let B_1^- and B_2^- be two contracted blocks neighboring the cut edge (a, b). Any move by the evader from B_1^- to B_2^-, or vice versa, must cross at least one of the truncated squares of $cover(a, b)$. □

We say that a cut edge is k-*covered* if we replace each pursuer in $cover(a, b)$ by k co-located pursuers. Clearly, any evader move that crosses a k-covered edge is k-critical,

and results in capture on pursuers' next move (cf. Lemma 30). At a high level our capture algorithm has the following form.

Algorithm HoleFreeCapture

1. Construct a k-block partition of P, using the Constrained Delaunay Triangulation. Place one pursuer on each cut edge to track $B(e)$, the current k-block containing the evader.

2. Perform a sweep of the block graph until the evader is trapped in an extended block $B^+(e)$ whose adjacent cut edges are all k-covered. With the pursuers in this position, any move by the evader exiting $B^+(e)$ is a k-critical move, leading to capture.

3. With the evader confined to an extended k-block, we use an additional set of $O(k)$ pursuers to find and capture the evader in $B^+(e)$.

By choosing $k = n^{1/2}$, this leads to a search and capture strategy using $O(n^{1/2})$ pursuers: there are $O(n/k) = O(n^{1/2})$ cut edges, each requiring one pursuer, and at most 3 groups of $O(n^{1/2})$ pursuers needed to sweep the block graph. It only remains to describe the details of Steps 2–3, which is the focus of the next two lemmas.

Lemma 35. *With $O(k)$ pursuers, we can confine the evader to an extended block.*

Proof. We use the fact that the block graph is a binary tree, whose nodes correspond to the k-blocks and whose edges correspond to cut edges, and that pursuers always know $B(e)$, the block containing the evader, by Lemma 28. Inductively, assume that $B(e)$ belongs to the subtree rooted at a node u, and the cut edge between u and its parent is k-covered; if u is root, then the parent is null. Let x and y be the two children of u, one of them may be null. For notational convenience, we use U, X, Y to denote the blocks corresponding to u, x and y, respectively.

We begin by repositioning two groups of $O(k)$ pursuers each to form k-covers of the cut edges (u, x) and (u, y), and then use a constant number of pursuers to search the constant-size subpolygons $cover(u, x)$ and $cover(u, y)$. By the end-game algorithm (cf. Lemma 36), *if the evader is in these subpolygons*, it is either captured or forced to exit it. The key observation is that *once the evader has exited* $cover(u, x)$ and $cover(u, y)$, the pursuers in the cover prevent the evader from crossing between the neighboring contracted blocks. Thus, after leaving the covers if the evader remains in U^+, then it is confined: moving to a neighboring contracted block forces a k-critical event (cf. Lemma 34). Because all three neighboring cut edges of U^+ are k-covered, the claim follows in this case. Otherwise, the evader must have moved outside the extended block U^+. Without loss of generality, assume that the evader is either in the extended block X^+ or in a descendant node of x. In either case, the evader cannot enter U^- because of the k-covering of (u, x). We can therefore free up the k-covering pursuers from edges

(u, v) and (u, y), and recursively search the subtree rooted at x. The search terminates within $O(n/k)$ such steps. □

3.2.6 The End Game

The last step of our algorithm deals with capturing the evader when it is confined to an extended k-block, using $O(k)$ pursuers. At a superficial glance, it may appear that this can be done by combining the localization algorithm of Guibas et al. [25] with a modified lion-and-man algorithm of Isler et al. [29]. Unfortunately, this strategy fails due a technical subtlety: the lion-and-man algorithm of [29] relies on a *small step size* assumption, which precludes the arbitrary speed with which the evader is allowed to move in our problem. However, since we have $O(k)$ pursuers available to us, we can design a simple direct algorithm for this end game.

Lemma 36. *Let P be a k-vertex simple polygon. Then, in $O(\mathrm{diam}(P)^2)$ steps $O(k)$ pursuers can capture an equally fast evader.*

Proof. The algorithm operates in two phases: a *preparation* phase followed by an *attack* phase. The preparation phase begins by triangulating P. We then assign one pursuer to each edge of the triangulation, including the polygon boundary edges, which positions itself at the projection of the evader. We define the projection $\pi_e(a, b)$ of the evader for an edge (a, b) as the closest point on the edge to the evader *measured by direct Euclidean distance ignoring the polygonal boundary*–this is either the foot of the perpendicular

from the evader's position or an endpoint a or b. Since the evader cannot leave P, and moves at most distance one in each move, each pursuer can arrive at its projection in $O(\mathrm{diam}(P))$ moves, which completes the preparation phase.

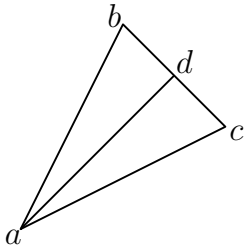


Figure 3.6: The end game: shrinking the triangle during the attack phase.

With each pursuer at the projection of its designated edge, the *attack* phase starts. Any move by the evader crossing a triangle is a critical move for some pursuer. Let p be a pursuer for whom this is a critical move. Since the entire polygon is collectively visible to the pursuers, the end position of the evader after the move is also known to p, and therefore p can capture the evader on its next move.

Thus, to avoid capture, the evader must remain confined to a single triangle, say, $\Delta(a, b, c)$. In this case, the three pursuers assigned to the triangle progressively "shrink" the area within which the evader lies, leading to eventual capture, as follows. Imagine sliding one of the edges of the triangle, say, (a, b) toward c by distance one, creating a "shrunken" triangle $\Delta(a', b', c)$. Position a new pursuer p' on the edge (a', b') at the projection $\pi_e(a', b')$ in $O(\mathrm{diam}(P))$ moves. Now, if the evader lies in the strip between (a, b) and (a', b'), then a single pursuer can eventually capture the evader by sweeping

this strip—the width of the strip is one, and the evader cannot cross the strip boundaries because they contain pursuers on projection points. If, however, the evader is in the triangle $\Delta(a', b', c)$, then we have successfully reduced the height of the triangle by one, which must lead to capture in $O(\mathrm{diam}(P)^2)$ steps. This completes the proof. □

This completes our discussion of the algorithm HoleFreeCapture. As discussed earlier, by choosing $k = n^{1/2}$, we achieve the following theorem, which is the main result of this section.

Theorem 10. *$O(n^{1/2})$ pursuers are always sufficient to capture an equally fast evader in any simple polygon of n vertices in $O(n \cdot \mathrm{diam}(P) + \sqrt{n} \cdot \mathrm{diam}(P)^2)$ moves.*

Proof. As the capture will clearly occur, we need only analyze the worst case number of moves. By invoking Lemma 29 the pursuers can determine the initial k-block containing the evader in $O(n \cdot \mathrm{diam}(P))$ moves. Additionally, the pursuers will cover at most $O(\sqrt{n})$ cut edges taking at most $O(\sqrt{n} \cdot \mathrm{diam}(P)^2)$ moves due to invocations of the end-game algorithm (Lemma 36) to force the evader out of k-covers. Finally, when the evader is captured in a k-block it takes $O(\mathrm{diam}(P)^2)$ moves for a total of $O(n \cdot \mathrm{diam}(P) + \sqrt{n} \cdot \mathrm{diam}(P)^2)$ moves. □

3.3 Capture in Polygons With Holes

In this section, we extend our results to polygonal environments with holes, also called multiply-connected polygons. We assume that the polygon contains h disjoint polygonal holes, and the total number of vertices including the holes is n.

3.3.1 An $\Omega(n^{2/3})$ Lower Bound Construction

We begin with a construction showing that in the worst-case at least $\Omega(n^{2/3})$ pursuers are needed to capture the evader. The proof follows the basic outline of Theorem 9, but requires a more complicated construction.

Theorem 11. *In the worst-case, at least $\Omega(n^{2/3})$ pursuers are needed to capture an equally fast evader in a multiply-connected polygon with n vertices.*

Proof. Our construction is based on a rectangular grid of r rows and c columns (see Figure 3.7). We convert this into a polygon by making each edge of the grid into a narrow corridor, so that the resulting polygon has $r \cdot c$ (rectangular) holes. Place a small notch in the middle of each corridor to block visibility across the notch. Next, at the top boundary of the grid, place the "comb" construction of Theorem 9, uniformly spaced so that there are n/c channel corridors in each of the c columns of the grid. We associate each group of n/c channel corridors with the grid column immediately preceding it. The height C of the comb corridors is chosen such that $C > WH$, where W and H,

respectively, are the width and the height of the grid (taking into account the notch detours). Finally, the movement speed of the players is set to $2C + WH$, which is strictly smaller than $4C$. This speed allows the evader to move between any two channel corridors in a single move but no pursuer can search more than two such corridors.

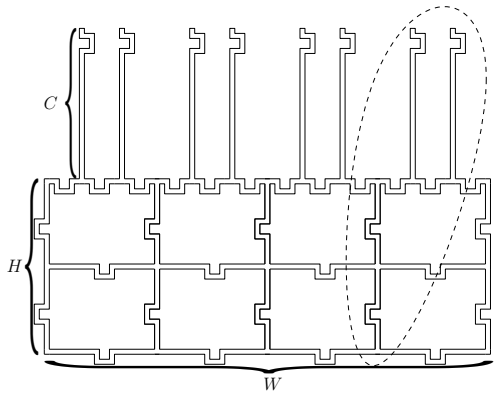


Figure 3.7: The lower bound construction for capture in polygon with holes ($r = 2$ and $c = 4$). An extended column is shown with an ellipse around it.

Define an *extended column* as the subpolygon consisting of the chain of r notched corridors associated with a grid column together with its n/c channel columns. See Figure 3.7. Given any placement of pursuers in the polygon, we call an extended column *uncovered* if no point of the extended column is visible to any pursuer. We claim that given any placement of $c/4$ pursuers in the polygon, (1) there are at least $3c/4$ uncovered extended columns, and (2) if $r = \sqrt{c}$, there is a group of $\sqrt{c}/2$ uncovered extended columns such that the evader can move between any two of them undetected.

The first claim follows from the fact that corridor notches limit a pursuer's visibility to at most one extended column. Since there are $c/4$ pursuers and c extended columns, at least $3c/4$ are uncovered. We prove the second claim by contradiction: assume the claim is false, and partition the uncovered extended columns into equivalence classes (groups) $g_1, g_2, \ldots, g_\ell$ such that the evader can move between two columns of the same group undetected but not between two columns of different groups. Because we have $3c/4$ uncovered extended columns, and by assumption each group g_i has fewer than $\sqrt{c}/2$ columns, the number of equivalence classes is $\ell > (3c/4)/(\sqrt{c}/2) = 3\sqrt{c}/2$. We can order these groups in their natural left-to-right order: all columns of one group must precede columns of the next group. We now claim that there must be at least one pursuer in every row between two consecutive groups: otherwise the evader can sneak between columns of two different groups, violating the equivalence class partition. Because the number of rows is $r = \sqrt{c}$, this implies there are at least $3\sqrt{c}/2 \cdot \sqrt{c} = 3c/2$ pursuers, contradicting our initial assumption of at most $c/4$ pursuers. Thus, claims (1) and (2) are both true.

Thus, there is a group of $\sqrt{c}/2$ uncovered extended columns for any placement of $c/4$ pursuers. The evader's strategy is to always move into one of these columns. Each of these columns contains n/c channel corridors for a total of $\frac{n}{2\sqrt{c}}$ channel corridors. If we choose $r = n^{1/3}$ and $c = n^{2/3}$, then $r = \sqrt{c}$, and we have $\frac{1}{2}n^{2/3}$ channel corridors for the evader to hide on its turn. Thus, if there are fewer than $c/4 = \frac{1}{4}n^{2/3}$ pursuers, the

evader can indefinitely avoid capture by repeatedly moving into ones of the uncovered channel corridors that is not searched by pursuers on their turn. The entire polygonal environment, with holes, has $O(n)$ vertices, and this completes the proof that $\Omega(n^{2/3})$ pursuers are required in the worst-case. □

In the rest of the section, we present an upper bound for the number of pursuers needed to capture the evader in a polygon with holes.

3.3.2 A k-block Partition of Polygons with Holes

We first extend the earlier notion of a k-block partition to polygons with holes. Our new partition has the structure of a planar graph, instead of a tree, and consists of two types of regions: triangles and k-block subpolygons (possibly with holes). The key property is that no two k-blocks are adjacent—they are adjacent only to triangles of the partition. More specifically, our partition satisfies the following properties:

1. the number of triangles (and blocks) in the partition is $O(n/k)$ if $h \leq n/k$, and $O(\sqrt{nh/k})$ otherwise,

2. the adjacency graph of the partition is planar, and is called the *block graph*, and

3. every k-block of the partition has only triangles of the partition as its neighbors.

We construct such a partition through recursive calls to the well-known planar separator theorem.

Lemma 37. *[18] Every planar graph $G = (V, E)$ on n nodes admits a partition of the nodes into three sets A, S, and B, such that neither A nor B has more than $2n/3$ nodes, S has at most $\sqrt{6n}$ nodes, and there are no edges with one endpoint in A and the other endpoint in B. The set of nodes S is called a* separator *of G.*

The only non-trivial part is that we want our separators to be of size $O(\sqrt{h})$, and not $O(\sqrt{n})$, but still want the two parts to be balanced in n. We do this by a suitable contraction of the triangulation graph of the polygon, and a recursive use of the separator theorem to achieve the balanced partition. We first need the following lemma as an intermediate result.

Lemma 38. *Given a triangulation of a polygon P with n vertices and h holes, we can find a set of $O(\sqrt{h})$ triangles whose removal partitions P into two (possibly disconnected) sub-polygons, each containing at most $2h/3$ holes and $2n/3$ vertices.*

Proof. The graph-theoretic *dual* of the triangulation is an $O(n)$ size planar graph, with a vertex for each triangle and an edge between two nodes if those triangles have a common boundary edge. In this graph, there is a *cycle* surrounding each of the h holes, and it is the structure of those cycles that is important to us. We reduce this triangulation graph to an $O(h)$ size planar graph, by repeatedly *contracting* vertices of degree 2, and deleting vertices of degree one, until all vertices have degree three. The resulting graph G has

h faces, each vertex has degree 3, and so by Euler's formula, it has $O(h)$ vertices and edges as well.

By the planar separation theorem, we can find a separator of size $O(\sqrt{h})$ that splits the graph into two parts, each containing at most $2h/3$ nodes, as well as $2/3$ of the faces of G. In the primal space of triangulation, the separator corresponds to $O(\sqrt{h})$ triangles, splitting the polygon into two pieces, each containing at most $2h/3$ holes. However, the split does not guarantee any balance for the number of polygon *vertices*. Thus, if either piece contains more than $2n/3$ vertices, we apply the algorithm recursively until no piece has more than $2n/3$ vertices. The total number of triangles used to achieve the desired partition follows the recurrence $T(h) = T(2h/3) + O(\sqrt{h})$, with $T(1) = 1$, which solves to $T(h) = O(\sqrt{h})$. In the base case, the subpolygon contains no holes, and a single triangle is sufficient to split the polygon into two pieces, each of size at most $2n/3$. This completes the proof. □

We repeatedly apply Lemma 38 to construct our k-block partition.

Lemma 39. *Every multiply-connected polygon with n vertices and h holes admits a k-block partition for any $3 \leq k \leq n$.*

Proof. We apply Lemma 38 recursively to our polygon P until each piece is a subpolygon of k vertices, possibly with holes. The recursive partition naturally corresponds to a binary tree, called the *partition tree*, whose leaves are the k-blocks and non-leaves are

the separators. The blocks corresponding to any two leaves necessarily are on opposite sides of a separator, and thus the partition has the desired adjacency property. It only remains to bound the total number of triangles and blocks. Each triangle is part of a separator, so the number of triangles equals the total size of all the separators used in the partition. The number of blocks (leaves of the partition tree) is upper bounded by the number of non-leaf nodes, which in turn is upper-bounded by the number of triangles.

First, consider the case when the number of holes is $h \leq n/k$. We classify the triangles into two groups, depending on whether or not the sub-polygon being split has holes. If the sub-polygon is hole-free, then a single triangle can split it in a balanced $\frac{1}{3}$–$\frac{2}{3}$ ratio, and so the total number of triangles used for splitting hole-free sub-polygons is $O(n/k)$. To bound the number of triangles used for splitting sub-polygons with holes, consider an intermediate sub-polygon P_j created during the partition, which has $h_j > 0$ holes. Call this sub-polygon *i-big* if $(2/3)^{i+1}h < h_j \leq (2/3)^i h$. There are at most $(3/2)^{i+1}$ i-big subpolygons because the subproblems at any level of the partition tree are pairwise disjoint, and there are a total of h holes shared among them. The maximum value of i with an i-big sub-polygon is $\log_{3/2} h$, and since the separator of an i-big polygon has size $O(\sqrt{(2/3)^i h})$, the total size of all the separators (number of triangles) created during the partition tree is

$$\sum_{i=0}^{\log_{3/2} h} \left(\frac{3}{2}\right)^{i+1} \cdot c\sqrt{\left(\frac{2}{3}\right)^{i} h},$$

for some constant c. One can easily verify that this sums to $O(h)$. Thus, the total number of triangles in the partition is $O(h + n/k) = O(n/k)$.

Now, assume that $h > n/k$, and consider an intermediate sub-polygon P_j with n_j vertices and h_j holes. In this case, call the sub-polygon *i-big* if either $(2/3)^{i+1}h < h_j \leq (2/3)^i h$ or $(2/3)^{i+1}n < n_j \leq (2/3)^i n$. (Intuitively, a subpolygon is i-big if either its number of holes or its number of vertices is large enough to force a split to the next level.) We claim that there are at most $2 \cdot (3/2)^{i+1}$ subpolygons that are i-big. This holds because all subproblems at any level of the partition tree are pairwise disjoint, and at most $h/(2/3)^{i+1}h = (3/2)^{i+1}$ polygons arise due to the condition on the number of holes, and at most $n/(2/3)^{i+1}n = (3/2)^{i+1}$ due to the condition on the number of vertices.

Since $(2/3)^{\log_{3/2}(n/k)}n = k$, each non-leaf sub-polygon created during recursive partitioning is i-big for some i where $0 \leq i < \log_{3/2}(n/k)$. Since the separator of an i-big polygon has size $O(\sqrt{(2/3)^i h})$, we can bound the total size of all the separators created during the partition tree as

$$\sum_{i=0}^{\log_{3/2}(n/k)} 2 \cdot \left(\frac{3}{2}\right)^{i+1} \cdot c\sqrt{\left(\frac{2}{3}\right)^{i} h},$$

for some constant c. One can easily verify that this sums to $O(\sqrt{nh/k})$, which completes the proof. □

The following lemma is straightforward.

Lemma 40. *Suppose a pursuer is placed in each triangle of a k-block partition of the polygon. Then, after any move of the evader that crosses a block or triangle boundary, the pursuers know the identity of $B(e)$, the block or triangle containing the evader.*

Additionally, the evader can be initially located using the following result of Guibas et al.

Lemma 41 ([25])**.** *Given an n vertex polygon P with h holes, $O(\sqrt{h} + \log n)$ pursuers can locate the evader in $O(n \cdot \mathrm{diam}(P))$ moves.*

3.3.3 Analysis of Capture in Polygons with Holes

We now have all the pieces in place to describe the outline of the capture strategy and derive our main result. Following our scheme for the polygons without holes, we construct the k-block partition using the *constrained Delaunay triangulation* of P so that any diagonal (edge of a triangle) can be covered using the construction of Lemma 33. In particular, we can k-cover all three edges of a triangle so that a k-critical evader move immediately leads to capture. At a high level, our capture algorithm has the following form.

Algorithm PolygonWithHolesCapture

1. Construct a k-block partition of P, using the Constrained Delaunay Triangulation. Place one pursuer in each separating triangle to track the current block, or the triangle, $B(e)$ containing the evader.

2. Position pursuers at nodes of the partition tree until the evader is trapped in an extended block $B^+(e)$, whose adjacent triangles are all k-covered. With the pursuers in this position, any move by the evader exiting $B^+(e)$ is a k-critical move, leading to capture.

3. With the evader confined to an extended k-block, we use an additional set of $O(k + \sqrt{h})$ pursuers to find and capture the evader in $B^+(e)$.

Only Steps 2 and 3 require explanation—the k-block partition is already described by Lemma 39. Step 3, in fact, is also easy because the end-game algorithm of Lemma 36 works even with holes: a polygon with k vertices, including holes, can always be triangulated using $O(k)$ triangles, and our end-game algorithm requires a constant number of pursuers per triangle. We note that $B(e)^+$ has $O(k + \sqrt{h})$ vertices because in the worst-case, a block may neighbor $O(\sqrt{h})$ separating triangles. Thus, the only remaining part is Step 2, which is analyzed in the following lemma.

Lemma 42. *With $O(k\sqrt{h})$ pursuers, we can confine the evader to an extended block.*

Proof. We first position pursuers to achieve k-covering of each of the $O(\sqrt{h})$ triangles for the separator at the root node of the partition tree. We then search the $O(\sqrt{h})$ covers, which either leads to the capture or evicts the evader from these covers. The important observation is that once the evader is outside all the covers associated with the root's separator, it is confined to the (extended) blocks of one side of the separator—any

crossing of the separator causes a k-critical event and leads to immediate capture.[1] Once the root node is covered, we recursively apply the algorithm to the child node whose subtree contains the block $B(e)$ with the evader, which the pursuers know by Lemma 40. The recursion stops when we reach a leaf node at which point the evader is confined to an extended k-block. Let us now examine the total number of pursuers needed in this search. Because the number of holes in a subproblem shrinks by factor $2/3$ at each level, the number of pursuers needed to k-cover all the separators along a root-to-leaf path has the following recurrence: $T(h) = T(2h/3) + O(k\sqrt{h})$, which solves to $O(k\sqrt{h})$. □

We can now prove our main result.

Theorem 12. *Let $f(n, h)$ be the number of pursuers needed to capture an equally fast evader in a polygon of n vertices and h holes. Then,*

$$f(n,h) \;=\; \begin{cases} O(n^{1/2} \cdot h^{1/4}) & \textit{if } h \leq n^{2/3} \\ O(h^{1/2} \cdot n^{1/3}) & \textit{otherwise} \end{cases}$$

Proof. By Lemma 42, we can confine the evader to a single extended k-block using $O(k\sqrt{h})$ pursuers, and then use the end-game algorithm to complete the capture with $O(k + \sqrt{h})$ additional pursuers. We choose the appropriate value of k, depending on the number of holes, to prove the result.

[1] The evader may exit all the covers but remain within a triangle. In that case, we treat the triangle as a trivial block.

When $h \leq n^{2/3}$, then we choose $k = n^{1/2}/h^{1/4}$. In this case, we have $h \leq n/k$. The block partition has $O(n/k) = O(n^{1/2} \cdot h^{1/4})$ triangles, each requiring one pursuer. The k-covering of triangles requires $O(k\sqrt{h}) = O(n^{1/2} \cdot h^{1/4})$ pursuers. Thus, the total number of pursuers is $O(n^{1/2} \cdot h^{1/4})$.

When $h > n^{2/3}$, then we choose $k = n^{1/3}$. In this case, $h > n/k$, and the block partition has $O(\sqrt{nh/k}) = O(n^{1/3} \cdot h^{1/2})$ triangles, each requiring a single pursuer. The k-covering of triangles needs additional $O(k\sqrt{h}) = O(n^{1/3} \cdot h^{1/2})$ pursuers, for the total of $O(n^{1/3} \cdot h^{1/2})$. This completes the theorem. □

The bounds of the preceding theorem can be combined into a single upper bound of $O(n^{5/6})$, giving the following Theorem.

Theorem 13. *Suppose P is a n vertex polygon with h holes, where n includes the vertices of the holes. Then $O(n^{5/6})$ pursuers can capture an equally fast evader in $O(n \cdot \mathrm{diam}(P) + \log(n) \cdot \mathrm{diam}(P)^2)$ moves.*

Proof. The worst case number of pursuers is given as a corollary of Theorem 12, thus we concern ourselves with the duration of the capture. By invoking Lemma 41 the pursuers can determine the initial k-block containing the evader in $O(n \cdot \mathrm{diam}(P))$ moves. Additionally, before confining the evader to an obstacle free k-block at most $O(\log n)$ sets of triangles must be k-covered, each taking $O(\mathrm{diam}(P)^2)$ moves when the evader must be evicted from a cover by Lemma 36. Thus, with the final invocation

of the End Game algorithm to capture the evader in a k-block, the worst case number of moves is $O(n \cdot \mathrm{diam}(P) + \log(n) \cdot \mathrm{diam}(P)^2)$. □

3.4 Minimum Feature Size Assumption

We now show that a *minimum feature size* property of the environment is sufficient to yield significantly better upper bounds for the capture problem. Specifically, we show that $O(\log n)$ pursuers are always sufficient to catch the evader in a simply-connected polygon of n vertices, and $O(\sqrt{h} + \log n)$ if there are h holes. The pursuers' winning strategy is deterministic, and succeeds in polynomial time. The minimum feature size of a polygonal environment is defined as follows.

Definition 3. Minimum Feature Size (MFS): The minimum feature size of a (multiply-connected) polygon P is the minimum distance between any two vertices, where the distance is measured by the shortest path within the polygon.

We assume that the minimum feature size of the environment is *lower bounded* by the maximum speed of the players: i.e., the environment has minimum feature size of at least one. One can check that the polygon used in our lower bound (Figure 3.2.1) violates the minimum feature size: the players' maximum speed is 1 but there are pairs of vertices that are within $1/\sqrt{n}$ of one another.

The primary reason that the minimum feature size allows a large reduction in the number of required pursuers is that a 1-critical move by the evader will result in its

capture, *regardless of knowledge of which k-block contains the evader*. Meaning, a square can be guarded with eight pursuers, even if *no additional pursuers* are deployed in P. The following lemma proves this fact.

Lemma 43. *Suppose that the minimum feature size of P is at least one. Then if the an evader's move becomes critical with respect to a pursuer p, then p can capture the evader on its next move.*

Proof. As shown in Lemma 30, if e's move to e' is critical for p at some point e_c, then it must be the case that $d(p, e') \leq d(e, e') \leq 1$. Thus, if the terminal position e' of the evader is *visible* to p, then p can capture the evader.

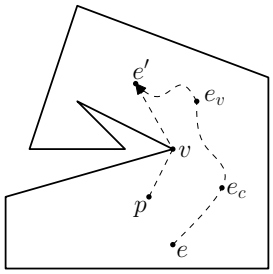


Figure 3.8: The proof of Lemma 43.

Let e_v be the last position during e's move where the evader is visible to p. Using the triangle inequality and the assumption $d(p, e_c) \leq d(e, e_c)$, we conclude that $d(p, e_v) \leq d(e, e_v)$. Notice then that the line segment (p, e_v) must contain a vertex of the environment, call it v, blocking p's visibility past the point e_v (see Figure 3.8). We claim that the shortest path homotopic to (p, e_v, e') is (p, v, e'), that is, it consists of a

single vertex v. Since the path (p, e_v, e') has length at most 1, the shortest path of the same homotopy also has length at most one, and the minimum feature size forbids two vertices with shortest path distance less than one. Thus, v is visible from both p and e', and $d(p, v) + d(v, e') \leq 1$. The pursuer p, therefore, can capture by first moving to v and then to e' in a single move. □

Due to the preceding lemma, the pursuers need only 1-cover each cut edge, which can be done with $O(1)$ pursuers. Further, it is no longer necessary to position pursuers on the cut edges to track the current k-block of the evader as a 1-critical move is sufficient to capture the evader without this knowledge. As a result, when sweeping the block graph the pursuers will not know which subtree to recursively search after covering the cut edges (cf. Lemma 35). When this occurs, the pursuers use the localization strategy of Lemma 29 to find the evader, and determine which subtree to search. Thus, by setting $k = 3$ our algorithm will confine the evader to an extended 3-block, at which point $O(1)$ pursuers can capture the evader using the end game algorithm, and we obtain the following theorem.

Theorem 14. *Suppose P is a simply connected n vertex polygon with MFS at least 1. Then $O(\log n)$ pursuers can capture an equally fast evader in $O(n^2 \cdot \mathrm{diam}(P) + n \cdot \mathrm{diam}(P)^2)$ moves.*

Proof. Over the course of the algorithm at most three cut edges are covered at any one time, each using $O(1)$ pursuers. An additional $O(\log n)$ pursuers are reused to search for the evader, and finally $O(1)$ pursuers are used to capture the evader when it is confined to an extended 3-block. Thus $O(\log n)$ pursuers suffice to capture the evader.

For each recursive covering of cut edges in Lemma 35 one node is removed from the block graph and one search occurs. Thus, over the course of the algorithm, the evader must be located at most n times. By Lemma 29 the n searches each have duration $O(n \cdot \mathrm{diam}(P))$, and take a total of $O(n^2 \cdot \mathrm{diam}(P))$ moves. Additionally the end game algorithm may be invoked $O(n)$ times to force the evader out of edge covers and once for the final capture taking $O(n \cdot \mathrm{diam}(P)^2)$ moves. Thus the evader will be captured in $O(n^2 \cdot \mathrm{diam}(P) + n \cdot \mathrm{diam}(P)^2)$ moves. □

When P contains holes, we again take advantage of the fact a 1-critical move is sufficient for capture and cover triangles with $O(1)$ pursuers and set a block size of 3. Once again, due to the absence of pursuers on each cut edge, when sweeping the partition tree it is necessary to search each of the subtrees to determine which one the evader is in, which can be done with Lemma 41.

The following theorem bounds the total number of pursuers and moves needed to capture the evader.

Theorem 15. *Suppose P n vertex polygon with h holes and MFS at least* 1. *Then $O(\sqrt{h} + \log n)$ pursuers can capture an equally fast evader in $O(n^2 \cdot \mathrm{diam}(P) + n \cdot \mathrm{diam}(P)^2)$ moves.*

Proof. We make one slight modification to our algorithm used in Theorem 12; when the pursuit reaches a point where the evader has been confined to a simply connected polygon, stop covering triangles and apply Theorem 14. By Lemma 38 each separator splits the polygon into subpolygons with at most $2/3$ as many holes, and thus the total number of triangles covered before reaching the simply connected polygon is $T(h) = T(2h/3) + O(\sqrt{h}) = O(\sqrt{h})$. Therefore $O(\sqrt{h})$ pursuers are needed to cover the triangles. Further, $O(\sqrt{h} + \log n)$ pursuers are reused to locate the evader, and $O(\log n)$ are used to capture the evader in a simply connected subpolygon for a total of $O(\sqrt{h} + \log n)$ pursuers.

The evader need only be located during each recursive partitioning, of which there are at most $O(\log(h))$, until a simply connected subpolygon is reached. The $O(\log(h))$ searches each have duration $O(n \cdot \mathrm{diam}(P))$ by Lemma 41 for a total of $O(n \cdot \log(h) \cdot \mathrm{diam}(P))$ moves. Additionally, the end game algorithm may be invoked n times to force the evader out of covered triangles taking $O(n \cdot \mathrm{diam}(P)^2)$ moves. Finally, invoking Theorem 14 takes $O(n^2 \cdot \mathrm{diam}(P) + n \cdot \mathrm{diam}(P)^2)$ moves. Thus the evader will be capture in $O(n^2 \cdot \mathrm{diam}(P) + n \cdot \mathrm{diam}(P)^2)$ moves.

□

3.5 A Randomized Pursuit Strategy

In our upper bounds so far, we have assumed that the evader can always predict the deterministic strategy of the pursuers. But now suppose that the pursuers have access to a source of randomness which the evader cannot predict. If they use this to randomize their movements, they can capture the evader in simply connected polygons with $O(1)$ pursuers, and $O(\sqrt{h})$ when there are h holes, even without the minimum feature size assumption [2].

The first step to achieving these bounds is the following lemma that shows that a 1-critical move is sufficient for a pursuer to capture the evader with probability $1/n$.

Lemma 44. *If the evader's move is critical with respect to a pursuer p, then p can capture the evader on its next move with probability $1/n$.*

Proof. As e's move to e' is critical with respect to p, we know that $d(p, e') \leq 1$. Thus, if e' is visible to p, it is captured. Otherwise, there is at least one vertex on the shortest path from p to e'. Thus, suppose there are m vertices within distance one of p. The pursuer uniformly at random chooses and moves along the shortest path to one of those m vertices. With probability $1/m \geq 1/n$ the pursuer gains visibility of e and has moved along the shortest path to e. Thus p can use to remainder of its move to capture the evader. □

[2] Applying the minimum feature size assumption reduces the expected time to capture by a factor of n, but does not change the number of required pursuers.

Suppose now that we use the MFS capture algorithm for simply connected polygons, except now edges are covered with probability $1/n$ using Lemma 44. If the evader performed a critical move it would have $1/n$ chance of being captured (if no critical move is performed it will be captured). If the evader avoided capture after the critical move, the pursuers simply restart the algorithm. Thus, the evader is expected to be captured in n rounds of the algorithm.

In order to actually reduce the number of pursuers required to capture the evader, we replace the $O(\log n)$ pursuers with a single pursuer using the following randomized strategy of Isler et al. [29].

Lemma 45. *Given a simply connected n-gon, a single pursuer has a randomized strategy that can locate the evader in $\mathrm{diam}(P)$ moves with probability at least $1/n$.*

Using the preceding lemma, we are able to prove the following theorem.

Theorem 16. *Suppose P is a simply-connected n-vertex polygon. Then, $O(1)$ pursuers can capture the evader in $O(n^3 \cdot \ln(n) \cdot \mathrm{diam}(P) + n^2 \cdot \mathrm{diam}(P)^2)$ expected moves with probability at least $1 - \frac{1}{n}$.*

Proof. There are n expected rounds of the algorithm before the evader is captured. In each round, the evader must be located at most n times (by the same reason as Theorem 14), for a total of n^2 localizations. When locating the evader, the probability of success of a single trial is at least $1/n$. Using the inequality $(1 + x) \leq e^x$, one can

easily show that after $3n\ln(n)$ trials of Lemma 45, the probability of not locating the evader is at most $1/n^3$. Then, by the union bound, the probability of failure in any of the n^2 localizations, is at most $n^2 \cdot 1/n^3 = 1/n$.

Thus, the n^2 localizations finish in $O(n^3 \cdot \ln(n) \cdot \mathrm{diam}(P))$ moves, with probability at least $1 - 1/n$. The remainder of the algorithm consists of executions of the end game algorithm and covering edges, both of which are deterministic, and their repetition over n rounds takes at most $O(n^2 \cdot \mathrm{diam}(P)^2)$ moves. Finally, $O(1)$ pursuers are used to cover the three cut edges, one pursuer is reused to locate the evader, and $O(1)$ to perform the end game algorithm and capture the evader, for a total of $O(1)$ pursuers. □

If P contains holes, we again use the MFS capture algorithm (Theorem 15) with the exception that triangles are covered with probability $1/n$. As in Theorem 15 the triangles are covered in order to confine the evader to a simply-connected polygon at which point the simply-connected algorithm is invoked (in this case the algorithm of Theorem 16). Finally, in order to reduce the number of required pursuers we replace the $O(\sqrt{h} + \log n)$ pursuer deterministic localization algorithm of Guibas et al. with the following randomized strategy of Isler et al. [29].

Lemma 46. *Given a multiply-connected n-gon with h holes, $O(\sqrt{h})$ pursuers have a randomized strategy which can locate the evader in $O(n \cdot \mathrm{diam}(P))$ moves with probability at least $1/n^2$.*

Using the preceding Lemma, we are able to prove the following Theorem.

Theorem 17. *Suppose P is a multiply connected n vertex polygon with h holes. Then, $O(\sqrt{h})$ pursuers can capture the evader in $O(n^5 \cdot \ln(n) \cdot \mathrm{diam}(P) + n^2 \cdot \mathrm{diam}(P))$ expected moves with probability at least $1 - \frac{1}{n}$.*

Proof. There are n expected rounds of the algorithm before the evader is captured. In the worst case, the evader must be located n times per round, for a total of n^2 localizations[3]. During one localization, the probability of success of a $O(n \cdot \mathrm{diam}(P))$ move trial is at least $\frac{1}{n^2}$. Using the inequality $(1 + x) \leq e^x$, one can easily show that after $3n^2 \ln(n)$ trials of Lemma 46, the probability of not locating the evader is at most $1/n^3$. Then, by the union bound, the probability of failure in any of the n^2 localizations, is at most $n^2 \cdot 1/n^3 = 1/n$.

Thus, the n^2 localizations finish in $O(n^5 \cdot \ln(n) \cdot \mathrm{diam}(P))$, with probability at least $1 - \frac{1}{n}$. The remainder of the algorithm consists of covering triangles, cut edges, and executing the end-game algorithm which are all deterministic, and their repetition over n rounds takes at most $O(n^2 \cdot \mathrm{diam}(P)^2)$ moves. Finally, $O(\sqrt{h})$ pursuers are used to guard the separating triangles, $O(\sqrt{h})$ are reused to locate the evader, and $O(1)$ pursuers by the algorithm of Theorem 16, for a total of $O(\sqrt{h})$ pursuers. □

[3]Some of the localizations will be in the invocation of Theorem 16, however, considering them localizations among obstacles greatly simplifies the analysis at the cost of a slight increase in the time bound

Chapter 4

Trackability with Imprecise Localization

4.1 Introduction

The problem of tracking a single known target is a classical one with a long history in artificial intelligence, robotics, computational geometry, graph theory and control systems. The underlying motivation is that many robotic applications including search-and-rescue, surveillance, reconnaissance and environmental monitoring have components that are best modeled as a tracking problem. The problem is often formulated as a *pursuit-evasion* game, with colorful names such as Man-and-the-Lion, Cops-and-Robbers, Hunter-and-Rabbit, Homicidal Chauffeur, and Princess-and-Monster [2, 7, 12, 31]. Visibility-based pursuit evasion [25, 70], in particular, has been a topic of great interest, in part due to its simple but realistic model: a team of pursuers is tasked with locating a single adversarial evader in an geometric environment with polygonal obstacles where pursuers learn the

evader's position only when the latter is in their line-of-sight. After two decades of research, tight bounds are known for detection or capture of the evader for many basic formulations of the problem [9, 25, 39], although the topic remains a rich subject of ongoing research [40, 56].

Most theoretical analyses of tracking, however, assume an idealized sensing model, ignoring the fact that all location sensing is noisy and imprecise in practice: the target's position is rarely known with complete and error-free precision. Although some papers have explored models to incorporate practical limitations of idealized visibility including angular visibility [33], beam sensing [58], field-of-view sensors [22], and range-bounded visibility [13], the topic of sensing noise or imprecision has largely been handled heuristically or through probabilistic techniques such as Kalman filters [32, 47, 67, 71]. One exception is [61], where Rote investigates a tracking problem under the *absolute error* model: in this model, the target's position is always known to lie within distance 1 of its true location, regardless of its distance from the tracker. The analysis in [61] shows that, under this noise model, the distance between the tracker and the target can grow at the rate of $\Theta(t^{1/3})$, where t is the time parameter. Our model, by comparison, deals with a more severe form of noise, with imprecision proportional to the distance from the tracker. In [44], Kuntsevich et al. consider the same relative error model as ours, but without any obstacles. Their work has a control-theoretic perspective, with a primary goal of deriving a bound on the time needed by the tracker to capture the target.

Our main contribution is to analyze the worst-case behavior of trackability as a function of the localization precision parameter λ.

Motivation and the Problem Statement. This chapter takes a small step towards bridging the gap between theory and practice of trackability, and analyzes the effect of noisy sensing. In particular, we consider a tracking agent P who wants to follow a moving target Q in d-dimensional Euclidean space using a noisy location sensor. For simplicity, we analyze the problem in two dimensions, but the results easily extend to d dimensions, as discussed in Section 4.5. We use the notation $Q(t)$ and $P(t)$ to denote the (true) positions of the target and the tracker at time t. We adopt a simple but realistic model of *relative* error in sensing noise: the localization error is *proportional* to the true distance between the tracker and the target. More precisely, the localization error is upper bounded as $||Q(t) - \tilde{Q}(t)|| \leq \frac{1}{\lambda}||P(t) - Q(t)||$ at all times t, where $\lambda \geq 1$ is the quality measure of *localization precision*. Thus, the closer the target, smaller the error, and a larger λ means better localization accuracy, while $\lambda = 1$ represents the completely noisy case when the target can be anywhere within a disk of radius $||P(t) - Q(t)||$ around $Q(t)$. *It is important to note that the parameter λ is used only for the analysis, and is not part of information revealed to the tracker.* In other words, the tracker only observes the approximate location $\tilde{Q}(t)$, and *not the uncertainty disk* containing the target. The relative error model is intuitively simple (farther the object, larger the measurement error) and captures the realism of many sensors: for instance, the

resolution error in camera-based tracking systems is proportional to the target's distance, and in network-based tracking, *latency* causes a proportionate localization uncertainty because of target's movement before the signal is received by the tracker.

We study the tracking problem as a game between two players, the tracker P and the target Q, which is played in *continuous time and space*: that is, each player is able to instantaneously observe and react to other's position, and the environment is the two-dimensional plane, with or without polygonal obstacles. Both the target and the tracker can move with equal speed, which we normalize to *one*, without loss of generality. With the unit-speed assumption, the following holds, for all times $t_1 \leq t_2$:

$$||Q(t_2) - Q(t_1)|| \leq |t_2 - t_1|, \quad ||P(t_2) - P(t_1)|| \leq |t_2 - t_1|$$

Under the *relative localization error* model, the reported location of the target $\tilde{Q}(t)$ always satisfies the following bound, where λ is the accuracy parameter:

$$||Q(t) - \tilde{Q}(t)|| \ \leq \ \frac{||P(t) - Q(t)||}{\lambda}$$

We measure the tracking performance by analyzing the distance function between the target and the tracker, namely, $D(t) = d(P(t), Q(t))$, over time, with $D(0)$ being the distance at the beginning of the game. Under error-free localization, the distance remains bounded as $D(t) \leq D(0)$. We analyze how $||D(t) - D(0)||$ grows under the relative error model, as a function of λ. Our main results are as follows.

Our Results. We show that the simple greedy strategy of "always move to the *observed* location of the target" achieves $D(t) \leq D(0) + t/\lambda^2$. That is, the target's distance from the tracker can grow at most at the rate of $O(\lambda^{-2})$, the inverse quadratic function of the localization parameter. We prove this rate to be worst-case optimal with a matching lower bound: a strategy for the target that ensures that, under the relative error model, it can increase its distance as $D(t) \geq D(0) + \Omega(t/\lambda^2)$.

We then extend this analysis to environments with polygonal obstacles, and show that the tracker can increase its distance by $\Omega(t)$ in time t *for any finite* λ. This is unsurprising because two points within a small margin of sensing error can be far apart in free-space, thereby fooling the tracker into "blind alleys." More surprisingly, however, if we adopt a localization error that is proportional to the *geodesic* distance (and not the Euclidean distance) between the target and the tracker, then the distance increases at a rate of $\Theta(\lambda^{-1})$. This bound is also tight within a constant factor: the tracker can maintain a distance of $D(t) \leq D(0) + O(t/\lambda)$ by the greedy strategy, while the target has a strategy to ensure that the distance function grows as at least $D(t) \geq D(0) + \Omega(t/\lambda)$.

Our analysis also helps answer some other questions related to tracking performance. For instance, a natural way to achieve good tracking performance in the presence of noisy sensing is to let the tracker move at a faster speed than the target. Then, what is the minimum speedup necessary for the tracker to reach the target (or, keep within a

certain distance of it)? We derive upper and lower bounds for this speedup function, which are within a constant factor of each other as long as $\lambda \geq 2$.

4.2 Tracking in the Unobstructed Plane

We begin with the simple setting in which a tracking agent P wants to follow a moving target Q in the two-dimensional plane without any obstacles. We show that the trivial "aim for the target's observed location" achieves essentially the best possible worst-case performance. We first prove the upper bound on the derivative $D'(t)$ of the distance function $D(t)$, and then describe an adversary's strategy that matches this upper bound.

4.2.1 Tracker's Strategy and the Upper Bound

Our tracker uses the following obvious algorithm, whose performance is analyzed in Theorem 18 below.

GREEDYTRACK. At time t, the tracker P moves directly towards the target's observed location $\tilde{Q}(t)$.

Theorem 18. *By using* GREEDYTRACK*, the tracker can ensure that* $D(t) \leq D(0) + O(t/\lambda^2)$*, for all* $t \geq 0$*.*

Proof. Consider the true and the observed positions of the target, namely $Q(t)$ and $\tilde{Q}(t)$, respectively, at time t, and let γ be the angle formed by them at $P(t)$. See Figure 4.1. Consider an arbitrarily small time period Δt during which P moves towards $\tilde{Q}(t)$ and

$Q(t)$ moves away from $P(t)$. We want to compute the derivative of the distance function, given as Equation (4.1).

$$D'(t) \;=\; \lim_{\Delta t \to 0} \frac{D(t+\Delta t) - D(t)}{\Delta t} \tag{4.1}$$

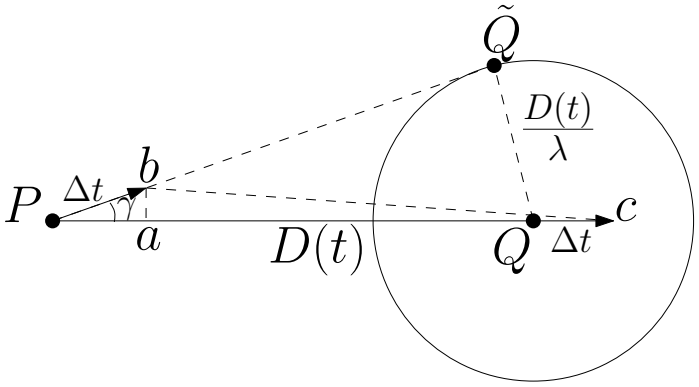


Figure 4.1: Proof of Theorem 18.

The new distance between the target and the tracker is given by bc in Fig. 4.1. In the triangle abc, we have $ab = \Delta t \sin\gamma$ and $ac = D(t) + \Delta t - \Delta t \cos\gamma$. We, therefore, can

bound $D(t+\Delta t)$ as follows (where the final inequality uses the fact $\sqrt{1+x} \leq 1 + \frac{x}{2}$):

$$\begin{aligned}
D(t+\Delta t) &= \sqrt{(\Delta t \sin\gamma)^2 + (D(t) + \Delta t - \Delta t\cos\gamma)^2} \\
&= \sqrt{\begin{array}{r}\Delta t^2 \sin^2\gamma + D(t)^2 + 2D(t)\Delta t(1-\cos\gamma) \\ + \Delta t^2 - 2\Delta t^2\cos\gamma + \Delta t^2\cos^2\gamma\end{array}} \\
&= \sqrt{\Delta t^2 + D(t)^2 + 2D(t)\Delta t(1-\cos\gamma) + \Delta t^2 - 2\Delta t^2\cos\gamma} \\
&= \sqrt{(D(t)+\Delta t)^2 - 2D(t)\Delta\cos\gamma + \Delta t^2 - 2\Delta t^2\cos\gamma} \\
&= \sqrt{\begin{array}{r}(D(t)+\Delta t)^2 - 2\Delta t(D(t)+\Delta t) + \Delta t^2 - 2D(t)\Delta\cos\gamma \\ - 2\Delta t^2\cos\gamma + 2\Delta t(D(t)+\Delta t)\end{array}} \\
&= \sqrt{\begin{array}{r}(D(t)+\Delta t-\Delta t)^2 - 2D(t)\Delta t\cos\gamma - 2\Delta t^2\cos\gamma \\ + 2\Delta t(D(t)+\Delta t)\end{array}} \\
&= \sqrt{D(t)^2 + 2\Delta t(D(t)+\Delta t)(1-\cos\gamma)} \\
&= D(t)\sqrt{1 + 2\Delta t(D(t)+\Delta t)(1-\cos\gamma)/D(t)^2} \\
&\leq D(t) + (\Delta t)(1+\Delta t/D(t))(1-\cos\gamma)
\end{aligned}$$

Returning to Equation (4.1), we get

$$D'(t) \;=\; \lim_{\Delta t\to 0}\frac{D(t+\Delta t)-D(t)}{\Delta t} \;\leq\; \lim_{\Delta t\to 0}(1+\Delta t/D(t))(1-\cos\gamma) \;=\; 1-\cos\gamma$$

Finally, since $\sin\gamma \leq \frac{1}{\lambda}$, we get $D'(t) \leq 1 - \sqrt{1 - \frac{1}{\lambda^2}}$, which simplifies by the Taylor series expansion:

$$D'(t) \leq 1 - (1 - \frac{1}{2\lambda^2} - \frac{1}{8\lambda^4} - \cdots) = \frac{1}{2\lambda^2} + \frac{1}{8\lambda^4} + \cdots \leq \frac{1}{\lambda^2}$$

This completes the proof that $D(t) \leq D(0) + t/\lambda^2$. □

4.2.2 Target's Strategy and the Lower Bound

We now show that this bound is asymptotically tight, by demonstrating a strategy for the target to grow its distance from the tracker at the rate of $D(t) \geq D(0) + \Omega(t/\lambda^2)$, for all $t \geq 0$. We think of the target as an adversary who can choose its observed location at any time subject only to the constraints of the error bound: $||Q(t) - \tilde{Q}(t)|| \leq \frac{1}{\lambda}(||P(t) - Q(t)||)$. (Recall that the tracker only observes the location $\tilde{Q}(t)$, and has no direct knowledge of either the parameter λ or the distance $||P(t) - Q(t)||$. Those quantities are only used in the analysis. However, the lower bound holds even if the tracker knows the uncertainty disk, namely, the localization error $\frac{1}{\lambda}(||P(t) - Q(t)||)$.)

In order to analyze the lower bound, we divide the time into *phases*, and show that the distance from the tracker increases by a *multiplicative factor* in each phase, resulting in a growth rate of $\Omega(1 + \lambda^{-2})$. If the ith phase begins at time t_i, then we let $d_i = ||Q(t_i) - P(t_i)||$ denote the distance between the target and the tracker at t_i. During

the ith phase, the target maintains the following invariant for a constant $0 < \alpha < 1$ to be chosen later.

Gap Invariant. Throughout the ith phase, the target moves along a path $Q(t)$ such that $||Q(t) - P(t)|| \geq \alpha d_i$, for all times t, and all reported locations satisfy $||Q(t) - \tilde{Q}(t)|| \leq \alpha d_i/\lambda$.

See Figure 4.2(a) for an illustration. Consider the isosceles triangle with vertices at $Q(t_i)$, q_a and q_b, whose base q_aq_b is perpendicular to the line $P(t_i)Q(t_i)$. The equal sides of the triangle have length $2d_i$, the base has length $2\alpha d_i/\lambda$, and let q_c be the midpoint of the base. The target's strategy is to move from $Q(t_i)$ to either q_a or q_b, and *report its location $\tilde{Q}(t)$ at the closest point on the line $Q(t_i)q_c$;* i.e. at all times, $\tilde{Q}(t)$ is the perpendicular projection of $Q(t)$ onto the line $Q(t_i)q_c$. By the symmetric construction, and the choice of the points q_a and q_b, the tracker cannot tell whether the target is moving to q_a or q_b. Thus, any deterministic tracker makes an incorrect choice in one of the two possible scenarios. For the *worst-case performance bound*, we can equivalently assume that the target *non-deterministically guesses* the tracker's intention, and moves to the better of the two possible locations, q_a or q_b. The tracker makes this choice based on whether the tracker is on or below the line $Q(t_i)q_c$, or not. In the former case, the target moves to q_a, and to to q_b otherwise. The ith phase terminates when the target reaches either q_a or q_b, and the next phase begins. (We note that, after i phases, there are 2^i possible choices made by the tracker, reflected in whether it is above or below

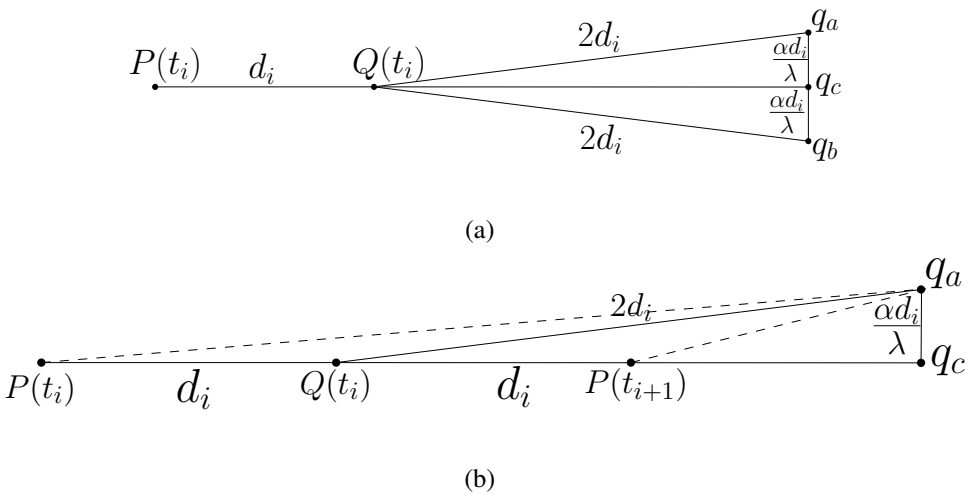


Figure 4.2: Target's strategy during the ith phase (a), and proofs of Lemmas 47 and 48 (b).

the line $Q(t_i)q_c$ at the conclusion of each phases. For each of these possible "worlds" there is a corresponding deterministic strategy of the target that "fools" the tracker in every phase, resulting in the maximum distance increase.) There is one subtle point worth mentioning here. It is possible that during the phase, the distance between the players may shrink if the tracker temporarily moves towards the same final location as the target—however, our Gap Invariant ensures that that the target's noisy location remains within the permissible error bound throughout the phase. The following lemma shows that this simple strategy of the target can maintain the Gap Invariant for any choice of $\alpha \leq 0.927$.

Lemma 47. *The target can maintain the Gap Invariant for any* $\alpha \leq 0.927$.

Proof. Consider an arbitrary phase i. By construction, we have $||Q(t) - \tilde{Q}(t)|| \leq \frac{\alpha d_i}{\lambda}$ throughout this phase, so we only need to show $D(t) \geq \alpha d_i$. There is one subtle point worth mentioning here. While the target's strategy will ensure that its distance from the tracker grows by a certain multiplicative factor *at the end of the phase*, the distance between the players may shrink during the phases. This happens when the tracker temporarily moves towards the same final location as the target. In spite of this temporary "lucky" guess by the tracker, we need to ensure that the target's noisy location remains within the permissible error bound throughout the phase. The constant α is introduced precisely to guarantee this validity, and we arrive at its value as follows.

Let d_{i+1} be the distance between P and Q if both moved toward q_a for the duration of phase i. Note that d_{i+1} is the length of the segment $P(t_i)q_a$ minus $2d_i$, as shown in Figure 4.2(b). The length of $P(t_i)q_a$ can be calculated from the right triangle $q_aP(t_i)q_c$, while the length of q_aq_c is known by construction. Finally, $Q(t_i)q_c$ has length d_i less than $P(t_i)q_c$. Thus, we have:

$$\begin{aligned} d_{i+1} &= \sqrt{(\sqrt{4d_i^2 - \frac{d_i^2\alpha^2}{\lambda^2}} + d_i)^2 + \frac{d_i^2\alpha^2}{\lambda^2}} - 2d_i \\ &= \sqrt{5d_i^2 - \frac{d_i^2\alpha^2}{\lambda^2} + 2d_i\sqrt{4d_i^2 - \frac{d_i^2\alpha^2}{\lambda^2}} + \frac{d_i^2\alpha^2}{\lambda^2}} - 2d_i \\ &= d_i\sqrt{5 + 4\sqrt{1 - \frac{\alpha^2}{4\lambda^2}}} - 2d_i \end{aligned}$$

In order to satisfy the Gap Invariant, we must choose an α such that the following inequality holds:

$$\begin{aligned}
\alpha d_i &\leq d_i\sqrt{5+4\sqrt{1-\frac{\alpha^2}{4\lambda^2}}}-2d_i \\
\alpha^2+4\alpha+4 &\leq 5+4\sqrt{1-\frac{\alpha^2}{4\lambda^2}} \\
\alpha^2+4\alpha+4 &\leq 5+4-\frac{\alpha^2}{2\lambda^2} \\
\alpha^2(1+\frac{1}{2\lambda^2})+4\alpha-5 &\leq 0
\end{aligned}$$

This gives the following upper bound:

$$\alpha \leq \frac{-4\pm\sqrt{16+4(1+\frac{1}{2\lambda^2})5}}{2(1+\frac{1}{2\lambda^2})} \leq \frac{1}{3}(\sqrt{46}-4)$$

□

The preceding lemma shows that our construction satisfies the Gap Invariant, and so we can now lower bound the distance growth during a single phase.

Lemma 48. *At the start of phase $i+1$, we have $d_{i+1} \geq d_i\sqrt{1+\frac{\alpha^2}{2\lambda^2}}$, where $\alpha = 0.927$ is an absolute constant.*

Proof. Suppose, without loss of generality, that the target is at q_a at the termination of the ith phase, which means the tracker is on or below the line $Q(t_i)q_c$. By the unit speed assumption, the target needs exactly $2d_i$ time for this move. The minimum value of

d_{i+1} is at least as large as if P had moved directly to q_c by distance $2d_i$, as shown in Figure 4.2(b). We can calculate d_{i+1} from the right triangle $q_a P(t_{i+1}) q_c$, as follows:

$$\begin{aligned}
d_{i+1} &\geq \sqrt{(\sqrt{4d_i^2 - \frac{\alpha^2 d_i^2}{\lambda^2}} - d_i)^2 + \frac{\alpha^2 d_i^2}{\lambda^2}} \\
&= \sqrt{d_i^2 - 2d_i\sqrt{4d_i^2 - \frac{\alpha^2 d_i^2}{\lambda^2}} + 4d_i^2 - \frac{\alpha^2 d_i^2}{\lambda^2} + \frac{\alpha^2 d_i^2}{\lambda^2}} \\
&= d_i\sqrt{5 - 4\sqrt{1 - \frac{\alpha^2}{4\lambda^2}}} \\
&\geq d_i\sqrt{5 - (4 - \frac{\alpha^2}{2\lambda^2})} \\
&= d_i\sqrt{1 + \frac{\alpha^2}{2\lambda^2}}
\end{aligned}$$

□

We can now prove the main result of this section.

Theorem 19. *Under the relative error localization model, a target can increase its distance from an equally fast tracker at the rate of* $\Omega(\lambda^{-2})$*. In other words, the target can ensure that* $D(t) \geq D(0) + \Omega(t/\lambda^2)$ *after any phase ending at time* t*.*

Proof. The target follows the phase strategy, where that after the ith phase that lasts $2d_i$ time units, the distance between the tracker and the target is at least $d_i\sqrt{1 + \frac{\alpha^2}{2\lambda^2}}$. Therefore, the distance increases during the ith phase by at least the following multiplicative factor (using a Taylor series expansion):

$$\frac{d_i\sqrt{1+\frac{\alpha^2}{2\lambda^2}}-d_i}{2d_i} = \frac{\sqrt{1+\frac{\alpha^2}{2\lambda^2}}-1}{2} \geq \frac{\alpha^2}{4\lambda^2}-\frac{\alpha^4}{16\lambda^4} = \Omega(\frac{1}{\lambda^2})$$

□

4.3 Trackability with a Faster Tracker

The results of the previous section establish bounds on the relative advantage available to the target by the localization imprecision. Its distance from the tracker can grow at the rate of $\Theta(\lambda^{-2})$ with time. A tracking system can employ a number of different strategies to compensate for this disadvantage. In this section, we explore one such natural mechanism: *allow the tracker to move at a faster speed than the target.* A natural question then is: what is the minimum speedup necessary to cancel out the localization noise as a function of λ? We give bounds on the necessary and sufficient speedups, which match up to small constant factors as long as $\lambda \geq 2$. The general form of the speedup function is $(1-\frac{1}{\lambda^2})^{-1/2}$. The following theorem proves the sufficiency condition.

Theorem 20. *Suppose the target moves with speed one, and the tracker has speed* $S = \sqrt{\frac{1}{1-1/\lambda^2}}$*, where* λ *is the localization precision parameter. Then, the tracker can maintain* $D(t) \leq D(0)$*, for all times* $t \geq 0$*.*

Proof. Our analysis closely follows the proof of Theorem 18, and calculates the increase in the distance during time Δt. During this time, the tracker is able to move $S\Delta t$, while

the target can move at most Δt. We can then calculate distance at time $t + \Delta t$ from the triangle abc (Fig. 4.1), where $ab = S\Delta t \sin\gamma$ and $ac = D(t) + \Delta t - S\Delta t \cos\gamma$, as follows:

$$
\begin{aligned}
D(t+\Delta t) &= \sqrt{(S\Delta t \sin\gamma)^2 + (D(t) + \Delta t - S\Delta t\cos\gamma)^2} \\
&= \sqrt{\begin{aligned} &S^2\Delta t^2 \sin(\alpha)^2 + D(t)^2 + 2D(t)\Delta t(1 - S\cos(\alpha)) + \Delta t^2 \\ &\qquad - 2\Delta t^2 S\cos(\alpha) + \Delta t^2 S^2 \cos(\alpha)^2 \end{aligned}} \\
&= \sqrt{S^2\Delta t^2 + D(t)^2 + 2D(t)\Delta t(1 - S\cos(\alpha)) + \Delta t^2 - 2\Delta t^2 S\cos(\alpha)} \\
&= \sqrt{S^2\Delta t^2 + (D(t) + \Delta t)^2 - 2D(t)\Delta t S\cos(\alpha) - 2\Delta t^2 S\cos(\alpha)} \\
&= \sqrt{\begin{aligned} &S^2\Delta t^2 + (D(t) + \Delta t)^2 - 2\Delta t(D(t) + \Delta t) + 2\Delta t(D(t) + \Delta t) \\ &\qquad + \Delta t^2 - \Delta t^2 - 2D(t)\Delta t S\cos(\alpha) - 2\Delta t^2 S\cos(\alpha) \end{aligned}} \\
&= \sqrt{\begin{aligned} &S^2\Delta t^2 + (D(t) + \Delta t - \Delta t)^2 + 2\Delta t(D(t) + \Delta t) - \Delta t^2 \\ &\qquad - 2D(t)\Delta t S\cos(\alpha) - 2\Delta t^2 S\cos(\alpha) \end{aligned}} \\
&= D(t)\sqrt{\begin{aligned} &1 + S^2\Delta t^2/D(t)^2 - \Delta t^2/D(t)^2 \\ &\quad + 2\Delta t(D(t) + \Delta t)(1 - S\cos(\alpha))/D(t)^2 \end{aligned}} \\
&\leq D(t) + S^2\Delta t^2/2D(t) - \Delta t^2/2D(t) + \Delta t(1 + \Delta t/D(t))(1 - S\cos\gamma)
\end{aligned}
$$

This allows us to bound $D'(t) \leq 1 - S\cos\gamma$, from which it follows that $D'(t) \leq 0$ as long as $S \geq \sqrt{\frac{1}{1-1/\lambda^2}}$. □

We now show that if $\lambda \geq 2$, this is the minimum speedup necessary as a function of λ, up to a small constant factor. We use the phase-based strategy of Theorem 19, however, the value of α determined by Lemma 47 is not sufficient to maintain the Gap Invariant in this case because of the higher speed of the tracker. Instead, the following lemma gives the sufficient choice of α.

Lemma 49. *Let $\lambda \geq 2$ and and $\alpha \leq 0.68$ be a constant. Then, the Gap Invariant can be maintained in any phase as long as $S \leq \frac{1}{\sqrt{1-1/\lambda^2}}$.*

Proof. Suppose, without loss of generality, that the target is at q_a at the termination of the ith phase, which means the tracker is below the line $Q(t_i)q_c$. By the unit speed assumption, the target needs exactly $2d_i$ time for this move. The minimum value of d_{i+1} is at least as large as if P had moved directly to q_c by distance $2Sd_i$, as shown in Figure 4.2(b). We can calculate d_{i+1} from the right triangle $q_aP(t_{i+1})q_c$, as follows:

$$d_{i+1} \geq \sqrt{(\sqrt{4d_i^2 - \frac{\alpha^2 d_i^2}{\lambda^2}} - (2Sd_i - d_i))^2 + \frac{\alpha^2 d_i^2}{\lambda^2}} \geq d_i\sqrt{(2S-3)^2 + \alpha^2(S-1/2)/\lambda^2} \tag{4.2}$$

In order to satisfy the Gap Invariant, we must choose an α such that the following inequality holds:

$$\alpha d_i \leq d_i\sqrt{5+4\sqrt{1-\frac{\alpha^2}{4\lambda^2}}} - 2Sd_i$$

$$\alpha^2 + 4\alpha S + 4S^2 \leq 5 + 4\sqrt{1-\frac{\alpha^2}{4\lambda^2}}$$

$$\alpha^2 + 4\alpha S + 4S^2 \leq 5 + 4 - \frac{\alpha^2}{2\lambda^2}$$

$$\alpha^2(1+\epsilon^2/2) + 4\alpha S + 4S^2 - 9 \leq 0$$

This gives the following upper bound when λ is minimum and S is maximum, which by assumption is 2 and $1/\sqrt{1-(1/2^2)}$, respectively.

$$\alpha \leq \frac{-4S + \sqrt{16S^2 - 4(1+1/2\lambda^2)(4S^2-9)}}{2(1+1/2\lambda^2)} \leq 0.68$$

□

We can now prove a lower bound on the increase in the distance during the ith phase.

Lemma 50. *If $\lambda \geq 2$, $\alpha \leq 0.68$, and $S \leq (1-1/\lambda^2)^{-1/2}$, then at the start of the $i+1$ phase, we have $d_{i+1} \geq d_i\sqrt{(2S-3)^2 + \alpha^2(S-1/2)/\lambda^2}$, where $\alpha = 0.68$ is an absolute constant.*

Proof. Suppose, without loss of generality, that the target is at q_a at the termination of the ith phase, which means the tracker is below the line $Q(t_i)q_c$. By the unit speed assumption, the target needs exactly $2d_i$ time for this move. The minimum value of d_{i+1} is at least as large as if P had moved directly to q_c by distance $2Sd_i$, as shown in Figure 4.2(b). We can calculate d_{i+1} from the right triangle $q_aP(t_{i+1})q_c$, as follows:

$$
\begin{aligned}
d_{i+1} &\geq \sqrt{(\sqrt{4d_i^2 - \frac{\alpha^2 d_i^2}{\lambda^2}} - (2Sd_i - d_i))^2 + \frac{\alpha^2 d_i^2}{\lambda^2}} \\
&= d_i \sqrt{5 + 4S^2 - 4S - 2(2S-1)\sqrt{4 - \frac{\alpha^2}{\lambda^2}}} \\
&\geq d_i \sqrt{5 + 4S^2 - 4S - 4(2S-1)(1 - \alpha^2/8\lambda^2)} \\
&= d_i \sqrt{4S^2 - 12S + 9 + S\alpha^2/\lambda^2 - \alpha^2/2\lambda^2} \\
&= d_i \sqrt{(2S-3)^2 + \alpha^2(S - 1/2)/\lambda^2}
\end{aligned}
$$

□

Remark. The preceding lemma can be used to calculate the maximum tracker speed for which the target can still force a non-negative distance for a specific λ as follows:

$$
\begin{aligned}
\sqrt{(2S-3)^2 + \alpha^2(S-1/2)/\lambda^2} &= 1 \\
4S^2 - 12S + \frac{\alpha^2 S}{\lambda^2} &= -8 + \frac{\alpha^2}{2\lambda^2} \\
2S - \left(\frac{12 - (\alpha/\lambda)^2}{4}\right)^2 - \left(\frac{12 - \alpha^2/\lambda^2}{4}\right)^2 &= -8 + \frac{\alpha^2}{2\lambda^2} \\
2S - \frac{12 - \alpha^2/\lambda^2}{4} &= -\sqrt{-8 + \frac{\alpha^2}{2\lambda^2} + (\frac{12 - \alpha^2/\lambda^2}{4})^2} \\
S &= \frac{-\sqrt{-8 + \frac{\alpha^2}{2\lambda^2} + (\frac{12-\alpha^2/\lambda^2}{4})^2} + \frac{12-\alpha^2/\lambda^2}{4}}{2}
\end{aligned}
$$

As λ gets large, the upper and lower bound are within a constant factor of each other. Indeed, with a more careful choice of α, we can show that the upper and lower bounds

are within a factor of 5.32 (as opposed to 10.23 for the above simple analysis) of each other for $\lambda \geq 2$, but we omit those details from this abstract.

4.4 Tracking in the Presence of Obstacles

The presence of obstacles makes the tracking problem considerably harder under the localization noise. The following simple example (Fig. 4.3) shows that the target can grow its distance from the tracker as $D(t) \geq D(0) + t$, for any finite value of λ. The obstacle consists of a single U-shaped non-convex polygon. Initially, the target is at distance $D(0)$ from the tracker, and the "width" of the obstacle is less than $D(0)/2\lambda$, so that the localization error is unable to distinguish between a target moving inside the U channel, or around its outer boundary. One can show that no matter how the tracker pursues, its distance from the target can grow linearly with time.

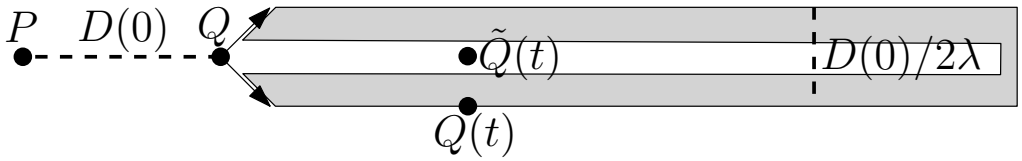


Figure 4.3: Impossibility of tracking among obstacles.

Path Proportionate Error. In order to get around this impossibility of tracking, we propose a *path proportionate error* measure, where the localization error is proportional to the *shortest path distance* between the target and the tracker, and not the Euclidean distance as used before. That is, the tracking signal and the physical movement of the

agents follow the same path metric. Formally, the localization error at time t always obeys the following bound:

$$d(Q(t), \tilde{Q}(t)) \leq \frac{d(P(t), Q(t))}{\lambda}$$

We show that the best tracking performance in this model is $D(t) = D(0) + \Theta(t/\lambda)$; that is the distance grows linearly with $1/\lambda$, as opposed to the inverse quadratic function for the unobstructed case.

4.4.1 Tracking Upper Bound

The tracker's strategy in this case is also greedy, except now the tracker makes short-term commitments in *phases*, instead of continuously changing its path towards the new observed location. In particular, for each phase, the tracker fixes its goal as the *observed position of the target at the start of the phase*, moves along the shortest path to this goal, and then begins the next phase.

MODIFIEDGREEDY. The initial phase begins at time $t = 0$. During the ith phase, which begins at time t_i, the tracker moves along the shortest path to the observed location of the target at t_i, namely, $\tilde{Q}(t_i)$. When tracker reaches $\tilde{Q}(t_i)$, the ith phase ends, and the next phase begins.

The upper bound on the tracking performance is given by the following theorem.

Theorem 21. *Using* MODIFIEDGREEDY*, the tracker can ensure that* $D(t) \leq D(0) + O(t/\lambda)$.

Proof. First note that because $d(\tilde{Q}(t_i), Q(t_i)) \leq D(t_i)/\lambda$, it follows that $t_{i+1} - t_i = D(t_i) + xD(t_i)$, where $\frac{-1}{\lambda} \leq x \leq \frac{1}{\lambda}$. Thus, the target's progress during the ith phase is upper bounded as $d(Q(t_i), Q(t_{i+1})) \leq D(t_i) + xD(t_i)$. Next, by applying the triangle inequality, the distance between P and Q at the beginning of phase t_{i+1} is upper bounded as

$$\begin{aligned} d(P(t_{i+1}), Q(t_{i+1})) &= d(\tilde{Q}(t_i), Q(t_{i+1})) \\ &\leq d(\tilde{Q}(t_i), Q(t_i)) + d(Q(t_i), Q(t_{i+1})) \\ &\leq \frac{D(t_i)}{\lambda} + D(t_i) + xD(t_i) \end{aligned}$$

Finally, the upper bound on the rate of distance increase can be derived as follows:

$$\begin{aligned} \frac{d(P(t_{i+1}), Q(t_{i+1})) - d(P(t_i), Q(t_i))}{t_{i+1} - t_i} &\leq \frac{D(t_i) + D(t_i)/\lambda + xD(t_i) - D(t_i)}{D(t_i) + xD(t_i)} \\ &= \frac{1/\lambda + x}{1 + x} \leq \frac{2}{\lambda + 1} \end{aligned}$$

where the final inequality uses the fact that the minimum value occurs when $x = 1/\lambda$. In conclusion, during each phase the distance between the tracker and the target increases by at most a factor of $\frac{2}{\lambda+1}$, giving the bound $D(t) \leq D(0) + O(\frac{t}{\lambda})$ □

4.4.2 Tracking Lower Bound.

Our final result is to prove that the trackability achieved by MODIFIEDGREEDY is essentially optimal. In particular, we construct an environment with polygonal obstacles and a movement strategy for the target that ensures $D(t) \geq D(0) + \Omega(t/\lambda)$. The construction of the polygonal environment is somewhat complicated and requires a carefully designed set of obstacles. The main schema of the construction is shown in Figure 4.4, where each edge of the "tree-like" diagram corresponds to a "channel" bounded by obstacles, and each face corresponds to a "gadget" consisting of a group of carefully constructed obstacles, with the outer face occupied entire by a single large obstacle.

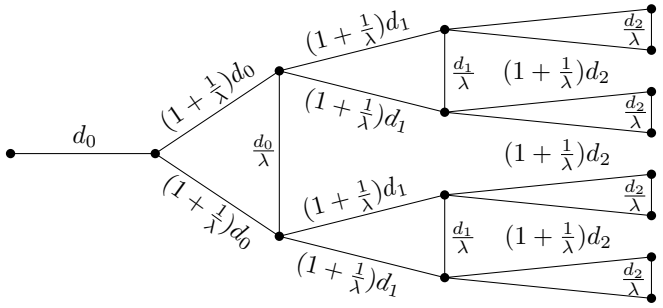


Figure 4.4: A high level schema for the lower bound construction. The numbers next to the edges denote the "path length" in the corresponding channels.

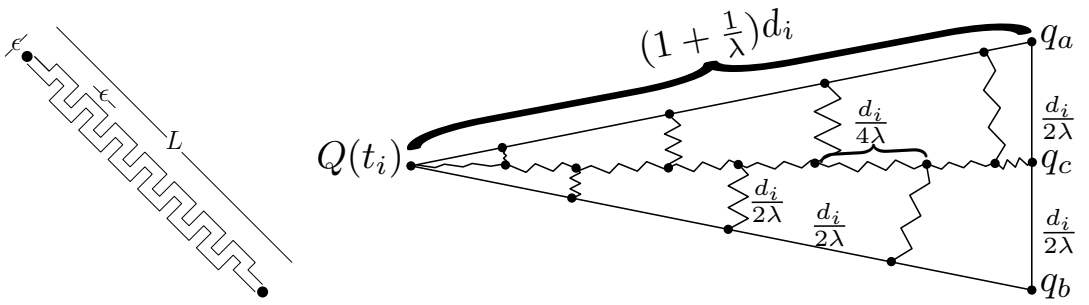
As in the proof of Theorem 19, the target moves either to top or the bottom point of the gadget during a phase, depending on the tracker's location. The gadget construction is such that the movement of the target along either path is indistinguishable to the tracker

because both paths are satisfied by a common set of observed locations throughout the path. Thus, by invoking the earlier equivalence principle, we may as well assume that the target knows the tracker's choices. If the target moves to the top, then the next phase occurs in the top gadget, otherwise the bottom, and so on.

To realize the geometric scheme of Figure 4.4, we replace each edge of the graph with a channel as shown in Figure 4.5(a). The desired edge length can be realized by adding any number of arbitrarily skinny bends such that the length of the shortest path through each channel equals the edge length. Each face is replaced with a set of obstacles, called a gadget, see Figure 4.5(b) for an abstract illustration. The jagged line between each pair of nodes corresponds to a channel such that shortest path through that channel has the given length. The target will move along the shortest path through either the top or bottom channel while reporting its location in the center channel. Meanwhile, the channels connecting the top and bottom to the center will guarantee that $d(Q(t), \tilde{Q}(t)) \leq \frac{1}{\lambda} d(Q(t), P(t))$ at all times t during a phase.

Gadget Construction and its Properties

We now describe the construction of our gadgets and establish the geometric properties needed for the correctness of our lower bound. Each gadget is constructed out of two building blocks, the bent channels seen in Figure 4.5(a), and intersections depicted in Figure 4.6(a). Each intersection has the property that the shortest path between any two of the points among a, b and c has length 2δ, where δ can be made arbitrarily close



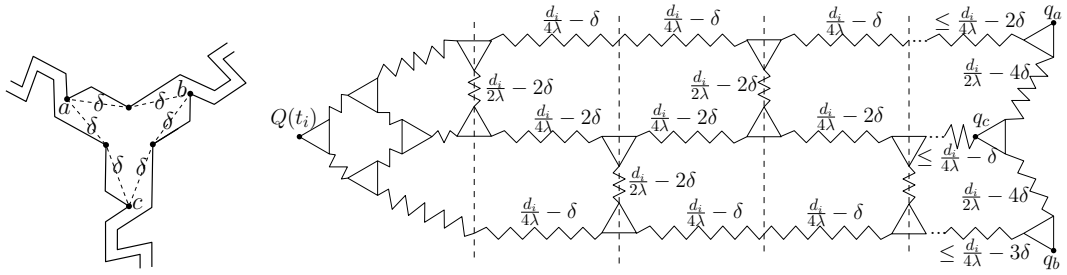
(a) (b)

Figure 4.5: The channel construction in (a). In (b) the shortest paths between nodes on the center path have length $\frac{d_i}{4\lambda}$, and the remaining all have length $\frac{d_i}{2\lambda}$.

to 0. Thus we can construct a channel that branches into two channels such that the path length through the intersection is the same regardless of the branch chosen. In Figure 4.6(b), we depict the construction of a gadget using only intersections (triangles) and channels (jagged lines).

As in the lower bound for the unobstructed case, the target starts the phase at $Q(t_i)$, and moves to q_a or q_b *while the observed location of the targets moves along the shortest path from* $Q(t_i)$ *to* q_c. In particular, let Π_a, Π_c, and Π_b denote the shortest paths from $Q(t_i)$ to q_a, q_c and q_b respectively. The following lemma establishes several properties needed for the feasibility of the target's strategy.

Lemma 51. *We can construct a gadget for each phase* i *such that (1)* Π_a*,* Π_c *and* Π_b *have length* $(1+\frac{1}{\lambda})d_i$ *and (2) for any point* x_c *at distance* ℓ *along* Π_c*, the corresponding*



(a) (b)

Figure 4.6: In (a) an example intersection such that the shortest path between any pair of a b and c has length 2δ. In (b) an example gadget construction, where each triangle corresponds to an intersection with corners representing the points a b and c. The horizontal channels have length $\frac{d_i}{4\lambda}$ between each pair of vertical dashed lines, except for the initial distance before the first line (which can be made arbitrarily small), and the remaining spillover distance after the last dashed line.

points x_a and x_b distance ℓ along Π_a and Π_b, respectively, satisfy $d(x_c, x_a) \leq \frac{d_i}{\lambda}$ and $d(x_c, x_b) \leq \frac{d_i}{\lambda}$.

Proof. By construction, the shortest path in each channel between the dashed lines in Figure 4.6(b) has length $\frac{d_i}{4\lambda}$, and therefore this construction can be extended until Π_a, Π_c and Π_b have length exactly $(1 + \frac{1}{\lambda})d_i$. Next, by the symmetry of the construction, we need only show that $d(x_c, x_a) \leq d_i/\lambda$. We ignore the case where x_c lies in the channels before the first dashed lines, as the length of such channels can be made arbitrarily small to guarantee that $d(x_a, x_c) \leq d_i/\lambda$. The maximum distance between x_a and x_c then occurs when x_a lies at the midpoint between two intersections in the top channel.

However, in this case one can easily verify that the following holds:

$$d(x_c, x_a) = \delta + \frac{d_i}{4\lambda} - 2\delta + 2\delta + \frac{d_i}{2\lambda} - 2\delta + 2\delta + \frac{d_i}{4\lambda} - \delta = \frac{d_i}{\lambda}$$

This completes the proof. □

Gap Invariant and the Proof of the Lower Bound

We now formulate the invariant maintained by the target so that its motion is valid under our (path proportionate) localization error and achieves the desired lower bound.

SP-Gap Invariant. Throughout the ith phase, the target moves along a path $Q(t)$ such that $D(t) \geq d_i$ for all times t, and all reported locations satisfy $d(Q(t), \tilde{Q}(t)) \leq \frac{d_i}{\lambda}$.

Lemma 52. *For the duration of phase i, SP-Gap Invariant is maintained.*

Proof. Whether Q moves along Π_a or Π_b, they are both shortest paths (and this cannot be shortcut by P), implying that $D(t) \geq d_i$ for the duration of the phase. Without loss of generality, suppose Q chooses Π_a. Then, after time t, both the target and its observed position have moved a distance of t along Π_a and Π_c, respectively. Therefore, by Lemma 51, we have $d(Q(t), \tilde{Q}(t)) \leq \frac{d_i}{\lambda}$. □

We can prove our lower bound.

Theorem 22. *The target's strategy guarantees that after each phase ending at time t, the distance function satisfies $D(t) \geq D(0) + \Omega(\frac{t}{\lambda})$.*

Proof. The proof is by induction on the phase i. The basis of the induction is $i = 0$. Since the localization error makes target's top and bottom paths indistinguishable to the tracker, the target can ensure that at the end of phase 0 the target is on the side of Π_c that is opposite P. Without loss of generality, suppose that that target has reached q_a. Then the best case for P is if it moved $\frac{d_0}{\lambda}$ along Π_c, which achieves $D(t_1) \geq D(0) + \frac{D(0)}{2\lambda}$.

Now assume by induction that after phase $i-1$ ends at time t_i, we have $D(t_i) \geq D(t_{i-1}) + D(t_{i-1})/2\lambda = d_i$. Suppose now that P has yet to reach the gadget corresponding to phase i when Q has finished phase i at time t_{i+1}. Then necessarily $D(t_{i+1}) \geq d_i + d_i/\lambda$, as that is the length Π_a and Π_b. Otherwise if P has moved into the gadget, then the inequality $D(t_i) \geq d_i$ ensures that the closest the target can be to the tracker is if P has moved $\frac{d_i}{\lambda}$ along Π_c, which implies $D(t_{i+1}) \geq D(t_i) + \frac{D(t_i)}{2\lambda}$.

Thus, in a round with duration $(1 + \frac{1}{\lambda})d_i$, the distance increases by at least $d_i/2\lambda$. Thus, in the ith phase, the distance increases by a factor of at least

$$\frac{d_i/2\lambda}{(1+\frac{1}{\lambda})d_i} = \frac{1}{2\lambda(1+\frac{1}{\lambda})} = \frac{1}{2+2\lambda}$$

Thus, at the end of any phase, we have the inequality $D(t) \geq D(0) + \Omega(t/\lambda)$, which completes the lower bound. □

4.5 Extension to d dimensions

Our analysis of trackability was carried out for 2-dimensional Euclidean plane, but the results generalize easily to d dimensions. Indeed, in the unobstructed case, our analysis of the upper bound only makes use of the triangle inequality: the region of interest is the triangle formed by $P(t)$, $Q(t)$, and $\tilde{Q}(t)$, and the target Q moves directly away from P. Thus, within an arbitrarily small time interval Δt, P and Q are moving within the two-dimensional plane of the triangle $P(t)Q(t)\tilde{Q}(t)$. The upper bound analysis therefore extend to any dimension $d \geq 2$. The same reasoning also holds in the presence of obstacles. Finally, the lower bound construction of $d = 2$ immediately implies that the trackability lower bound holds in all dimensions $d \geq 2$.

4.6 Simulation Results

In our first simulation, we use a GPS trace of a hike available from [1]. Using the scale of the GPS coordinate system, the total length of the trace is 0.51, and we place the tracker at an initial distance of 0.014 away from the target (Fig. 4.7), so that their initial separation is about 2.5% of the entire trarectory length. During the simulation, the target follows the GPS trace, the tracker moves directly toward the current reported location of the target, and they both have the same speed. The localization error for this simulation is set to $\lambda = 3$, a fairly high level of imprecision. At each instant, the revealed location

$\tilde{Q}$ of the target makes the largest allowable angle (deviation) from the PQ line. In our simulation, we consistently chose $\tilde{Q}$ to be the rightward point of tangency. However, results were similar or better if $\tilde{Q}$ is chosen using some other rule such as, leftward point, or randomly chosen between left and right. In Figure 4.7 we depict the paths followed by the players and observe that despite the initial distance between P and Q, and the large localization error, the tracker P quickly reduces its distance to Q. In fact, the gap continues to shrink, becoming almost zero, after only about $1/4$ of the trace. Figure 4.8 zooms into the initial portion of the trajectory to more clearly show the tracking path.

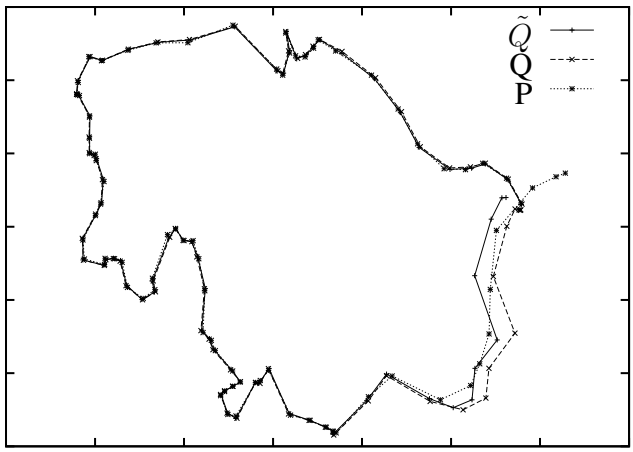


Figure 4.7: Depiction of the trajectories of P, Q, and $\tilde{Q}$.

Our second simulation uses a synthetic trajectory to force a worst-case (adversarial) tracking behavior: instead of moving along a fixed path, the target Q always moves *directly away from* P. The tracker moves directly toward the observed location $\tilde{Q}$, which

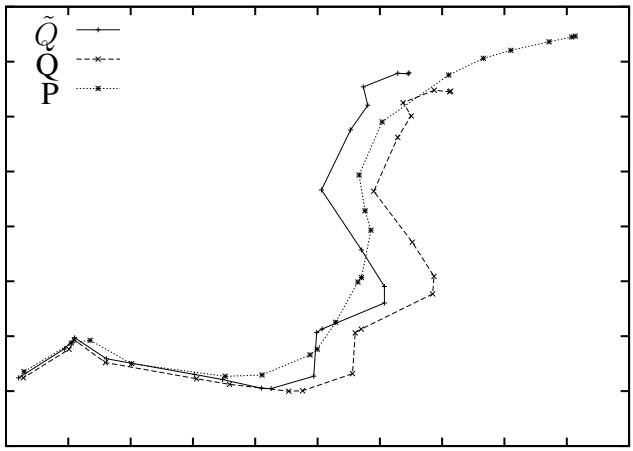


Figure 4.8: A zoomed-in view to illustrate the quick tracking convergence.

as in the previous simulation is chosen as the rightward point of tangency at maximum distance from Q. The error parameter is again set to $\lambda = 3$ and the simulation begins with P positioned at the origin and Q at the point $(10, 10)$. The result is shown in Figure 4.9. Essentially, P always moves to the right of Q's true location, and as a result Q moves further to the left at each step. This results in a *spiralling trajectory* in which the distance between P and Q is increasing by approximately $.05$ per time unit.

In another variation of this similation, the initial conditions are the same, except that $\tilde{Q}$ is chosen uniformly at random among all possible locations of $\tilde{Q}$. In this case, we found that the distance between tracker and target grows only by about $.005$ per time unit, namely, an order of magnitude better than the adversarial target of the first simulation.

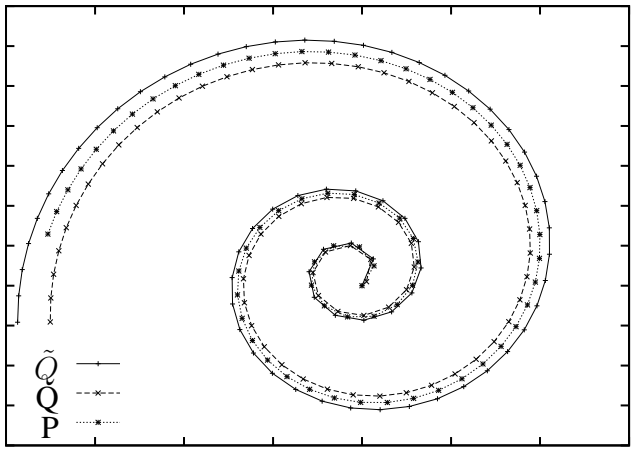


Figure 4.9: Paths taken by P, Q, and $\tilde{Q}$ take in a worst case simulation.

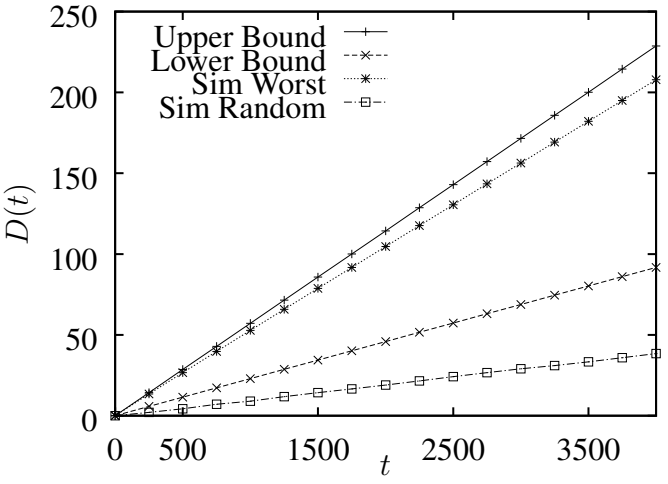


Figure 4.10: Growth in distance over time for simulations and proved bounds.

Finally, Figure 4.10 graphs the increase in distance over time for this simulation setup. The curves labeled upper and lower bounds show the theoretical limits established in Section 4.2. SIM WORST and SIM RANDOM show the results for the spiralling simulation, both with the worst-case target trajectory and the random target trajectory. We observe that in the worst case where $\tilde{Q}$ is always chosen at the maximum possible distance from Q, the distance growth is very close to our upper bound, but if $\tilde{Q}$ is chosen randomly, the distance increase is about half of the theoretical (adversarial) lower bound.

Conclusion

In this dissertation we studied several variants of geometric pursuit evasion, with a focus on finding combinatorial bounds on the number of pursuers that are sufficient and necessary for capturing an adversarial evader. We began with a model in which the pursuers are equipped with powerful sensors giving perfect knowledge of the evader's location in a polygonal environment. By following a natural progression in regards to both the sensing capabilities and complexity of environments considered, we are able to advance those results to more feasible sensing models and more realistic models of the real world such as polyhedral surfaces.

We began in Chapter 1 by considering the complete information pursuit evasion problem set in polygonal environments and gave two algorithms showing that three pursuers are always sufficient to capture an evader. Further, we proved this bound is tight by constructing an example where three pursuers are required. In Chapter 2, we extended these results to polyhedral surfaces and showed that 4 pursuers always suffice (upper bound), and that 3 are sometimes necessary (lower bound), for any polyhedral

surface with genus zero. Generalizing this bound to surfaces of genus g, we prove the sufficiency of $(4g + 4)$ pursuers. Finally, we show that 4 pursuers also suffice under the "weighted region" constraints, where the movement costs through different regions of the (genus zero) surface have (different) multiplicative weights. While open questions remain such as establishing a tight bound for polyhedral surfaces, and a lower bound for non-zero genus polyhedron, the primary question remains to find bounds when the location of the evader is not already known by the pursuers, which we address in the following chapter for polygonal environments.

In Chapter 3 we studied visibility-based pursuit evasion, where the pursuers only know the location of the evader when it is in direct line of sight. We begin my making only the minimalist assumption that pursuers and the evader have the same maximum speed. When the environment is a simply-connected (hole-free) polygon of n vertices, we show that $\Theta(n^{1/2})$ pursuers are both necessary and sufficient in the worst-case. When the environment is a polygon with holes, we prove a lower bound of $\Omega(n^{2/3})$ and an upper bound of $O(n^{5/6})$ pursuers, where n includes the vertices of the hole boundaries.

We then showed that with additional assumptions these bounds can be drastically improved. Namely, if the players movement speed is small compared to the features of the environment, we give a deterministic algorithm with a worst case upper bound of $O(\log n)$ pursuers for simply-connected n-gons and $O(\sqrt{h} + \log n)$ for polygons with

h holes. In addition to obtaining tight lower bounds, it remains a challenging problem to extend these results to polyhedral surfaces.

Finally, in Chapter 4 we further reduced the sensing capabilities of the pursuers by incorporating sensor noise. In particular, we adopt a simple but realistic model: the localization error is proportional to the true distance between the tracker and the target. We gave an algorithm for the tracker to following the target, and showed that this strategy is asymptotically optimal in the Euclidean plane, both with and without obstacles. An interesting direction for future work is to investigate the feasibility of extending our results on capture to this model in order to account for sensor noise present in real world applications.

Bibliography

[1] Openstreetmap gps trace. `http://www.openstreetmap.org/user/filot/traces/1657587`, 2014.

[2] M. Aigner and M. Fromme. A game of cops and robbers. *Discrete Applied Mathematics*, 8(1):1 – 12, 1984.

[3] L. Aleksandrov, M. Lanthier, A. Maheshwari, and J.-R. Sack. An ϵ-approximation algorithm for weighted shortest paths on polyhedral surfaces, 1998.

[4] S. Alexander, R. Bishop, and R. Ghrist. Pursuit and evasion in non-convex domains of arbitrary dimensions. In *Proc. of Robotics: Science and Systems*, Philadelphia, USA, August 2006.

[5] S. Alexander, R. Bishop, and R. Ghrist. Capture pursuit games on unbounded domains. *Enseign. Math. (2)*, 55(1-2):103–125, 2009.

[6] L. Alonso. "lion and man": Upper and lower bounds. *ORSA Journal on Computing*, 4(4):447 – 457, 1992.

[7] S. Alpern, R. Fokkink, R. Lindelauf, and G.-J. Olsder. The "princess and monster" game on an interval. *SIAM J. on Control and Optimization*, 47(3):1178–1190, 2008.

[8] D. Bhadauria and V. Isler. Capturing an evader in a polygonal environment with obstacles. In *Proceedings of the Joint Conference on Artificial Intelligence (IJCAI)*, pages 2054–2059, 2011.

[9] D. Bhadauria, K. Klein, V. Isler, and S. Suri. Capturing an evader in polygonal environments with obstacles: The full visibility case. *International Journal of Robotics Research*, 31(10):1176–1189, Sept. 2012.

[10] D. Bienstock and P. Seymour. Monotonicity in graph searching. *J. Algorithms*, 12(2):239–245, 1991.

[11] B. Bollobás, G. Kun, and I. Leader. Cops and robbers in a random graph. *Journal of Combinatorial Theory, Series B*, 103(2):226 – 236, 2013.

[12] S. D. Bopardikar, F. Bullo, and J. Hespanha. A cooperative homicidal chauffeur game. In *46th IEEE Conference on Decision and Control*, pages 4857 –4862, 2007.

[13] S. D. Bopardikar, F. Bullo, and J. P. Hespanha. On discrete-time pursuit-evasion games with sensing limitations. *IEEE Transactions on Robotics*, 24(6):1429–1439, 2008.

[14] J. Chen and Y. Han. Shortest paths on a polyhedron. In *Proc. of 6th Symposium on Computational Geometry*, pages 360–369, New York, NY, USA, 1990. ACM.

[15] L. P. Chew. Constrained delaunay triangulations. In *Proceedings of the third annual symposium on Computational geometry*, SCG '87, pages 215–222, New York, NY, USA, 1987. ACM.

[16] E. Chiniforooshan. A better bound for the cop number of general graphs. *Journal of Graph Theory*, 58(1):45–48, 2008.

[17] N. Dendris, L. Kirousis, and D. Thilikos. Fugitive-search games on graphs and related parameters. In E. Mayr, G. Schmidt, and G. Tinhofer, editors, *Graph-Theoretic Concepts in Computer Science*, volume 903 of *Lecture Notes in Computer Science*, pages 331–342. Springer Berlin / Heidelberg, 1995.

[18] H. N. Djidjev. On the problem of partitioning planar graphs. *SIAM Journal on Algebraic and Discrete Methods*, 3(2):229–240, 1982.

[19] J. Erickson and K. Whittlesey. Greedy optimal homotopy and homology generators. In *Proc. SODA*, pages 1038–1046, 2005.

[20] F. V. Fomin, P. A. Golovach, and J. Kratochvíl. On tractability of cops and robbers game. In *IFIP TCS*, pages 171–185, 2008.

[21] P. Frankl. Cops and robbers in graphs with large girth and cayley graphs. *Discrete Appl. Math.*, 17(3):301–305, June 1987.

[22] B. P. Gerkey, S. Thrun, and G. J. Geoffrey. Visibility-based pursuit-evasion with limited field of view. *I. J. Robotic Res.*, 25(4):299–315, 2006.

[23] A. S. Goldstein and E. M. Reingold. The complexity of pursuit on a graph. *Theoretical Computer Science*, 143(1):93 – 112, 1995.

[24] L. Guibas, J. Hershberger, D. Leven, M. Sharir, and R. Tarjan. Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica*, 2(1):209–233, 1987.

[25] L. J. Guibas, J.-C. Latombe, S. M. LaValle, D. Lin, and R. Motwani. Visibility-based pursuit-evasion in a polygonal environment. *IJCGA*, 9(5):471–494, 1999.

[26] L. Guilamo, B. Tovar, and S. LaValle. Pursuit-evasion in an unknown environment using gap navigation trees. In *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 4, pages 3456–3462 vol.4, 2004.

[27] B. Halpern. The robot and the rabbit–a pursuit problem. *The American Mathematical Monthly*, 76(2):140–145, 1969.

[28] J. Hershberger and S. Suri. "Vickrey pricing and shortest paths: What is an edge worth?". In *43th FOCS*, 2002.

[29] V. Isler, S. Kannan, and S. Khanna. Randomized pursuit-evasion in a polygonal environment. *Robotics, IEEE Transactions on*, 21(5):875 – 884, 2005.

[30] V. Isler, S. Kannan, and S. Khanna. Randomized pursuit-evasion with local visibility. *SIAM Journal on Discrete Mathematics*, 1:26–41, 2006.

[31] V. Isler and N. Karnad. The role of information in the cop-robber game. *TCS*, 399(3):179 – 190, 2008.

[32] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82:35–45, 1960.

[33] N. Karnad and V. Isler. Bearing-only pursuit. In *Proc. IEEE Int. Conf. on Robotics and Automation*, 2008.

[34] N. Karnad and V. Isler. Lion and man game in the presence of a circular obstacle. In *IROS'09*, pages 5045–5050, 2009.

[35] A. Kehagias, G. Hollinger, and S. Singh. A graph search algorithm for indoor pursuit/evasion. *Mathematical and Computer Modelling*, 50(910):1305 – 1317, 2009.

[36] M. Kirousis and C. H. Papadimitriou. Searching and pebbling. *Theor. Comput. Sci.*, 47:205–218, November 1986.

[37] K. Klein and S. Suri. Complete information pursuit evasion in polygonal environments. In *Proc. of 25th Conference on Artificial Intelligence*, pages 1120–1125, 2011.

[38] K. Klein and S. Suri. Catch me if you can: Pursuit and capture in polygonal environments with obstacles. In *Proc. of 26th Conference on Artificial Intelligence*, pages 2010–2016, 2012.

[39] K. Klein and S. Suri. Capture bounds for visibility-based pursuit evasion. In *Proc. of the 29th Symposium on Computational Geometry*, SoCG '13, pages 329–338, New York, NY, USA, 2013. ACM.

[40] K. Klein and S. Suri. Pursuit evasion on polyhedral surfaces. In *Proc. of 24th International Conference on Algorithms and Computation (ISAAC)*, 2013.

[41] A. Kolling, A. Kleiner, M. Lewis, and K. Sycara. Pursuit-evasion in 2.5d based on team-visibility. In *Proc. Int. Conf. on Intelligent Robots and Systems*, pages 4610 –4616, 2010.

[42] S. Kopparty and C. V. Ravishankar. A framework for pursuit evasion games in R^n. *Information Processing Letters*, 96:114–122, November 2005.

[43] A. Kovshov. The simple pursuit by a few objects on the multidimensional sphere. *Game Theory & Applications II*, pages 27–36, 1996.

[44] V. M. Kuntsevich and A. V. Kuntsevich. Analysis of the pursuit-evasion process for moving plants under uncertain observation errors dependent on states. In *Proc. of the 15th International Federation of Automatic Control*, 2002.

[45] A. LaPaugh. Recontamination does not help to search a graph. *J. ACM*, 40(2):224–245, 1993.

[46] S. M. Lavalle and J. E. Hinrichsen. Visibility-based pursuit-evasion: The case of curved environments. In *IEEE Transactions on Robotics and Automation*, pages 1677–1682, 1999.

[47] L. Liao, D. Fox, J. Hightower, H. Kautz, and D. Schulz. Voronoi tracking: Location estimation using sparse and noisy sensor data. In *Proc. of International Conference on Intelligent Robots and Systems (IROS*, 2003.

[48] J. E. Littlewood. *Littlewood's Miscellany*. Cambridge University Press, 1986.

[49] C. S. Mata and J. Mitchell. A new algorithm for computing shortest paths in weighted planar subdivisions. In *Proc. 13th Symposium on Computational Geometry*, pages 264–273, 1997.

[50] N. Megiddo, S. L. Hakimi, M. R. Garey, D. S. Johnson, and C. H. Papadimitriou. The complexity of searching a graph. *J. ACM*, 35:18–44, January 1988.

[51] A. Melikyan. Geometry of pursuit-evasion games on two-dimensional manifolds. *Annals of the International Society of Dynamic Games*, 9:173–194, 2007.

[52] J. S. B. Mitchell and C. H.Papadimitriou. The weighted region problem: finding shortest paths through a weighted planar subdivision. *Journal of the ACM*, 38(1):18–73, Jan. 1991.

[53] J. S. B. Mitchell, D. M. Mount, and C. H. Papadimitriou. The discrete geodesic problem. *SIAM Journal on Computing*, 16(4):647–668, Aug. 1987.

[54] E. Nardelli, G. Proietti, and P. Widmayer. Finding the most vital node of a shortest path. *TCS.*, 296(1):167–177, 2003.

[55] S. Neufeld and R. Nowakowski. A game of cops and robbers played on products of graphs. *Discrete Mathematics*, 186(13):253 – 268, 1998.

[56] N. Noori and V. Isler. Lion and man with visibility in monotone polygons. *International Journal of Robotics Research*, 2013.

[57] R. Nowakowski and P. Winkler. Vertex-to-vertex pursuit in a graph. *Discrete Mathematics*, 43(2-3):235 – 239, 1983.

[58] S.-M. Park, J.-H. Lee, and K.-Y. Chwa. Visibility-based pursuit-evasion in a polygonal region by a searcher. In *Proceedings of the International Colloquium on Automata, Languages and Programming (ICALP)*, pages 281–290. Springer-Verlag, 2001.

[59] T. D. Parsons. Pursuit-evasion in a graph. In Y. Alavi and D. R. Lick, editors, *Theory and Application of Graphs*, pages 426–441. Springer-Verlag, Berlin, 1976.

[60] A. Quilliot. Some results about pursuit games on metric spaces obtained through graph theory techniques. *European Journal of Combinatorics*, 7(1):55 – 66, 1986.

[61] G. Rote. Pursuit-evasion with imprecise target location. In *Proc. of 14th ACM-SIAM Symposium on Discrete algorithms*, SODA '03, pages 747–753, Philadelphia, PA, USA, 2003. Society for Industrial and Applied Mathematics.

[62] S. Sachs, S. Rajko, and S. M. LaValle. Visibility-based pursuit-evasion in an unknown planar environment. *International Journal of Robotics Research*, 23(1):3–26, 2004.

[63] B. S. W. Schroder. The copnumber of a graph is bounded by $\lfloor$ 3/2 genus(g)$\rfloor$ +3. In *Categorical Perspectives Proc. of the Conference in Honor of George Streckers 60th Birthday*, pages 243–263, 2001.

[64] A. Scott and B. Sudakov. A bound for the cops and robbers problem. *SIAM Journal on Discrete Mathematics*, 25(3):1438–1442, 2011.

[65] P. D. Seymour and R. Thomas. Graph searching and a min-max theorem for tree-width. *Journal of Combinatorial Theory Series B*, 58(1):22–33, May 1993.

[66] J. Sgall. Solution of david gale's lion and man problem. *Theor. Comput. Sci.*, 259(1-2):663–670, 2001.

[67] X. Sheng, Y.-H. Hu, and P. Ramanathan. Distributed particle filter with GMM approximation for multiple targets localization and tracking in wireless sensor network. In *Proc. of the 4th International Symposium on Information Processing in Sensor Networks*, 2005.

[68] J. R. Shewchuk. Lecture notes on delaunay mesh generation. 1999.

[69] N. Stiffler and J. O'Kane. Shortest paths for visibility-based pursuit-evasion. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 3997–4002, 2012.

[70] I. Suzuki and M. Yamashita. Searching for a mobile intruder in a polygonal region. *SIAM J. Comput.*, 21:863–888, October 1992.

[71] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust monte carlo localization for mobile robots. *Artif. Intell.*, 128(1-2):99–141, 2001.

[72] J. Thunberg and P. Ögren. A mixed integer linear programming approach to pursuit evasion problems with optional connectivity constraints. *Autonomous Robots*, 31(4):333–343, Nov. 2011.

[73] B. Tovar and S. M. LaValle. Visibility-based pursuit-evasion with bounded speed. *International Journal of Robotics Research*, 32(13), 2013.

[74] M. Vieira, R. Govindan, and G. Sukhatme. Scalable and practical pursuit-evasion with networked robots. *Intelligent Service Robotics*, 2(4):247–263, 2009.

[75] J. Yen. Finding the k shortest loopless paths in a network. *Management Science*, 17(11):pp. 712–716, 1971.