REACHABILITY-BASED CONTROL FOR HUMAN AND AUTONOMOUS AGENTS IN ADVERSARIAL GAMES

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF AERONAUTICS AND ASTRONAUTICS
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Haomiao Huang

June 2012

This dissertation is online at: http://purl.stanford.edu/tv028wc3847

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Claire Tomlin, Primary Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Stephen Rock

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Daniel Klein

Approved for the Stanford University Committee on Graduate Studies.

Patricia J. Gumport, Vice Provost Graduate Education

This signature page was generated electronically upon submission of this dissertation in electronic format. An original signed hard copy of the signature page is on file in University Archives.

Abstract

Solutions to multi-agent adversarial games are useful in many applications where control actions must be taken in the presence of opposing agents, moving obstacles, or other external disturbances. These applications range from collision avoidance for autonomous cars and air traffic control to team coordination in the military and police. In addition to controlling robotic or autonomous agents, solutions to these games can play an essential role in providing guidance to humans, who may be participating either in supervisory roles or as agents themselves. The critical challenges to obtaining such solutions are the need to robustly account for adversarial actions in scenarios with many agents, and to do so in a computationally tractable fashion. In addition, to be helpful to humans the solutions must be presented so that they fit within the decision-making framework of the humans.

This thesis develops solution methods for two types of related multi-agent adversarial games: reach-avoid games, where an attacking agent attempts to reach a target while avoiding other, defending agents, and pursuit-evasion games, where an evading agent attempts to avoid capture by pursuing agents. First, a Hamilton-Jacobi-Isaacs reachability formulation is used to compute guaranteed, optimal solutions to a reach-avoid game involving a single attacker and defender. Then an open-loop reachability formulation is introduced to allow solutions to be found in real time for games involving multiple agents on a side. An efficient method for computing open-loop safe-reachable sets is introduced and used to compute solutions for an attacking agent in reach-avoid games with multiple defenders, and for pursuing agents in pursuit-evasion games involving a single evader and multiple pursuers.

These reachability-based game solutions are used to develop intuitive, automated

tools to guide human agents in adversarial games as well as to control robotic assets like unmanned aerial vehicles (UAVs). The tools are validated using a novel testbed incorporating smartphones and UAVs in a system with human agents. These experiments demonstrate the use of the tools in realistic scenarios involving varied terrain and noisy communications. In these experiments, the human agents not only directly utilize the computed solutions, but are also able to modify the plans using the associated reachability information when communications failure and other factors render the assumptions made by the automation invalid.

Acknowledgements

The road to a PhD is long and winding, and I would never have reached this point without the support and encouragement of the many people who have helped me along the way.

I would like to express my deepest and sincerest gratitude to my advisor, Professor Claire Tomlin. She's been an incredible mentor, giving me enough independence to find my own way, but always providing guidance and support to keep me from getting lost in the wilderness. She's been instrumental not only in my research progress, but also in shaping me as a researcher. Finally, her attention to detail in all of the many non-academic aspects of academic life, from her impeccable courtesy to her willingness to engage the bureaucracy on behalf of her students, stand as an example of the kind of mentor I aspire to be.

I'd also like to thank the members of my thesis committee, Professors Steven Rock, Dan Klein, Chris Gerdes, and Leonidas Guibas. Their guidance and mentorship was invaluable in shaping the course of my PhD. In particular, I'd like to thank the members of my reading committee, Professor Rock and Professor Klein. My conversations with Professor Rock have always been a great inspiration, and have served me as constant reminders of the relationship between research and the real world. I also wish to thank Professor Klein for giving me the opportunity to work with him and his students during the Starcraft AI competition, and for giving me one of the clearest and most cogent explanations I've ever heard of what a PhD thesis is and represents.

This thesis would also not have been possible without the support of the staff members of the Department of Aeronautics and Astronautics at Stanford, and in the

Electrical Engineering and Computer Sciences Department at Berkeley. I want to thank Sherann Ellsworth and Dana Parga, who always took care of us and made it possible for us to do the research that we do. I'd also like to thank Jay Subramanian for making sure that we wayward Berkeley expats weren't lost when we left Stanford, and I'd like to thank Ralph Levine, Robin Murphy, and Lynn Kaiser for all of their help as well. On the other side of the Bay, I want to thank Claudia Martinez-Schwarz, Alice Wong, Jessica Gamble, Gary Given, and Cindy McKinnon for making the transition to Berkeley go so smoothly.

I had the privilege of working with some extraordinarily talented colleagues in the Hybrid Systems Lab at Stanford and Berkeley. They provided guidance, mentorship, good humor, and friendship, and without them I would never have gotten through the PhD. I thank Gabe Hoffman and Steve Waslander for all that they've taught me, about UAVs and about being grad students, and for introducing me to the lab. I want to thank Kaushik Roy, for his sense of fun, and for some nuggets of wisdom that I will always remember. I want to thank the other students who worked with me on STARMAC and collaborated with me on many research projects: Jeremy Gillula, Pat Bouffard, Eugene Li, Vijay Pradeep, Tony Mercer, and most especially Mike Vitus, whose work ethic and attention to detail amazes and inspires me. This thesis would not have been possible without collaborators and colleagues: Wei Zhang, Dŭsan Stipanović, Selina Pan, and Jerry Ding, who helped me get into adversarial games in the first place, Ryo Takei, whom I owe for the open-loop formulations, and Zhengyuan Zhou, who has both helped me and challenged me to produce better research. I also want to thank all of the other members of the lab, past and present, who have worked with me, helped me, and hung out with me over the years: Sleiman Itani, Robin Raffard, Young-hwan Chang, Insoon Yang, Anil Aswani, and Maryam Kamgarpour.

I also want to extend a special thank you to the undergraduates I've had the opportunity to work with, whose assistance made the experimental results in this thesis possible: Zhengyuan, Steven Xu, Andrew Sy, and Scott Hoag, for building the capture-the-flag game and then conducting all of the experiments with me, and Sridatta Thatipamala, for building the Contrails air traffic control game with me.

Of course, none of this would have been possible without the support, love, and caring of my family: my mother, whose wisdom has been my truest guide in life, my father, who set me on this path so many years ago when he first took me to look at satellites in the night sky, and my brother, who has over the course of his 22 years gone from infant, to friend, and now fellow researcher as well.

Finally, I want to express my most heartfelt, sincere, and loving thanks to my fiancée, Lydia Thé. She's been here for me during the highest highs and the most grueling lows, and has brought me more happiness than anyone could ever wish for.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This thesis presents a set of tools for efficiently computing control actions for agents in multi-agent adversarial games. These tools, based on reachability analysis, are designed with an emphasis on control for human agents. Specifically, they are envisioned for use in contexts where human agents are supported by autonomous assets, e.g. robots or automated decision aids. The human agents may follow the computed solutions directly, but the reachability analysis also provides contextual information about the limits of the computed solutions. This allows human agents the flexibility to incorporate the solutions within larger problems not directly addressed by the automated tools, or to modify them with information not available to the automation if necessary. This work presents algorithmic foundations of these tools along with experimental results demonstrating their use by human and autonomous agents.

1.1 Overview

Automated systems that sense, reason about, and act upon the physical world are rapidly becoming ubiquitous. In addition to traditional single-agent systems like mobile robots and unmanned aerial vehicles (UAVs), these systems are expanding into a broad range of multi-agent and embedded contexts. They may be used in air traffic control to direct large groups of aircraft, in energy grids to manage networks of energy consumers and suppliers, and in command centers to coordinate security

and defense assets.

Paradoxically, as these systems become more powerful and autonomous they demand greater interaction with humans. Complex, potentially safety-critical tasks cannot be entirely automated, and humans will be involved as operators, users, and in particular, as components of the system. For example, an air traffic control command center must manage human pilots as well as unmanned aircraft, and human police officers and soldiers will work alongside robotic assets. In these cases, human performance can be improved by automated guidance, but to be useful the guidance must fit into the human decision-making process, and overall system performance should degrade gracefully if the automation fails.

A number of important applications that benefit from automated guidance fall within the context of multi-agent adversarial games. A multi-agent adversarial game pits a group of one or more agents against another group with competing objectives. Common examples of multi-agent adversarial games are pursuit-evasion games, where one group of agents is attempting to capture another, and reach-avoid games, where one or more agents on one team attempt to reach a target while another team attempts to intercept them. In addition to obvious applications in security and defense, adversarial games are important tools for safety, in aircraft flight and driving for example. Uncertainty in other vehicles' movements and external forces like wind can be treated as adversarial disturbances acting against a controlled vehicle. Solutions from these games can then be used to design systems for automated collision and obstacle avoidance, or for warning operators of potential danger.

Automated solutions can simplify complex games for humans, as the interplay of multiple, moving agents can tax the cognitive abilities of even experienced participants. On the other hand, multi-agent games are computationally complex and difficult to solve even with heavy computing power. The algorithms depend on simplifications and assumptions that are undermined by noisy sensing, equipment failure, and the sheer complexity of real-world environments. The goal is not to design automation that will be 100% correct, which is impossible, but to design tools that provide human agents with good assistance and recommendations, and clearly indicate when and where these recommendations are useful and correct. This allows the

human to optimize the use of the automation tools.

The research in this dissertation addresses the challenge presented above with novel algorithms for several multi-agent adversarial games developed using reachability analysis. In addition to providing control strategies, these reachability-based methods are used to generate easily understood visual displays that help human users compensate for the limitations of the automated solutions. Tools designed using these algorithms are validated experimentally in adversarial games played with human agents on the **BE**rkeley **A**utonomy and **R**obotics in **CA**pture-the-flag **T**estbed (BEARCAT).

1.2 Multi-Agent Games of Interest

Two related multi-agent adversarial games are considered in this thesis: pursuit-evasion games and reach-avoid games. These games are linked by the common thread that one group of agents is attempting to avoid the other, but differ in the goals of the avoiding agents. The theoretical results in this thesis are concerned with fully observed games where the positions of the agents are known to each other, although application of this work to games with noisy or missing observations are seen and discussed in experimental results.

1. **Pursuit-Evasion Games**

   A pursuit-evasion game is a game in which one or more agents, denoted pursuers, are attempting to capture another set of agents, denoted evaders. An evader is considered to be captured if it enters a pre-defined capture set around each pursuer, typically a circle centered around the pursuer. This thesis considers pursuit-evasion games between multiple pursuers and a single evader, where the pursuers' objective is to capture the evader as quickly as possible, and the evader's objective is to remain un-captured for as long as possible. Analysis of pursuit-evasion games focuses on answering the following questions: can the evader ever be captured, and what are the pursuer and evader strategies to minimize and maximize the capture time, respectively?

2. **Reach-Avoid Games**

   A reach-avoid game is one in which one or more agents, denoted attackers, are attempting to reach one or more target sets in the state space as quickly as possible while the other agents, denoted defenders, are attempting to prevent the attackers from reaching the targets. This thesis considers reach-avoid games between multiple defenders and a single attacker, where the attacker's objective is to arrive at the targets as quickly as possible, and the defenders' objective is to either capture the attacker or prevent the attacker from arriving at the targets by threat of capture. Analysis of reach-avoid games focuses on answering the following questions: can the attacker reach the targets in finite time, and what are the attacker and defender strategies to minimize and maximize the final arrival time, respectively?

1.3 Motivating Applications

Many robotics and controls applications require coordination of agents moving to achieve a goal in the presence of external disturbances, adversarial opposition, or simply moving obstacles that must be avoided. These applications include piloting an aircraft, driving a ground vehicle, air traffic control, and path planning for teams of human and robotic agents. Analyzing certain aspects of these applications in the context of a multi-agent adversarial game formulation can provide solutions to a number of relevant problems.

1.3.1 Vehicle Collision Avoidance

Self-driving cars are being deployed on the highways [1], and autonomous aircraft will soon be incorporated into the civil airspace [2]. As these vehicles move from the laboratory to commercial use, they will require collision avoidance algorithms to safely operate with other vehicles on the road and in the air. In addition, with cheaper and more advanced sensing, the same technology can be deployed in human controlled vehicles to decrease the likelihood of accidents. The challenge for an operator, whether

human or autonomous, is that control actions must be considered against all possible actions of other vehicles.

For safety-critical situations, a worst-case approach can be taken by treating other vehicles as adversarial and attempting to collide with the controlled vehicle. The other vehicles are now cast as pursuers in a pursuit-evasion game, and the collision avoidance task becomes that of finding a successful evasion strategy. The advantage of this worst-case approach is that a successful evasion strategy found in these circumstances is naturally conservative; it would avoid the other vehicles even if they were *trying* to collide. Solutions to multi-agent pursuit-evasion games can then be directly applied as collision avoidance strategies.

1.3.2 Air Traffic Control

Air traffic control is another area seeing rapid growth in automation where human-friendly solutions to multi-agent adversarial games are beneficial. Demand for air travel is growing steadily, increasing air traffic density around important airports. Maintaining safety in the face of additional usage is one source of pressure for additional automation, and the promise of efficiencies gained through the use of emerging technologies such as GPS is another.

Pursuit-evasion games related to collision-avoidance as discussed above are relevant to the tactical level of separation assurance, when aircraft are a few miles apart. Additionally, air traffic control may benefit from solutions to reach-avoid games at longer time horizons, for example when planning the arrival of aircraft at airports while avoiding adverse weather patterns. A conservative approach treating the motion of these weather patterns as adversarial will generate solution trajectories that guarantee the safe arrival of the aircraft.

A key aspect of designing these tools is the incorporation of human agents, as automatically generated plans must be implemented by human controllers and pilots. Safety considerations demand that human controllers maintain supervisory control of the aircraft routing process, and human pilots are operating most of the aircraft. Although the automation can generate superior solutions under nominal operating

conditions, there will be times when the computational tools fail, for example when aircraft equipment malfunctions in a way not anticipated by the designers of the automation. In these circumstances, human controllers are only able to take the correct action if they understand when the automation solutions are valid and when they must be ignored. Another consideration is that the automation may also fail entirely, for example due to power failure, requiring human controllers to take over. This possibility requires the human controllers to understand the automated solutions they are implementing, in the sense of knowing why certain actions are taken, so that they may pick up where the automation left off.

1.3.3 Human and Human-Robot Teams

Tasks such as security and search-and-rescue often require teams of agents to act collectively in adversarial situations. The utility of solutions to pursuit-evasion and reach-avoid games is self-evident in security and defense applications, where police or military assets might be deployed to capture a criminal or prevent a threat from reaching a critical target. Automated tools can be developed using game solutions that guide police officers and soldiers in completing these tasks.

However, the multi-agent planning tools that can result from game solutions are also relevant in applications that are not strictly adversarial. In teams where humans and robots must work together, for example with robotic construction equipment or automated farming vehicles, clear and timely communication of intent between human and robotic elements is not always possible. Disturbances such as actuator noise or sensing error may be modeled as causing one set or another of agents to act adversarially, and reach-avoid solutions can be used in these circumstances to plan safe paths for robots to avoid humans, and vice-versa.

1.4 Capture-the-Flag Test Environment

Evaluating algorithms for human agents in adversarial games requires a realistic environment where these algorithms may be implemented and tested. Much of the

research conducted in this thesis is evaluated using the game of capture-the-flag, played with human and autonomous agents supported by computing assets.

Capture-the-flag is a two-sided game played by teams of mobile agents constrained to remain within a bounded game area, divided into two regions allocated to each side. Each team owns a flag located in their territory that can be captured by an opposing agent. A typical configuration for the game is illustrated in Figure 1.1, showing the two sides and their flags. While inside the opposing team's territory, an agent can be intercepted if tagged by an opponent. The objective of the game is to capture the opponent's flag and return to safety while protecting one's own flag from the enemy. Each flag is also typically surrounded by a forbidden region to prevent a defending agent from simply staying at the flag location and making flag capture impossible. The game as typically played also incorporates jails (labeled gaols in the illustration), where captured agents are held until freed by another agent on their team.

A capture-the-flag game offers an ideal research arena to evaluate adversarial game strategies and their use by human agents. Many pursuit-evasion and reach-avoid scenarios occur naturally during the game, while important real-world considerations such as limited sensing, communications failure, and rough terrain are also present. The capture-the-flag game allows the algorithms and strategies produced in the research to be evaluated in a real-world context. On the other hand, the use of a simplified game scenario eliminates many complexities that are tangential to the research questions, such as maintenance of vehicles or legal restrictions involved in working with air traffic controllers. This allows easy and repeated experimentation, leading to improved research results.

1.5 Related Work in Adversarial Games

The utility of practical solutions to multi-agent adversarial games has long been apparent, and there has been significant research into this topic. Given the complexity of these games, there has been no "one-size fits all" approach. Instead, the approaches

Figure 1.1: **An illustration depicting a typical game of capture-the-flag, showing the relevant game regions and agents (image courtesy of** `www.goforyourlife.victoria.gov.au`**).**

in the literature trade off between optimality, guarantees, game complexity, and computational speed. Optimality in this context is with respect to the time required to complete the game, either to achieve capture in pursuit-evasion or to reach a target in reach-avoid. Guarantees refer to the ability to find a solution when one exists, for example for a pursuer in pursuit-evasion to always capture the evader when starting from a configuration where capture is possible no matter what the evader does. Optimal solutions are guaranteed, but guaranteed solutions may not be optimal. In addition to the number of agents, the complexity of the game refers to the game domain in terms of obstacles, as well as the dynamics of the agents.

This section presents an overview of some relevant past work on pursuit-evasion and reach-avoid games, in particular exploring the specific trade-offs associated with each approach, and discusses the relationship of the work in this thesis to the larger

context.

1.5.1 Geometric and Analytic Solutions

For certain games and game configurations, it is possible to construct strategies for the agents geometrically. Alonso et al. [3] analytically determine upper and lower bounds on capture time for the "Lion and the Man" problem, where a pursuer and evader with equal speeds play in a bounded circular game domain. Kopparty and Ravishankar [4] give a pursuit strategy with bounds on capture time for games with multiple pursuers and a single evader in open domains or with a small set of half-space constraints. In this case, capture is assured if the evader is contained in the convex hull of the pursuers. Alexander et al. showed that pure pursuit (where the pursuit strategy is to instantaneously minimize the geodesic distance to the evader) guarantees capture in a number of different geometries [5].

For reach-avoid games, a class of methods has been proposed for safe motion-planning in the presence of moving obstacles by computing the future set of states an obstacle may occupy, given the dynamics of the controlled agent and the obstacles. These forward reachable sets are then treated as obstacles in the joint state-time space, and paths are planned which avoid these states [6, 7, 8]. They are well suited for scenarios in which the obstacle motion can be unpredictable or even adversarial, or where the application requirements place hard constraints on safety, in which case one needs to account for the worst-case possibility that the disturbances may actively attempt to collide with the agent.

In general, such geometric methods are computationally efficient in generating control strategies, but are limited to relatively simple game configurations without complex static obstacle configurations. Inhomogeneous speed constraints, such as that occasioned by varying terrain, also present a challenge. In addition, guarantees may be found for certain game types and configurations, but optimality cannot usually be shown.

1.5.2 Solutions to the Hamilton-Jacobi-Isaacs Equation

The most complete approach to either pursuit-evasion and reach-avoid games is to formulate the problem as a differential game and then solve the appropriate related Hamilton-Jacobi-Isaacs(HJI) partial-differential equation(PDE) [9, 10, 11, 12]. Each game is defined by an appropriate value function, and the opposing agents either seek to minimize or maximize this value. For example, for a pursuit-evasion game the value would be the capture time, with the evader attempting to maximize the time and pursuers attempting to minimize. For a reach-avoid game, the value is the arrival time of the attacker at the target, which the attacker seeks to minimize and the defenders maximize. This value can be computed via a related HJI equation, with appropriate boundary conditions. Solutions are typically found either using the method of characteristics or via numerical approximation on grids.

The method of characteristics proceeds in the following manner. For a given terminal state, the value is known, e.g. the time-to-capture for a pursuit-evasion game is zero at the moment of capture. Given a particular terminal state, the HJI PDE simplifies to an ordinary differential equation (ODE), which may be solved by integrating backwards in time. This gives a single optimal trajectory for the system with a known terminal condition. Isaacs presents particular solutions for a number of specific games via this method [9], and more games are covered in a similar manner by Başar and Olsder [11].

The characteristic solutions are useful in understanding optimal solutions qualitatively, but can be of limited utility in extracting control inputs. The method is limited by the necessity to integrate backward from a terminal point, which can make it difficult to generate strategies when only the current, initial state of the agents is known. In some cases when the game configuration is simple, for example agents moving in unconstrained free-space, the terminal conditions can be directly related to initial conditions via observation [9]. State constraints such as obstacles are also difficult to handle, and the ODE solutions break down in the presence of discontinuities in the value function. These typically appear in cases where symmetry presents two equally valid and optimal options that are available to an agent, for example when an agent must move around an obstacle and either moving to the left or right

result in the same arrival time.

The HJI PDE can also be globally approximated on a grid representing the state space by computing the viscosity solution to the HJI PDE at every node in the grid [12, 13]. Control inputs can then be extracted through numerical differentiation of the value function. This approach has been successfully applied to a number of applications, including a pursuit-evasion game with non-holonomic agents for air traffic collision avoidance [12] and a reach-avoid game modeling planning for UAV refueling [14]. This approach is flexible and powerful, and is used in some of the work presented in this thesis.

HJI computation on grids is limited by the size of the problem: computing solutions to HJI equations is computationally infeasible for large problems, as the grid required for approximating the value function grows exponentially as the size of the state space increases. Even smaller problems such as 2-agent, kinematic games, typically cannot be solved in real time, requiring pre-computation. This computation is performed offline for the entire state space, and the solutions stored for later use online. Numerical issues may also introduce errors into the PDE solutions.

1.5.3 Discrete Games

In addition to games played in continuous time on continuous spaces, there has also been research into discrete games played on graphs, with the agents taking turns to move. This work on pursuit-evasion games on graphs has also led to results for games in continuous spaces, particularly for a class of games known as visibility pursuit-evasion.

In discrete pursuit-evasion, the pursuers and evader are limited to the nodes of a graph, and take turns moving a number of links each turn. Capture is achieved when a pursuer occupies the same node as the evader. Parsons was one of the first to formalize a pursuit-evasion game played in discrete time on graphs, and characterized the number of pursuers required to capture an evader in a tree [15]. Aigner and Fromme later showed that in general finite, planar graphs, three pursuers are sufficient and necessary to capture an evader [16].

The graph results underlie solutions to the related problem of visibility pursuit-evasion in continuous domains, where a group of searchers is attempting to bring an evader into their field of view [17, 18, 19]. Typically, the sensing is modeled as vision with unbounded range (in effect an infinite capture radius when not occluded). This allows these continuous visibility games to be transformed into discrete pursuit-evasion games, and solved accordingly.

The main difficulty with regards to discrete games is computational complexity. Graphs may be characterized in that the number of pursuers required for a particular game domain can be found, but implementable control strategies can be difficult to synthesize. For either pursuit-evasion on graphs or visibility pursuit-evasion, solution strategies are usually found by searching over a large set of discrete actions. The exponential branching that this requires limits the size of problems that can be practically solved, especially when multiple agents are involved on a side.

1.5.4 Model-Predictive Control

When the computation required to find optimal or guaranteed strategies is intractable due to either problem complexity or the need for real time solutions, a form of model-predictive control (MPC) is often employed. In an MPC formulation, an optimization problem is solved over the control actions of one side while using a model to predict opponent actions. This solution is implemented for a short time period, and the optimization is then re-solved using the new agent states. Feedback via this re-computation is used to correct for errors in the prediction model.

This strategy has been used for a number of games, for example in complex pursuit-evasion games such as air combat, where the roles of the agents may change over time. Sprinkle et al. used nonlinear model-predictive control to generate control inputs for an unmanned aerial vehicle [20], while McGrew et al. used approximate dynamic programming for a similar application [21].

In the reach-avoid context, a defensive strategy for intercepting multiple attackers with multiple defenders was developed by Earl and D'Andrea using mixed-integer linear programming, with the assumption that attackers proceed toward the target

in straight lines [22]. Another method to solve for optimal defender control actions was proposed by Chasparis and Shamma using linear programming under an assumed attacker strategy in the form of a linear feedback law [23].

In general, MPC approaches work best when the predictive model used is a good approximation of the actual strategy of the opponent. When this is true, control inputs can be quickly and efficiently generated for the controlled agents using standard optimization tools. However, proofs of optimality and guarantees for the solutions are usually not available.

1.5.5 Summary and Discussion

The overview above highlights the trade-offs between optimality, guarantees, problem complexity, and computational speed that must be made in control design in adversarial games. Optimal, guaranteed solutions come at the cost of either simplified problems, slow computation, or both, while real time computation usually requires simple problems or loss of optimality and completeness.

The research in this thesis is no exception to this rule. As the ultimate goal of this work is to provide useful, practical solutions, optimality may be sacrificed for computationally efficient solutions that nonetheless possess guarantees with respect to capture or arrival at a target. For the pursuit-evasion game, this means finding strategies that guarantee capture or escape, and for reach-avoid games this means strategies that either guarantee reaching a target or successfully defending a target. The work presented in this thesis attempts to address the following limitations in the current literature:

1. Existing methods for reach-avoid games do not give guaranteed solutions for domains with arbitrary obstacle configurations, especially in games with many agents.

2. Current methods can not be used to effectively coordinate large groups of agents in bounded domains with guaranteed results.

3. Most existing methods do not provide contextual information, which is important for human agents (to be explored in the following section).

4. Many techniques that work in real time require knowing or making assumptions about the opponent control strategies.

In addition, although the strategies presented are for scenarios where the state of all agents is known, these strategies still have utility when sensing is limited, and this is also explored.

1.6 Background on Humans as Decision-Makers

Designing control algorithms that guide human agents can be very different from designing algorithms for robotic or computerized agents. In many real-world environments, human intuition and perception are still far superior to anything a technological approach can produce. The goal of the research in this work is to augment, and not replace human decision-making. This section presents some motivating background on human decision-making, and the reachability approach used in this thesis for guiding humans. This is not meant to be an exhaustive overview of the literature on human decision-making, but to present some context for the methods presented in this thesis.

1.6.1 Background on Human Decision-Making

A review of the relevant literature suggests that human decision-making is decompositional and hierarchical: complex problems are decomposed into or related to simple problems with known solutions, and these known solutions are then modified and composed into solutions for the larger problem. For example, in aircraft maneuvering human pilots will fly using maneuver primitives, which are composed together into complex trajectories to meet mission requirements [24, 25]. In air traffic control, studies have shown that experienced controllers managing a single large sector composed of many aircraft will group aircraft into multiple “events” involving subsets

of aircraft [26]. The identification of these events then suggests particular strategies which are used to generate specific commands to aircraft. Military scenarios are often decomposed in terms of hierarchical objectives and various strategies and tactics to achieve those objectives [27].

Moreover, the human brain is itself deeply hierarchical, and this framework may underlie much of human sensing, cognition, and motor control [28]. For example, visual input consisting of facial features are aggregated to identify faces, while time sequences of auditory input are aggregated to identify voices. Voices and faces are then aggregated at higher levels to identify specific people, and so on [29].

In order to work successfully in a complex situation, a human decision maker needs to understand how various decomposed components fit into the larger plan. In a maneuver sequence it is important make sure that each maneuver terminates in a way that the next desired maneuver can start, and in air traffic control it is vital that corrections made to avoid a conflict for one aircraft do not lead to other conflicts later. This understanding of the larger context is known as situational awareness, sometimes abbreviated as SA, and has been found to be an important component of human decision-making. Loss of situational awareness is one of the largest factors in errors in systems where humans are working with automation [30].

1.6.2 Reachability for Decision-Making in Adversarial Games

A core concept used in the work in this thesis is the idea of *reachability*, and the computation of *reachable sets*. The concept of reachability is the following: suppose a system is in some configuration x_1, and there is some other configuration x_2. x_2 is said to be *reachable* from x_1 if there exists a control input that drives the system from x_1 to x_2 in finite time, regardless of adversarial disturbance or opposition. The reachable set associated with x_2 is the set of all states that can be controlled to x_2, regardless of the actions of the adversarial disturbance.

A simple, intuitive example of reachability can be seen in the game of capture-the-flag, where it is necessary for an attacker who has reached the opponent's flag to know whether the flag can be safely retrieved without being intercepted by a defender. The

concept of reachability has utility in many contexts beyond that of adversarial games, as the adversarial disturbance need not be opposing agents. For example, reachability has been used to derive landing conditions for an aircraft where an aircraft may land safely regardless of external disturbances such as wind [31].

Reachability and reachable sets are natural, intuitive tools to use with human agents and hierarchical control frameworks. Reachable sets, in particular, have several uses for a human agent. First, the reachable set allows the human to know instantly whether a particular goal is achievable, which is useful when there are multiple possible goals. In Oishi et al. reachable sets were used to help a pilot make a choice between landing and several methods of aborting the landing [31]. If the required conditions for a particular goal are met, the automated strategy may be used directly, but reachable sets are also valuable in cases in which the automated strategy cannot be followed. In these cases, the reachable sets allow the human to determine where and when the automation *can* be followed for a guaranteed result. This gives the human the option of attempting to control the system into this set, knowing that the once in the reachable set the goal is achievable by directly following the automated solutions. The sets can also be used to create intuitive visualizations for the human agent, and in general are useful methods for helping the human understand the relationship of automatically computed solutions to the larger context. These concepts are explored in more detail in the experimental sections associated with the solution strategies discussed in this thesis.

1.7 Contributions

This research combines the insights discussed in Section 1.6 with algorithmic contributions to computing solutions for multi-agent adversarial games, creating solutions that can be incorporated within the decision-making of human agents. The contributions of this work can be categorized into three general areas: algorithmic solutions to pursuit-evasion and reach-avoid games, experimental validation of reachability-based tools for guiding human agents, and novel applications of reachability to UAV control.

1. **Solutions to reach-avoid games**

Three related adversarial games are solved using reachability methods. First, a version of capture-the-flag involving two agents is formulated as a reach-avoid game and solved using Hamilton-Jacobi-Isaacs reachability, giving guaranteed solution strategies in bounded domains with arbitrary obstacle configurations. Next, an open-loop reach-avoid game involving multiple defending agents is formulated and then solved using a novel, modified fast-marching algorithm, giving guaranteed solutions in real time for attacking agents in domains with arbitrary obstacles. Finally, a decentralized, real time algorithm is developed for cooperative pursuit of a single evader by multiple pursuers in bounded, simply-connected planar domains, with proof of guaranteed capture when the domain is convex. In all of these results, no modeling or assumptions are made about opponent control strategies. The work on Hamilton-Jacobi reachability was first presented in [32]. The open-loop reach-avoid formulation has been presented in [33] and [34], and the cooperative pursuit algorithm was presented in [35].

2. **Experimental validation of tools for human agents**
   Tools for human agents are designed based on the solution algorithms and evaluated through experiments with the **BE**rkeley **A**utonomy and **R**obotics in **CA**pture-the-flag **T**estbed (BEARCAT). BEARCAT is a novel testbed for research into automated assistance for human agents in adversarial games, consisting of smartphones connected to off-board computation and quadrotor UAVs. The development of BEARCAT is presented, along with experiments showing the use of the tools in realistic scenarios. The contextual information from reachability allows human agents to optimize the use of the automated solutions while mitigating the impact of imperfect sensing and simplified problem formulations. The development, control, and aerodynamics of quadrotor UAVs related to this work was first presented in [36].

3. **Reachability for UAV control**
   A method is developed for designing provably safe maneuver sequences for a

quadrotor UAV, and experimental results are presented demonstrating the robustness of a designed maneuver sequence in the presence of external disturbances. This work was first presented in [37]. The reachability solution to capture-the-flag with known agent states is used to guide UAV search when the location of an opposing agent is unknown, with an experimental demonstration of the search strategy in a game of capture-the-flag.

Chapter 2

Experimental Platform: BEARCAT

Experimental validation for the algorithms discussed in this thesis is performed using the **BE**rkeley **A**utonomy and **R**obotics in **CA**pture-the-flag **T**estbed (BEARCAT). The primary purpose of BEARCAT is to provide a flexible testbed where human agents can participate in an adversarial game such as capture-the-flag, while receiving guidance from computational tools and working with autonomous agents such as UAVs. The testbed allows experiments to be performed by playing reach-avoid and pursuit-evasion games on the Berkeley campus. The major components of BEARCAT are illustrated in Figure 2.1, consisting of HTC Incredible smartphones [38], laptop computers, and quadrotor UAVs. The different components are networked via wireless communications, providing a complete system for both tracking agents and providing them with access to resources such as computed game solutions and UAV sensing.

Smartphones

HTC Incredible smartphones are used in BEARCAT as user interfaces and to provide tracking resources to the human agents, essentially serving as each agent's gateway to the networked system. The smartphones run the Android operating system [39], and are equipped with GPS, 3-axis accelerometers, 3-axis magnetic compass, and full internet connectivity via the 3G data network as well as an interactive touch-screen.

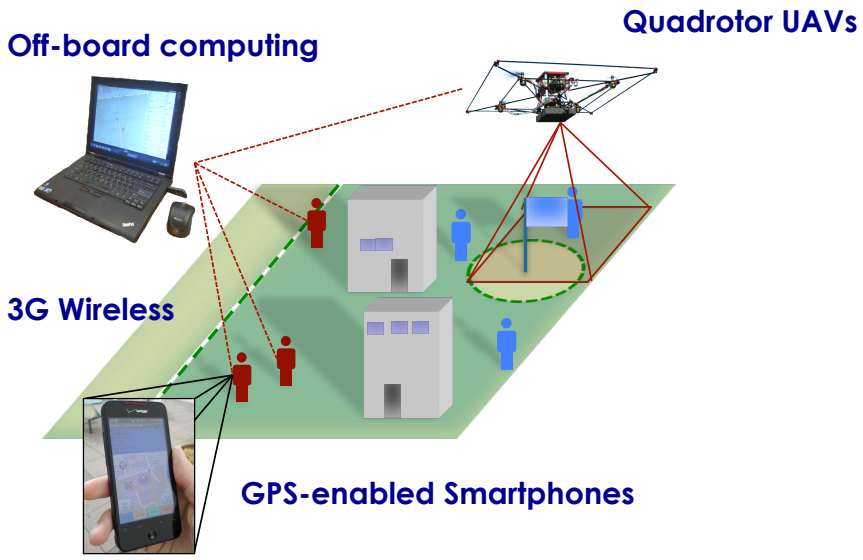


Figure 2.1: **Components of the BEARCAT experimental platform.**

The Android OS is supported by a powerful and flexible development environment, giving full access to all of the sensors and user interaction tools. An application written using this framework allows the phones to report user position to each other and to off-board computing, communicating via the 3G network. For each agent, the application provides a visual display that shows the local map and the location and orientations of all the agents. It also allows the phones to serve as user interface devices for the agents, allowing commands and other information to be sent to the UAVs and the off-board computing. The phone is shown in use in Figure 2.2.

The smartphone application is crucial to the experiments described in this thesis. In addition to providing tracking, the application is used to display reachability information to the agents during the experiments. The agents are able to view computed reachable sets and recommended movement directions, as well as information pertaining to the location and state of the UAVs.

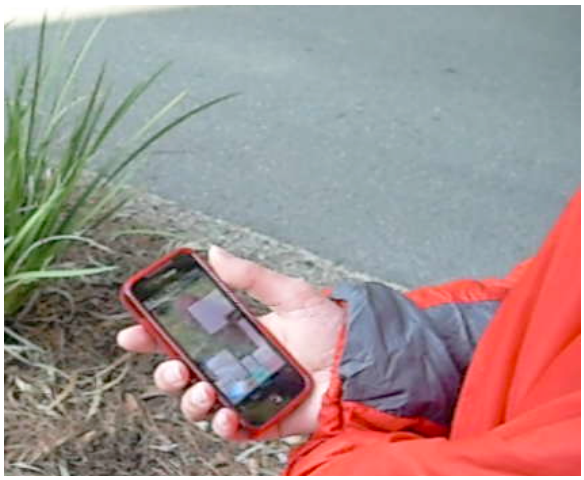


Figure 2.2: **HTC Incredible smartphone in use during a BEARCAT experiment.**

Off-board computing

Off-board computing in the form of a ground-station laptop provides oversight of the agents and extra computational resources. It is used for evaluating various algorithms to aid the agents and to control the UAVs. The off-board computing typically consists of either a Macbook Pro laptop, running Mac OS X 10.6, or a Lenovo Thinkpad T420 running Windows 7. The laptop communicates with the phones via a Pantech UML290 USB modem [40] over the 3G wireless data network, giving data rates comparable with IEEE 802.11 (WiFi) wireless. Communications with the quadrotor UAVs occurs both over WiFi or by mounting a 3G USB modem on the UAVs. It should be noted that due to policies implemented by Verizon, the service provider used for these devices, certain forms of packet communications are unreliable over 3G, and thus during the experiments described in this thesis communications between the phones and laptop is over 3G, but most communication with UAVs is over WiFi.

A ground-station application written in Java is used to manage the experiments. This software controls the UAVs, and also allows supervisory observation of the human agents. Global information such as agent visibility is computed within this program, and it is used to manage communications between the agents as well as adjusting experimental parameters for the different trials.

Quadrotor UAVs

Quadrotor UAVs form the final component of BEARCAT, serving as aerial sensors. Quadrotor helicopters are an increasingly popular rotorcraft concept for unmanned aerial vehicle (UAV) platforms. These vehicles use two pairs of counter-rotating, fixed-pitch rotors located at the four corners of the aircraft, as shown in Figure 2.3. Their use as autonomous platforms has been envisaged in a variety of applications, both as individual vehicles and in multiple vehicle teams, including surveillance, search and rescue, and mobile sensor networks.

Quadrotor UAVs have two main advantages over comparable vertical take off and landing (VTOL) UAVs, such as helicopters. First, quadrotors can use fixed pitch rotors and direct control of motor speeds for vehicle control, simplifying design and maintenance by eliminating complex mechanical control linkages for rotor actuation. Second, the use of four rotors ensures that individual rotors are smaller than the equivalent main rotor on a helicopter for a given airframe size. The smaller rotors store less kinetic energy during flight and can be enclosed within a protective frame, permitting flights indoors and in obstacle-dense environments with reduced risk of damage to the vehicles, their operators, or surroundings. These added safety benefits greatly accelerate the design and test flight process by allowing testing to take place indoors or out, by inexperienced pilots, and with a short turnaround time for recovery from incidents.

Two different quadrotor UAVs are used in the experiments described in this thesis: the Stanford Testbed of Autonomous Rotorcraft for Multi-Agent Control (STARMAC), shown in Figure 2.3, and the Ascending Technologies Pelican, shown in Figure 2.4. STARMAC was one of the first quadrotor research platforms to be developed, with the aim of being an easy-to-use and reconfigurable proving ground for novel algorithms for multi-agent applications [36]. The electronics and sensing architecture used by STARMAC is typical of the pattern now used by most modern quadrotor UAVs, and is illustrated in Figure 2.5. An inertial measurement unit (IMU) with 3-axis accelerometers and 3-axis micro-electromechanical-systems (MEMS) gyroscopes coupled with a low-level embedded processor provide attitude stabilization, and a GPS receiver provides position data. A high-level, laptop-class processor running

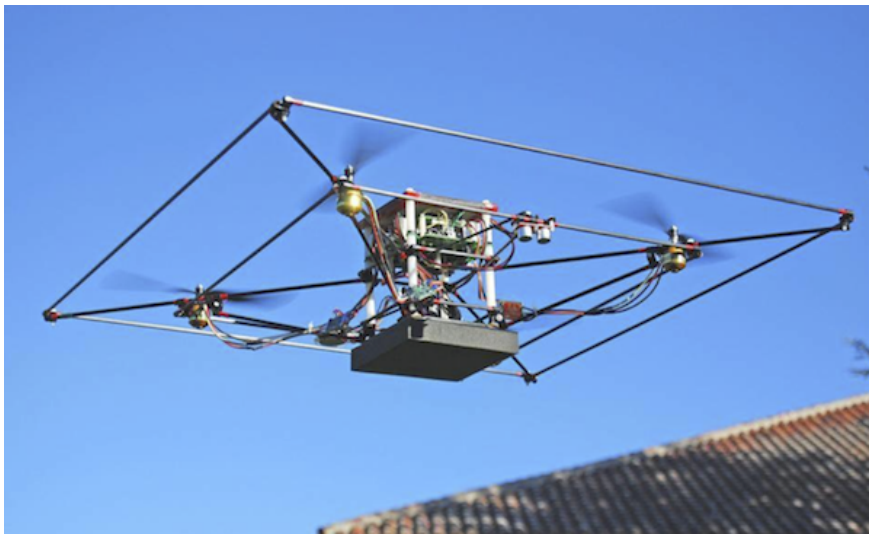


Figure 2.3: **STARMAC quadrotor UAV.**

Linux controls the vehicle in position, and can be used to interface with a number of sensors, including the Videre Systems stereo vision camera [41], various USB cameras, the Hokuyo URG-04LX laser range finder [42], and the Tracker DTS digital avalanche rescue beacon and receiver [43].

STARMAC has been used to test a number of algorithms, including experiments for collision avoidance [44], information-theoretic control for cooperative search [45, 46], dynamically feasible trajectory generation[47], and verification of provably safe aerobatic maneuvers[48, 37]. In each case, the flexibility and convenience of the quadrotor design in general and the precision flight capabilities of STARMAC in particular have enabled rapid evaluation of new technologies. The STARMAC quadrotor was used in some of the experiments describe in this thesis, including early results with reachability for maneuver sequencing, and experience with STARMAC greatly influenced the selection of the commercial quadrotor platform now used in BEARCAT.

The current generation of quadrotor UAV used in BEARCAT is the Ascending Technologies Pelican, a commercial product designed with research applications in mind [49]. The Pelican's electronics architecture is similar to the pattern established by STARMAC, with a low-level embedded processor that stabilizes the vehicle in attitude and position and a high-level, laptop-class Intel Atom processor that handles

Figure 2.4: **Ascending Technologies Pelican quadrotor UAV.**

more complex tasks. The Pelican's Atom board runs Ubuntu Linux, allowing a number of sensors like webcams and laser rangefinders to be easily integrated. In addition, the Atom board incorporates built-in WiFi communications, and other commercial communications devices with Linux support can be easily integrated.

For BEARCAT, the Atom processor is primarily tasked with handling communications with the ground station laptop either via WiFi or 3G wireless using the Pantech UML290 USB modem. GPS waypoints are transmitted to the flyer via WiFi or 3G, and then relayed to the low-level processor. The quadrotor is able to fly to and hover over waypoints with an accuracy on the order of 1 to 2 meters.

Communications architecture

The use of existing communications and networking infrastructure in BEARCAT allows a disparate set of computing, sensing, and robotic systems to be brought together simply and effectively. Specifically, the use of the 3G wireless phone data network allows all components of BEARCAT to communicate with each other at high speeds during field experiments without the need for dedicated communications equipment. As 3G phone coverage is widespread, experiments can be carried out over

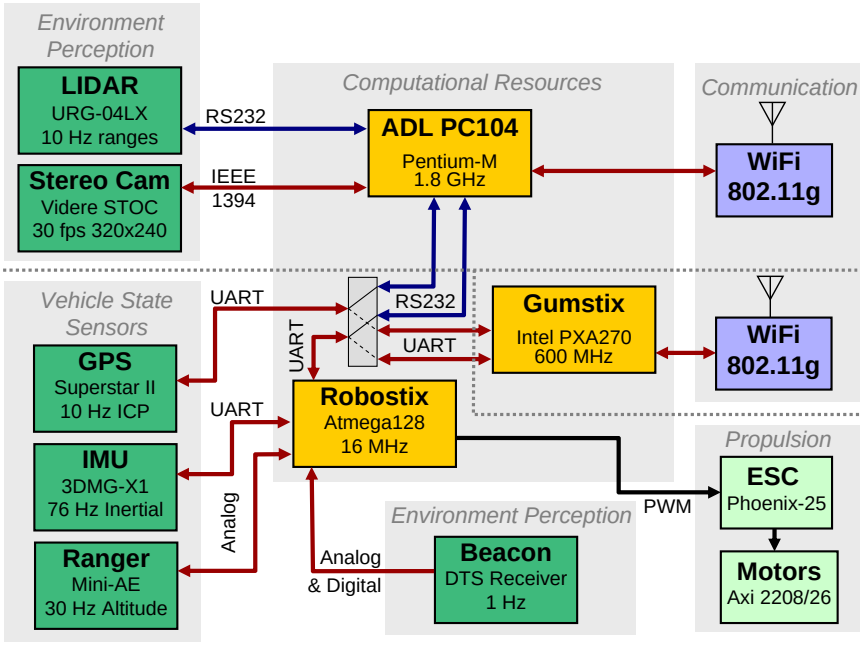


Figure 2.5: **STARMAC electronics architecture, typical of modern quadrotor UAVs.**

the entirety of campus while maintaining connectivity. Indeed, if desired, experiments can be conducted that take place across entire cities or even the country.

The communications architecture for BEARCAT is laid out in Figure 2.6. The phones are directly connected to the 3G network, and computers such as the ground station laptop or the high-level processors on the UAVs may connect to the 3G network via USB modems. As the 3G network is designed to maintain user security, devices cannot directly communicate with each other as their local internet protocol (IP) addresses are unknown. Instead, it is necessary to set up a communications server with a known web address acting as an intermediary. BEARCAT devices then contact the server via the internet so that the server can build a map of device identities to local IPs. Devices then transmit data packets to the server, which routes them accordingly.

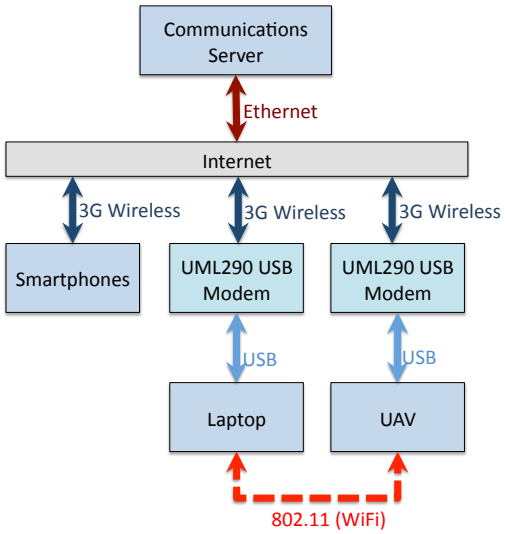


Figure 2.6: **BEARCAT communications architecture.**

Summary

Many experiments were carried out using the BEARCAT experimental platform to validate the algorithms developed in this research. These experiments under real-world conditions were crucial in examining both the strengths and limitations of these algorithms. In particular, communications failures and GPS errors often challenged the basic assumptions made in the solutions to the various games, and the actions of human agents often introduced elements of unpredictability. By testing under these conditions, valuable insights were gained into the practical use of these tools. Results from these experiments will be discussed throughout this thesis to illustrate the use of the algorithmic tools developed here.

Chapter 3

1 vs. 1: Hamilton-Jacobi-Isaacs Reachability

The game of capture-the-flag presents an excellent framework in which to examine solutions for adversarial games. Although the full game is complex and multi-layered, capture-the-flag in its simplest form can be condensed into a reach-avoid game between two agents over two different stages: an attacker seeking to first reach a flag and subsequently return it to safety, and a defender attempting to prevent this by intercepting the attacker. This reduction focuses on the adversarial interaction between the attacking and defending sides, and allows the game to be solved using methods from differential games. The solution can then be used to provide guidance to human or autonomous agents.

This chapter presents the formulation of capture-the-flag as a multi-stage reach-avoid game between two agents, and describes an approach for characterizing the winning initial conditions and strategies for each agent using Hamilton-Jacobi-Isaacs (HJI) reachability analysis. The positions of the agents are assumed to be known, but no model is assumed for opponent actions beyond the limits imposed by the dynamics. The approach described here is able to compute optimal, guaranteed strategies for bounded domains with arbitrary obstacle configurations for two agents. Solutions can be found for a game taking place over multiple stages, with different target sets for the attacker in each stage. Although the solutions are not computed in real

time, results for a particular game domain may be precomputed and recalled for use in real time. In addition, this approach also produces clear and intuitive tools for informing human decision-making through visualizations of the winning regions and recommended control actions. This 1 vs. 1 game is presented in the context of the game of capture-the-flag, but the formulation and solution presented are applicable to many general reach-avoid and pursuit-evasion games where the objective is for one agent to arrive at a target region while avoiding another agent.

The HJI formulation for reachability in adversarial games presented in this chapter is the most complete solution method to the games addressed in this thesis, although its practical utility can be limited by computational concerns. As such, this formulation and the concepts presented in this chapter will serve as a foundation from which the following chapters will build upon.

The chapter proceeds by first defining the 1 vs. 1 capture-the-flag problem and illustrating the desired solution approach via a 1-dimensional example in Section 3.1. Next, the Hamilton-Jacobi-Isaacs reachability method for finite time horizons is introduced in Section 3.2, along with a discussion of the relative merits of the finite and infinite horizon approaches, and their extension to multi-stage games. The finite horizon reachability method is then applied to the game of capture-the-flag in Section 3.3, with simulations presented in Section 3.4 to illustrate the solutions so generated. Experimental validation of the solutions with human agents in the BEARCAT platform are presented (without the use of UAVs) in Section 3.5, including detailed examination of the use of the system when the assumption of perfect state knowledge does not hold. A discussion of the theoretical and experimental results concludes the chapter. The algorithmic results underlying this work were first presented in [32].

3.1 1 vs. 1 Capture-the-Flag

The game considered here is a simplified version of capture-the-flag with a single agent on each team and full observability. The objective of the first agent, called the attacker, is to arrive at the flag and subsequently return to a safe region. The objective of the second agent, called the defender, is to prevent the attacker from

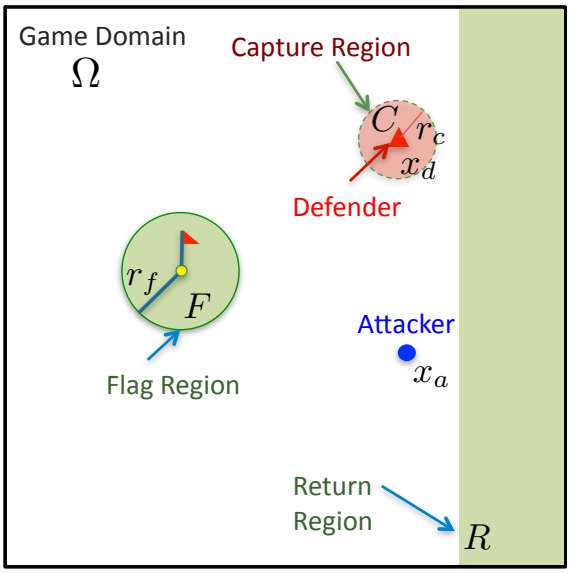


Figure 3.1: **The basic configuration of the capture-the-flag game.**

completing these tasks. The problem then is to characterize the sets of attacker and defender configurations for which there exists a winning strategy for either agent, as well as to find the strategies that ensure those winning conditions.

3.1.1 Problem Formulation

The game domain is assumed to be some finite closed set $\Omega \subset \mathbb{R}^2$. The set Ω can be regarded as the allowable free space for the agents to occupy, and in general may not be simply connected. For example obstacles for both agents may be represented as holes in Ω. The "safe" return region for the attacker is a set $R \subset \Omega$, and the flag is located in a set $F \subset \Omega$. For simplicity of illustration, the sets Ω and R in the examples discussed below will be defined as rectangular regions, with F defined as a circular region with radius r_f (see Figure 3.1). However, the computational methods described in this chapter are not limited to these simple geometric shapes.

The state of the game is described by the vector $\mathbf{x} = (x_a, x_d) \in \mathbb{R}^4$, where x_a and x_d are the planar positions of the attacker and defender, respectively, and $\Omega^2 \in \mathbb{R}^4$

denotes the joint domain of both agents. The equations of motion are given by

$$\dot{x}_a = u, \qquad u \in U,$$
$$\dot{x}_d = d, \qquad d \in D.$$

where u and d are the inputs of each agent, constrained to lie within sets U and D, respectively. The initial state is described by $\mathbf{x}(0) = (x_a^0, x_d^0) = \mathbf{x}^0$. For this example, U and D are represented by speed limits $v_{a,max}$ and $v_{d,max}$, with $||u||_2 \leq v_{a,max}$ and $||d||_2 \leq v_{d,max}$. It is not necessary to assume that $v_{a,max} = v_{d,max}$, and the maximum speeds may vary throughout Ω.

As per the rules of the game described in Section 1.4, the defender is constrained to remain outside of the flag and return regions, while the attacker can move freely through either. The attacker is considered to be intercepted by the defender if the attacker comes within some capture set centered on the defender, defined here as $C = \{\mathbf{x} \mid ||x_a - x_d||_2 \leq r_c\}$ where r_c is a constant capture radius.

Victory for the attacking agent is attained by meeting all of the following conditions, assuming the game takes place over some finite time horizon $[0, T_f]$ with a time duration T_c allocated for flag capture and T_r allocated for flag return:

- Flag capture: $x_a(T_c) \in F$ for some finite time horizon T_c where $0 \leq T_c \leq T_f$
- Flag return: $x_a(T_c + T_r) \in R$ for some finite time horizon T_r where $T_r + T_c \leq T_f$
- Avoiding defender capture:
  $x_a(t) \in \Omega \bigwedge \mathbf{x} \notin C$ for all time $t \in [0, T_r]$

In addition, the attacker also wins if at any time the defender violates the rules of the game by entering the flag or return regions. Victory for the defending agent is achieved by preventing the attacker from achieving any of the above conditions by T_f while obeying the constraint $x_d \in \Omega \setminus (F \cup R)$. It should be remarked that when T_f is small, the options of the attacking agent may be severely restricted by the time limit. Correspondingly, this also leads to simplistic delaying strategies by the defending agent. Motivated by this consideration, the work presented here will primarily be concerned with scenarios where T_f is large, and as T_f approaches ∞.

It is assumed that the positions of both agents are fully observable to each other, and that the defender input $d(t)$ may be a function of the attacker input $u(t)$ at each time t. The latter assumption corresponds to a choice in the order of play to prevent an infinite regression of second-guessing between the agents. Thus the solution is technically conservative on the part of the attacker, in that the defender has access to the attacker's control inputs. However, for the agent dynamics considered here, the opposite order of play leads to an equivalent problem formulation and identical solutions. The attacker input is determined by an attacker strategy $\mu(\mathbf{x}(t), t)$, with $u(t) = \mu(\mathbf{x}(t), t)$. The defender strategy is similarly determined by a defender strategy $\gamma(\mathbf{x}(t), u(t), t)$. The sets of admissible attacker and defender strategies are defined as $\mathbb{U} = \{\mu : \mathbb{R}^4 \times [0, T_f] \to U\}$ and $\mathbb{D} = \{\gamma : \mathbb{R}^4 \times U \times [0, T_f] \to D\}$, respectively.

The goal here is to provide a solution to this two-agent game of capture-the-flag in terms of both the winning regions and winning strategies for each agent. For the attacker, this means determining the subset of initial configurations $W_A \subset \mathbb{R}^4$ for which there exists a feasible attacker strategy for achieving the victory conditions, regardless of the strategy of the defending agent. Furthermore, for any configuration in W_A, it is also necessary to determine a winning strategy that ensures meeting the attacker victory conditions. Clearly, for any permissible initial configuration outside this set, there exists some defender strategy so as to prevent the attacker from achieving victory, regardless of the attacker strategy. Thus, the defender winning set W_D is given simply as $\Omega^2 \setminus W_A$.

The game occurs over two stages: when the attacker is attempting to achieve flag capture, and then subsequently when the attacker is attempting to achieve flag return. In building up towards the winning sets, some intermediate winning sets for the flag capture and flag return stages of the game will also be characterized. The sets are defined as follows:

- **Flag Capture Set** F_A: configurations from which the attacker can achieve flag capture while avoiding defender interception.
- **Stop Capture Set** F_D: $\Omega^2 \setminus F_A$, where the defender can prevent flag capture by the attacker.

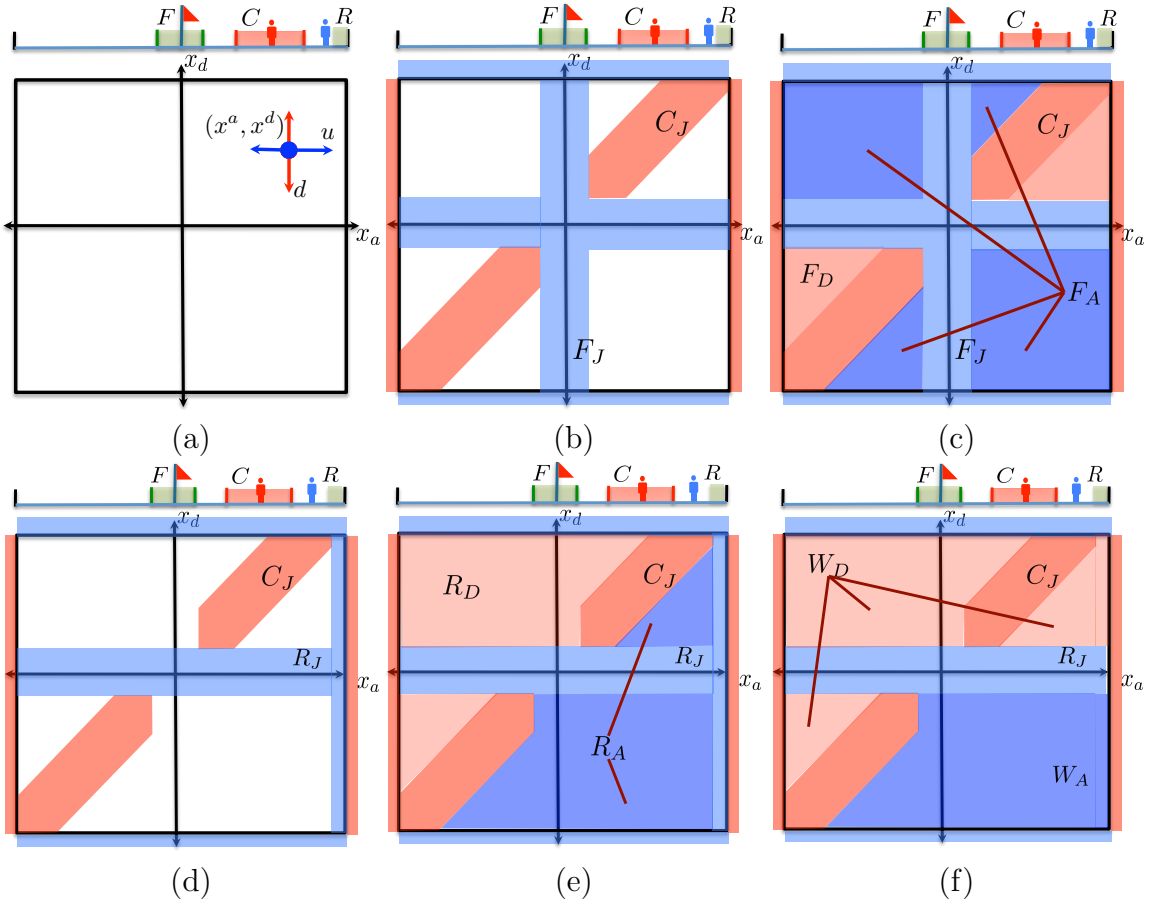


Figure 3.2: **1-dimensional example illustrating (a) the joint state space and the current state of the game, along with the agents' inputs, the targets and winning configurations for the agents for (b), (c) flag capture and (d), (e) flag return, and (f) the winning configurations for the full game.**

- **Flag Return Set** R_A: configurations from which the attacker can achieve flag return while avoiding defender interception.
- **Stop Return Set** R_D: $\Omega^2 \setminus R_A$, where the defender can prevent flag return by the attacker.

3.1.2 1-Dimensional Example

A 1-dimensional example game is presented in Figure 3.2 to clarify the region definitions and the concept of winning regions presented in the preceding text. Figure 3.2a shows the joint state space for a 1-dimensional game, with the attacker coordinate plotted on the horizontal axis and the defender coordinate plotted on the vertical axis. The current state of the game as illustrated is shown along with the possible inputs of the two agents. Note that the attacker input u moves the state along the x_a (horizontal) axis, and similarly the defender input d moves the state along the x_d (vertical) axis. It is obvious that in the initial configuration shown in Figure 3.2a there is no way for the attacker to reach the flag without being intercepted by the defender.

Figure 3.2b illustrates the desired terminal configurations for each agent in the flag capture stage of the game. The target configurations for the attacker are denoted F_J and are highlighted in blue, representing the set of states where the attacker has entered F, or the defender has forfeited by either exiting the game region or entering F or R. Similarly, the defender's target configurations C_J are highlighted in red, denoting the states where the attacker has either been intercepted by entering C or has exited Ω. From this, the winning regions for the flag capture portion of the game can be determined by inspection as those identified in Figure 3.2c. Intuitively, the attacker winning configurations F_A correspond to states where the defender is not between the attacker and the flag, and F_D comprises the remainder of the states.

Similarly, the target configurations for the flag return stage are shown in Figure 3.2d with winning configurations for each agent illustrated in Figure 3.2e. Note that in order for the attacker to win the full game, it must first reach the flag and then return it to safety. This requires the flag capture portion of the game to end with the joint state in R_A, that is, the attacker must reach the flag in such a way that it can safely return while avoiding interception. Thus the full winning configuration for the attacker W_A will be the darker blue region as shown in Figure 3.2f, with W_D as the light red region.

The 1-d example is discussed to give some intuition into the nature of the winning regions in the joint state space of the agents and what they represent. For this simple

example, the regions can be found trivially by inspection, but the computation of such sets in $\mathbb{R}^4$ is a more complex matter. The agents are not constrained to move on a line, but may move around each other and around obstacles in the state space. Instead, the full game will be characterized using reachability to find the desired sets. The reachability formulation is described below, and then used to find solutions for capture-the-flag.

3.2 Computing Reachable Sets via Hamilton-Jacobi-Isaacs Equations

Capture-the-flag falls naturally under the framework of differential games due to the competing objectives of the attacking and defending agents [11]. This allows the winning regions of the attacker and defender to be characterized via the results of numerical Hamilton-Jacobi-Isaacs reachable set calculations [12].

Although it is possible to use geometric analysis to construct solutions for simple configurations of the game regions, the numerical approach allows arbitrarily complex domains to be addressed, including domains with obstacles which both agents must avoid. Moreover, this framework naturally allows for the sequential aspect of the game, where the attacker must first reach the flag zone and then subsequently return to safety. Such sequencing can be very difficult to analyze using geometric analysis.

The solution presented will be based on the use of a time dependent reachability formulation over finite time horizons. The use of a finite horizon formulation better aligns with the limited time nature of actual games of capture-the-flag, and is also preferred over a static, infinite horizon formulation for numerical reasons, due to the presence of infinite values in the infinite horizon solution. The finite horizon approach will be presented in detail, and its relationship to the infinite horizon problem and the trade-offs involved in the two approaches will also be discussed. The generalization of reachability methods to games played over multiple stages will also be addressed. For the purposes of this discussion, general, generic dynamics and time horizons will be used, and specific application to capture-the-flag will be demonstrated in the following

section.

3.2.1 Finite Horizon Reachability

The two sides in a general differential game will be referred to as the *control* and the *disturbance*. This choice of naming reflects the fact that although the solution for a general game finds the control inputs for both sides, the game is often being solved for an application that requires controlling a specific side. The other side is then regarded as a disturbance whose adversarial actions must be accounted for and rejected by the control. This chapter considers solutions for both sides, although subsequent chapters focus on one particular side. Note that each side in a reach-avoid game may consist of multiple agents.

The continuous dynamics are assumed to be modeled by the ordinary differential equation

$$\dot{\mathbf{x}} = f(\mathbf{x}, u, d),\ \mathbf{x}(0) = \mathbf{x}^0 \tag{3.1}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the system state, u is the input of the control, d is the input of disturbance, and $\mathbf{x}^0$ is the initial condition. The input ranges of the control and disturbance are denoted by U and D, respectively.

The game is assumed to be played over a finite horizon T. The control selects inputs $u(t), t \in [0, T]$, satisfying $u(t) \in U$, possibly as a function of the state $\mathbf{x}(t)$, according to a strategy $\mu(\mathbf{x}(t), t)$ with $u(t) = \mu(\mathbf{x}(t), t)$. Similarly, the disturbance selects inputs $d(t), t \in [0, T]$, satisfying $d(t) \in D$. However, this selection is allowed to be a function of both the state $\mathbf{x}(t)$ and the input u of the control, with the strategy denoted $\gamma(\mathbf{x}(t), u(t), t)$. This order of play reflects an conservatism by the control, as the desire usually is to find a strategy that is successful even in the worst case. The set of permissible strategies for the control and disturbance are denoted by $\mathbb{U} = \{\mu : \mathbb{R}^n \times [0, T_f] \to U\}$ and $\mathbb{D} = \{\gamma : \mathbb{R}^n \times U \times [0, T_f] \to D\}$, respectively.

Now consider a target set $\mathcal{T}$ and an undesired set K, with $\mathcal{T}, K \subset \mathbb{R}^n$ and time horizon $[0, T]$. The problem is to compute the set of initial conditions $\mathbf{x}^0 \in \mathbb{R}^n$ for which there exists some choice of control strategy $\mu \in \mathbb{U}$, such that regardless of the choice of disturbance strategy $\gamma \in \mathbb{D}$, the state trajectory $\mathbf{x}(\cdot)$ under equation (3.1)

satisfies $\mathbf{x}(t) \in \mathcal{T}$ for some $t \in [0, T]$ and $\mathbf{x}(s) \notin K$, $\forall s \in [0, t]$. In other words, this is the set of initial conditions from which the system can be steered into $\mathcal{T}$ within finite time by the control, while avoiding K at all times, regardless of the disturbance strategy. This set is denoted by $\mathcal{RA}_T(\mathcal{T}, K)$, and will be referred to as the *reach-avoid set* over $[0, T]$.

Under suitable conditions, given in [12] and [50], the reach-avoid set can be obtained from the solution of a constrained Hamilton-Jacobi-Isaacs (HJI) partial differential equation (PDE), using a level set representation of sets. Under this representation, a set $G \subset \mathbb{R}^n$ is defined implicitly as the sub-level set of a function $J_G : \mathbb{R}^n \to \mathbb{R}$, such that $G = \{\mathbf{x} \mid J_G(\mathbf{x}) \leq 0\}$. Set operations proceed straightforwardly using the *max* and *min* functions, where $A \cup B = \min(J_A, J_B)$ and $A \cap B = \max(J_A, J_B)$ and other set functions follow accordingly [50].

The reachability computation is performed via the solution to a terminal-cost differential game beginning at time $t = 0$ and ending at time $t = T$. The level set representation $J_\mathcal{T}$ of the target region $\mathcal{T}$ can be used to define a final cost at $t = T$, where $J_\mathcal{T}(\mathbf{x}) \leq 0$ implies that $\mathbf{x} \in \mathcal{T}$ and $J_\mathcal{T}(\mathbf{x}) > 0$ implies $\mathbf{x} \notin \mathcal{T}$. To drive the system into $\mathcal{T}$, the control is modeled as attempting to minimize the terminal cost $J_\mathcal{T}$, and the disturbance as maximizing. Assume that the game begins at time $t = 0$ with state $\mathbf{x}^0$ and ends at $t = T$. Both the control and disturbance play optimally, and at the end of the game the system is at a state $\mathbf{x}(T)$. If $J_\mathcal{T}(\mathbf{x}(T)) \leq 0$, then the control has successfully guided the system into $\mathcal{T}$ and $\mathbf{x}^0 \in \mathcal{RA}_T(\mathcal{T}, K)$, otherwise the disturbance has won and $\mathbf{x}^0 \notin \mathcal{RA}_T(\mathcal{T}, K)$. The reachability computation is then equivalent to finding, for an initial state $\mathbf{x}^0$, what the terminal cost will be if the game were to begin at $\mathbf{x}^0$ and both sides play optimally until T.

This can be encapsulated by defining a cost function $J(\mathbf{x}, t)$, which for state $\mathbf{x}$ at time t is

$$J(\mathbf{x}, t) = \min_{\mu \in \mathbb{U}} \max_{\gamma \in \mathbb{D}} J_\mathcal{T}(\mathbf{x}(T)) \tag{3.2}$$

That is, the current value $J(\mathbf{x}, t)$ for a particular state $\mathbf{x}$ at a time t reflects the final value of the game at time $t = T$ if the game were to proceed optimally starting at $\mathbf{x}$. Reach-avoid sets for a game occurring over $[0, T]$ can then be found by computing

$J(\mathbf{x}, 0)$ for all points $\mathbf{x}$.

This cost function $J(\mathbf{x}, t)$ is the viscosity solution [51] of the following constrained terminal value HJI PDE,

$$\frac{\partial J}{\partial t} + \min\left[0, H\left(\mathbf{x}, \frac{\partial J}{\partial \mathbf{x}}\right)\right] = 0,\ J(\mathbf{x}, T) = J_{\mathcal{T}}(\mathbf{x}) \tag{3.3}$$

subject to

$$J(\mathbf{x}, t) \geq -J_K(\mathbf{x})$$

where H is the optimal Hamiltonian

$$H(\mathbf{x}, p) = \min_{u \in U} \max_{d \in D} p^T f(\mathbf{x}, u, d)$$

and $J_{\mathcal{T}}, J_K$ are the level set representations of $\mathcal{T}$ and K, respectively, with $p = \frac{\partial J}{\partial \mathbf{x}}$. This computation proceeds backward in time, starting from the final value $J(\mathbf{x}, T)$ and integrating backward until $J(\mathbf{x}, 0)$ is found.

The reach-avoid set can now be found, as

$$J(\mathbf{x}, 0) \leq 0 \Rightarrow J_{\mathcal{T}}(\mathbf{x}(T)) \leq 0 \Rightarrow x \in \mathcal{RA}_T(\mathcal{T}, K)$$

and the reach-avoid set is given by

$$\mathcal{RA}_T(\mathcal{T}, K) = \{\mathbf{x} \mid J(\mathbf{x}, 0) \leq 0\}$$

An accurate numerical solution to equation (3.3) can be computed using the Level Set Toolbox for MATLAB [52]. As the grid required to represent the state space grows exponentially with the number of continuous states, this method may be limited computationally. Currently, systems of up to 4 dimensions can be handled easily, and solutions have been found for some systems with 5 dimensions [53]. Details of the theoretical formalism may be found in [12, 50], and a schematic description of the approach, including a dynamic programming derivation of the approach and some geometric insights, may be found in [37].

3.2.2 Infinite Horizon Reachability

The finite horizon reachability formulation requires the solution of a time dependent HJI equation backward in time over the entire time horizon. If the game is to be played over very long horizons that approach infinity, it is also possible to address the infinite horizon problem directly. However, although this approach is theoretically sound there are practical issues (discussed below) that make the time dependent approach preferable in this particular application.

The solution to the infinite horizon reachability problem proceeds via a static (time independent) HJI equation, where the value is the time required for arrival at the target set $\mathcal{T}$ while avoiding K. The control is attempting to minimize this time and the disturbance is attempting to maximize. The optimal value of the game $\mathcal{V}_{\mathcal{T}}(\mathbf{x})$ for a state $\mathbf{x}$ is then:

$$\mathcal{V}_{\mathcal{T}}(\mathbf{x}) = \min_{\mu \in \mathbb{U}} \max_{\gamma \in \mathbb{D}} \{t \mid \mathbf{x}(t) \in \mathcal{T}, \mathbf{x}(s) \notin K, \forall s \in [0,t]\}. \tag{3.4}$$

For states where $\mathcal{T}$ cannot be reached while avoiding K, $\mathcal{V}_{\mathcal{T}}(\mathbf{x})$ is infinite. The value so defined satisfies the following Hamilton-Jacobi-Isaacs equation [9]:

$$\min_{u \in U} \max_{d \in D} \left[\frac{\partial \mathcal{V}_{\mathcal{T}}^T}{\partial \mathbf{x}} f(\mathbf{x}, u, d) \right] + 1 = 0 \tag{3.5}$$

subject to the following boundary conditions

$$\begin{aligned} \mathcal{V}_{\mathcal{T}}(\mathbf{x}) &= 0, \mathbf{x} \in \mathcal{T} \\ \mathcal{V}_{\mathcal{T}}(\mathbf{x}) &= \infty, \mathbf{x} \in K. \end{aligned}$$

The time dependent and time independent HJI equations are related in the following manner: where $\mathcal{V}_{\mathcal{T}}$ is differentiable, T-level sets of $\mathcal{V}_{\mathcal{T}}$ are equal to the 0-level sets of the time dependent value $J(\mathbf{x}, 0)$ for the same game (with same $\mathcal{T}$ and K) played over a time horizon $[0, T]$, specifically

$$\mathcal{V}_{\mathcal{T}}(\mathbf{x}) = T \Rightarrow J(\mathbf{x}, 0) = 0. \tag{3.6}$$

This corresponds naturally to the intuition that both $\mathcal{V}_{\mathcal{T}}(\mathbf{x}) = T$ and $J(\mathbf{x}, 0) = 0$ imply that the system can be driven into $\mathcal{T}$ in at most time T, starting from $\mathbf{x}$. Thus the finite horizon reach-avoid set $\mathcal{RA}_T(\mathcal{T}, K)$ can be found using the infinite horizon value as

$$\mathcal{RA}_T(\mathcal{T}, K) = \{\mathbf{x} \mid \mathcal{V}_{\mathcal{T}}(\mathbf{x}) \leq T\}$$

and the infinite horizon reach-avoid set $\mathcal{RA}(\mathcal{T}, K)$ can be found as

$$\mathcal{RA}(\mathcal{T}, K) = \{\mathbf{x} \mid \mathcal{V}_{\mathcal{T}}(\mathbf{x}) < \infty\}$$

Note that the infinite horizon reach-avoid set $\mathcal{RA}(\mathcal{T}, K)$ can also be found by computing the finite time reachable set $\mathcal{RA}_T(\mathcal{T}, K)$ over a time horizon T and allowing $T \to \infty$.

The practical use of the infinite horizon formulation is affected by two factors. First, equation (3.5) cannot be directly solved numerically due to the presence of infinite values. Instead, solution typically proceeds via the Kruzkov Transform [54]

$$\mathcal{U}_{\mathcal{T}}(\mathbf{x}) = 1 - e^{-\mathcal{V}_{\mathcal{T}}(\mathbf{x})},$$

thus mapping values $0 - \infty$ to $0 - 1$ instead. This transformation allows for efficient solution via numerical methods, however numerical diffusion in the solution introduces small errors that makes it difficult to precisely locate the boundary between $\mathcal{V}_{\mathcal{T}} = \infty$ and $\mathcal{V}_{\mathcal{T}} < \infty$. For example, consider a case where $\mathcal{U}_{\mathcal{T}}(\mathbf{x})$ is 1 for some $\mathbf{x}$, implying that $\mathcal{V}_{\mathcal{T}}(\mathbf{x}) = \infty$ and that the point $\mathbf{x}$ is outside of the reachable set. In this case, a small numerical error may render $\mathcal{U}_{\mathcal{T}}(\mathbf{x})$ to be less than 1, erroneously giving a finite value for $\mathcal{V}_{\mathcal{T}}(\mathbf{x})$ and placing the point inside the reachable set. Similarly, numerical errors in the opposite direction may falsely exclude points from the set. This makes it difficult to determine where the game is actually winnable for a given side, as the boundary of the infinite horizon reachable set cannot be precisely ascertained.

Another issue is that the static value represents the game from the perspective of a particular side. For states where the disturbance wins the game by preventing the system from reaching $\mathcal{T}$, the value $\mathcal{V}_{\mathcal{T}}(\mathbf{x})$ is infinite and thus the gradient $\frac{\partial \mathcal{V}_{\mathcal{T}}}{\partial \mathbf{x}}$ is

undefined. The optimal input for each side is typically computed via the Hamiltonian, which depends upon the gradient of the value function. This means that to extract control inputs the game must be solved separately with a new value defined. For example, one may consider the disturbance as attempting to reach the avoid region K and solve the game with the minimization and maximization reversed. Note that even this is insufficient, as there may exist states from which $\mathbf{x}$ cannot be driven into either $\mathcal{T}$ or K, which according to the original definition of the game still results in victory for the disturbance.

The additional complications engendered by the infinite horizon formulation make it less suitable for the reachability computations of interest than the finite horizon approach, thus the solution to capture-the-flag will proceed using the finite horizon reachability formulation.

3.2.3 Multi-Stage Games

The final issue that must be addressed is the extension of the reachability formulations discussed above to multi-stage games, where a sequence of target sets $\mathcal{T}_s = (\mathcal{T}_1, \ldots, \mathcal{T}_{N_s})$ must be reached in order, while avoiding a sequence of avoid sets $K_s = (K_1, \ldots, K_{N_s})$ for N_s separate stages. To complete the entire sequence, it is clearly insufficient for the control to drive the system into a target $\mathcal{T}_k$ during the kth stage and hope for the best. In order to subsequently reach $\mathcal{T}_{k+1}$, the system must also be in the appropriate reach-avoid set for $\mathcal{T}_{k+1}$. The construction of the reach-avoid set for the entire sequence must be constructed from the final target set backwards to the first.

For the infinite horizon case, let $\mathcal{RA}_{k+}$ represent the set of states that can reach the entire sequence of targets $\mathcal{T}_{k+} = (\mathcal{T}_k, \ldots, \mathcal{T}_N)$ in order while avoiding the entire sequence of avoid sets $K_{k+} = (K_k, \ldots, K_N)$. The initial reach-avoid set $\mathcal{RA}_{1+}$ can be found inductively from $\mathcal{RA}_{N_s+}$. Clearly, $\mathcal{RA}_{N_s+} = \mathcal{RA}(\mathcal{T}_{N_s}, K_{N_s})$. Stepping backward, each $\mathcal{RA}_{k-1+}$ can be found from $\mathcal{RA}_{k+}$ as

$$\mathcal{RA}_{k-1+} = \mathcal{RA}(\mathcal{RA}_{k+} \cap \mathcal{T}_{k-1}, K_{k-1})$$

and $\mathcal{RA}_{1+}$ can be found accordingly.

For a finite horizon game occurring over horizon $[0, T_f]$, with a sequence of stage time horizons $T_1, \ldots, T_{N_s}$ where $\sum_{i=1}^{N_s} T_i = T_f$, the multi-stage reachable set $\mathcal{RA}_{T_f,1+}$ can be found similarly. Again, each stage of the game must end in the reach-avoid set for the next stage, thus the construction of the initial set begins from the final stage and works backward. In this case, the final stage reachable set is $\mathcal{RA}_{T_{N_s},N_s+} = \mathcal{RA}_{T_{N_s}}(\mathcal{T}_{N_s}, K_{N_s})$ and each intermediate stage can be found as

$$\mathcal{RA}_{T_{k-1+},k-1+} = \mathcal{RA}_{T_{k-1}}(\mathcal{RA}_{T_{k+},k+} \cap \mathcal{T}_{k-1}, K_{k-1})$$

where $T_{k+} = \sum_{j=i}^{N_s} T_j$ is the remaining time for all future stages after stage k.

With these constructions, it is possible to apply reachability to multi-stage reach-avoid games like capture-the-flag. The solution to capture-the-flag using finite-horizon reachability will now be presented.

3.3 Characterization via Reachability

The solution to the 1 vs. 1 capture-the-flag game can now be found by characterizing the attacker victory sets F_A, R_A, and W_A using the reach-avoid operator. The corresponding defender victory sets F_D, R_D, and W_D can be obtained via the relations described in Section 3.1. Under the differential game setting, the attacker takes on the role of the control, while the defender takes on the role of the disturbance. For the system dynamics of capture-the-flag, the optimal Hamiltonian is

$$H(\mathbf{x}, p) = \min_{u \in U} \max_{d \in D} \left[\frac{\partial J^T}{\partial x_a} u + \frac{\partial J^T}{\partial x_d} d \right] \tag{3.7}$$

where J is defined as in equation (3.2). Note that, although formally the framework is conservative on behalf of the control with regard to the disturbance, in this case the dynamics of the two agents decouple, and the solution is identical no matter the order of play.

3.3.1 Constructing the Winning Sets

The construction of the winning sets will proceed backward, starting with the flag return stage in order to determine the terminal conditions for flag capture that could result in successful return. First, consider the flag return stage independently, as taking place over some time horizon $[0, T_r]$. The winning condition for the attacker during this stage is to arrive in the return region R within $[0, T_r]$, while avoiding interception by the defender. This portion of the game terminates at a time t with $0 \leq t \leq T_r$ if the attacker arrives in R or is intercepted by the defender. If the attacker is unable to reach R by T_r, the defender wins.

The rules require both agents to stay within the game region Ω and the defender to remain outside of F and R. The constraints for each agent are encoded as part of the winning conditions for the other, so that the attacker wins if the defender violates its constraints and vice versa.

The formal definition of the winning conditions for each agent follows. Let $\mathbf{x}$ be the joint configuration (x_a, x_d), Ω^C the complement of Ω in $\mathbb{R}^2$, and $G_A = \left\{\mathbf{x} \mid x_a \in \Omega^C\right\}$, then the attacker winning conditions as described above are $O_R \vee O_D$, where

$$
\begin{aligned}
O_R &= \{\exists t \in [0, T_r], x_a(t) \in R \wedge \mathbf{x}(s) \notin C \cup G_A, \forall s \in [0, t]\} \\
O_D &= \big\{\exists t \in [0, T_r], x_d(t) \in F \cup R \cup \Omega^C \wedge \mathbf{x}(s) \notin C \cup G_A, \\
&\quad \forall s \in [0, t]\}
\end{aligned}
$$

Define the sets $G_D = \left\{\mathbf{x} \mid x_d \in F \cup R \cup \Omega^C\right\}$ and $G_R = \{\mathbf{x} \mid x_a \in R\}$. Then the attacker target set in the flag return stage can be defined as the set $R_J = G_R \cup G_D$, and the defender target set is $C_J = C \cup G_A$. The flag return set can then be computed as

$$R_A = \mathcal{RA}_{T_r}(R_J, C_J) \tag{3.8}$$

Now consider the flag capture stage of the game, with a time horizon of $[0, T_c]$. If the second stage of the game is ignored, then the goal of the attacker is to simply arrive in the flag region F within $[0, T_c]$ while avoiding interception by the defender, and the goal of the defender is to intercept the attacker or prevent the attacker from

reaching F. Using the same constraint encoding as before, the attacker winning condition for this stage is given by $O_F \vee O_D$, where

$$O_F = \{\exists t \in [0, T_c], x_a(t) \in F \wedge \mathbf{x}(s) \notin C \cup G_A, \forall s \in [0, t]\}$$

and O_D is modified with time horizon T_c. Define the set $G_F = \{\mathbf{x} \mid x_a \in F\}$. Then the attacker target set in the flag capture stage can be defined as $F_J = G_F \cup G_D$, and the flag capture set can be computed as

$$F_A = \mathcal{RA}_{T_c}(F_J, C_J) \tag{3.9}$$

When both stages of the game are considered, it is clearly insufficient for the attacker to simply arrive in F. If the defender chooses a strategy such that the attacker arrives at the flag but the overall state $\mathbf{x}$ is in a configuration outside the flag return set R_A, then the defender can prevent the attacker from returning safely to R. Instead, the multi-stage solution must be used, with $\mathcal{T}_s = (F, R)$ and $K_s = (C_J, C_J)$, with the same avoid set in each stage. The time horizons are (T_c, T_r) with $T_c + T_r = T_f$. The attacker's winning set W_A is then simply $\mathcal{RA}_{T_f,1+}$ with respect to $\mathcal{T}_s$ and K_s as defined above.

In this two stage game, the results can be written explicitly. The attacker must reach the set of configurations $\tilde{R}_A = G_F \cap R_A$ during the first stage of the game, reflecting that the attacker must reach the flag and be in a safe return configuration. This corresponds to the modified attacker objectives during the flag capture stage $\tilde{O}_F \vee O_D$, where

$$\tilde{O}_F = \left\{\exists t \in [0, T_c], \mathbf{x}(t) \in \tilde{R}_A \wedge \mathbf{x}(s) \notin C \cup G_A, \forall s \in [0, t]\right\}$$

From this, the modified attacker target set in flag capture must be $\tilde{F}_J = \tilde{R}_A \cup G_D$ and the attacker winning set is then

$$W_A = \mathcal{RA}_{T_c}(\tilde{F}_J, C_J).$$

3.3.2 Strategies

Winning strategies in regions of the state space where the solution J to equation (3.3) is differentiable can be synthesized based upon the approach described in [12]. For reach-avoid and pursuit-evasion games, the solution to the HJI equation is typically differentiable away from some singular surfaces where the winning strategies are not uniquely defined [11]. With arbitrarily small perturbations from these surfaces, the solution becomes differentiable, allowing application of the techniques described here.

Under this setting, for $\mathbf{x} \in \mathcal{RA}_T(\mathcal{T}, K)$, the strategy that allows the control to achieve the desired objectives regardless of the disturbance strategy is described by

$$\mu(\mathbf{x}, t) = \arg\min_{u \in U} \max_{d \in D} p(\mathbf{x}, t)^T f(\mathbf{x}, u, d),\ t \in [0, T] \tag{3.10}$$

where $p = \frac{\partial J}{\partial \mathbf{x}}$ and the dependence of the state on t is dropped for clarity. Similarly, for $\mathbf{x} \notin \mathcal{RA}_T(\mathcal{T}, K)$, the disturbance strategy that is guaranteed to prevent the control from achieving the desired objectives is given by

$$\gamma(\mathbf{x}, u, t) = \arg\max_{d \in D} p(\mathbf{x}, t)^T f(\mathbf{x}, u, d),\ \ t \in [0, T] \tag{3.11}$$

Given the numerical computation of the set $\mathcal{RA}_T(\mathcal{T}, K)$, the derivatives of J are not available in closed form. Nonetheless, approximate strategies can be obtained by computing numerical derivatives of J.

For the capture-the-flag problem, the optimal Hamiltonian is given by equation (3.7). Let $p_u = \frac{\partial J}{\partial x_a}$ and $p_d = \frac{\partial J}{\partial x_d}$. From this the explicit winning strategies can be computed as

$$\mu(\mathbf{x}, t) = -v_{a,max} \frac{p_u(\mathbf{x}, t)}{||p_u(\mathbf{x}, t)||_2} \tag{3.12}$$

$$\gamma(\mathbf{x}, u, t) = \gamma(\mathbf{x}, t) = v_{d,max} \frac{p_d(\mathbf{x}, t)}{||p_d(\mathbf{x}, t)||_2} \tag{3.13}$$

Note how the choice of agent order does not affect the agent strategies in this problem as the optimal defender and attacker inputs are independent of each other.

Over short time horizons, time varying strategies can be synthesized for the flag

capture and flag return stages of the game as per equations (3.12) and (3.13), using W_A and R_A, respectively. Under the attacker control strategy constructed from W_A, the trajectory $\mathbf{x} = (x_a, x_d)$ is guaranteed to enter $\tilde{R}_A$ within $[0, T_c]$, satisfying the flag capture objective. At the time instant when $\mathbf{x}(t) \in \tilde{R}_A$, the attacker can switch to the control strategy constructed from R_A and safely return to R, thus winning the game.

In cases in which the solution to equation (3.3) converges, namely $\lim_{t\to\infty} J(\mathbf{x}, t) = J_\infty(\mathbf{x})$ for some $J_\infty : \mathbb{R}^n \to \mathbb{R}$, time-invariant control strategies approximating equations (3.10) and (3.11) can be constructed. Specifically, suppose J is computed numerically at time steps $k\tau, k = 0, 1, ...$ for a time step τ. An index k_f can be selected at which $||J(\cdot, k_f\tau) - J(\cdot, (k_f + 1)\tau)||_\infty < \epsilon$, for some tolerance $\epsilon > 0$. If ϵ is chosen sufficiently small, it can be reasonably assumed that $J(\cdot, t) \approx J(\cdot, k_f\tau)$, $\forall t > k_f\tau$. This allows the long term strategies of the two agents to be approximated by

$$\mu(\mathbf{x}) \approx \arg\min_{u\in U} \max_{d\in D} p(\mathbf{x}, k_f\tau)^T f(\mathbf{x}, u, d)$$
$$\gamma(\mathbf{x}) \approx \arg\max_{d\in D} p(\mathbf{x}, k_f\tau)^T f(\mathbf{x}, u, d)$$

For this application, the sets R_A and W_A typically converge numerically for sufficiently large T_r and T_c, leading to approximate long term strategies for the attacker and defender.

The methods presented above are sufficient to construct a solution to the game of 1 vs. 1 capture-the-flag, giving both the initial conditions that lead to victory for each agent as well as the optimal inputs necessary to achieve victory. The following sections will illustrate the approach in simulation, and then show how these computations can be used in experimental games with human agents.

3.4 Simulation Results

The HJI reachability solution to capture-the-flag is illustrated here via an example with $v_{a,max} = v_{d,max} = 1$, $r_f = 1$, $r_c = 0.5$, and $T_c = T_r = 12$. The value function is computed using the Level-Set Toolbox [52]. For the particular set of parameters,

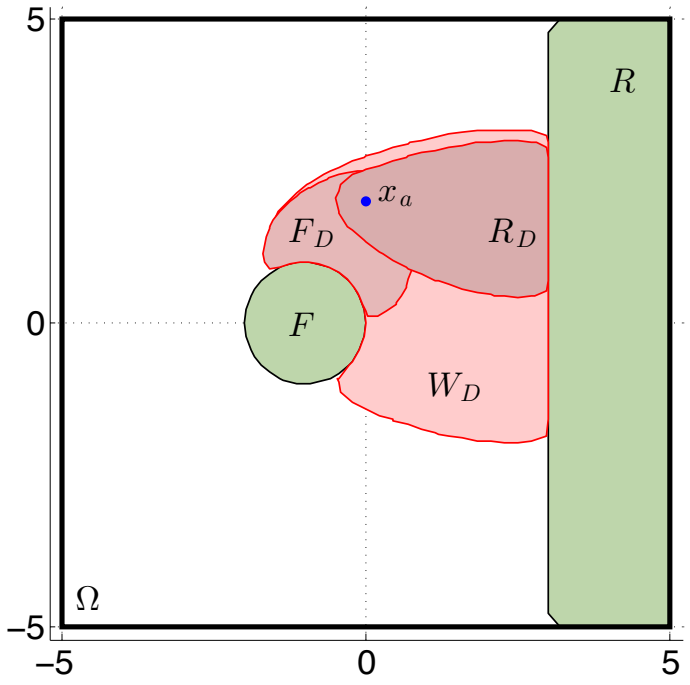


Figure 3.3: **The overall winning region W_D of the defender for $x_a = (0, 2)$, with R_D and F_D super-imposed for comparison.**

the value functions for the two stages of the game both converge to fixed points after about 6 time units. The actual winning regions lie in the 4-D joint configuration space $\mathbf{x}$, but for visualization purpose, slices are shown in 2-D with x_a fixed. The defender-winning sets F_D, R_D and W_D, corresponding to configurations from which the defender can always prevent the attacker from achieving the desired objectives, are shown in Figure 3.3, with the attacker fixed at $(0, 2)$. Observe that W_D is much larger than simply $F_D \cup R_D$, reflecting strategies where the defender uses the time during the flag capture stage to arrive at a configuration that blocks the attacker's return path.

Using the reachable sets, simulations were conducted where the attacker and defender chose controls according to the strategies discussed in Section 3.3.2. Figure 3.4 shows two example scenarios, one for the flag capture stage and the other for the flag return stage. In both cases, the defender started within the winning set and successfully intercepted the attacker. More interesting scenarios involving the interplay

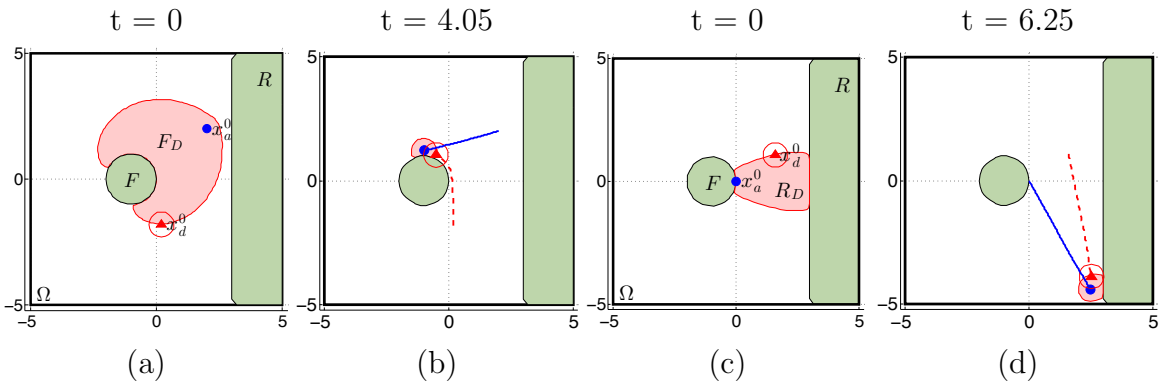


Figure 3.4: **Two simulations showing cases where the defender (red triangle, dashed line) successfully intercepted the attacker (blue circle, solid line). (a) shows a scenario for flag capture only, with the resulting trajectories shown in (b). (c) and (d) show a similar scenario for flag return.**

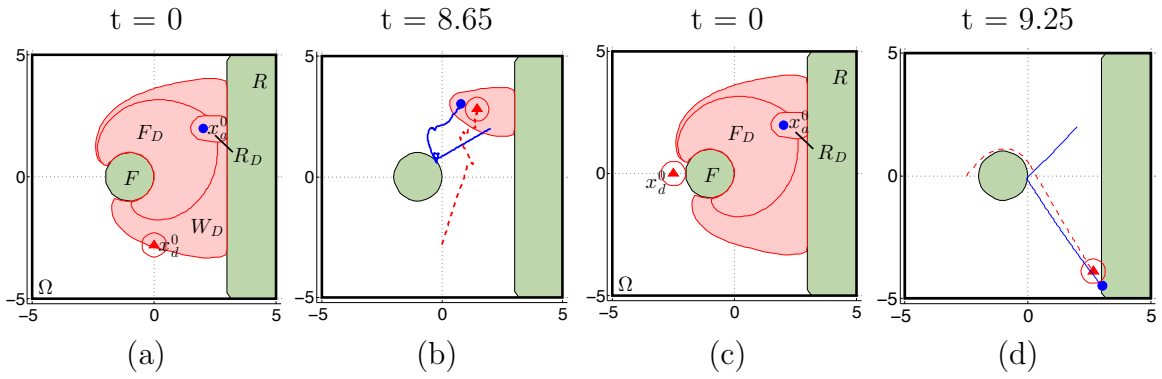


Figure 3.5: **Two scenarios showing combined winning regions W_D for both game modes. In (a) the defender (red triangle, dashed line) started inside W_D and successfully prevented the attacker (blue circle, solid line) from returning with the flag, as seen in (b). (c) shows a case where the defender began outside of W_D, and was unable to prevent the attacker's successful flag capture and return, as shown in (d).**

between the flag capture and flag return stage are shown in Figure 3.5. In Figure 3.5a, the defender started within the overall winning set W_D, but outside the set F_D. The trajectories of the simulation in Figure 3.5b show that the defender did not try to prevent flag capture. Instead it used the time it took for the attacker to reach the flag to get into position to prevent flag return, thereby winning the game. On the other hand, when the defender began outside W_D, as in Figure 3.5c, the simulated trajectory in Figure 3.5d shows that the attacker was able to successfully capture the flag and return to the safe region, without being intercepted by the defender at any point. Note how in this example the attacker did not take the shortest direct path to F, but rather reached F at a point that is closer to R, making the subsequent return path shorter and avoiding the defender.

The computation time of the reachable sets is strongly tied to the grid size and the numerical scheme for obtaining the spatial derivatives. With 45 points in each dimension, each set took approximately 1 hour to compute on an Apple Macbook Pro laptop with a 2.66 Ghz Core i7 processor and 8 GB of RAM. Sets with 25 points in each dimension can be computed in as little as 4 minutes.

It should be noted that some numerical errors are inevitable due to the necessity of solving the HJI PDE on a discrete grid. In particular, the numerical differentiation scheme is poorly equipped to handle sharp set boundaries, corresponding to discontinuities in the spatial derivative. Due to this reason, it was observed that near the intersection of F_D with F, the zero sub-level set sometimes slightly under-approximated the stop capture set, resulting in defender trajectories that began just outside of F_D (below the grid resolution) that nevertheless captured the attacker. More accurate solutions can be obtained with finer spatial discretization, at the cost of higher computational load. Another possible approach is to over-bound the numerical error by some small $\epsilon > 0$ and either choose the $-\epsilon$ or $+\epsilon$ level set boundaries to ensure winning conditions for the attacker and defender, respectively.

In addition, in certain cases the value of J on the interior of $\mathcal{RA}_T(G, A)$, that is, where $J < 0$, may not be correctly preserved, leading to incorrect gradients and correspondingly incorrect optimal inputs. The numerical scheme used here was sometimes susceptible to this error, as the scheme was designed primarily to compute the

boundary of the reachable sets accurately. Thus the scheme does not always correct for certain numerical issues that introduce errors in the value on the interior of the sets, as they do not affect the propagation of the boundary. In these cases, the approximate optimal inputs may be found by storing the optimal inputs at points in the joint state space as the boundary of the set passes over them. This process is described below.

Note that from equation (3.6) the level sets of the infinite horizon value function $\mathcal{V}_\mathcal{T}(\mathbf{x}) = T$ are equivalent to the level sets of the finite horizon value $J(\mathbf{x}, 0)$ for a game over a horizon of T. Thus, the outward normals of the sets are equal, and $\frac{\partial J}{\partial \mathbf{x}} = \frac{\partial \mathcal{V}_\mathcal{T}}{\partial \mathbf{x}}$ on the set boundary. Since the optimal infinite horizon control inputs are

$$\mu(\mathbf{x}) = \arg\min_{u \in U} \max_{d \in D} \frac{\partial \mathcal{V}_\mathcal{T}^T}{\partial \mathbf{x}} f(\mathbf{x}, u, d)$$
$$\gamma(\mathbf{x}) = \arg\max_{d \in D} \frac{\partial \mathcal{V}_\mathcal{T}^T}{\partial \mathbf{x}} f(\mathbf{x}, u, d)$$

the approximate infinite horizon control inputs can be calculated for points in the interior of the reachable set as

$$\mu(\mathbf{x}) = \arg\min_{u \in U} \max_{d \in D} \frac{\partial J(\mathbf{x}, T^*)^T}{\partial \mathbf{x}} f(\mathbf{x}, u, d)$$
$$\gamma(\mathbf{x}) = \arg\max_{d \in D} \frac{\partial J(\mathbf{x}, T^*)^T}{\partial \mathbf{x}} f(\mathbf{x}, u, d)$$

where

$$T^* = \min\{t \mid J(\mathbf{x}, t) = 0\}.$$

As the boundary of the finite horizon reachable set moves during the computation of the value $J(\mathbf{x}, t)$, the appropriate infinite horizon control inputs for $\mathbf{x}$ are captured at the moment when the boundary of the set passes over $\mathbf{x}$.

3.5 Experimental Results

The reachability method presented above was experimentally validated in a series of tests using the BEARCAT platform (without UAVs). The tests were performed in

various sections of the UC Berkeley campus, involving only the flag return stage of the game. As the main purpose was to test the use of the reachable sets, only a single stage of the game was considered in order to shorten testing time and simplify the testing logistics. Each agent was equipped with an HTC Incredible smartphone that displayed the positions of the two agents as well as visualizations of the reachable sets and optimal inputs. Figure 3.6 shows a screenshot of a phone from the attacker perspective.

Tests were conducted with the agents receiving reachability guidance and without. When receiving reachability guidance, each agent could see the opponent's winning set as a function of the agent's own position. As the optimal input is always to move at the maximum speed, the optimal input was displayed as a suggested heading. For example, in Figure 3.6 the reachable set displayed is the set of winning defender positions relative to the current attacker position. The optimal heading is displayed as a circular ring with an arrow in the direction of desired movement, against which the agent can compare the current heading of the phone.

The solution to the game was pre-computed, with the 4-D value function and partial derivatives stored on a PC laptop ground station that communicated with the phones via 3G wireless. Agent GPS positions and heading were transmitted to the laptop, which then performed the appropriate interpolation and transmitted the 2-D reachable set information and optimal direction to each agent as required.

Two sets of experiments were performed to validate the use of reachability information in the game. The first set of trials was played in a small, open area with no obstacles with the agents walking, and was meant to test the feasibility of the phones and the reachability computations. The second set of trials was played on a larger area with a number of buildings serving as obstacles with running agents, and was meant to evaluate the game in a more realistic capture-the-flag game setting.

3.5.1 Small Area Trials

Several trials were conducted in a small, obstacle-free area to evaluate the phone technology and communications architecture. For these tests, the phones communicated

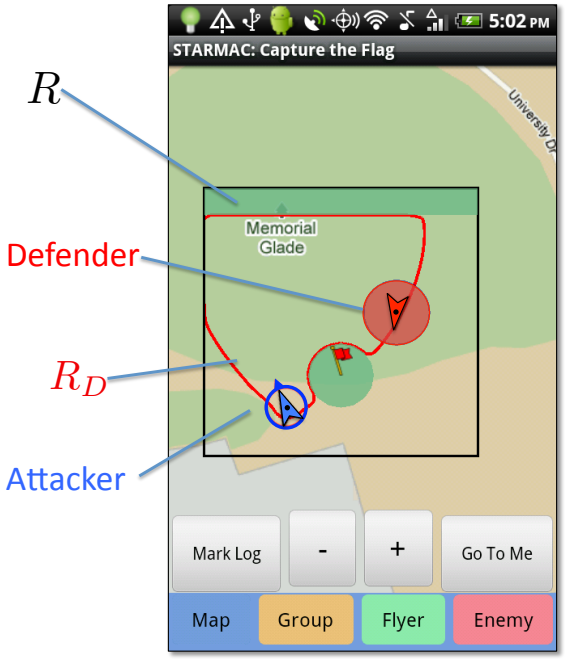


Figure 3.6: **Screenshot of the attacker's phone, showing the agents, the return region R, and the defender's winning region R_D with respect to the current attacker position. The blue ring around the attacker shows the attacker's optimal input, displayed as a direction to travel at max speed.**

with the laptop ground station via WIFI, and the agents were within visual contact of each other at all times. In these tests, the game region was defined as a 50m x 50m square, with the return region defined as a 5m deep strip running the width of the game region at the northern (upper) portion of the region (see Figure 3.7). The speed of both agents was limited to $3m/s$, and the agents were instructed to maintain a walking pace. Trials were conducted where reachability guidance was provided (a) to both agents, and (b) to one agent only. These tests demonstrated that the hardware could provide reachability guidance for the agents in real time, and that the reachability information could be used to play the game.

Reachability guidance for both agents

Four trials were conducted with both agents receiving full reachability information, two with the defender in a winning initial condition and two with the attacker in a

winning initial condition. In all cases, the game played out as predicted, as the agents were able to play according to their optimal inputs, resulting in victory or defeat according to the initial conditions. Figure 3.7 shows GPS data and images recorded during one of these trials. This trial began with the attacker in a winning position. The MATLAB plot for each snapshot shows the position of the attacker (blue circle), the defender (red triangle), the flag region (green circle), and the return region (green rectangle). The attacker was attempting to reach R at the upper (northernmost) extreme of the game domain, defined by the green rectangle. The pink region shows the defender's winning region R_D plotted for the current attacker position. Figure 3.7a shows the initial condition for this trial, where the defender was outside of R_D. The attacker therefore was guaranteed to win (reach R) by following the optimal input as displayed by the phone. The remaining sequence illustrates the game being played out. Notice that although the agents were attempting to follow the displayed inputs as best they could, the actual path they traced out was not smooth, due to time delays in communication, GPS tracking error, and errors in following the directed heading. Nonetheless, the agents were able to follow the phone instructions and complete the game.

Reachability guidance for a single agent only

As the game is deterministic, when both agents play optimally the result of the game is determined completely from the initial conditions. A more interesting case is when only one agent has access to the reachability information, and the other agent is left to play freely, without guidance. For the purposes of this discussion, the agent with reachability information will be referred to as the aided agent, and the agent without will be referred to as the unaided agent.

Several trials were carried out in this manner, varying the aided and unaided agents as well as the initial conditions. Trials where the aided agent was in a winning position proved predictably uninteresting, as simply following the optimal heading was sufficient to ensure victory. More illuminating results emerged from tests where the aided agent began in a losing configuration.

Trials where the defender was the aided agent in a losing configuration did not

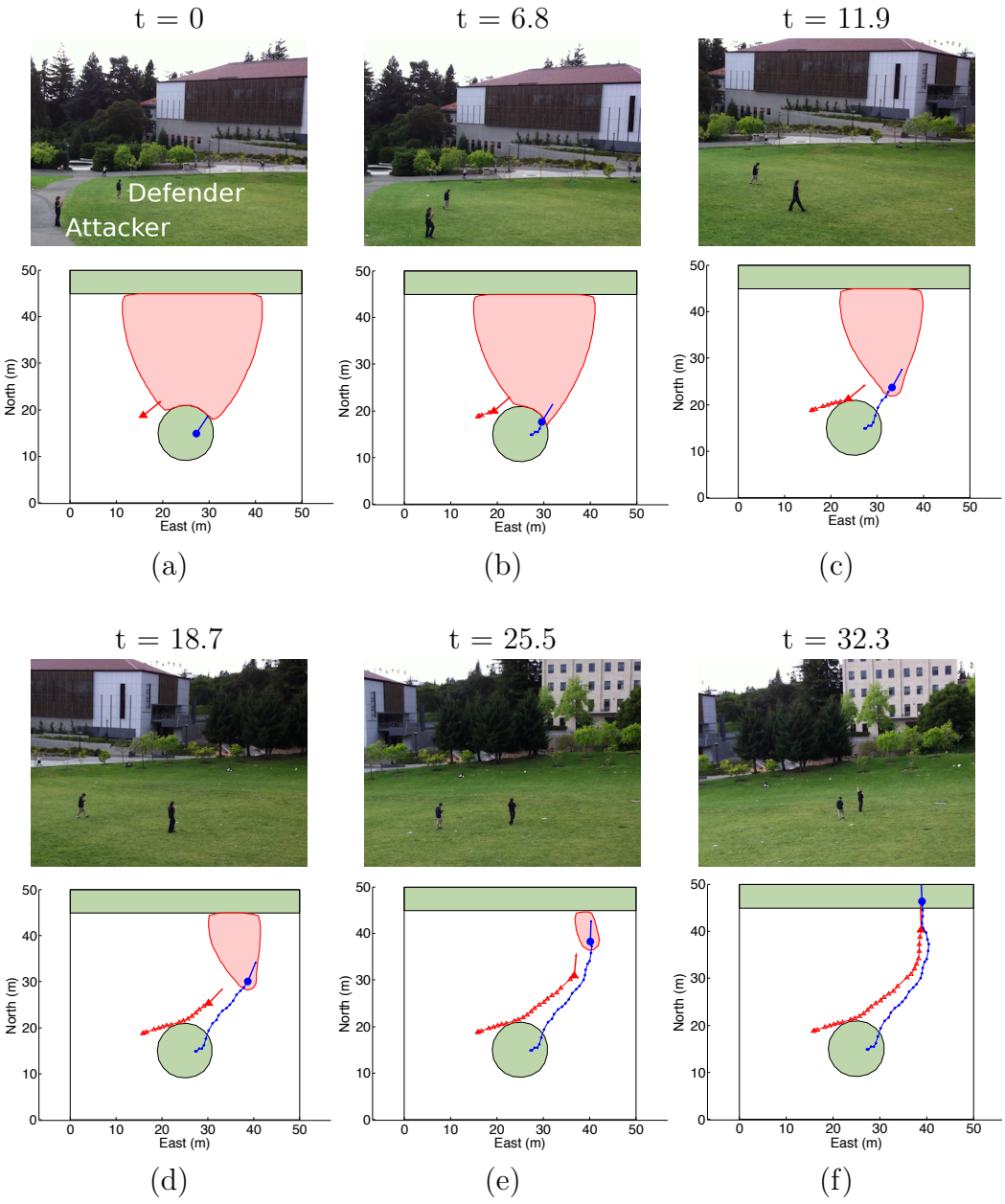


Figure 3.7: **Sequence of agent positions and reachable set data during an experiment where both agents had access to reachability information. The attacker (blue circle) was able to safely reach the return region and avoid the defender (red triangle) by following the optimal heading recommendations.**

produce unexpected results. In such cases the attacker could elude the defender by simply heading away from the defender toward the corner of the return region. However, an interesting new strategy emerged when the attacking agent was the aided agent and in a losing initial position. In the first trial, the defender chose an incorrect intercept angle and was unable to intercept the attacker. However, as the optimal headings reported by the reachability information is deterministic, on the subsequent trial the defender made an adjustment and was able to successfully intercept the attacker. This led the attacker to innovate by introducing an element of uncertainty into the game by "faking out" the defender. This is illustrated in Figure 3.8.

For the third trial, shown in Figure 3.8, the attacker did not initially follow the optimal heading. Instead, the attacker moved in the opposite direction in a slightly erratic manner, as seen in Figure 3.8b. The defender was forced to move in an attempt to come between the attacker and R. However, the unpredictable movement of the attacker caused the defender to exit the winning region R_D, as seen in Figure 3.8c, allowing the attacker to revert to following the optimal heading and subsequently winning the game. The visual display of reachability information allowed the attacker to instantly ascertain whether or not the "fake-out" maneuver had been successful. Once the attacker could see that the defender had exited R_D, the optimal heading could be directly followed for guaranteed victory. Two more trials were conducted, and in each case the attacker was able to successfully maneuver past the defender using this strategy. Note however that this maneuver was only successful with asymmetric reachability guidance. Subsequent tests of the "fake-out" maneuver when both agents received reachability guidance showed that the reachability guidance for the defender was able to correctly maintain the defender in a winning blocking position.

This maneuver highlights an important aspect of the reachability tools. Although the attacker could not rely upon the reachability optimal heading when starting the game in a losing configuration, the visualization gave the agent enough information to decide when the heading recommendation was valid. The solutions not only provided optimal inputs to the agents, but the reachable set visualizations also provided contextual information that let the agent know when the automation recommendations could and could not directly provide a winning solution.

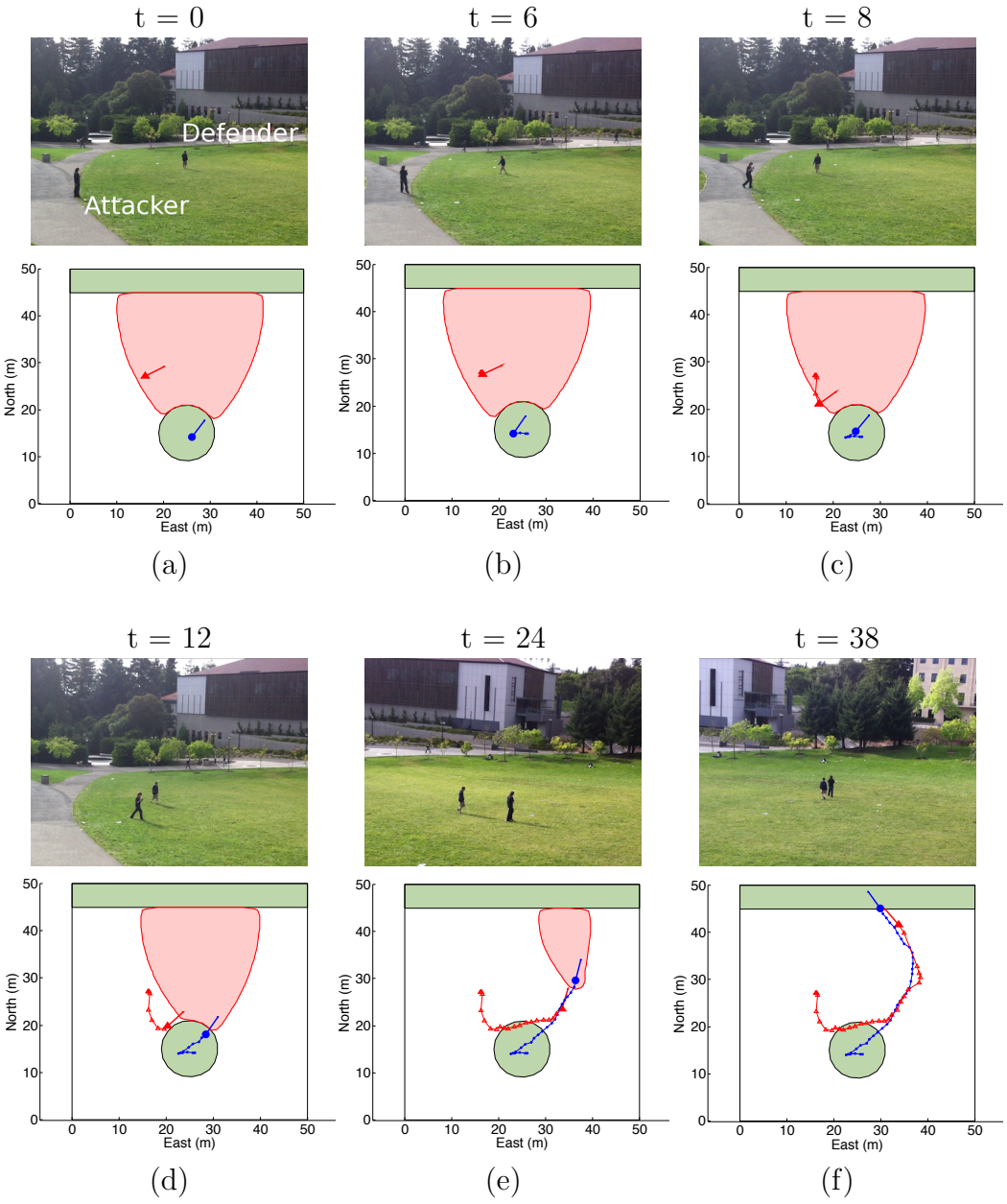


Figure 3.8: **Sequence of agent positions and reachable set data where the defender (red triangle) did not have reachability information, but did begin the game within the defender winning region. The attacker (blue circle) moved unpredictably, causing the defender to move outside the winning region, resulting in an attacker victory.**

3.5.2 Large Area Tests

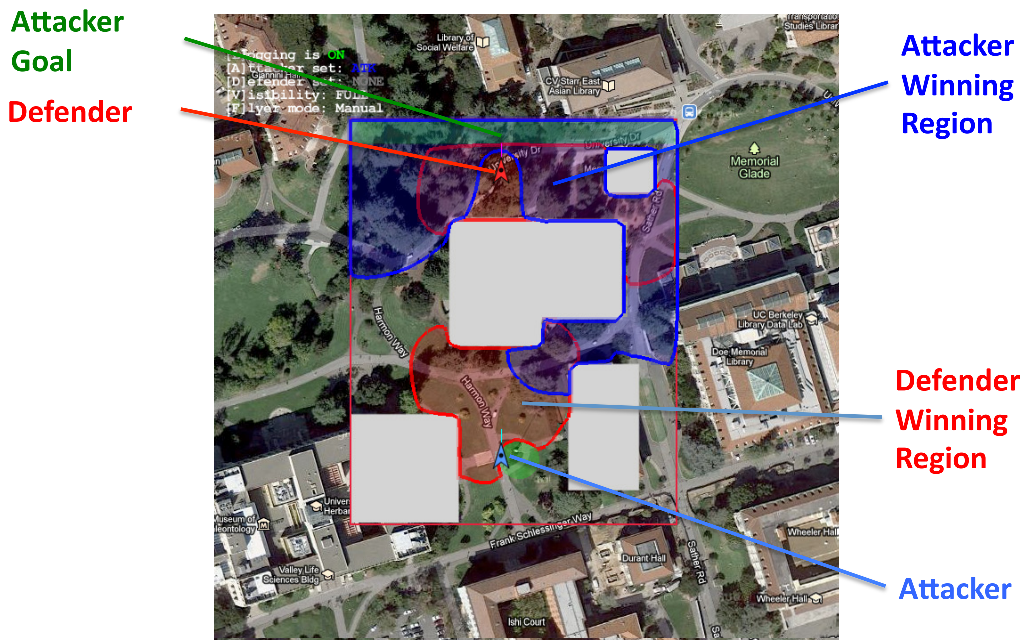


Figure 3.9: **Screenshot of the laptop showing the game area with building obstacles, the agents, and their respective winning regions with respect to the other agent (map image courtesy of `maps.google.com`).**

The second set of trials was carried out in a larger region, with buildings serving as obstacles to movement. Although agent positions were still displayed on the phones, the agents could not always see each other visually. The game area was approximately 200m x 180m, with a number of buildings serving as obstacles. Agent speeds for these trials were capped at $5m/s$, and the agents were permitted to run while playing. Figure 3.9 shows a screen capture from the ground station, displaying the game region with building obstacles, the positions of the two agents, and the respective winning regions. The blue region displays $R_A(x_d)$, the set of attacker positions which are winning relative to the current defender location, and the red area is $R_D(x_a)$, the set of defender positions that are winning relative to the current attacker location.

The main purpose of these experiments was to validate the use of the reachability information in a larger, more realistic game environment with greater agent speeds.

The results of the experiments demonstrated that the reachability support was still useful and valid in the presence of obstacles, and at higher speeds, given that the assumptions made in the computations held. However, the experiments also showed that complications as a result of the more realistic game conditions can introduce some uncertainty into the game, which cannot be accounted for via the current framework. Although the reachability information was still used in playing the game, these factors combined to reduce the direct utility of the reachability guidance as compared to the earlier trials.

Communications delay and failure were major sources of uncertainty. In these more complex tests, phone-to-phone and phone-to-laptop communications occurred over the 3G wireless network, not WIFI. The 3G wireless link was not always perfectly reliable, as signal quality could degrade when moving between buildings or simply due to unknown external factors. A number of trials were affected by unpredictable delays or packet loss, rendering agent positions unreliable. In these cases, the agents could not rely on the phones to correctly report the location of the other agent, rendering the reachability information less useful.

The terrain was another factor, as the game was played in a much more varied area than before. Although large features such as buildings could be easily incorporated as obstacles into the game, smaller terrain features such as fences and ledges could not be easily identified from overhead imagery, and it was hard to identify which areas of vegetation were passable and which were not. This meant that the optimal headings could not always be directly followed, as certain small obstacles were not accounted for. A mitigating factor is that, since there was one major building between the flag and the return region, the agents' choices were essentially between moving either left or right around the building. As a result of this the optimal headings were mostly ignored by the agents, and the reachability information was instead used as a "go/no-go" tool, to decide when the attacker could safely make an attempt to run toward the target set.

Although these difficulties combined to reduce the efficiency and accuracy of the reachability guidance, the reachability solutions nonetheless played a useful role in the game. In fact, the uncertainty introduced by these factors made for a much

more exciting game with more room for human innovation. As in the "fake-out" maneuver, the attacker could win even from a defender-winning initial configuration by maneuvering the defender into a losing position. Since both agents had reachability guidance this could only be accomplished by taking advantage of the uncertainties in the agent positions. The fact that the agents did not have perfect information helped the attacker to out-maneuver the defender. However, it also created an additional difficulty for the attacker in being able to recognize when the defender had, in fact, been out-maneuvered.

All of these factors combining had the interesting result of making gameplay much more dependent on deception and managing opponent information, with the reachability tool used as a decision-aid rather than a primary guide. Figure 3.10 shows an example of such a game. The initial condition, seen in Figure 3.10a, was actually an attacker-winning configuration. However, the agents were not in line-of-sight of each other, and with the communications issues the attacker could not completely trust the information displayed. Instead, both agents moved to get better visibility, shown in Figure 3.10b, bringing the two agents into direct view of each other. Note that the attacker stayed close to the southwest corner of the central building, which, although not visible from the overhead views, was a fairly cluttered area. This caused the defender to move south (down) in an attempt to better determine the location of the attacker, gambling on being able to move back into a winning configuration before the attacker could make a decision to go for the target set. Instead, this move rendered the defender more fully visible to the attacker, who was also able to visually establish that the defender was correctly reported by the phone to be well outside of $R_D(x_a)$. The attacker, now confident of victory, moved toward the target in the opposite direction, as seen in Figure 3.10c, and successfully arrived at R without being intercepted, as shown in Figures 3.10d-f.

3.6 Discussion

The reachability-based approach presented in this chapter gives a complete solution to the game of 1 vs. 1 capture-the-flag, where a single agent is attempting to reach a

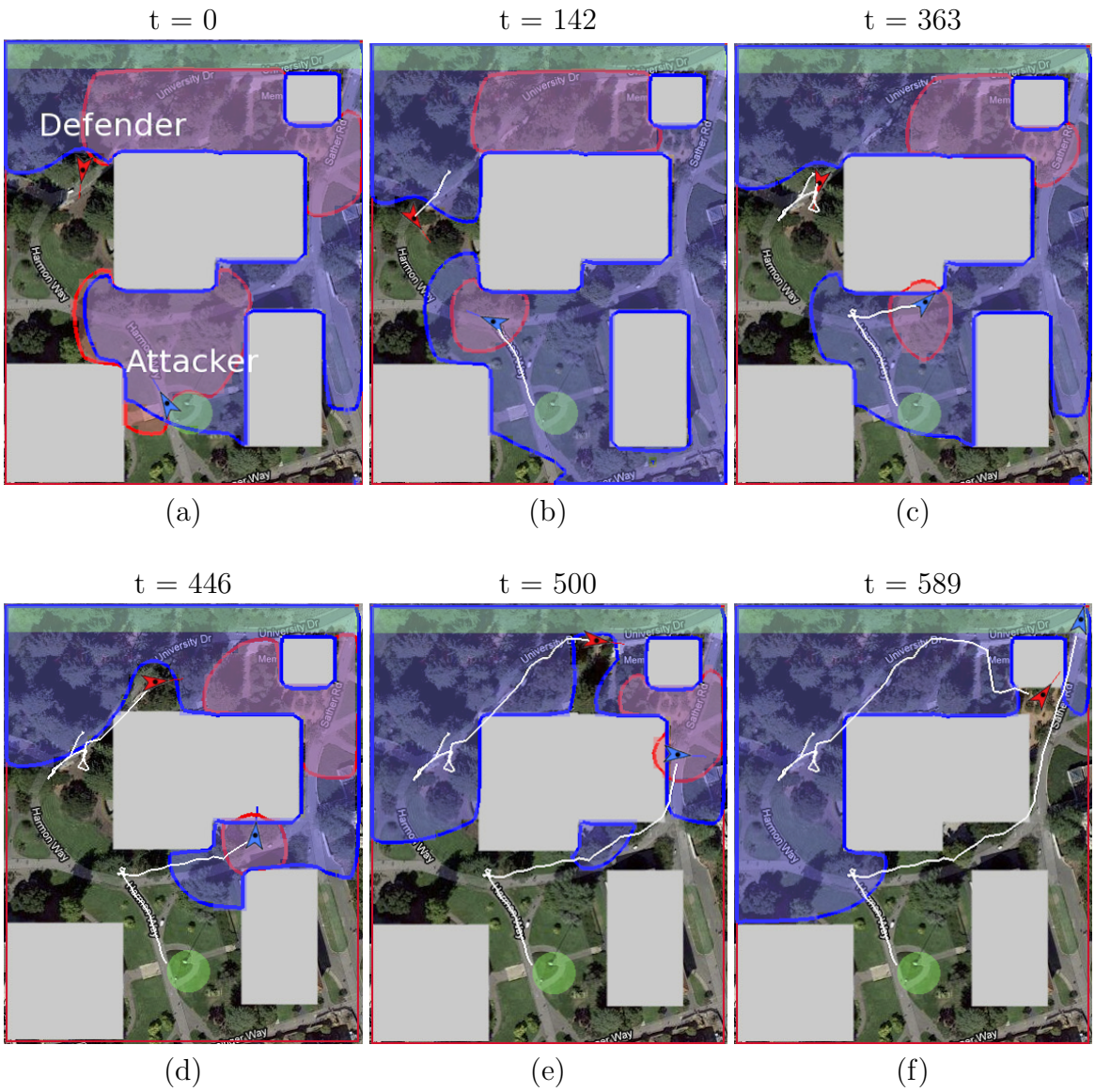


Figure 3.10: **Sequence of screenshots from the ground station laptop showing the game played in a large area with obstacles, at full running speeds, with the attacker in blue, starting from the bottom of the screen, and the defender in red, at the top (map images courtesy of `maps.google.com`).**

target and subsequently return to safety while avoiding another agent. The solution presented gives the set of initial conditions from which each agent can win, and also gives a method for computing optimal inputs for each agent. The reachability method is flexible and fairly powerful, and extends to a variety of other applications beyond pursuit-evasion-style games, as will be shown in Chapter 6, where it is used in the context of UAV control. The primary strengths of this approach from a theoretical standpoint are that it finds guaranteed, optimal solutions, that it can be used for arbitrary game geometries, and that it naturally handles multi-stage games.

Although not currently implemented, the approach discussed above can naturally be generalized to a symmetric, two-sided game of capture-the-flag, where the two agents may choose to switch between playing the roles of the attacker and defender. This scenario would require the reachability value function to be computed for each combination of roles. Once computed, these value functions can be directly used by each agent to determine which choice of roles leads to victory.

There are two main drawbacks to the numerical reachability approach presented in this chapter. The first is the need to compute the HJI value function numerically on a grid. As the number of agents grows, the number of grid nodes grows exponentially in the continuous states of the agents. Games with larger numbers of agents require different methods, as will be addressed in the following chapters. In addition, grid discretization limits the resolution of features that can be contained in the game space, for example small obstacles such as trees and fences could not be included in the obstacle set for the experiments. The second factor with this approach is the limited ability to directly address games with uncertainty and limited information. This would require formulations over sets of possible states, greatly increasing the computational demands on any solution.

Nonetheless, the experimental validation has shown that despite these limitations, the reachability solutions can be useful tools for human agents. The reachability solutions not only provide recommendations as to the optimal direction of movement, but also give a sense of context as to when those recommendations can be relied upon to assure victory. As the large area tests showed, these tools are useful even when the assumption that the state of all agents is constantly accessible cannot be relied upon.

The following chapters will illustrate methods that address some of the weakness of this approach in scenarios with many agents and with uncertainty.

Chapter 4

1 vs. Many: Open-Loop Reachability

Consider the problem of a single attacker attempting to reach a target in the presence of 1 or more defenders. Such a scenario is relevant not only to reach-avoid games like capture-the-flag, but also to motion planning problems where an agent must navigate to a goal in the presence of moving obstacles. To guarantee safety in worst-case scenarios, such obstacles may be considered as adversarial defenders attempting to collide with or capture the agent. By treating the agent as an attacker and the obstacles as defenders, a safe solution path for the attacker, if found, guarantees that the moving obstacles will never collide with the agent.

The previous chapter presented a method for computing solutions to reach-avoid games with two agents that allows the sets of winning configurations for each agent, as well as optimal inputs for the agents, to be found. As the number of agents in a game grows, and with it the size of the grid required to represent the state space, the numerical Hamilton-Jacobi-Isaacs solutions become intractable. Though optimal solutions via HJI equations cannot be found, other methods can be used to quickly provide feasible inputs.

This chapter presents an algorithm for computing a guaranteed safe path for a single attacking agent to reach a sequence of target sets, while avoiding interception by one or more defenders, in domains with arbitrary obstacle configurations. The

problem is posed as an open-loop differential game (also known as a *static game* [55]). In an open-loop game, the agents pick their inputs *before* the game is played, then play out their input sequences in an open-loop sense. As in the preceding chapter, the method described here is conservative from the perspective of the attacker, although in this case the order of play does affect the solution. The attacker's inputs are selected and made known to the defenders before the defenders select their input sequences. The conservatism implies that sometimes a winning path may not be found, but when a path is found, it is guaranteed to reach the target and can be computed in real time.

In contrast to the HJI reachability approach, the simpler information pattern of the open-loop game (the choice of input sequences depends only on the initial condition) reduces the computational task to solving low dimensional Hamilton-Jacobi-Bellman equations in the state space of each agent, rather than a high dimensional HJI equation in the joint state space of all the agents. This reduction is particularly effective for games with many agents. Furthermore, the HJB formulation allows the use of efficient and robust numerical algorithms, such as the fast marching method (FMM), allowing solutions to be found in real time. However, the current formulation does not allow the strategies of the defenders to be explicitly computed.

The organization of this chapter is as follows. Section 4.1 describes the open-loop game formulation of the safe motion planning problem and its solution technique using a modified fast marching method. For clarity, a single-stage problem with a single target set and a single defender is first explored and then extended to the multiple defender case. The formulation is then naturally extended to multi-stage games in Section 4.2, in which the agent is given a sequence of target sets to be visited in order. Simulation results are presented in Section 4.3. Section 4.4 concludes with a discussion of the algorithm and results. The approach discussed in this chapter was first presented in [33] and [34].

4.1 The Open-Loop Reach-Avoid Game

The analysis of the open-loop reach-avoid game begins with an example for a single attacker and defender with a single target region, which is then be generalized to multiple defenders and sequential targets. As in Chapter 3, the control computation is performed on behalf of the attacker, with the defender playing the role of a disturbance that must be conservatively protected against. However, in the open-loop game the order of play does affect the solution.

The game is therefore played in the following manner: first, the attacker decides upon an input sequence, attempting to reach the target while considering the worst-case actions of the defender. Then the defender chooses its input sequence, with full knowledge of the attacker's inputs. If the attacker is able to find an input sequence that allows it to reach the target even with this advantage given to the defender, then the attacker is guaranteed to reach the target no matter what the defender does. This section formulates the open-loop pursuit-evasion game, and then presents an algorithm which finds the solution to the game on behalf of the attacker by computing the set of points reachable by the attacker while avoiding interception by the defender. This solution is then generalized to multiple defenders.

4.1.1 Open-Loop Game Formulation

Both the attacker and the defender are confined to a domain $\Omega \subset \mathbb{R}^2$. Ω may include holes, representing obstacles in the state space. Let $\mathbf{x} = (x_a, x_d) \in \mathbb{R}^4$ represent the joint states of the attacker and defender, respectively, at a given time $t \geq 0$, with initial states $\mathbf{x}(0) = (x_a^0, x_d^0) = \mathbf{x}^0$. The dynamics of the two agents are as follows:

$$\begin{aligned} \dot{x}_a &= u, \qquad u \in U, \\ \dot{x}_d &= d, \qquad d \in D. \end{aligned} \tag{4.1}$$

where u and d are the inputs of the attacker and defender, respectively and U, D are the sets of control values for each agent with $U = \{u \mid ||u||_2 \leq v_{a,max}\}$ and $D = \{d \mid ||d||_2 \leq v_{d,max}\}$. The maximum speeds $v_{a,max}$ and $v_{d,max}$ are not necessarily

constant throughout Ω. Let $\mathbb{U}_o = \{\mu\colon [0,\infty) \to U\}, \mathbb{D}_o = \{\gamma\colon [0,\infty) \to D\}$, be the sets of admissible *open-loop* input sequences. For this chapter, the attacker and defender strategies μ and γ will be limited to the set of open-loop input sequences, with $\mu \in \mathbb{U}_o$ and $\gamma \in \mathbb{D}_o$.

Each input sequence will result in the agent moving along a particular path. A path $x_a(\cdot)$ is said to be admissible if it is a solution to equation (4.1) for some $\mu \in \mathbb{U}$ such that $x_a(t) \in \Omega$, $\forall t \geq 0$. Admissible paths $x_d(\cdot)$ for the defender may be defined similarly. To simplify notation, the dependency of the paths $x_a(\cdot)$ and $x_d(\cdot)$ on the strategies μ and γ and the initial conditions x_a^0 and x_d^0 will not be written.

Let $\mathcal{T} \subset \Omega$ denote a closed target set. The attacker's objective is to reach $\mathcal{T}$ as quickly as possible, while avoiding interception by the defender. The attacker is intercepted if it enters some capture set attached to the defender: $x_a(t) \in C(x_d(t))$ for some time $t \geq 0$ and joint capture set $C = \{\mathbf{x} \mid ||x_a - x_d||_2 \leq r_c\}$ with capture radius r_c. For simplicity of notation the capture set in $\mathbb{R}^2$ as a function of the defender location $C(x_d(t))$ will be written as $C(t)$.

The attacker is attempting to minimize the time required to reach the target, and the defender attempts to maximize, and the game is completely determined by the *initial positions* of the agents. Thus the open-loop value for the game with respect to an initial condition $\mathbf{x}^0$ is

$$\overline{\mathcal{V}}_{\mathcal{T}}(\mathbf{x}^0) = \min_{\mu\in\mathbb{U}_o} \max_{\gamma\in\mathbb{D}_o} \{t \mid x_a(t) \in \mathcal{T}, x_a(s) \notin C(s), \forall s \in [0,t]\}. \tag{4.2}$$

It is assumed here that the minimum of the empty set is infinity. Note that the value so defined is nearly identical to the infinite time horizon HJI reachability value, with the only difference being the type of strategies.

Equation (4.2) is a conservative estimate of the value from the attacker's perspective and serves as an upper bound on the value. If $\overline{\mathcal{V}}_{\mathcal{T}}(\mathbf{x}^0) < \infty$, the attacker is guaranteed an input sequence $\mu^* \in \mathbb{U}_o$ that will reach $\mathcal{T}$ at time $t = \overline{\mathcal{V}}_{\mathcal{T}}(\mathbf{x}^0)$ while avoiding interception, *for any* action by the defender. On the other hand, $\overline{\mathcal{V}}_{\mathcal{T}}(\mathbf{x}^0) = \infty$ only means that knowing the attacker's choice of inputs, the defender has a choice of inputs resulting in interception of the attacker. Solving the open-loop

game requires computing $\overline{\mathcal{V}}_{\mathcal{T}}(\mathbf{x}^0)$ and the safe inputs/path from a given x_a^0 to $\mathcal{T}$ if such a path can be found.

4.1.2 Safe-Reachable Sets

The open-loop value $\overline{\mathcal{V}}_{\mathcal{T}}(\mathbf{x}^0)$ can be found by characterizing the set of points in the state space that the attacker can reach no matter what the defender does. Given an initial condition $\mathbf{x}^0$, a point $y \in \Omega$ is *safe-reachable* if there exists some $\mu \in \mathbb{U}_o$ and $t \geq 0$ such that $x_a(t) = y$ and $x_a(s) \notin C(s)$ for all $s \in [0, t]$ and $\gamma \in \mathbb{D}_o$. Such a path $x_a(\cdot)$ will be referred to as a *safe-reachable path*. The *safe-reachable set* $\mathcal{S}$ of the attacker is then defined as

$$\mathcal{S} = \{y \in \Omega \mid y \text{ is safe-reachable}\}.$$

Clearly, $\overline{\mathcal{V}}_{\mathcal{T}}(\mathbf{x}^0) < \infty$ if and only if there exists a safe-reachable point in the target set, namely $\mathcal{S} \cap \mathcal{T} \neq \emptyset$.

The safe-reachable set can be found in the following way. Define the minimum time-to-reach function $\varphi \colon \Omega \to \mathbb{R}$ for the attacker, constrained to the set $\mathcal{S}$ as:

$$\varphi(y) = \min\{t \mid x_a(t) = y, x_a(s) \in \mathcal{S}, \forall s \in [0, t]\}. \tag{4.3}$$

Similarly, the minimum time-to-reach function $\psi \colon \Omega \to \mathbb{R}$ for the defender in Ω is:

$$\psi(y) = \min\{t \mid x_d(t) = y, x_d(s) \in \Omega, \forall s \in [0, t]\}. \tag{4.4}$$

and a minimum time-to-capture function can be defined as

$$\psi^c(y) = \min\{t \mid y \in C(x_d(t)), x_d(s) \in \Omega, \forall s \in [0, t]\}. \tag{4.5}$$

The minimum time-to-capture function is particularly important in that it represents, for a point y, the minimum time required for the defender to capture the attacker if $x_a = y$. $\mathcal{S}$ is therefore

$$\mathcal{S} = \{y \mid \varphi(y) < \psi^c(y)\}. \tag{4.6}$$

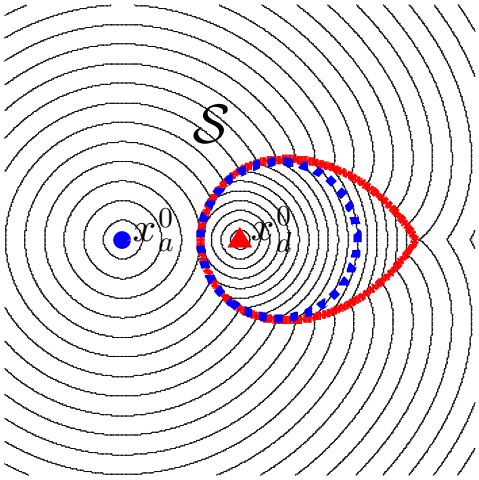


Figure 4.1: **$\mathcal{S}$ is the subset partitioned by the solid red curve containing x_a^0. Level lines of φ on $\mathcal{S}$ and ψ_c on $\Omega\backslash\mathcal{S}$ are plotted. The dotted blue circle is the set of equal time-to-reach points of the attacker and the defender when capture is not considered.**

Figure 4.1 illustrates $\mathcal{S}$, φ and ψ for an example where the attacker is twice as fast as the defender. In this case $C(x_d(t)) = x_d(t)$. As is shown here, in general the safe-reachable set is not necessarily the same as the equal time-to-reach points for the two agents, as there may be points that are only reachable for the attacker if it moves past the defender. For comparison, the Apollonius circle showing the equal time-to-reach points of the attacker and the defender is overlaid.

The value $\overline{\mathcal{V}}_{\mathcal{T}}$ can now be found using φ, via the following result:

Theorem 1. $\overline{\mathcal{V}}_{\mathcal{T}}(\mathbf{x}^0) = \min\{\varphi(y) \mid y \in \mathcal{T}\}$.

Proof. First, consider the case in which $\overline{\mathcal{V}}_{\mathcal{T}}(\mathbf{x}^0) = \infty$. This implies that for any attacker input sequence $\mu \in \mathbb{U}_o$, there exists a defender input sequence $\gamma \in \mathbb{D}_o$ such that the defender intercepts the attacker before the target set $\mathcal{T}$ is reached. Therefore, $\mathcal{S} \cap \mathcal{T} = \emptyset$, and there is no path that reaches the target without being captured, and from equation (4.3), $\varphi(y) = \infty$ for all $y \in \mathcal{T}$, as desired.

If $\overline{\mathcal{V}}_{\mathcal{T}}(\mathbf{x}^0) < \infty$, let $x_a^*(\cdot)$ be the path corresponding to the input

$$\mu^* = \arg\min_{\mu \in \mathbb{U}_o} \max_{\gamma \in \mathbb{D}_o} \{t \mid \mathbf{x}(t) \in \mathcal{T}, x_a(s) \notin C(s), \forall s \in [0,t]\}.$$

Observe that $x_a^*(t) \in \mathcal{S}$ for all $t \leq \overline{\mathcal{V}}_{\mathcal{T}}(\mathbf{x}^0)$; indeed, if $x_a^*(s) \notin \mathcal{S}$ for some $s \leq \overline{\mathcal{V}}_{\mathcal{T}}(\mathbf{x}^0)$, there then exists some defender input sequence $\gamma \in \mathbb{D}_o$ that successfully intercepts the attacker, implying the contradiction $\overline{\mathcal{V}}_{\mathcal{T}}(\mathbf{x}^0) = \infty$. Thus, $\overline{\mathcal{V}}_{\mathcal{T}}(\mathbf{x}^0)$ is the minimum time-to-reach to the target set $\mathcal{T}$ along admissible paths contained in the set $\mathcal{S}$, which is of course $\min\{\varphi(y) \mid y \in \mathcal{T}\}$. □

Given this, the problem of evaluating $\overline{\mathcal{V}}_{\mathcal{T}}(\mathbf{x}^0)$ now depends on computing φ. Observe that, as a consequence of this, the problem of computing a value for $\overline{\mathcal{V}}_{\mathcal{T}}(\mathbf{x}^0)$ in the high dimensional *joint state space* of the two agents has been reduced to solving for φ in the lower dimensional state space of *the attacking agent only*. This result not only speeds the computation for the game with two agents, but has important consequences for finding solutions when additional defenders are introduced into the game.

From optimal control theory, the minimum time-to-reach function is the viscosity solution to a Hamilton-Jacobi-Bellman (HJB) equation [54]. For the particular dynamics at hand, the minimum time-to-reach φ as defined in equation (4.3) satisfies the HJB equation

$$-\min_{u \in U} \{\nabla \varphi(y) \cdot u\} = 1, \tag{4.7}$$

with boundary conditions

$$\varphi(x_a^0) = 0, \tag{4.8}$$

$$\varphi(y) = \infty, \quad y \in \Omega \backslash \mathcal{S}. \tag{4.9}$$

While problems of the form of equations (4.7)-(4.9) can be solved in many ways when $\mathcal{S}$ is known *a priori*, the challenge here lies in the fact that $\mathcal{S}$ itself depends on φ. Instead, both the set $\mathcal{S}$ and the values φ will be computed simultaneously, using an algorithm inspired by the fast marching method (FMM).

4.1.3 Modified Fast Marching Method

The safe-reachable set and minimum time-to-reach functions are computed on a grid using a modified version of the fast marching method (FMM). The structure of the algorithm is as follows. First, the minimum time-to-reach for every point in Ω is

computed for the defender. Simultaneously, the minimum time-to-capture is also computed. The set of safe-reachable points for the attacker is found in the following way. Beginning with the initial attacker position, the set is expanded outward, computing the time-to-reach for neighboring points and incorporating only points that can be reached by the attacker before they can be captured by the defender. The procedure terminates when all remaining points along the boundary are either outside of Ω or cannot be reached by the attacker before being captured by the defender, and the results give both the safe-reachable set and the minimum time-to-reach for both agents for all points in the set.

Both computing the time-to-reach for the agents and expanding the set make use of FMM. FMM [56, 57, 58], is a single-pass method used to numerically approximate the *Eikonal equation*

$$v(y)\|\nabla\varphi(y)\| = 1, \quad y \in \Omega \setminus \mathcal{O}, \tag{4.10}$$

with Dirichlet boundary conditions $\varphi = \infty$ on $\partial\Omega$ and $\varphi(y) = b(y), \forall y \in \mathcal{O}$, for some closed set $\mathcal{O} \subset \Omega$ and some boundary condition $b(y)$.

The Eikonal equation can be used to compute the minimum time-to-reach from an initial condition or to a target set. Note that equation (4.10) is equivalent to equation (4.7) for the system dynamics here and if the boundary condition in $\mathcal{O}$ is $g(y) = 0, \forall y \in \mathcal{O}$. That is, if v is a positive scalar function representing the maximum speed, then the solution φ to equation (4.10) gives the minimum time-to-reach starting from point y and ending in the region $\mathcal{O}$, or starting from $\mathcal{O}$ and ending at y.

The value φ in equation (4.10) is approximated by a grid function $\varphi_{i,j}$ on a uniform 2-D Cartesian grid $\mathcal{G}$, where $\varphi_{i,j} \approx \varphi(y_{i,j})$, $y_{i,j} = (ih, jh) \in \mathcal{G}$ and h is the grid spacing. To simplify the treatment of the boundary conditions, assume that $\partial\Omega$ is well-discretized by the grid points $\partial\mathcal{G} \subset \mathcal{G}$. The solution to equation (4.10) is found via the finite difference approximation,

$$v(y_{i,j})\sqrt{(\varphi_{i,j} - \min\{\varphi_{i\pm1,j}, \varphi_{i,j}\})^2 + \quad (\varphi_{i,j} - \min\{\varphi_{i,j\pm1}, \varphi_{i,j}\})^2} = 1 \tag{4.11}$$

The solution to equation (4.11) for $\varphi_{i,j}$ can be found through the quadratic formula,

and is called $\varphi^*_{i,j} = \varphi^*_{i,j}(\varphi_{i\pm1,j}, \varphi_{i,j\pm1}, v(y_{i,j}))$. The scheme in equation (4.11) is consistent and stable, and converges to the viscosity solution of equation (4.10) as $h \to 0$ [59].

The FMM algorithm computes the solution to equation (4.11) for the entire grid by sequentially computing the value for each grid node in a particular order. The value for each node is computed only a small number of times, resulting in significant computational savings over the more general HJI solution methods referenced in Chapter 3. At each iteration, FMM partitions the grid $\mathcal{G}$ into Accepted (nodes where the approximation is solved and value is known), Narrow-Band (candidate nodes to be added to Accepted) and Far Away (neither Accepted nor Narrow-Band). The key principle exploited by the FMM is that of "causality" [56], which states that the solution at a node only depends on adjacent nodes that have smaller values. This defines an ordering of the nodes, in increasing values of $\varphi_{i,j}$; this ordering is realized by adding the smallest valued candidate in Narrow-Band into Accepted at each step, until all nodes are in Accepted. Essentially the algorithm begins with the Accepted set as the known boundary conditions, then expands the set, computing the value for points near the boundary (the Narrow-Band) until the values for all points on $\mathcal{G}$ have been found. Schematically, the generic FMM is executed as follows:

1. Set $\varphi_{i,j} = \infty, \forall y_{i,j} \in \mathcal{G}$ and label them as Far Away.

2. Set $\varphi_{i,j} = b(y_{i,j}), \forall y_{i,j} \in \mathcal{O}$, and label them as Accepted.

3. For all nodes $y_{i,j}$ adjacent to a node in Accepted, set $\varphi_{i,j} = \varphi^*_{i,j}$ and label them as Narrow-Band.

4. Choose a node $y^{\min}$ in Narrow-Band with the smallest $\varphi_{i,j}$ and label it as Accepted.

5. Set $\varphi_{i,j} = \varphi^*_{i,j}$ for all non-Accepted nodes adjacent to $y^{\min}$ from the previous step, and (re-)label them as Narrow-Band.

6. Repeat from step 4, until all nodes are labeled Accepted.

By choosing $\mathcal{O}$ to represent some small neighborhood of a point y, such that at least one $y_{i,k}$ falls within $\mathcal{O}$, the time-to-reach of all points to or from y can be computed. The algorithm terminates in a finite number of iterations, since the total number of nodes is finite. On a grid with M nodes, the complexity is $O(M \log M)$ and the algorithm naturally extends to three or higher dimensions [57].

A modified version of FMM can be used to compute the solution to equations (4.7)-(4.9) along with the set $\mathcal{S}$. Assume that equation (4.4) is approximated by $\psi_{i,j} \approx \psi(y_{i,j})$ at each $y_{i,j} \in \mathcal{G}$, calculated using the unmodified FMM. For any capture set C, a capture mask $\tilde{C}(y) = \{x \mid y \in C(x)\}$ can be defined. Then the time-to-capture for any node $y_{i,j}$, denoted $\psi^c_{i,j}$, can be found as $\min_{x \in \tilde{C}(y_{i,j})} \psi_{i,j}(x)$. The key observation is that *any node $y_{i,j}$ captureable by the attacker in time $t = \varphi_{i,j} \geq \psi^c_{i,j}$ cannot belong in $\mathcal{S}$*. This calls for a simple modification to the FMM; namely, implement the method with the boundary condition in equation (4.8) and insert immediately after step 4:

4.5) At node $y^{\min}_{i,j}$, if $\varphi_{i,j} \geq \psi^c_{i,j}$, then set $\varphi_{i,j} = \infty$.

To see why this modification is sufficient, suppose that, at the start of step 3 of the current iteration, all Accepted nodes have the correct $\varphi_{i,j}$ value. Suppose also that $y_{i,j}$ is the smallest element in Narrow Band and $\varphi^*_{i,j} \geq \psi^c_{i,j}$. Since $\varphi^*_{i,j}$ is computed using neighboring Accepted nodes (which are assumed to have the correct values), this implies that an optimal path in $\mathcal{S}$ would take longer than $\psi^c_{i,j}$ to reach $y_{i,j}$. Then, $\varphi_{i,j} = \infty$, since $y_{i,j}$ cannot be safe-reachable. On the other hand, if $\varphi^*_{i,j} < \psi^c_{i,j}$, there is a safe-reachable path in $\mathcal{S}$ that takes less than $\psi^c_{i,j}$ time to reach $y_{i,j}$. Thus, after step 4.5, $\varphi_{i,j} < \infty$ if and only if $y_{i,j} \in \mathcal{S}$. Since all Accepted nodes are initially correct (step 2), the above argument holds inductively until all Accepted nodes are computed correctly.

The result of the computation above is a grid $\mathcal{G}$ with nodes where $\varphi_{i,j}$ approximates the value of φ for each node $y_{i,j}$. The safe-reachable set $\mathcal{S}$ can then be approximated as

$$\mathcal{S} \approx \{y_{i,j} \mid \varphi_{i,j} < \infty\}.$$

Note that with more general dynamics that are not isotropic, the FMM is not

directly applicable. However a similar causality-ordering procedure is possible via the Ordered Upwind Method (OUM) [60], allowing the method to be eventually extended to anisotropic [60] and non-holonomic [61] dynamics.

4.1.4 Extracting the Attacker's Optimal Control and Path

If $\overline{\mathcal{V}}_{\mathcal{T}}(\mathbf{x}^0) < \infty$, the next step is to identify the attacker's optimal input sequence

$$\mu^* = \arg\min_{\mu\in\mathbb{U}_o}\max_{\gamma\in\mathbb{D}_o}\{t \mid \mathbf{x}(t) \in \mathcal{T}, x_a(s) \notin C(s), \forall s \in [0,t]\}.$$

Note that μ^* is not necessarily unique, but all such inputs yield the same value $\overline{\mathcal{V}}_{\mathcal{T}}(\mathbf{x}^0)$. In fact, μ^* is simply the optimal input sequence for the attacker corresponding to a time-optimal safe-reachable path from x_a^0 to a point

$$x_a^f \in \arg\inf_{y\in\mathcal{T}\cap\mathcal{S}} \varphi(y),$$

that is, a point in $\mathcal{T}$ that can be safely reached in the shortest time. Thus, φ can be used to extract the optimal input sequence $\mu^* = \mu^*(y)$ where it is smooth. For the velocity limited dynamics used here, $\mu^*(y) = -\nabla\varphi(y)/\|\nabla\varphi(y)\|$. In general, $\nabla\varphi(y)$ may not exist at some $y \in \Omega$, representing points at which no unique optimal input exists.

The optimal path $x_a^*(\cdot)$ of the attacker can be computed by solving the ordinary differential equation

$$\dot{x}_a^*(t) = -\mu^*(x_a^*(t)) \tag{4.12}$$

from $t = \varphi(x_a^f)$ to $t = 0$ (backward in time), with a terminal condition $x_a^*(\varphi(x_a^f)) = x_a^f$. Due to the construction of φ, this will result in $x_a^*(0) = x_a^0$. This can be approximated by any standard numerical ODE solver.

Note that the position and inputs for the defender are not explicitly computed in this solution. As the attacker is able to reach the target using this open-loop input no matter what the defender does, the actions of the defender can be safely ignored. This does mean, however, that the open-loop formulation presented here cannot be used to control the defender.

4.1.5 Multiple Defenders

The open-loop game solution developed above can now be used for scenarios with multiple defenders.

The addition of multiple defenders is a straightforward adaptation of the open-loop game solution. Consider the presence of multiple defenders indexed $1, \ldots, N$, with distinct initial states and dynamics. The objective for the attacker is to find a path to the target that avoids possible capture by any of the defenders. Thus, the attacker must be able to reach points in the safe-reachable set in less time than any defender. Let ψ_k^c be the minimum time-to-capture function for defender k. The attacker must reach the point y in less time than the *minimum* of all of the time-to-capture functions. Therefore

$$\psi^c(y) = \min_{k=1,\ldots,N} \psi_k^c(y), \forall y \in \Omega$$

and $\mathcal{S}$ can be constructed in an identical fashion to the single defender case.

The key advantage to using the modified FMM solution for open-loop games over the methods presented in Chapter 3 is that solutions for games with multiple defenders are still computed on the lower dimensional state space for each agent. Additional defenders only require the computation of ψ_k^c for each additional defender, so computations scale linearly with the number of defenders rather than exponentially.

4.2 Open-Loop Multi-Stage Games

The previous section presented an algorithm for computing the minimum time-to-reach to a single target set, subject to a collision avoidance constraint with respect to one or more defenders. Like the reachable sets in Chapter 3, extending this solution to a multi-stage game with multiple target sets is not simply a direct concatenation of solutions to individual stages (see Figure 4.2).

Conceptually, the extension of the open-loop formulation and solutions to multi-stage games is similar to the method in Section 3.2.3. Working backward from the final target set, the solution for each preceding stage is constructed to ensure that the

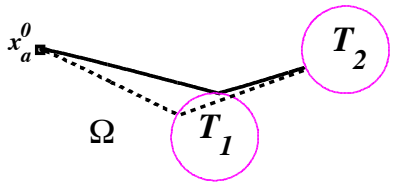


Figure 4.2: **The concatenation of two single-stage optimal paths (dotted) does not coincide with the two-stage optimal path (solid).**

next target in the sequence is reachable. Unlike the reach-avoid sets in Section 3.2.3, which are computed in the joint state space of all agents, $\mathcal{S}$ is computed in the attacker's state space only, and the movements of the defenders are not explicitly computed. Instead, the set $\Omega \setminus \mathcal{S}$ reflects all the states that *could* be captured by a defender. To construct the solution to a multi-stage open-loop game, the attacker must avoid all states that the defenders may reach during *all* stages of the game. Thus, the open-loop solution to a multi-stage game will depend on computing φ to correctly account for the defenders during each stage of the game.

For clarity and simplicity, the solution to open-loop multi-stage games will be explored in detail with a two-stage game. Generalizing to an arbitrary number of stages is a straightforward extension, and is briefly described in section 4.2.3.

4.2.1 Open-Loop Value for a Two-Stage Game

Let $\mathcal{T}_1, \mathcal{T}_2 \subset \Omega$ be disjoint target sets, such that the attacker's objective is to first reach $\mathcal{T}_1$, and then reach $\mathcal{T}_2$. As before, the attacker's goal is to accomplish this in minimum time, while avoiding all possible locations where it may be captured by the defender. It is assumed that φ_1 and $\mathcal{S}_1$ have been computed using the method discussed in Section 4.1.3, and that $\min_{y \in \mathcal{T}_1} \varphi_1(y) < \infty$ so that $\mathcal{T}_1$ is safe-reachable.

Suppose the attacker travels along a time-optimal safe-reachable path to a point $z \in \mathcal{T}_1$ during the first stage; this will take (at most) $\varphi_1(z)$ time. Next, the attacker's goal is to find a time-optimal safe-reachable path from z to $\mathcal{T}_2$. The open-loop value function is the minimum of the total time spent in the two stages, over all possible intermediate points $z \in \mathcal{T}_1$:

$$\overline{\mathcal{V}}_2(\mathbf{x}^0) = \min_{z \in \mathcal{T}_1} \left[\varphi_1(z) + \min_{\mu \in \mathbb{U}_o} \max_{\gamma_2 \in \mathbb{D}_o} \{ t \mid x_a(t) \in \mathcal{T}_2, x_a(s) \notin C(s), \forall s \in [\varphi_1(z), t] \} \right], \quad (4.13)$$

where, $\gamma_2(t) = \gamma(t + \varphi_1(z)), \forall t \geq 0$. The time shift in the defender's input sequence indicates that, at the beginning of the second stage, the defender could be at any point that is reachable in time $\varphi_1(z)$. This modification captures the requirement that the attacker must avoid all states that the defender may reach during both stages of the game.

The open-loop value for the two-stage game in equation (4.13) can now be characterized by computing the appropriate minimum time-to-reach function for the attacker. Given $\mathbf{x}^0$ and $\mathcal{T}_1$, a point $y \in \Omega$ is *second-stage safe-reachable* if it is reachable by a path of the attacker that first visits an intermediate point in $\mathcal{T}_1$, and avoids capture by the defender for all $\gamma \in \mathbb{D}_o$. The second-stage safe-reachable set can then be defined as

$$\mathcal{S}_2 = \{ y \in \Omega \mid y \text{ is second-stage safe-reachable} \}.$$

The analog of equation (4.3) on $\mathcal{S}_2$ is:

$$\begin{aligned} \varphi_2(y) = \min \big\{ t + \varphi_1(z) \mid \; & z \in \mathcal{T}_1, t \geq 0, x_a(t) = y, \\ & x_a(0) = z, x_a(s) \in \mathcal{S}_2, \forall s \in [\varphi_1(z), t] \big\}. \end{aligned} \quad (4.14)$$

In the above equation, as with equation (4.13), it is assumed that the attacker takes a time-optimal safe-reachable path to the intermediate point $z \in \mathcal{T}_1$.

The main result for the two-stage game is analogous to Theorem 1.

Theorem 2. $\overline{\mathcal{V}}_2(\mathbf{x}^0) = \min\{\varphi_2(y) \mid y \in \mathcal{T}_2\}$.

The proof is omitted, as it is a direct application of the definitions given above, using a similar line of argument as in the proof of Theorem 1. As with the single-stage game, the modified FMM can be used to approximate φ_2 on a (Cartesian) grid $\mathcal{G}$.

The set $\mathcal{S}_2$ and φ_2 satisfy the coupled relation:

$$\mathcal{S}_2 = \{y \in \Omega \mid \varphi_2(y) < \psi^c(y)\}, \tag{4.15}$$

$$\begin{cases} -\min\limits_{u\in U}\{\nabla\varphi_2(y)\cdot u\} = 1, & y \in \mathcal{S}_2\backslash\mathcal{T}_1, \\ \varphi_2(y) = \varphi_1(y), & y \in \mathcal{T}_1, \\ \varphi_2(y) = \infty, & y \in \Omega\backslash\mathcal{S}_2. \end{cases} \tag{4.16}$$

$(\varphi_2)_{i,j}$ (the approximation of φ_2 on a grid $\mathcal{G}$) can be computed by performing the modified FMM described in section 4.1.3 with the boundary conditions in equation (4.16).

4.2.2 Extracting the Optimal Control and Path

The optimal input sequence and path for the attacker can now be computed sequentially, working backward from the second target to the first, then to the initial condition. Suppose that φ_1 and φ_2 have been computed, and that $\overline{\mathcal{V}}_2(\mathbf{x}^0) < \infty$. Let $x_a^f \in \arg\inf\{\varphi_2(y) \mid y \in \mathcal{T}_2\}$. For all $y \in \mathcal{S}_2$, the optimal input $\mu_2^* = \mu_2^*(y)$ can be extracted from φ_2 as in Section 4.1.4. In the second stage, $x_a^*(\cdot)$ can be computed by solving equation (4.12), with μ^* replaced by μ_2^*, backward in time from $t = \varphi_2(x_a^f)$, with the terminal condition x_a^f. The ODE is solved (backward in time) until $x_a^*(t) \in \mathcal{T}_1$. From this point onwards, the optimal input sequence $\mu^* = \mu^*(y)$ for $y \in \mathcal{S}_1$ is extracted from φ_1, until $t = 0$, when $x_a^*(0) = x_a^0$.

4.2.3 Games with Arbitrarily Many Stages

The methods from the two-stage game can be used to generalize directly to a N_s-stage open-loop game for $N_s \geq 3$. Suppose the attacker's objective is to find a time-optimal path that visits target sets $\mathcal{T}_1, \ldots, \mathcal{T}_{N_s} \subset \Omega$ consecutively, while avoiding capture by

the defender. For each $k = 1, 2, \ldots, N_s$, the coupled relation for the k-th-stage safe-reachable set $\mathcal{S}_k$, and minimum time-to-reach function φ_k is given by

$$\begin{aligned} &\mathcal{S}_k = \{y \in \Omega \mid \varphi_k(y) < \psi^c(y)\}, \\ &\begin{cases} -\min\limits_{u \in U} \{\nabla \varphi_k(y) \cdot u\} = 1, & y \in \mathcal{S}_k \backslash \mathcal{T}_{k-1}, \\ \varphi_k(y) = \varphi_{k-1}(y), & y \in \mathcal{T}_{k-1}, \\ \varphi_k(y) = \infty, & y \in \Omega \backslash \mathcal{S}_k. \end{cases} \end{aligned} \tag{4.17}$$

Furthermore, in an similar fashion as Theorems 1 and 2, the k-th open-loop value function can be shown to satisfy $\overline{\mathcal{V}}_k(\mathbf{x}^0) = \min\{\varphi_k(y) \mid y \in \mathcal{T}_k\}$. The extraction of the attacker's optimal inputs and path proceeds in the same manner. Note that the computational complexity scales linearly in the number of stages, since the modified FMM is invoked once at every stage. The total complexity for a N_s-stage game on a grid with M nodes is $O(N_s M \log M)$.

4.3 Simulation results

The algorithm was tested in simulation in a 2-stage game, representing capture-the-flag, on a 2-D map of size 400^2 pixels, shown in Figure 4.3, covering a portion of the UC Berkeley campus. The map data $m_{i,j}$, $i, j \in 1, \ldots, 400$ have values 0 in the obstacles (buildings) and 1 in the walkways; other regions have intermediate values in $(0, 1)$ selected in proportion to the estimated density of vegetation. This represents the fraction of the maximum speed each agent was allowed to travel at on that pixel, with 1 being maximum speed and 0 representing impenetrable obstacles. Two simulations were conducted: the first with one defender and the second with two defenders.

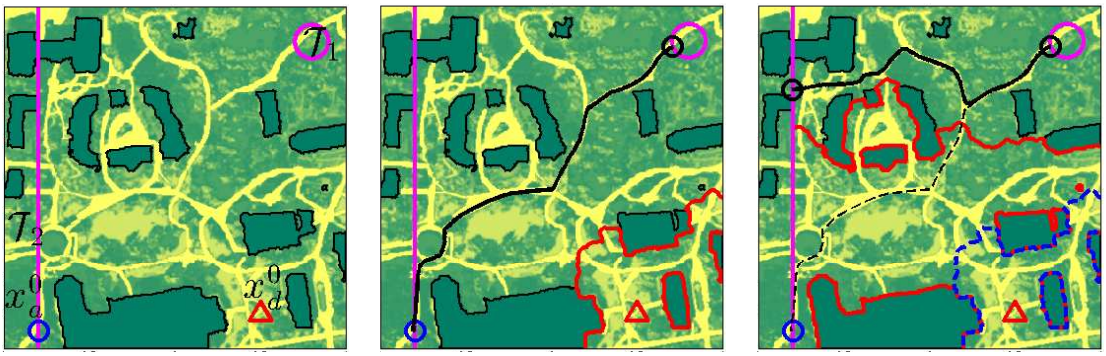
The attacker's speed $v_{a,max}$ at each grid node $y_{i,j}$ was set to $5m_{i,j}$ and $2m_{i,j}$, in the first and second stages, respectively. Figure 4.4 shows the example with a single defender. The defender's speed $v_{d,max}$ was set to $m_{i,j}$ at each grid node $y_{i,j}$. Figure 4.4b shows the first stage of the game, with the boundary of $\mathcal{S}$ outlined in red. This marks the farthest extent that the defender could reach, and thus the attacker was able to arrive at the target $\mathcal{T}_1$. Note that the path of the attacker reflects the

Figure 4.3: **A segment of the UC Berkeley campus used for the simulations (map image courtesy of `maps.google.com`).**

varying speed constraints, and the attacker did not move in a straight line but rather used the walkways to reach the target as quickly as possible. Figure 4.4c shows the second stage of the game, where the attacker's speed has been reduced. The attacker is nonetheless able to reach the second target.

Fig. 4.5 shows the case in which a second defender with speed $0.5m_{i,j}$ was added to the previous simulation. As this defender was rather slower, it made a relatively small impact on the attacker's safe-reachable set in the first stage. However, its presence did cause the attacker to make a detour in its path to $\mathcal{T}_2$.

All computations were performed on a desktop computer with 3.33 GHz Intel Core-II duo processors. The code was implemented in C using the Matlab MEX compiler to allow function calls within Matlab. Both tests were completed (including the computation of optimal paths) in less than 0.5 seconds each. It should be emphasized that the implementation and the computational efficiency are independent of variations in the speed function, and the addition of an extra defender did not increase the computation time substantially. In fact, overhead for the function calls significantly outweighed the difference in computation time occasioned by the



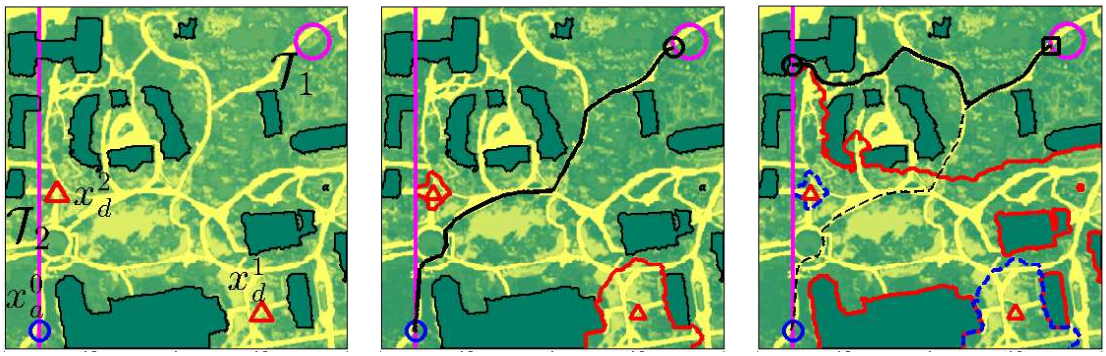
(a) (b) (c)

Figure 4.4: **Simulation showing (a) the target sets, obstacle boundaries (black contours), and initial states of the attacker and the defender, (b) $\mathcal{S}_1$ (red curve) and the safe-reachable path to $\mathcal{T}_1$, and (c) $\mathcal{S}_2$ (red curve) and the safe-reachable path (black curve) from $\mathcal{T}_1$ to $\mathcal{T}_2$. As a comparison, the attacker's optimal path (black dotted curve) and the defender's reachable set from the first stage (blue dotted curve) are plotted.**

additional defender.

4.4 Discussion

The open-loop game and solution presented in this chapter are an example of an approach that can quickly generate feasible solutions for a game with many agents. Formulating the problem as an open-loop reach-avoid game allows the value function to be characterized via an HJB equation directly in the state space of individual agents, as opposed to the joint state space of all agents. Compared with the HJI approach, this results in considerable computational savings, at the expense of conservatism with respect to the possible actions of the defenders. This conservatism means that for certain initial conditions, the open-loop solution value will be greater the HJI value, requiring more time to arrive at a target. In the worst case, this can result in an infinite open-loop value and thus no attacker inputs, even though the closed-loop HJI solution will give a finite arrival time and guide the attacker successfully to the target. More research will need to be conducted on the best that can be done in such situations, as well as in situations where the attacker cannot win if



(a) (b) (c)

Figure 4.5: **Numerical results for the same setting as in Fig. 4.5, but with an additional defender, slower than the first.**

the defenders play optimally, but may exploit some sub-optimal defender actions to eventually achieve victory.

Nevertheless, the speed with which solutions can be found can be a major advantage over the HJI solutions, in particular allowing solutions to be found in real time and with multiple defenders. As the numerical results demonstrated, the open-loop solution is efficient, accurate, and readily adaptable to complicated domain geometry and inhomogeneous agent speeds. Where the open-loop solution does exist, the attacker path found is guaranteed to be safe from capture by the defenders. The FMM algorithms can be used to quickly check if an open-loop solution exists. If it does, then this solution can be used directly, with a guarantee of safety, without resorting to other sub-optimal solutions that may not have any provable properties. In addition, the real time computation opens up some other interesting possibilities for future work, including the incorporation of sensing updates on obstacles in the domain, and the use of an iterative scheme to recompute the optimal path in an MPC-like scheme.

Currently, the open-loop formulation as presented does not generate solutions for the defender, as it only solves the game from one side. The following chapter will discuss the opposite perspective, addressing the problem of controlling multiple agents to capture a single evading agent.

Chapter 5

Many vs. 1: Safe-Reachable Area Cooperative Pursuit

For adversarial games with many agents, the issue of cooperation between agents is often a source of considerable difficulty. The necessity of computing solutions over the joint input space of multiple agents can greatly increase the computational complexity of the problem. The preceding chapters have presented solution strategies for a single agent acting against one or more opponents. This chapter focuses on deriving control strategies for a number of agents coordinating on one team.

The problem considered here is a multi-agent pursuit-evasion game, with a number of pursuers attempting to capture a single evader in a simply connected planar region. The pursuers may have speed equal to or faster than the evader. The objective is to find a successful cooperative pursuit strategy for the pursuing agents. Although the cooperative pursuit problem can be formulated as a differential game, the addition of multiple pursuers means that numerical methods such as the Hamilton-Jacobi-Isaacs (HJI) formulation in Chapter 3 are not tractable. The open-loop formulation from Chapter 4 is also not directly applicable: except for some restricted cases, the evader will always be able to evade indefinitely if it knows the planned control inputs for the pursuers. However, a conceptually similar approach to the open-loop game will be taken: instead of considering the high dimensional joint state space of all agents, an approach will be taken that allows computations to occur only in the state space of

single agents.

This chapter presents a decentralized, guaranteed pursuit strategy where the pursuers cooperatively minimize the area of the evader's safe-reachable set, analogous to the attacker's safe-reachable set as defined in Chapter 4. The safe-reachable set can be computed using the modified FMM algorithm presented there, or analytically for convex game domains when the pursuers and evader have equal speeds [62]. The pursuers compute control inputs independently given the safe-reachable set and the location of the evader, which is the only shared information, resulting in real time computation of the control inputs. For convex domains and equal speeds, this pursuit strategy results in guaranteed capture of the evader in finite time regardless of the strategy or inputs of the evader.

Each pursuer influences the evader's safe-reachable set only on a portion of the safe-reachable set's boundary, analogous to the shared Voronoi boundary in a Voronoi decomposition. Thus each pursuer's input decouples from that of the other pursuers and can be computed independently, but their inputs are coupled through the safe-reachable set and the evader's position, giving rise to cooperation between the pursuers. Empirically, the algorithm is shown to result in effective cooperation between pursuing agents, resulting in superior performance to techniques such as the pure pursuit strategy, where the pursuers attempt to minimize the instantaneous distance to the evader. This ability to engender cooperation is one of the key advantages of the strategy proposed in this chapter. Pure pursuit is the optimal single-pursuer strategy in an open environment, and is chosen as a standard of comparison due to its simplicity and ubiquity [5]. Although more complex cooperative pursuit-strategies have been proposed in the literature, they are difficult to adapt to general configurations with obstacles.

The following sections will present the pursuit strategy in detail. The pursuit-evasion problem in question is defined in Section 5.1, and its formulation as a differential game is discussed. Section 5.2 lays out the pursuit strategy, first for the case of equal-speed pursuers and evader in a convex domain, and then for the more general case of non-convex domains and unequal speeds. A number of simulations are presented comparing the performance of the proposed strategy against the pure

pursuit strategy. Section 5.3 describes these simulations and some comparative tests that were conducted for evaluating the algorithm. Experimental results are also presented for games involving human agents using the BEARCAT platform. These are described in Section 5.4. These experiments demonstrate the feasibility of the safe-reachable area minimization strategy not only for autonomous agents, but also as a tool for guiding and coordinating human agents. Finally, Section 5.5 concludes the chapter with a discussion of the algorithm and results. The pursuit algorithm in this chapter was first presented in [35].

5.1 The Cooperative Pursuit Problem

Consider a multi-agent pursuit-evasion game involving N pursuers and a single evader, taking place in an open, simply connected region Ω in $\mathbb{R}^2$. Let $x_e \in \mathbb{R}^2$ be the position of the evader and $x_p^i \in \mathbb{R}^2$ be the position of pursuer i. The equations of motion are

$$\begin{aligned} \dot{x_e} &= d,\ x_e(0) = x_e^0, \\ \dot{x_p}^i &= u_i,\ x_p^i(0) = x_p^{i,0},\ i = 1, ..., N, \end{aligned} \tag{5.1}$$

where d and u_i are the velocity control inputs of the evader and pursuers, respectively, and $x_e^0, x_p^{i,0} \in \Omega$ are the initial evader and pursuer positions. The respective agent inputs are constrained to lie within sets U_i for the pursuers and D for the evader. For the purposes of this chapter, U_i and D are assumed to be the following:

$$D = ||d||_2 \leq v_{e,max},\ U_i = ||u_i||_2 \leq v_{i,max},\ \forall t \geq 0, \tag{5.2}$$

for some maximum speeds $v_{e,max}$ for the evader and $v_{i,max}$ for each pursuer. The motions of the evader and pursuers, as described by equation (5.1), are also constrained to lie within the region Ω, with

$$x_e(t),\ x_p^i(t) \in \Omega,\ \forall t \geq 0. \tag{5.3}$$

Any velocity input $d(t)$ or $u_i(t)$ which satisfies the constraints in equations (5.2) and (5.3) is called an admissible input for the evader or pursuer i, respectively.

The goal of the pursuers is to capture the evader by having at least one of the pursuers bring the evader within a distance r_c of the pursuer. Let δ_i be the distance between pursuer i and the evader, and $\delta_{\min} = \min_i ||x_e - x_p^i||_2$ be the minimum separation between the evaders and pursuers. Let $\mathbf{x} = (x_e, x_p^1, ...x_p^N)$ be the joint state of all agents, and let C be the joint set of *all* states in which the evader is captured, that is

$$C = \{\mathbf{x} \mid \delta_{\min} \leq r_c\}$$

The capture condition for the pursuers is then given by

$$\mathbf{x}(t) \in C. \tag{5.4}$$

To achieve this capture condition, each pursuer selects control inputs using a pursuit strategy $\mu_i(x_e, x_p^1, ..., x_p^N)$, based upon observations of the evader and pursuer positions at each time instant, resulting in the closed-loop system dynamics:

$$\begin{aligned} \dot{x_e} &= d,\ x_e(0) = x_e^0, \\ \dot{x_p}^i &= \mu_i(x_e, x_p^1, ..., x_p^N),\ x_p^i(0) = x_p^{i,0},\ i = 1, ..., N \end{aligned} \tag{5.5}$$

The evader may use some strategy $\gamma(x_e, x_p^1, ..., x_p^N)$ to avoid the pursuers. Any strategy μ_i which satisfies the constraints from equations (5.2) and (5.3) is referred to as an admissible pursuit strategy for pursuer i, and similarly for γ. The sets of admissible strategies for the pursuers and evaders are denoted $\mathbb{U}$ and $\mathbb{D}$, respectively.

A precise statement of the problem for the multi-agent pursuit-evasion game can now be given as the following: for any initial configuration $x_e^0,\ x_p^{i,0} \in \Omega$ satisfying $\delta_{min}(0) > r_c$, find an admissible choice of pursuit strategy μ_i for each pursuer i such that, regardless of any admissible choice of evader input d, the capture condition equation (5.4) is satisfied for some time $T < \infty$. In this case, the control computation is performed on behalf of the pursuers, and the evader is the disturbance.

5.1.1 Differential Game Formulation

The pursuit-evasion game presented above can be formulated as a differential game, similar to the reach-avoid games presented in the preceding chapters. The minimum time-to-capture can be defined as

$$T_c(\mathbf{x}) = \min\{t \mid \mathbf{x}(t) \in C, \mathbf{x}(s) \notin C, \forall s \in [0, t)\}$$

For notational simplicity, let μ_p represent the joint pursuer strategy, which collects together the individual pursuer strategies, with admissible strategy set $\mathbb{U}_p$. The differential game is a game in which the pursuers attempt to minimize T_c and the evader attempts to maximize it, with the value of the game as

$$\mathcal{V}(\mathbf{x}) = \min_{\mu_p \in \mathbb{U}_p} \max_{\gamma \in \mathbb{D}} T_c(\mathbf{x})$$

This value is equal to the viscosity solution to the following Hamilton-Jacobi-Isaacs equation

$$\min_{u_i \in U_i} \max_{d \in D} \left[\sum_i \frac{\partial \mathcal{V}^T}{\partial x_p^i} u_i + \frac{\partial \mathcal{V}^T}{\partial x_e} d \right] + 1 = 0 \tag{5.6}$$

subject to

$$\mathcal{V}(\mathbf{x}) = 0, \mathbf{x} \in C$$

and can be approximated using the appropriate numerical methods (see Section 3.2.2). Let

$$p_e = \frac{\partial \mathcal{V}^T}{\partial x_e}$$

and

$$p_i = \frac{\partial \mathcal{V}^T}{\partial x_p^i}$$

The optimal evader and pursuer control inputs can be found then as $\frac{p_e}{||p_e||_2}$ and $\frac{p_i}{||p_i||_2}$, respectively.

Numerically solving (5.6) on a grid yields good approximations of the optimal strategies when the solution is tractable, but this approach suffers from the same computational limits as the techniques discussed in Chapter 3, where exponential

growth of the grid needed to compute the HJI solution limits the game to two agents. Instead, alternative methods must be found.

The approach taken in this chapter will be conceptually similar to that of Chapter 4, where the solution was found in the lower dimensional state space of an individual agent to reduce computational complexity. The open-loop formulation cannot be directly used, since the control computation is being performed for the pursuers. In general, if the evader is permitted to know the pursuers' strategies and the pursuers are not able to react to the evader's actions, the evader will be able to evade indefinitely. Instead, the pursuit strategy will still be executed in a closed-loop sense, but utilizing some of the concepts from the open-loop game.

5.2 Pursuit via Safe-Reachable Area Minimization

The pursuit strategy proposed is based on the concept of the safe-reachable set as presented in Chapter 4. The evader's safe-reachable set $\mathcal{S}_e$ is defined as the set of all points in Ω that the evader can directly move to without being captured by a pursuer. The strategy is designed so as to decrease the area of $\mathcal{S}_e$ over time. Intuitively, as this area decreases towards zero, the capture condition will be satisfied. The evader's safe-reachable set relative to the pursuers can be defined in an identical manner as the attacker's safe-reachable set with respect to a group of defenders from Sections 4.1.2 and 4.1.5. When the pursuer and evader speeds are equal, the safe-reachable set is equivalent to the generalized Voronoi decomposition of the agents [62].

Since $\mathcal{S}_e$ depends only on the position of the agents, the area A of $\mathcal{S}_e$ depends only on the locations of the pursuers relative to the evader, and this dependence is smooth whenever the pursuer locations are in Ω. The time derivative of A is given by

$$\frac{dA}{dt} = \frac{\partial A}{\partial x_e}\dot{x_e} + \sum_{i=1}^{N} \frac{\partial A}{\partial x_p^i}\dot{x_p}^i \tag{5.7}$$

Now consider a cooperative pursuit strategy that jointly minimizes $\frac{dA}{dt}$. According to equation (5.7), this joint objective can be decoupled into the individual objectives of minimizing $\frac{\partial A}{\partial x_p^i}\dot{x_p}^i$ for each pursuer i. A pursuer i is said to share an *active boundary*

with the evader if there is a portion of the boundary of $\mathcal{S}_e$ where $\psi_i^c(y) < \psi_j^c(y)$ for all points y in this portion and all other pursuers j. In other words, there is a portion of the boundary of $\mathcal{S}_e$ that is defined purely by the position of pursuer i. This is analogous to having a shared Voronoi boundary with the evader in a Voronoi decomposition.

Let $\mathcal{N}_e$ be the set of all such pursuers that share an active boundary with the evader. A only depends on the pursuers in $\mathcal{N}_e$, so $\frac{\partial A}{\partial p_i} = 0$ for all $i \notin \mathcal{N}_e$. Any pursuer i which is not does not share an active boundary with the evader may simply use the pure pursuit strategy. On the other hand, for each pursuer $i \in \mathcal{N}_e$, the choice of pursuit strategy which minimizes equation (5.7) is given by:

$$u_i^* = \mu_i^*(x_e, x_p^1, \ldots, x_p^N) \triangleq -v_{i,max} \frac{\frac{\partial A}{\partial x_p^i}}{||\frac{\partial A}{\partial x_p^i}||_2}.$$

Computing $\frac{\partial A}{\partial x_p^i}$ for each pursuer depends on the geometry of the game domain Ω and the relative speeds of the agents. An analytic solution is found below for convex domains with equal agent speeds, and a more general algorithm based on fast marching methods is presented for non-convex domains with unequal agent speeds.

5.2.1 Convex Domains with Equal Speeds

For games played in convex domains between agents of equal speeds, the safe-reachable set of the evader is equal to the evader's Voronoi partition. Let $S(D) = \{\mathcal{S}_e, S_1, \ldots, S_N\}$ be the Voronoi partition of Ω generated by the points $\{x_e, x_p^1, \ldots, x_p^N\}$:

$$\begin{aligned}
\mathcal{S}_e &= \{y \in \Omega \mid \|y - x_e\| < \|y - x_p^i\|, \forall i \leq N\}, \\
S_i &= \big\{y \in \Omega \mid \|y - x_p^i\| \leq \\
&\qquad \min\{\|y - x_e\|, \|y - x_p^j\|\}, \forall j \neq i\big\}, i \leq N
\end{aligned}$$

Let $\mathcal{N}_e$ be the set of pursuer indices that are Voronoi neighbors of the evader. The edge shared by $\mathcal{S}_e$ and S_i, $i \in \mathcal{N}_e$ is called the *line of control* for pursuer i and is denoted by B_i, where L_i is the length of B_i (see Figure 5.1). The area A of the

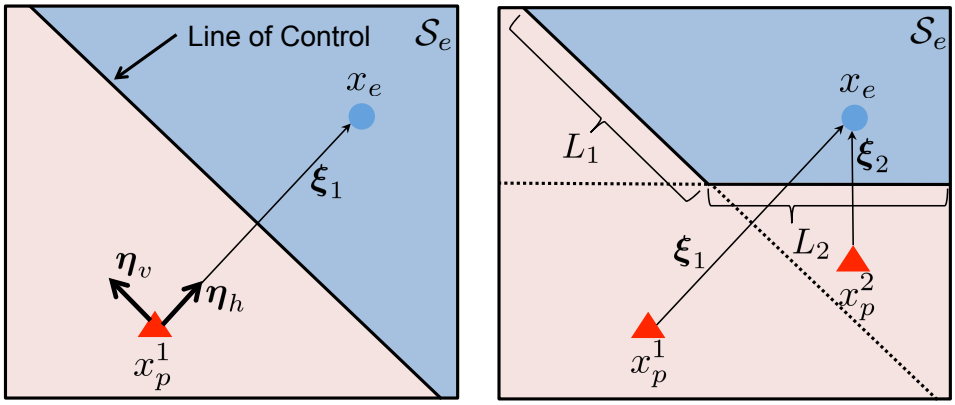


Figure 5.1: **Illustrations showing the evader's safe-reachable set $\mathcal{S}_e$ in a convex domain with equal agent speeds (a) for a single pursuer and evader and (b) with an additional pursuer.**

Voronoi cell $\mathcal{S}_e$ containing the evader can be calculated as

$$A(x_e, x_p^1, ..., x_p^N) = \sum_{j=1}^{n_e} \left(x_j^e y_{j+1}^e - x_{j+1}^e y_j^e \right), \tag{5.8}$$

where n_e is the number of vertices of $\mathcal{S}_e$ and $\{(x_j^e, y_j^e)\}_{j \leq n_e}$ is the set of vertices of $\mathcal{S}_e$ and $n_e + 1$ wraps around to the first vertex. It can be observed that by appropriate re-scaling of the dynamics in equation (5.5), the region Ω, and the capture radius r_c, it is sufficient to consider this problem for the case where $v_{e,max} = v_{i,max} = 1$. Thus, for the rest of the convex, equal-speed analysis, it is assumed without loss of generality that $v_{e,max} = v_{i,max} = 1$ in equation (5.2).

Convex pursuit strategy

In this case, an analytic expression for the pursuer strategies μ_i^*, $i \in \mathcal{N}_e$ can be found using a particular choice of local coordinate system. First, let $\boldsymbol{\xi}_i(x_e, x_p) = x_e - x_p^i$ be the displacement vector pointing from the location of pursuer i towards the location of the evader. When there is no ambiguity, its arguments will be omitted and this vector will be denoted simply by $\boldsymbol{\xi}_i$. As $\delta_{min}(0) > r_c$ until capture, at which $\delta_{min}(T) = r_c$, $||\boldsymbol{\xi}_i|| \geq r_c$ for all $i \leq N$ and $t \in [0, T]$. Define $\boldsymbol{\eta}_h^i = \frac{\boldsymbol{\xi}_i}{\|\boldsymbol{\xi}\|_i}$ and let $\boldsymbol{\eta}_v^i \in \mathbb{R}^2$ be a unit

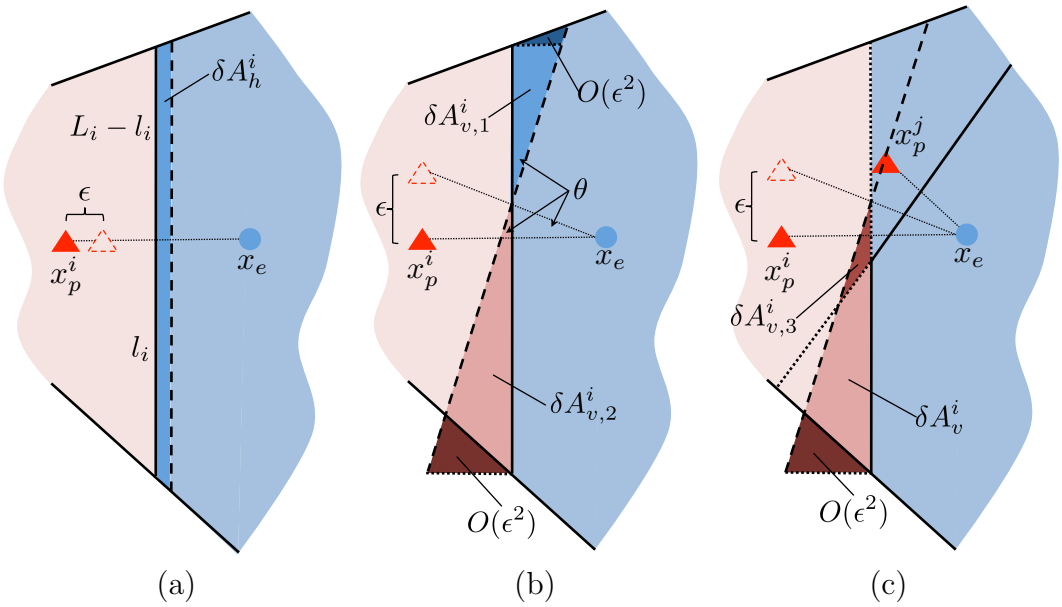


Figure 5.2: **Variational change in area of the evader's safe-reachable set with respect to (a) a perturbation toward the evader, (b) perturbation parallel to the line of control, and (c) when another pursuer is present and $\boldsymbol{\xi}_i$ no longer intersects B_i, as in Figure 5.1b.**

vector orthogonal to $\boldsymbol{\eta}_h^i$, as shown in Figure 5.1a. The vectors $\{\boldsymbol{\eta}_h^i, \boldsymbol{\eta}_v^i\}$ define a local coordinate system that depends on the locations of x_e and x_p^i. For any $y \in \mathbb{R}^2$ and $(x_e, x_p^1, ..., x_p^N)$ such that $y + x_p^i \in \Omega$, define

$$A_i^+(y)|_{(x_e, x_p^1, ..., x_p^N)} = A(x_e, x_p^1, ..., x_p^i + y, ..., x_p^N).$$

Define $D_h^i A$ and $D_v^i A$ as the directional derivatives of A along $\boldsymbol{\eta}_h^i$ and $\boldsymbol{\eta}_v^i$, then

$$\begin{cases} D_h^i A|_{(x_e, x_p^1, ..., x_p^N)} = \lim_{\epsilon \to 0} \frac{A_i^+(\epsilon \cdot \boldsymbol{\eta}_h^i)|_{(x_e, x_p^1, ..., x_p^N)} - A}{\epsilon} \\ D_v^i A|_{(x_e, x_p^1, ..., x_p^N)} = \lim_{\epsilon \to 0} \frac{A_i^+(\epsilon \cdot \boldsymbol{\eta}_v^i)|_{(x_e, x_p^1, ..., x_p^N)} - A}{\epsilon}, \end{cases} \tag{5.9}$$

where $A(x_e, x_p^1, ..., x_p^N)$ is denoted by A for brevity. From this expression, the partial derivative of A with respect to x_p^i is given by

$$\frac{\partial A}{\partial x_p^i} = D_h^i A \cdot \boldsymbol{\eta}_h^i + D_v^i A \cdot \boldsymbol{\eta}_v^i. \tag{5.10}$$

Lemma 3. *For any $i \in \mathcal{N}_e$, it is true that*

$$D_h^i A = -\frac{L_i}{2},$$
$$D_v^i A = \frac{l_i^2 - (L_i - l_i)^2}{2||\boldsymbol{\xi}_i||},$$

where L_i is the length of the line of control B_i and l_i is the length of the segment of B_i on the side of the intersection of $\boldsymbol{\xi}_i$ with B_i opposite to $\boldsymbol{\eta}_v^i$, as shown in Figure 5.2.

Proof. **Perturbation along $\boldsymbol{\eta}_h^i$:** A perturbation ϵ in the pursuer's position toward the evader moves the line of control $\frac{\epsilon}{2}$ toward the evader, and generates a corresponding change in the area of the evader's Voronoi cell δA_h^i, as shown in Figure 5.2a. This change in area is

$$\delta A_h^i = -\frac{L_i \epsilon}{2} + O(\epsilon^2),$$

where the $O(\epsilon^2)$ term depends on the angle of intersection between B_i and the boundaries of the Voronoi cell $\mathcal{S}_e$. From this expression, the directional derivative of A along $\boldsymbol{\eta}_h^i$ can be calculated as

$$D_h^i A = \lim_{\epsilon \to 0} \frac{\delta A_h^i}{\epsilon} = -\frac{L_i}{2}.$$

Perturbation along $\boldsymbol{\eta}_v^i$: There are two different scenarios for perturbation along $\boldsymbol{\eta}_v^i$, corresponding to the two pursuer configurations shown in Figure 5.1b. In one case, as for x_p^2 in Figure 5.1b, $\boldsymbol{\xi}_i$ intersects B_i. A perturbation of ϵ, as shown in Figure 5.2b, will cause the evader's Voronoi cell to shrink above the new intersection by $\delta A_{v,1}^i$ and

grow below it by $\delta A^i_{v,2}$. Let $\delta A^i_v = \delta A^i_{v,2} - \delta A^i_{v,1}$, with

$$\delta A^i_{v,1} = \frac{1}{2}((L_i - l_i) - \frac{\epsilon}{2})^2 \frac{\epsilon}{\|\boldsymbol{\xi}_i\|} + O(\epsilon^2),$$
$$\delta A^i_{v,2} = \frac{1}{2}(l_i + \frac{\epsilon}{2})^2 \frac{\epsilon}{\|\boldsymbol{\xi}_i\|} + O(\epsilon^2),$$

where the terms $O(\epsilon^2)$ again depend on the angle of intersection between B_i and the boundaries of the Voronoi cell $\mathcal{S}_e$. Thus, the resulting changes in area will be

$$\delta A^i_{v,1} = \frac{(L_i - l_i)^2 \epsilon}{2||\boldsymbol{\xi}_i||} + O(\epsilon^2)$$
$$\delta A^i_{v,2} = \frac{l_i^2 \epsilon}{2||\boldsymbol{\xi}_i||} + O(\epsilon^2)$$

which implies

$$D^i_v A = \lim_{\epsilon \to 0} \frac{\delta A^i_v}{\epsilon} = \frac{l_i^2 - (L_i - l_i)^2}{2||\boldsymbol{\xi}_i||}.$$

The second case is that of x^1_p in Figure 5.1b, where $\boldsymbol{\xi}_i$ no longer intersects B_i due to the presence of other pursuers. As shown in Figure 5.2c, the change in area is

$$\delta A^i_v = \delta A^i_{v,2} - \delta A^i_{v,3},$$

where $\delta A^i_{v,2}$ is calculated as before and

$$\delta A^i_{v,3} = \frac{1}{2}(l_i - L_i + \frac{\epsilon}{2})^2 \frac{\epsilon}{\|\boldsymbol{\xi}_i\|} + O(\epsilon^2).$$

Note that here $l_i \geq L_i$. Letting $\epsilon \to 0$ then again it is true that

$$D^i_v A = \lim_{\epsilon \to 0} \frac{\delta A^i_v}{\epsilon} = \frac{l_i^2 - (l_i - L_i)^2}{2||\boldsymbol{\xi}_i||}.$$

□

With the above lemma, the proposed strategy μ^*_i can be rewritten in the local

coordinate system as

$$\mu_i^* = -\left(\frac{\alpha_h^i}{\sqrt{|\alpha_h^i|^2 + |\alpha_v^i|^2}} \cdot \boldsymbol{\eta}_h^i + \frac{\alpha_v^i}{\sqrt{|\alpha_h^i|^2 + |\alpha_v^i|^2}} \boldsymbol{\eta}_v^i\right), \tag{5.11}$$

where α_h^i and α_v^i are given by

$$\alpha_h^i = -\frac{L_i}{2},\ \alpha_v^i = \frac{l_i^2 - (L_i - l_i)^2}{2||\boldsymbol{\xi}_i||}.$$

Lemma 4. *It can be shown that under this choice of pursuit strategy, u_i always points toward the interior of Ω, thus satisfying the constraint from equation* (5.3)*. The proof is straightforward but requires some amount of algebra and is omitted.*

Proof of guaranteed capture

The pursuit strategy outlined above for the convex, equal speed game is guaranteed to capture the evader in finite time, regardless of any admissible evader input d. It can be seen that if this holds for the case of a single pursuer ($N = 1$), then the conclusion also extends to the case of multiple pursuers ($N > 1$). Indeed, for the case of $N > 1$, one can choose any pursuer i which is a Voronoi neighbor of the evader and use the arguments for the case of $N = 1$ to show that the capture condition will be satisfied. This section presents the proof for a single pursuer. Correspondingly, the notation from above will carry through without the indices i.

First, observe that as A approaches zero, the evader's Voronoi cell approaches either a line or a point. Either of the two cases clearly implies $\|x_e - x_p\|_2 \to 0$. The strategy here is then to show that, under the proposed pursuit strategy μ^* and any admissible evader control input d, either the area A or the distance δ is guaranteed to decrease until the capture condition is met.

In terms of preliminaries, by Lemma 3,

$$\frac{\partial A}{\partial x_p} = \alpha_h \boldsymbol{\eta}_h + \alpha_v \boldsymbol{\eta}_v.$$

It can be also verified in a similar manner as the proof of Lemma 3 that the partial

derivative $\frac{\partial A}{\partial x_e}$ in the local coordinate system is given by

$$\frac{\partial A}{\partial x_e} = \alpha_h \boldsymbol{\eta}_h - \alpha_v \boldsymbol{\eta}_v. \tag{5.12}$$

Also recall that the variable L in the statement of Lemma 3 depends on the spatial locations of the evader and the pursuer, as well as the geometry of the region Ω. For this proof, make the following definitions of parameters $l_{\min}$ and $l_{\max}$, which depend solely on the geometry of Ω:

$$\begin{cases} l_{\min} = \inf_{x_e \in \Omega, x_p \in \Omega} L(x_e, x_p) \\ l_{\max} = \sup_{x_e \in \Omega, x_p \in \Omega} ||x_e - x_p||_2. \end{cases} \tag{5.13}$$

Since Ω is bounded and the game ends upon capture, it is true that $l_{\min} > r_c$, $l_{\max} < \infty$, and $L \leq l_{max}$.

The following result shows that the area A is always non-increasing under the pursuit strategy μ^* for a single pursuer.

Lemma 5. *Under the proposed pursuit strategy $\mu^*(x_e, x_p)$, the area A satisfies $\frac{dA}{dt} \leq 0$ for any admissible evader control input. Furthermore, $\frac{dA}{dt} = 0$ if and only if the evader follows the following strategy:*

$$\gamma^*(x_e, x_p) = \frac{\alpha_h \boldsymbol{\eta}_h - \alpha_v \boldsymbol{\eta}_v}{\sqrt{\alpha_h^2 + \alpha_v^2}}. \tag{5.14}$$

Proof. For an arbitrary d with $\|d\| \leq 1$:

$$\begin{aligned} \frac{dA}{dt} =& \frac{\partial A}{\partial p} \mu^*(x_e, x_p) + \frac{\partial A}{\partial e} d \\ =& -\sqrt{\alpha_h^2 + \alpha_v^2} + \left(\alpha_h \boldsymbol{\eta}_h - \alpha_v \boldsymbol{\eta}_v\right)^T d \leq 0, \end{aligned}$$

where equality holds if and only if $d(t) = \gamma^*(x_e(t), x_p(t))$. □

To prove that the capture condition is achieved in finite time, it is necessary to show that the distance between the pursuer and the evader is strictly decreasing

whenever the area A is constant. For this purpose, define

$$z(x_e, x_p) = \|\xi(x_e, x_p)\|^2 = (x_e - x_p)^T(x_e - x_p).$$

Clearly, the variable z is the squared Euclidean distance between the evader and pursuer. From the preceding discussions, the range of z lies in $[r_c^2,\ l_{\max}^2]$. The following result shows that $\dot{z} < 0$ whenever $\dot{A} = 0$.

Lemma 6. *If $\dot{A} = 0$, then under the pursuit strategy μ^*, the following holds:*

$$\frac{dz}{dt} = -\frac{4z}{\sqrt{z + (l_{max} - l_{min})^2}} \leq \frac{-4r_c^2}{\sqrt{r_c^2 + l_{\max}^2}}.$$

Proof. By Lemma 5, $\dot{A}(t) = 0$ if and only if $d(t) = \gamma^*(x_e(t), x_p(t))$. Thus, if the pursuer input is selected according to the strategy μ^*, then whenever $\dot{A} = 0$, then

$$\begin{aligned}
\dot{z} =& 2(x_e - x_p)^T(\dot{x_e} - \dot{x_p}) \\
=& 4\boldsymbol{\xi}^T\left(\frac{\alpha_h}{\sqrt{\alpha_h^2 + \alpha_v^2}}\boldsymbol{\eta}_h\right) \\
=& \frac{-2L\|\boldsymbol{\xi}\|}{\sqrt{\frac{L^2}{4} + \frac{(l^2 - (L-l)^2)^2}{4\|\boldsymbol{\xi}\|^2}}} \\
=& -\frac{4z}{\sqrt{z + (2l - L)^2}} \\
\leq& \frac{-4r_c^2}{\sqrt{r_c^2 + l_{\max}^2}},
\end{aligned}$$

where the second equality follows from the fact that $\boldsymbol{\xi}^T\boldsymbol{\eta}_v = 0$, and the last inequality follows from the monotonicity of the function $\frac{4z}{\sqrt{z+(2l-L)^2}}$ for $z \geq 0$. □

By this result, z is strictly decreasing whenever the area A remains constant. However, there remains the possibility that z is increasing on time intervals where A is strictly decreasing. The question then becomes whether there exists an evader control that can keep z inside $[r_c^2,\ l_{\max}^2]$ while preventing A from reaching 0. The following result proves that this is not the case, by exploiting certain properties of

the proposed pursuit strategy.

Lemma 7. *Under the pursuit strategy μ^*, if $\dot{A} \geq -\beta$ for some positive constant $\beta > 0$, then $\dot{z} \leq -c(\beta)$, where the bound $c(\beta)$ is given by*

$$c(\beta) = \frac{\sqrt{2}r_c^2}{l_{\max}} - \frac{4l_{\max}}{l_{\min}}\beta. \tag{5.15}$$

Proof. Under strategy μ^*, the following identities hold

$$\begin{cases} \dot{A} = -\sqrt{\alpha_h^2 + \alpha_v^2} + (\alpha_h \boldsymbol{\eta}_h - \alpha_v \boldsymbol{\eta}_v)^T d \\ \dot{z} = 2(x_e - x_p)^T d - \frac{2z}{\sqrt{z+(2l-L)^2}} \end{cases}$$

Now suppose $\dot{A} \geq -\beta$. From the relations $\boldsymbol{\eta}_h = \frac{x_e - x_p}{\|x_e - x_p\|}$, $\alpha_h = -\frac{L}{2}$, and $\alpha_v \boldsymbol{\eta}_v^T d \geq -|\alpha_v|$, it is true that

$$\begin{aligned} (x_e - x_p)^T d &\leq -\frac{2\|x_e - x_p\|}{L}\left[\sqrt{\alpha_h^2 + \alpha_v^2} - \beta - |\alpha_v|\right] \\ &\leq \frac{2\|x_e - x_p\|\beta}{L} \leq \frac{2l_{\max}}{l_{\min}}\beta, \end{aligned}$$

which implies that

$$\dot{z} \leq \frac{4l_{\max}}{l_{\min}}\beta - \frac{\sqrt{2}r_c^2}{l_{\max}}.$$

□

Notice that this lemma also implies $\dot{A} < -\beta$ whenever $\dot{z} > -c(\beta)$. For the rest of this section, it is assumed that the β parameter in Lemma 7 is chosen such that $c(\beta) > 0$. Now the previous results in this section will be combined to show that under μ^*, the area A or the distance δ decreases until the capture condition is met.

Consider an "energy" function E

$$E = kA + Z$$

for some positive constant k. Clearly, $E = 0$ if and only if $A = 0$ and $z = 0$, both of which imply that capture occurs. A proof will now be presented that E will decline to zero as t increases.

Theorem 8. *Under the pursuit strategy μ^*, if the capture condition has not been achieved before time $t > 0$, then for some positive constant $\beta > 0$*

$$E(t) \leq E(0) - c(\beta)t$$

where $E(0)$ is the initial energy and $c(\beta)$ is defined as in (5.15).

Proof. Lemmas 6 and 7 imply that one of the following conditions must be true at any given time:

1. $\dot{A} \geq -\beta$ and $\dot{z} \leq -c(\beta)$, or
2. $\dot{z} > c$ and $\dot{A} < -\beta$

Note that the derivative of E is

$$\dot{E} = k\dot{A} + \dot{z}$$

and $\dot{A} \leq 0$ for all time. Then, under condition 1, $\dot{A} \leq 0$ and $\dot{z} \leq -c(\beta)$, thus $\dot{E} \leq -c(\beta)$. The rate of change of z is limited by the maximum speed of the two agents and the geometry of the domain. Since $\dot{z} = 2\delta\dot{\delta}$, $\delta \leq l_{max}$, and $\dot{\delta} \leq 2$, then $\dot{z} \leq 4l_{max}$. Now, let $k = \frac{4l_{max}+c(\beta)}{\beta}$. Under condition 2, $\dot{A} < -\beta$ and of course $\dot{z} \leq 4l_{max}$, thus $\dot{E} \leq -k\beta + 4l_{max}$, therefore again $\dot{E} \leq -c(\beta)$, guaranteeing that the energy will decrease to 0 in finite time, leading to capture. □

5.2.2 Non-Convex Domains and Unequal Speeds

For non-convex domains and unequal speeds, it is more difficult to construct the safe-reachable set geometrically. Instead, the modified fast-marching method (FMM) presented in Chapter 4 can be used to compute the safe-reachable set on a grid. The area can then be approximated using the grid and the gradient computed numerically.

The details of the modified FMM algorithm can be found in Section 4.1 and will not be repeated here. A sketch of the algorithm is as follows: first, the standard

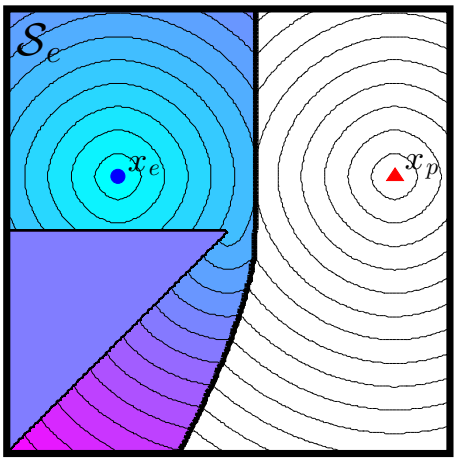


Figure 5.3: **An example of safe-reachable set computed in a non-convex game domain, with contours plotted for equal time-to-reach values for each agent.**

FMM algorithm is used to compute the time-to-capture $\psi_i^c(y_{i,j})$ for every node $y_{i,j}$ on the grid. The modified FMM is then used to compute the safe-reachable set $\mathcal{S}_e$, beginning with the current position of the evader x_e, and successively adding points that can be reached in time $\varphi(y_{i,j})$ with $\varphi(y_{i,j}) < \psi_i^c(y_{i,j}), \forall i$. An example of the computation performed for a non-convex region with a triangular obstacle and point capture ($r_c = 0$) is shown in Figure 5.3, with equal time-to-reach values plotted for the two agents. Note that the boundary of $\mathcal{S}_e$ is no longer a straight line, since the evader must first move to the corner of the obstacle in order to reach points in the lower portion of the game.

Let N_s be the number of grid nodes for which $\varphi(y_{i,j}) < \psi_i^c(y_{i,j}), \forall i$. The area of $\mathcal{S}_e$ can be approximated as

$$A \approx N_s h^2$$

where h is the grid spacing. The gradient with respect to pursuer movements can be numerically approximated by perturbing each pursuer by h horizontally and vertically

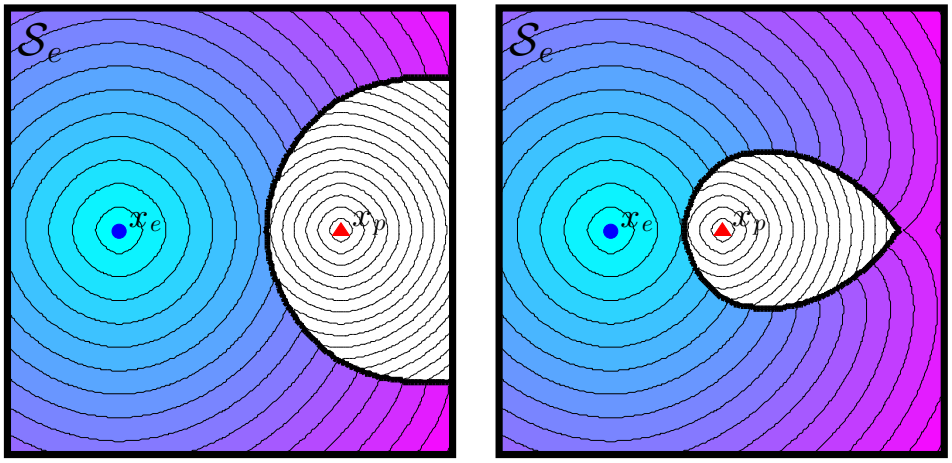


Figure 5.4: **An illustration of a scenario where the pursuer (red triangle) is slower than the evader (blue circle). Note how $\mathcal{S}_e$ grows larger as the pursuer gets closer to the evader.**

on the grid and recomputing A. The pursuit strategy is identical, with

$$u_i^* = \mu_i^*(x_e, x_p^1, \ldots, x_p^N) \triangleq -v_{i,max} \frac{\frac{\partial A}{\partial x_p^i}}{||\frac{\partial A}{\partial x_p^i}||_2}.$$

Some Remarks on Non-Convex Pursuit and Unequal Speeds

There are two things to note about using safe-reachable area pursuit in non-convex domains with unequal speeds. The first is that the algorithm is only effective in cases where the pursuers are at least as fast as the evader. This is due to the fact that, if the pursuer is slower than the evader, the evader's safe-reachable set actually shrinks in area as the pursuer goes farther from the evader, as shown in Figure 5.4. Thus area minimization will likely not lead to capture.

A second point is that currently no proof has been found to guarantee capture in the non-convex case, although the area minimization strategy works well empirically. The major obstacle in this case is that the game is no longer symmetric with respect to the area of $\mathcal{S}_e$. This is illustrated in Figure 5.5, which shows the safe-reachable set for an evader that must turn around a non-convex obstacle. This configuration gives an advantage to the evader, as it moves in the same direction whether it is

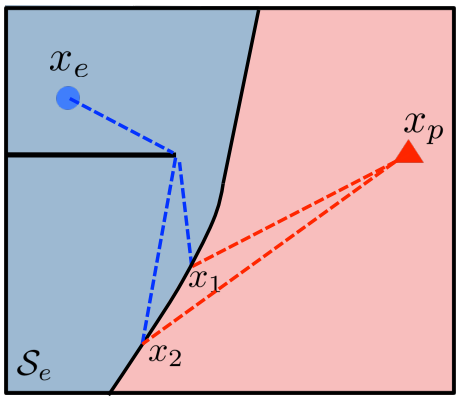


Figure 5.5: **Illustration of the asymmetry between the evader and the pursuer when the domain is non-convex.**

headed for the point x_1 or x_2. Thus, if the evader were to move distance ϵ toward x_1, it has also decreased its distance to x_2 by ϵ. If the pursuer moves to maintain x_1 on the boundary of the safe-reachable set, then it can move ϵ toward x_1, but it will have moved some distance less than ϵ toward x_2, and x_2 will have become part of the evader's safe-reachable set. In this manner, the evader is able to increase the area of its safe-reachable set regardless of the pursuer input.

5.3 Simulation Results

A number of simulations were conducted to evaluate the proposed safe-reachable area minimization pursuit strategy. The performance of the pursuit strategy was compared in a sequence of trials against two other pursuit algorithms: pure pursuit, where the pursuers attempt to instantaneously minimize the distance between each pursuer and the evader, and a numerical approximation of the optimal Hamilton-Jacobi-Isaacs solution on a grid. Several simulation examples will be used to highlight some qualitative properties of the safe-reachable area pursuit strategy, and then the quantitative results of the numerical trials will be presented.

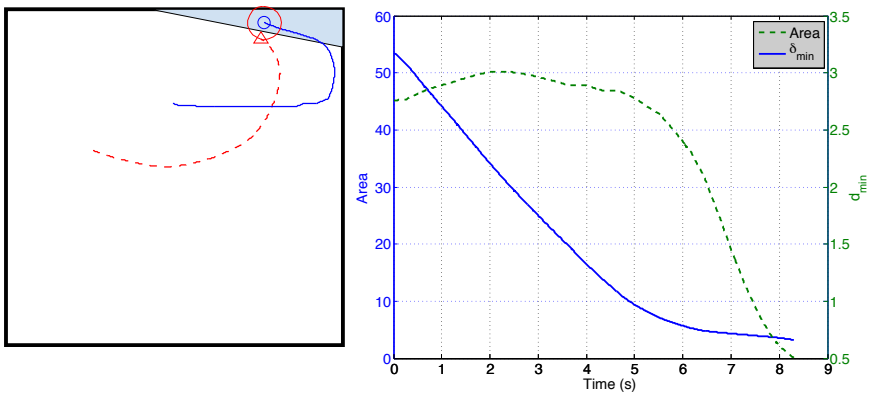


Figure 5.6: **Simulation results for a single pursuer (triangle, dashed line) and evader (circle, solid line), showing (a) the trajectories and (b) the area of $\mathcal{S}_e$ and δ_{min} over time.**

5.3.1 Illustrative Examples

A few simulations are presented here to highlight some qualitative aspects of the safe-area pursuit algorithm. The first set of simulations were conducted in a 10 x 10 square using the convex, analytic pursuit algorithm. The maximum speed was 1 for all agents, with capture radius of 0.5, and time steps of 0.01. In these simulations the trajectory of the evader was controlled by human input, and pursuers that did not have a line of control bordering on the evader's safe-reachable set were commanded to head straight for the evader.

An example trajectory for a game with a single pursuer is shown in Figure 5.6. The critical trade-off between area and distance is highlighted by this example. Note that initially the pursuer did not move directly toward the evader, and thus the distance between the agents did not decrease, but the area decreased very quickly. Near the end of the game the area decreased slowly while the distance decreased very quickly.

Figure 5.7 shows a comparison between the pure pursuit strategy and safe-reachable area pursuit for a scenario with 3 pursuers and highlights the cooperation in the area-minimization strategy. Pure pursuit is shown in Figure 5.7a, and safe-reachable area pursuit is shown in Figure 5.7b. The pursuers began closely grouped, and in pure pursuit they acted independently, resulting in a prolonged chase. With the safe-reachable

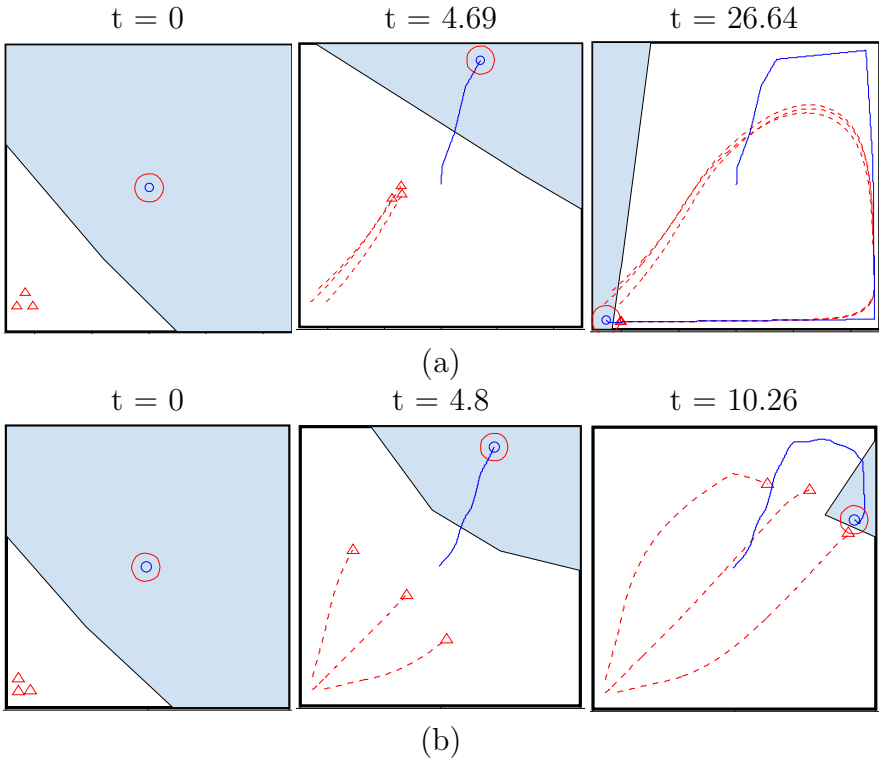


Figure 5.7: **A scenario with 1 evader (blue circle) and 3 pursuers (triangles, dotted lines) using (a) pure pursuit, and (b) area minimization, highlighting the cooperation enforced by the area-minimizing pursuit strategy when the pursuers begin tightly spaced.**

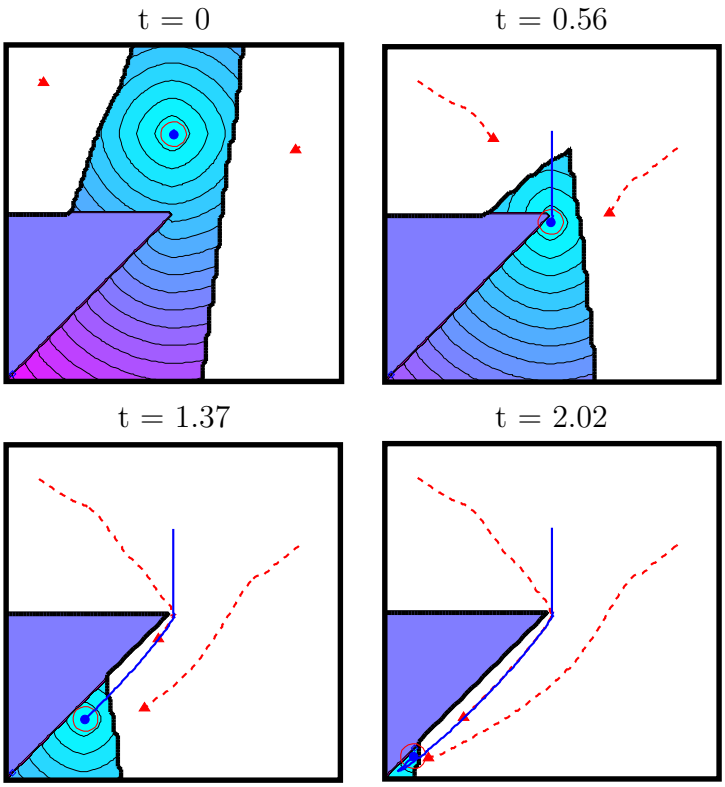


Figure 5.8: **An example scenario showing 2 pursuers and 1 evader in a non-convex environment, solved using the modified FMM algorithm.**

area strategy, the pursuers gradually separated to surround the evader. The cooperative behavior effectively contained the evader, limiting its movements until capture was achieved.

Pursuit in a non-convex environment is shown in Figure 5.8, which shows 2 pursuers (red triangles) pursuing a single evader (blue circle) in a simple, non-convex region with a triangular obstacle. The evader's safe-reachable set $\mathcal{S}_e$ is shown at each time, with equal time-to-reach contours plotted within $\mathcal{S}_e$.

The simulations were conducted in Matlab on a Macbook Pro laptop with a 2.2 Ghz Intel Core i7 processor with 8 GB of RAM, with computation per time-step of less than 1ms to calculate inputs for all the pursuers in the analytic, convex case, and about 100ms for each pursuer using FMM. Note that some small errors are introduced by discretization of the control scheme when distances between the evader

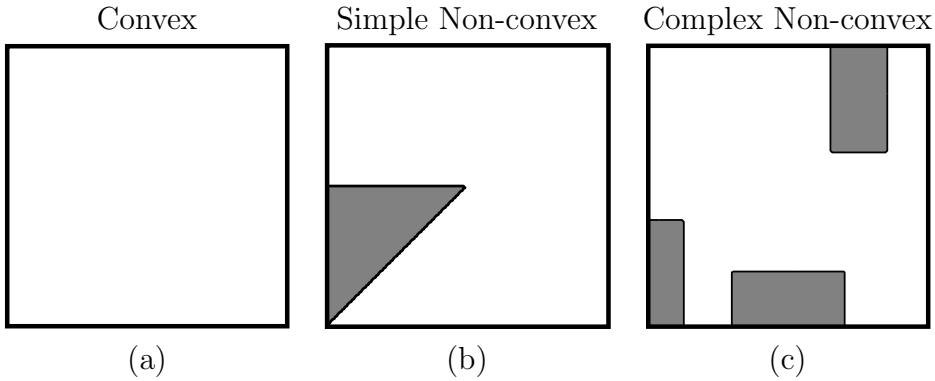


Figure 5.9: **Domains used for the comparison simulations.**

and pursuers are comparable to the distance traveled by an agent in a single time step. Reducing the time step alleviates the problem without eliminating it entirely, and increasing the capture radius also reduces the chance of this problem occurring. It is possible that some relationship can be found between step size, velocity, and the capture radius to formally guarantee this in a discrete-time situation.

5.3.2 Comparison Tests

The results of the comparison tests conducted to evaluate the safe-reachable area pursuit strategy will now be presented. Two groups of trials were conducted. The first set of trials matched safe-reachable area pursuit, pure pursuit, and the numerical HJI strategy against each other in tests with one pursuer and one evader. For these tests, a numerical approximation to the HJI equation was computed on a 40x40 grid for a simple non-convex region, shown in Figure 5.9b, and the pursuer and evader strategies were evaluated by numerical differentiation. The three different pursuit strategies were then evaluated against the approximate optimal evader strategy for 500 initial conditions generated randomly.

The results of this test are displayed in Figure 5.10. In general, the area-minimization strategy performed slightly worse than the numerical HJI pursuit strategy, and the pure pursuit strategy performs worse than either of the others. However, it should be noted that the numerical HJI strategy depends on numerical differentiation of the approximated value function on a grid, and numerical errors can lead to sub-optimal

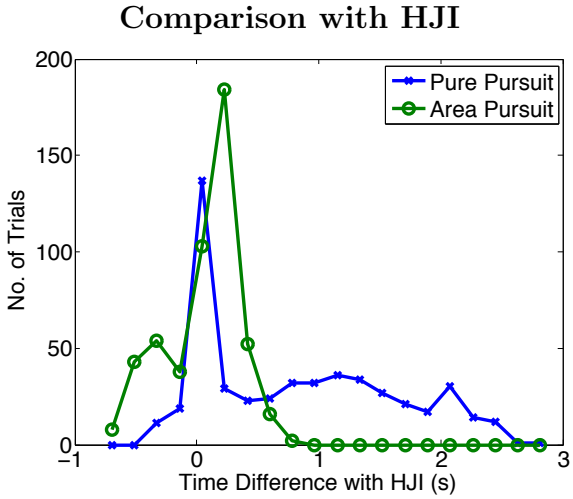


Figure 5.10: **Histogram showing the difference in capture times in each single pursuer trial between the area minimization strategy, the pure pursuit strategy, and the numerical Hamilton-Jacobi-Isaacs pursuit strategy.**

pursuer and evader actions. For example, in 29% of trials the area minimization pursuit strategy out-performed the "optimal" HJI pursuer, and similarly in 8.5% of trials the pure pursuit strategy resulted in faster capture-times.

In addition to the numerical issues discussed above, due to computational complexity, the HJI solution can only be found for the case of a single pursuer. To evaluate the performance of the pursuit strategy with multiple pursuers, a further series of tests were conducted comparing safe-reachable area pursuit with pure pursuit. For these tests, the evader strategy was defined as the following: the evader selects as its target point y^* the farthest point from itself in $\mathcal{S}_e$, that is

$$y^* = \max_{y \in \mathcal{S}_e} \delta_g(y, x_e)$$

where δ_g is the geodesic distance between y and x_e. Once the evader reaches a certain distance (set here as half of the capture radius) from y^*, a new target is selected and the evader will proceed toward this target.

Three sets of trials were conducted with 1, 2, and 3 pursuers, with one set in a square, convex region using the analytically derived pursuit formula and two others

in non-convex regions using FMM, shown in Figure 5.9. For each set, 500 random initial conditions of pursuer and evader positions were generated. The results of the tests are summarized in Figure 5.11, showing histograms of the difference in time between trials, defined as the time required for the safe-reachable area strategy minus the time required for pure pursuit for each initial condition.

Table 5.1 shows the fraction of trials in each case where the area pursuit strategy resulted in faster capture times than pure pursuit. Figure 5.11a shows the distribution of results in the convex environment. In this scenario, the safe-reachable area pursuit strategy resulted in clearly superior performance, with the vast majority of trials resulting in faster capture times. The distribution of times seems somewhat bimodal in these trials, with a number of trials where area minimization pursuit and pure pursuit performed similarly, and then a large group where the area minimization pursuit strategy was clearly superior.

The results for the non-convex scenarios are shown in Figure 5.11b for the simple non-convex environment, and in Figure 5.11c for the complex non-convex environment. The simple non-convex case still resulted in a large majority of trials where the area minimization pursuit strategy gave faster capture, although in a smaller percentage of trials than the convex scenario. The complex non-convex case showed a decline in the performance of the area minimization strategy relative to the pure pursuit strategy. This is due to the fact that the obstacles create areas where the width of the free space is of similar size to the capture radius, thus the pure pursuit can still "trap" the evader, lessening the advantage conferred by the safe-reachable area strategy. In fact, it is to be expected that as the space becomes more and more similar to a single long, narrow corridor, the pure pursuit and area-pursuit strategies should have identical performance. This is especially evident in the trials with only 1 pursuer, where only about 50% of trials resulted in faster capture with safe-reachable area pursuit, with a long tail of trials where the area pursuit performed much worse than pure pursuit. These typically occurred in trials where the evader was able to escape from a confined portion of the game domain against the area pursuer, in part due to numerical errors in the area differentiation. Nonetheless, the area minimization strategy showed a clear superiority in the trials with 2 and 3 pursuers, demonstrating

	1 pursuer	2 pursuers	3 pursuers
Convex	91.6%	98.2%	96.6%
Simple Non-convex	67.2%	86.2%	86.4%
Complex Non-convex	48.2%	72.6%	76.4%

Table 5.1: **Percentage of trials for which the safe-reachable area minimization pursuit strategy out-performed the pure pursuit strategy.**

the benefit of cooperation.

5.4 Experimental Results

Experiments were conducted using the safe-reachable set area minimization pursuit strategy on the BEARCAT platform, using the smartphones only. The purpose of the tests was to evaluate the feasibility of the pursuit strategy as a tool for guiding human agents in a cooperative pursuit task. The game was played by two pursuers and a single evader in a small, convex field measuring about 80m x 40m. Agents were tracked using HTC Incredible smartphones, with the agent positions, game region, and the evader safe-reachable set also displayed on the phones. Pursuer strategies were computed using the analytic formulation directly on the phones, and the optimal heading direction was displayed for each pursuer, along with that pursuer's active boundary of the safe-reachable set.

Results from one of these experiments is shown in Figure 5.12. In this experiment, the pursuers began the game on the western side of the field, with the evader on the eastern side. The camera was placed to the north of the field, looking south. The pursuers were able to successfully trap and capture the evader.

An important point to note is that during these experiments, the GPS positions of the agents were not always perfectly reliable, and as a result the optimal headings were not always correct. Nonetheless, the pursuers were able to use the boundaries of the evader safe-reachable set to guide their actions. Instead of following the optimal headings exactly, each pursuer's active boundary of $\mathcal{S}_e$ gave an idea of that pursuer's area of responsibility, such that each pursuer could still reduce the area of $\mathcal{S}_e$ by attempting to "push" the boundary toward the evader. In addition, the visual display

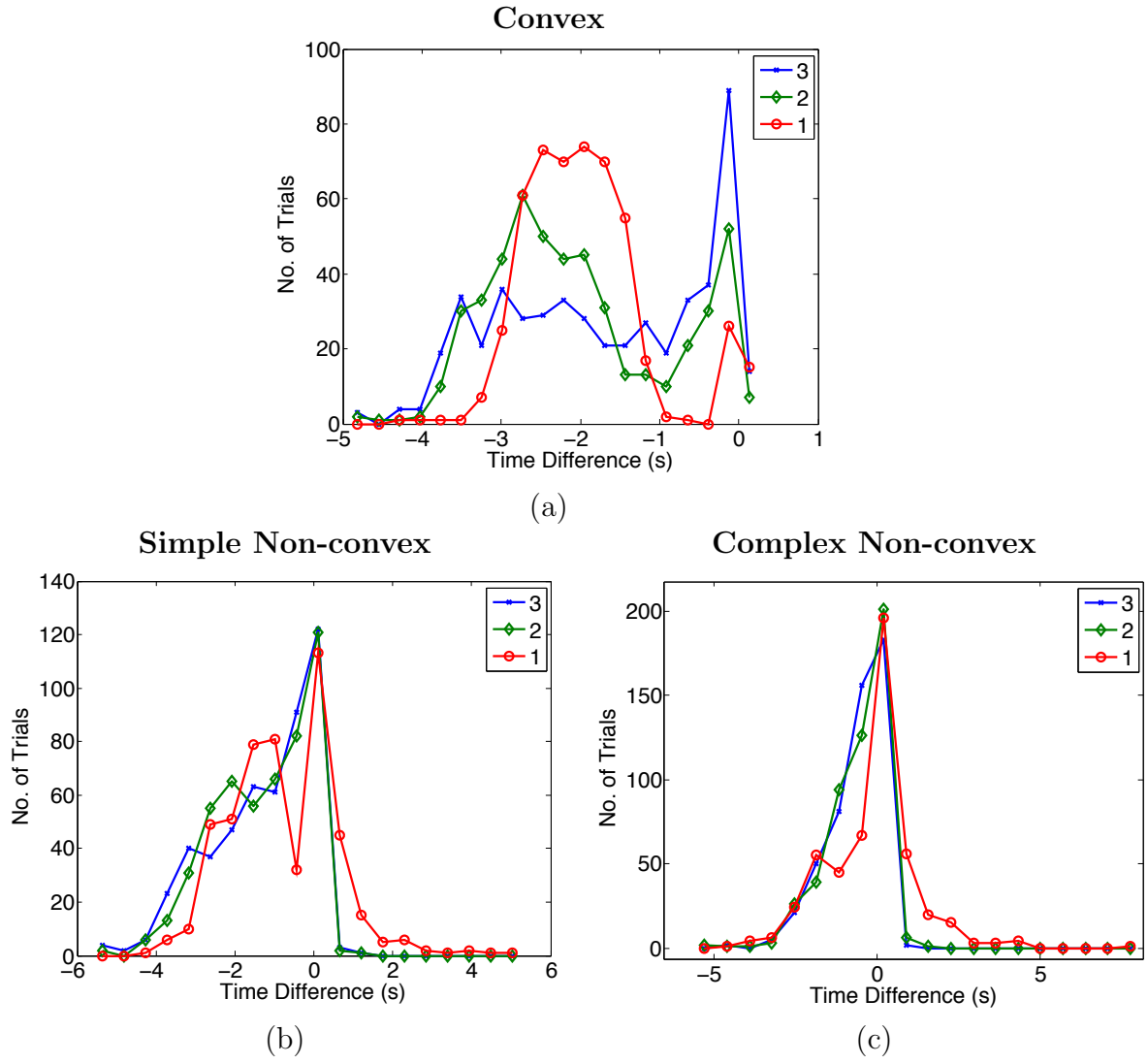


Figure 5.11: **Histograms showing the number of trials versus difference in capture time between the pure pursuit and safe-reachable area minimization strategies for (a) the convex game domain, (b) the simple non-convex domain, and (c) the complex non-convex domain.**

of the safe-reachable set allowed for implicit cooperation between the human pursuers, without the need for verbal communication. Thus, even though the agents were unable to always directly used the computed optimal headings, the safe-reachable set still enabled the pursuers to efficiently coordinate in capturing the evader.

5.5 Discussion

The safe-reachable area minimization strategy presented in this chapter has several important strengths. Like the open-loop game in Chapter 4, the construction of the safe-reachable set allows a high-dimensional problem to be reduced to lower dimensions, easing the computational burden. Additionally, each pursuer can compute its inputs independently, allowing the strategy to run efficiently in real time. Yet information is shared through the set itself, enabling cooperation between the pursuers and reducing capture time. The safe-reachable set itself also encompasses global information about the game domain, giving an advantage over a strategy like pure pursuit that ignores the presence of obstacles and boundaries.

One weakness of the strategy as presented is that safe-reachable area pursuit cannot handle more general domains that are not simply connected, where obstacles form holes in the domain. The numerical evaluation of the area gradient may also be improved. From the simulations and experiments it was noted that the behavior of the pursuers was typically smoother with the analytic input formula in convex domains than with the FMM computation.

Overall, though, safe-reachable area minimization seems to hold promise as a cooperative pursuit solution approach. As seen before in the capture-the-flag reachability solutions in Chapter 3, the visualization of the reachable set is a useful tool for human agents. Although GPS noise and time delay renders the optimal headings sometimes unreliable in practice, the visualization of the set itself allows the agents to compensate for these disturbances and head in the correct general direction to capture the evader. In general, this result supports the idea that reachable sets are effective visual tools for assisting human agents.

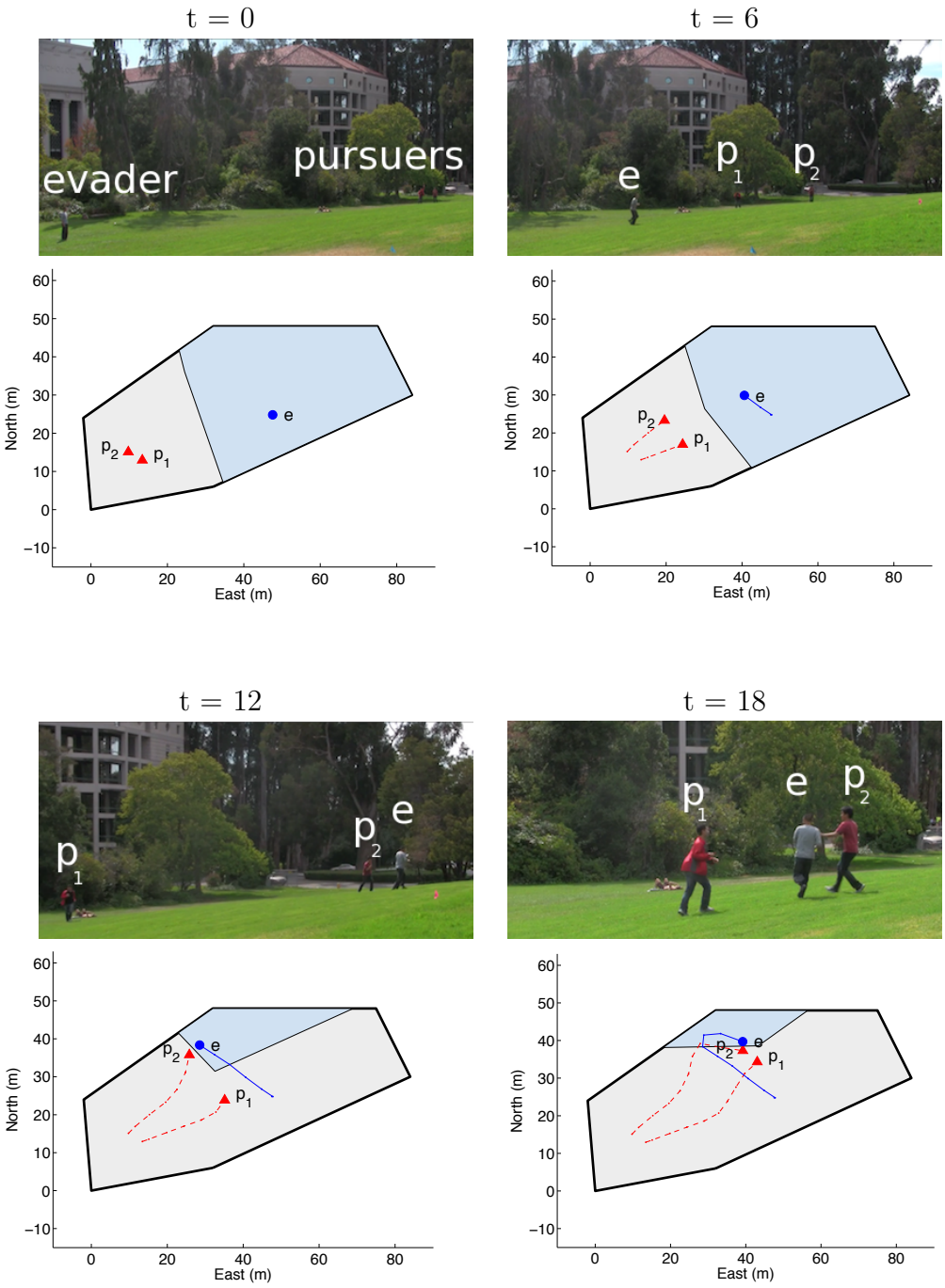


Figure 5.12: **Video stills and data plotted for an experiment using the safe-reachable set area minimization pursuit strategy in BEARCAT. For visual clarity, the evader is labeled e and the pursuers are labeled p_1 and p_2.**

Chapter 6

Reachability for UAV Control

The reachability-based results discussed in the preceding chapters have applications beyond directing agents in adversarial games. In addition to multi-agent games, the reach-avoid formulation presented in Chapter 3 can play a useful role in designing systems with robust safety requirements. Reachable sets can also be utilized beyond directly guiding the agents playing the game.

This chapter describes two applications of reachability to problems of UAV control. The first problem is that of designing a sequenced control input that can safely take a maneuvering UAV through a series of maneuvers while observing safety requirements in the presence of external disturbances. Control for a quadrotor UAV attempting a backflip maneuver will be used to demonstrate how the Hamilton-Jacobi-Isaacs reachability tools can be used to compute reach-avoid sets of states that are guaranteed to arrive at a desired target set while avoiding undesired configurations. The formulation of this control problem, its solution, and related experimental results are described in Section 6.1. These results were first presented in [37].

The second problem to be considered is the task of using a UAV to provide sensing for a human agent in a limited-information version of the 1 vs. 1 capture-the-flag game. The reachable sets computed for the perfect-information game are used to provide guidance to the UAV in searching for an opposing agent, allowing it to disregard possible locations for the unseen agent that cannot affect the outcome of the game. This problem is discussed in Section 6.2.

6.1 Aerobatic Maneuver Design and Execution

As the demands on performance grow, UAVs require increasingly sophisticated control systems to take advantage of their full range of capabilities. This section addresses one such challenge, designing safe aerobatic maneuvers. The full nonlinear dynamics are simplified into a hybrid model and reachability analysis is used to design and implement a backflip maneuver for a quadrotor helicopter.

6.1.1 Related Work

Many approaches to the control of highly maneuverable aircraft have used statistical learning techniques, for example by copying an expert pilot's example trajectory either through machine learning or via manual creation of approximate trajectories [25, 63, 64], or through iterative schemes which update the control laws at each step using information from experimental runs [65, 66]. These methods have been able to push the envelope of what is possible with autonomous control, but since they lack performance guarantees about their stability and robustness, their use must be limited to situations where safety is not critical.

An alternate approach that allows more rigorous formal analysis is hybrid decomposition. In this method, the behavior of the system is approximated as a finite set of discrete modes representing the dynamics in specific regimes or portions of the state space. An important consideration in the design and control of systems with switched dynamics is the safety of transitions between modes. For example, in the case of aircraft maneuver sequences it is necessary to ensure that an aircraft completing one maneuver is able to begin the next maneuver without being in an unsafe or infeasible configuration.

This has been accomplished in a variety of ways in the past. Previous helicopter maneuvering work used "trim states" such as steady flight or hover that the vehicle is required to return to after a maneuver before beginning another [67]. In robotic manipulation, sequences of specially derived Lyapunov functions have been used to

guarantee that a defined sequence can be followed [68] as well as analytically calculating regions where a given motion is guaranteed to place a part in a desired configuration [69]. There has also been extensive work in the hybrid systems literature on the construction of switching regions for mode switching [70, 71, 72]. Specifically, partitions or manifolds in the state space have been found that are regions of attraction for particular modes or controllers.

Much of the existing work has focused on switching under nominal conditions or sensing uncertainty and has not explicitly considered external disturbances. Additionally, in most cases the particular method to ensure continuity between modes has been specific to the application at hand. These methods have also not considered separate safety requirements such as obstacle avoidance. In this section, a general method using reachable sets is developed that can be used to design aerobatic maneuvers for a quadrotor helicopter. This method accounts for external disturbances as well as separate safety requirements, and is flexible enough to be adapted to many nonlinear systems. In particular, the HJI differential game formulation of reachable sets introduced in Chapter 3 is used to construct maneuvers that safely transition through a sequence of modes to perform a backflip maneuver while arriving at a target state and avoiding unsafe states en route. While the actual simulation and experimental trajectories are similar to those achieved in the literature using other methods, the reachable sets provide both formal guarantees of safety and attainability, as well as information about how much the maneuvers can deviate from those given and still maintain those guarantees.

6.1.2 Reachability Formulations

The safety analysis in this section proceeds using reach-avoid formulations as developed in Chapter 3. However, for the application considered here it is necessary to consider the cases of reaching a target and avoiding an undesired set separately. The latter requires a slightly modified formulation, which will be explored subsequently. The dynamics of the system are assumed to be

$$\dot{\mathbf{x}} = f_i(\mathbf{x}, u, d),\ \mathbf{x}(0) = \mathbf{x}^0$$

with two opposing sides, one playing the role of the control, with input $u \in U$, and the other playing the role of the disturbance, with input $d \in D$. U and D denote the compact sets of possible inputs, respectively. The subscript i denotes that the dynamics of the system may differ in different modes or sections of a maneuver. The control input is a function of the control strategy μ, and the disturbance uses the disturbance strategy γ. These are analogous to the attacker and defender strategies described in Chapter 3. The goal of the control is to either arrive in a desired state configuration, a process to be referred to as attainability, or to keep the system out of some undesired state configuration, which will be called safety. The goal of the disturbance in both cases is to oppose the control.

Capture sets for attainability

In attainability, the goal of the controller is to drive the system into some desired set $\mathcal{T}$, while the disturbance is assumed to be attempting to drive the system away. This is similar to the reach-avoid situation discussed in Chapter 3, except there is no undesired set to avoid. The capture set with respect to a time horizon T can be computed directly using the reach-avoid operator presented in Chapter 3. For simplicity of notation the capture set will be referred to simply as

$$\mathcal{RA}_T(\mathcal{T}) = \mathcal{RA}_T(\mathcal{T}, \emptyset).$$

Avoid sets for safety

Although the capture set can be computed straightforwardly using the reach-avoid operator, the avoid set computation for safety requires a modification of the reach-avoid computation. For safety, the goal is to prevent the system from entering an undesired set K, and the disturbance is assumed to be attempting to drive the system into this unsafe area of the state space. The avoid set relative to K for some time horizon T is denoted $\mathcal{A}_T(K)$, and is defined as the region of the state space where, for any control strategy μ, there exists some disturbance strategy γ such that $\mathbf{x}(t) \in K$ for some $t \in [0, T]$.

The avoid set can be computed using a reversal of the roles of control and disturbance in the reach-avoid formulation presented previously. In this case, the reach-avoid computation is performed with the disturbance attempting to drive the system into the avoid set and the control trying to stay out. Thus $\mathcal{A}_T(K)$ can be computed as

$$\mathcal{A}_T(K) = \tilde{\mathcal{RA}}_T(K, \emptyset)$$

where the reach-avoid operator is modified slightly to reverse the order of play. Recall that the standard reach-avoid set presented in Chapter 3 is defined as

$$\mathcal{RA}_T(R, K) = \{\mathbf{x} \mid J(\mathbf{x}, 0) \leq 0\}$$

where

$$J(\mathbf{x}, 0) = \min_{\mu} \max_{\gamma} J_{\mathcal{T}}(\mathbf{x}(T))$$

where μ and γ are the control and disturbance strategies, respectively, over the game time horizon. The modified reach-avoid operator is defined similarly, with

$$\tilde{\mathcal{RA}}_T(\mathcal{T}, K) = \left\{\mathbf{x} \mid \tilde{J}(\mathbf{x}, 0) \leq 0\right\}$$

where

$$\tilde{J}(\mathbf{x}, 0) = \max_{\mu} \min_{\gamma} J_{\mathcal{T}}(\mathbf{x}(T))$$

and the roles of the control and disturbance are reversed, but the game is still conservative with respect to the disturbance.

Once appropriately formulated, the cost functions are again computed numerically using the Level Set Toolbox developed at the University of British Columbia [73].

Reachable sets without optimal control

In many cases (such as with the aerobatic maneuver example to be described in Section 6.1.3) it can be convenient to use a particular predetermined controller or control strategy. This can easily be incorporated into the reachable set analysis by

defining a controller

$$u = \mu_i(t, \mathbf{x}) \tag{6.1}$$

specified for the currently active mode. This controller can then be subsumed into a modified form of the system dynamics, such that

$$\dot{\mathbf{x}} = f_i(t, \mathbf{x}, \mu_i(t, \mathbf{x}), d) = \hat{f}_i(t, \mathbf{x}, d). \tag{6.2}$$

The optimal Hamiltonians for safety and reachability can be modified by removing the max and min, respectively, over u, and leaving only the optimization over d.

6.1.3 Maneuver Sequencing Using Reachable Sets

Capture and avoid sets (as described in Section 6.1.2) can be used to construct safe sequences of maneuvers, similar to the multi-stage games formulation in Section 3.2.3. Starting with the final (desired target) set $\mathcal{T}_n$, the dynamics for the final (n^{th}) maneuver can be used to generate a backwards reachable set set $\mathcal{RA}_T(\mathcal{T}_n)$ for that maneuver (i.e. all points in the state space guaranteed to reach $\mathcal{T}_n$ within time horizon T). This forms a capture set for the n^{th} maneuver, and the capture sets for each preceding target can be generated as in Section 3.2.3. Avoid sets for safety can be generated in a similar fashion. Starting from an initial unsafe set, the boundary of the full avoid set can be progressively propagated backward using the sequence of mode dynamics.

Note that this method is not guaranteed to always produce a solution (though if a solution is produced, it is guaranteed to be feasible). For example, if the disturbances are so great that the reachable sets for two subsequent maneuvers do not overlap, then there will be no guaranteed feasible way to transition between them. Since the reachable sets are generated before initiating the maneuver, however, this does not put the system in any danger (unless the maneuver is initiated anyway, despite the lack of achievability guarantees). In fact, when used properly this lack of a solution could enable the designer to spot problems in the maneuver design or the model of the system ahead of time, and correct them if possible.

Problem Formulation

To demonstrate the validity of this approach for maneuver sequencing, this method is used to develop a backflip maneuver for the STARMAC quadrotor helicopter. As the backflip is a planar maneuver the quadrotor's dynamics are modeled in a 2D plane, since the out-of-plane dynamics can be stabilized without affecting the maneuver (as supported by analysis and testing). The planar dynamics are given by:

$$\frac{d}{dt}\begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \\ \phi \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ -\frac{1}{m}C_D^v\dot{x} \\ \dot{y} \\ -\frac{1}{m}\left(mg + C_D^v\dot{y}\right) \\ \dot{\phi} \\ -\frac{1}{Iyy}C_D^\phi\dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 \\ d_x \\ 0 \\ d_y \\ 0 \\ d_\phi \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ -\frac{1}{m}\sin\phi & -\frac{1}{m}\sin\phi \\ 0 & 0 \\ \frac{1}{m}\cos\phi & \frac{1}{m}\cos\phi \\ 0 & 0 \\ -\frac{l}{Iyy} & \frac{l}{Iyy} \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} \tag{6.3}$$

where the state variables x, y, and ϕ represent the vehicle's lateral, vertical, and rotational motion, respectively; d_x, d_y and d_ϕ are disturbances; and constant system parameters are given by m for the vehicle's mass, g for gravity, C_D^v for translational drag, C_D^ϕ for rotational drag, and I_{yy} for the moment of inertia. For simplicity the vehicle's drag is modeled as linear with respect to velocity, an assumption that was experimentally shown to be sufficiently accurate.

It should be noted that in situations not captured by the 2D model (e.g. if an un-modeled disturbance overpowers the vehicle's out-of-plane stabilization) the guarantees described in this section would obviously no longer hold. The same statement is true for any guarantee regarding stability and robustness: if the model used to generate the guarantee does not closely match the actual system, then the guarantee

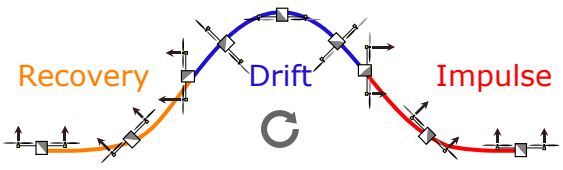


Figure 6.1: **The backflip maneuver, broken into three modes. The vehicle travels from right to left, spinning clockwise as it does so. The size of each arrow indicates the relative thrust from each rotor.**

may be invalid. However, a great deal of prior field testing has shown these modeling assumptions to be reasonable under the conditions in which the quadrotors typically operate.

For ease of analysis and visualization, it is useful to decompose high dimensional systems into multiple lower dimensional systems (assuming the system's dynamics decouple). Six-dimensional problems are difficult to visualize and time-intensive to compute, slowing the design process, so the system's states are divided into three sets for independent analysis. The rotational dynamics are analyzed to ensure the attainability of the backflip, the vertical dynamics are analyzed to ensure safety (i.e. that the vehicle remains above some minimum altitude), and the horizontal dynamics are ignored as they are not relevant to successfully completing the maneuver.

Backflip attainability

For the purpose of guaranteeing attainability, the backflip is divided into three modes as shown in Figure 6.1: impulse, in which the rotation of the vehicle is initialized; drift, where the vehicle freely rotates and falls under gravity; and recovery, which brings the vehicle to a controlled hover condition. This division is driven largely by the fact that unlike a standard helicopter, a quadrotor's blades have a fixed pitch, which means that a quadrotor is only capable of generating thrust in one direction. As a result, whenever the vehicle is inverted any thrust generated by its rotors must propel it downward. Thus, to successfully complete a backflip maneuver on the STARMAC vehicle with a slow rotational rate (e.g. around 400°/sec), it is necessary to turn off the motors while the vehicle is inverted to prevent the vehicle from propelling itself

into the ground.

Each mode is designed using the method described in the beginning of Section 6.1.3. The target sets $\mathcal{T}_3$, $\mathcal{T}_2$, and $\mathcal{T}_1$ in the $(\phi, \dot{\phi})$ space for the recovery, drift, and impulse modes respectively, are $(0 \pm 5°, 0 \pm 10°/\text{sec})$ – essentially a stable hover configuration, $(110 \pm 20°, -180 \pm 185°/\text{sec})$, and $(310 \pm 10°, -287 \pm 58°/\text{sec})$, as shown in Figure 6.2. As described in Section 6.1.2, a fixed controller (in this case a standard PD controller on ϕ) is used to drive the vehicle during the recovery mode, and a D-DD controller is used during the impulse mode.

Finally, $\mathcal{RA}_{T_3}(\mathcal{T}_3)$, $\mathcal{RA}_{T_2}(\mathcal{T}_2)$, $\mathcal{RA}_{T_1}(\mathcal{T}_1)$, are calculated as described in Section 6.1.2 using worst case disturbances. In this case, the set of allowable disturbances is assumed to be rectangular, with

$$d_{i,min} \leq d_i \leq d_{i,max} \tag{6.4}$$

along each axis $i = x, y, \phi$. The magnitude of these disturbances has a significant impact on the resulting reachable sets; if the potential worst case disturbances are too large, then a solution may not exist that will allow the vehicle to reach the target set. The symmetry of the quadrotors used in these experiments means that the aerodynamic center is very close to the center of mass. Therefore the wind conditions the quadrotors typically experience do not impart a significant rotational moment, which is confirmed experimentally in test flights. Thus for the recovery and impulse modes, the primary disturbance source is noise in the thrust produced by the motors, which is determined from previous flight data to be less than 5% of the total nominal thrust [36]. This disturbance is reduced for the drift mode due to the lack of motor noise as the motors are turned off.

Actuator saturation is modeled by constraining the thrusts to lie within the range

$$0 \leq \mathcal{F}_i \leq \mathcal{F}_{max} \tag{6.5}$$

for motors $i = 1, 2$ where $\mathcal{F}_{max}$ is the maximum thrust produced by the motors. For the quadrotors and controllers discussed in this paper, the typical operating regime is around 40% of maximum thrust with control inputs of at most 20% magnitude, thus

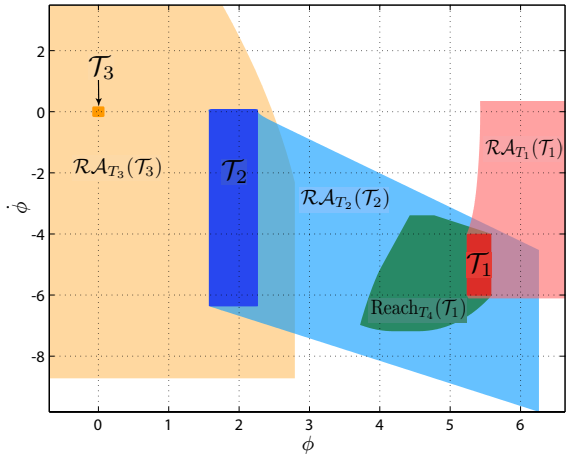


Figure 6.2: **Composite capture sets of the backflip maneuver are plotted in the ϕ (radians) vs $\dot{\phi}$ (radians/second) plane. The backflip maneuver starts in the region labeled $\mathcal{RA}_{T_1}(\mathcal{T}_1)$ and ends in the region labeled $\mathcal{T}_3$.**

actuator saturation, while accounted for in the calculations, did not have an impact on the flight experiments except for motor turn-off in the drift phase.

It was originally assumed that the motors would turn off instantaneously when the vehicle entered the drift mode; some initial experiments proved that assumption incorrect. As a result an additional mode was added for the purpose of analysis. In this mode, the motor turn off is modeled as a linear decay in the vehicle's angular acceleration, i.e.:

$$u_{\text{turnoff}} = \min\{\alpha t + \ddot{\phi}, 0\}/I_{yy} \tag{6.6}$$

where the parameter α is found using linear regression. These dynamics are then propagated forward from the target set of the impulse mode, $\mathcal{T}_1$; the resulting level set (labeled $Reach_{T_4}(\mathcal{T}_1)$ in Figure 6.2) contains all possible states the vehicle could be in while the motors are turning off. Thus, as long as this set is contained in the drift set, $\mathcal{RA}_{T_2}(\mathcal{T}_2)$, attainability of the backflip is once again guaranteed.

Backflip safety

To ensure the vehicle would perform the backflip safely, a similar procedure to that described for attainability is used. First, a final unsafe set K_3 is chosen to represent

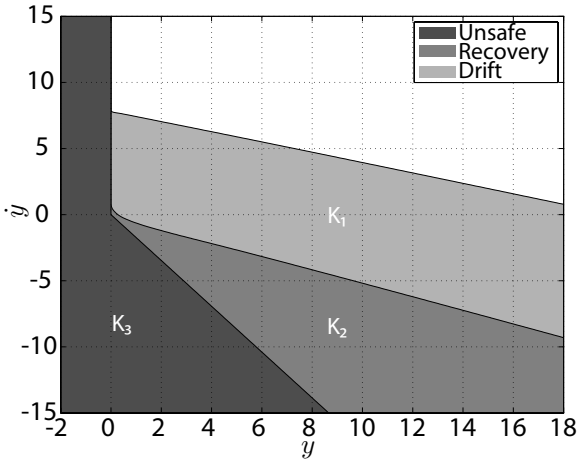


Figure 6.3: **Unsafe vertical sets of the backflip maneuver are plotted in the y (meters) vs $\dot{y}$ (meters/second) plane. As long as the vehicle begins a given mode outside that mode's unsafe set, safety is guaranteed.**

all configurations the vehicle would need to avoid during the recovery mode: the union of the region $y \leq 0$ (representing the ground) and the region $\dot{y} \leq -1.73y$ (representing the vehicle falling in such a way that it will hit the ground in a little over half a second). Because the vehicle's rotational and vertical dynamics are coupled during powered thrust, however, it is first necessary to find a way to decouple them so that safety could be analyzed solely in the vertical state space. This decoupling is accomplished by taking advantage of the fact that the recovery mode is designed to use a fixed control law. As a result, a nominal trajectory with some bounds on deviation can be generated that can then be incorporated into the system dynamics. This allows the backward reachable set to be computed as usual by propagating it backward for a fixed time T_{rec}, based on the maximum time that the rotational part of the recovery mode can take. The resulting level set $K_2 = \mathcal{A}_{T_{rec}}(K_3)$ indicates all the configurations from which it would be unsafe for the vehicle to enter the recovery mode.

In the drift mode, the rotational and vertical dynamics decouple, and so the unsafe set for the drift mode, $K_1 = \mathcal{A}_{T_d}(K_2)$ is generated by propagating backward the unsafe set for the recovery mode using the vertical dynamics. Once again, this is done for a fixed time T_d, based on the maximum length of the maneuver as calculated

from the rotational dynamics. The resulting level set represents all the configurations in which it would be unsafe for the vehicle to enter the drift mode.

For the impulse mode, it is assumed that there is no loss in altitude because the impulse mode is designed so that the vehicle's thrust will always be upward during this mode. The resulting unsafe sets are shown in Figure 6.3; as long as the vehicle begins each mode outside the unsafe set for that mode, the overall safety of the system is guaranteed. To ensure that the vehicle begins the entire maneuver outside of these unsafe sets, an additional preliminary climb mode is added before the impulse mode, in which the vehicle accelerates upward until it reaches a safe altitude and velocity. The addition of this climb mode guarantees that as long as the disturbances stays within the bounds used when calculating the capture and avoid sets, the vehicle will always be able to switch to the next mode when it reaches the target set of the current mode.

6.1.4 Results

A mosaic of one of the demonstrations of the backflip maneuver is shown in Figure 6.4. Figure 6.4a depicts the quadrotor after the initial climb mode which is the start of the impulse mode, and Figure 6.4(b) is at the end of the impulse mode and at the beginning of the drift mode. Figures 6.4b-f display the entire drift portion of the maneuver and Figure 6.4(e) shows the quadrotor inverted. Finally, Figures 6.4f-j display the recovery mode of the backflip maneuver which successively returned the quadrotor to a safe condition of $\phi = 0^\circ$ and $\dot{\phi} = 0^\circ/\text{sec}$. Figure 6.5 shows the trajectory of the vehicle corresponding to the experimental trial displayed in Figure 6.4; the labeled points correspond to the frames in the mosaic. Video of the backflip maneuver can be viewed at `http://hybrid.eecs.berkeley.edu/aerobatics.html`.

Attainability: attitude results

Figure 6.6 shows the $(\phi, \dot{\phi})$ trajectory through the designed capture sets for the backflip maneuver. Figure 6.6a shows the simulated trajectory and Figure 6.6b displays three experimental validations. As the figure illustrates, the trajectories are contained

Figure 6.4: **A mosaic of the successful demonstration of the backflip maneuver. (a): The quadrotor has finished the climb portion of the backflip and is starting the impulse mode. (b): The quadrotor has finished the impulse stage and is entering the drift portion. (b)-(f): The drift stage of the backflip. (f): The drift mode is concluding and the recovery mode has started. (f)-(j): The recovery mode is safely returning the quadrotor to its hovering position.**

within the capture sets for each maneuver. The transition between the impulse and drift modes is denoted by a black diamond, and the transition between the drift and recovery modes is indicated by a black square. The switches between the maneuvers are contained within each of their target regions, $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively. When the quadrotor switched into the drift mode, it took approximately 0.2 seconds for the motors to spin down, which explains why the quadrotor was still accelerating at the beginning of the drift maneuver. Table 6.1 displays the time spent in each mode for each experimental validation. Figure 6.7 displays the pitch of the quadrotor throughout the maneuver, which was within $\pm 5^\circ$ for almost the entire maneuver. This validates the assumption that the backflip maneuver can be modeled in the 2D $(\phi, \dot{\phi})$ plane.

Safety: Altitude Results

Figure 6.8 displays the unsafe vertical reachable sets and the switching points for the three experimental validations of the maneuver. The upper (blue) points correspond

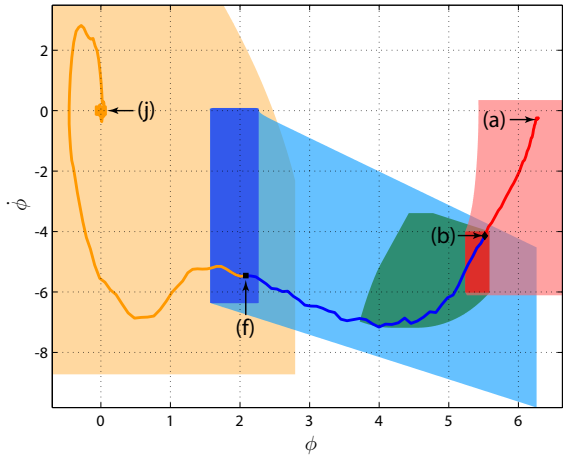


Figure 6.5: **The trajectory of the vehicle corresponding to the mosaic shown in Figure 6.4. The labeled points correspond to the frames in the mosaic.**

	Impulse (sec.)	Drift (sec.)	Recovery (sec.)
Trial 1	0.55	0.55	2.56
Trial 2	0.50	0.55	5.54
Trial 3	0.55	0.61	4.70

Table 6.1: **The amount of time spent in each mode for each experimental trial of the backflip maneuver.**

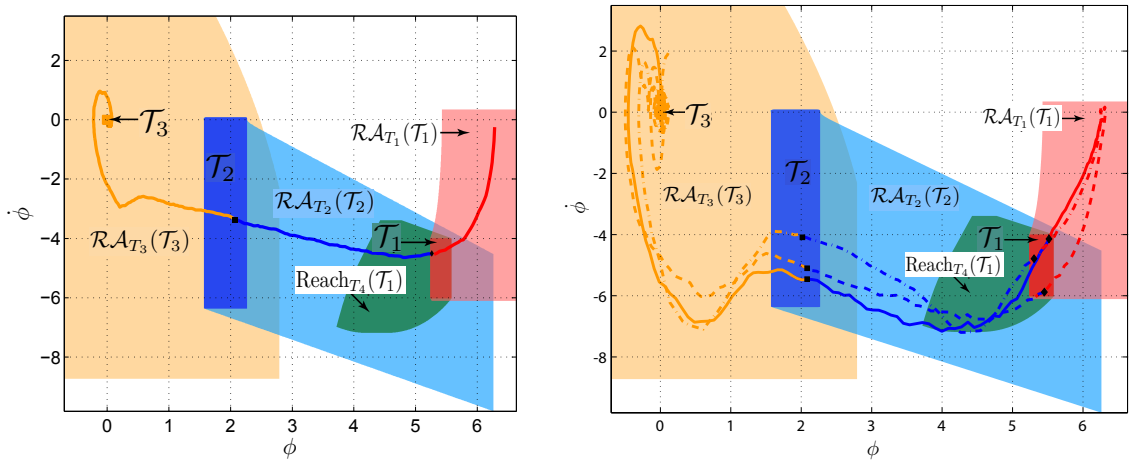


Figure 6.6: **Simulation and experimental validation of the designed backflip maneuver overlaid on the composite reach sets. The transitions from the impulse to drift mode are shown as black diamonds which are contained in region $\mathcal{T}_1$, and the transitions from the drift to the recovery mode are indicated by the black squares that are confined to region $\mathcal{T}_2$. (a) The simulated trajectory of the vehicle. (b) Three experimental validations (solid, dash and dash-dot lines) of the backflip maneuver.**

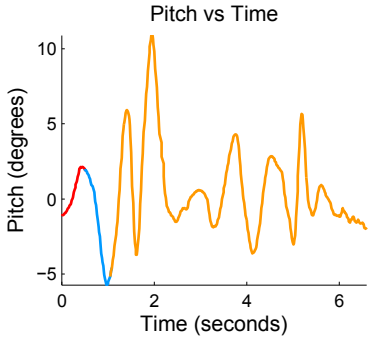


Figure 6.7: **Experimental data from a single run of the backflip maneuver showing the out-of-plane pitch of the quadrotor throughout the maneuver.**

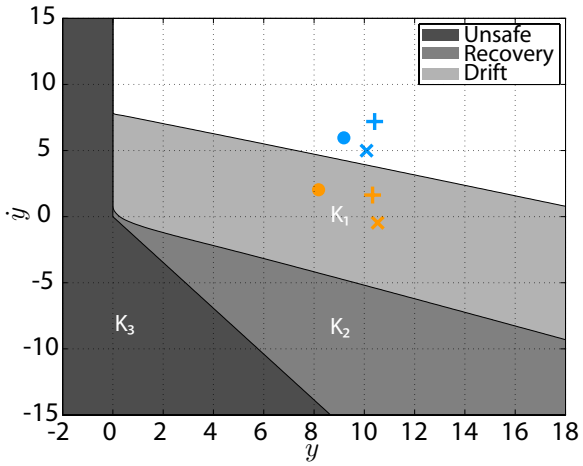


Figure 6.8: **Three experimental validations (×, +, and •) of the backflip maneuver. The light blue symbols (in the white region) correspond to when the vehicle entered into the drift mode and the orange symbols (in the medium-gray region) correspond to when the vehicle entered into the recovery mode. Since all points are outside of their respective reachable set, it is safe for the vehicle to execute the maneuver.**

to entering into the drift mode and the lower (orange) points correspond to entering into the recovery mode. As the figure illustrates, all the points are outside of their respective unsafe set and therefore the vehicle can safely perform the maneuver without hitting the ground.

Finally, it should be noted that while the results of three trials are presented here, several additional trials were also conducted with varying levels of success. However, the failures of the unsuccessful trials were due solely to factors outside the scope of the reachable set analysis. For example, some trials failed because of human error, in the form of bugs in the code running on the vehicle. Others were unsuccessful due to hardware malfunctions (e.g. a broken sonic ranger, or saturation of the IMU's gyroscopes). These sorts of failures are typical when making the transition from theory to "real" engineering. However, they also serve a useful purpose underscoring the importance of understanding the limitations that arise when applying guarantees generated by provably safe techniques to real-world robots.

6.2 Reachability-Guided UAV Search

The capture-the-flag problem presented and solved in Chapter 3 assumes that both agents have full knowledge of the state of the opposing agent, a condition that does not always hold. In an actual adversarial game, the agents would have to rely on their own sensing and any external sensing assets to locate the other agent. Indeed, as shown in the experimental results section of that chapter, communications errors can result in loss of information even when the agents are broadcasting their positions as part of the game. However, the reachability information computed for the fully observed game can still serve a useful purpose when the opposing agent's position is unknown.

Consider the problem of controlling a UAV to assist a agent by searching for that agent's opponent. The opposing agent may be located anywhere within a large expanse of unobserved space, but not all unobserved positions are equally likely to be occupied. A probabilistic search method may be attempted, but it is difficult to create a distribution of opponent locations without a model of the opponent's likely strategies. Instead, the reachable sets for the fully observed game can be used to guide the search, as the reachability information can be used to determine the winning region for the opponent given the agent's current position. The opponent only poses a danger to the agent if it is found within this set, thus the opponent's winning reachable set can be used to guide the UAV in its search pattern.

The following sections present an approach to using reachability information to guide UAV search in a 1 vs. 1 capture-the-flag game. Related work is discussed and the search task is formulated, with experimental results presented at the end.

6.2.1 Related Work

The problem of searching an area with one or more UAVs has been well studied in the literature. There are, broadly, two major, related approaches to the problem. The first, most common approach is to formulate the problem probabilistically. This approach has its roots in some of the earliest operations research, when models of optimal search were constructed for finding U-boats in convoy protection during

WWII [74]. In such cases, a model is first built of the search agent's sensing, along with a distribution over likely target positions. A search is then performed either using some limited look-ahead planning or via a set of pre-determined strategies, with the target location probabilities updated in a Bayesian manner. This probabilistic search strategy has been successfully applied in a number of scenarios with both stationary and randomly moving targets [75, 76, 77, 46]. However, in a game scenario such strategies are complicated by the absence of a good model for the target's motions, as reliable target distributions are difficult to obtain in such a case.

Another approach to the search problem is to treat the problem deterministically, and track the set of all possible target locations. The searching agent's sensor footprint is treated as deterministically reducing areas where the target may be. A number of strategies have been proposed for such problems, focused around controlling the extent of the possible target locations [78, 79, 80]. These approaches treat the entire target area equally, with the search agent sweeping across the area in an effort to eventually reduce the target area to zero.

The search strategy utilized in this work builds on an intuition of the previous work, namely that, without a target distribution all target locations must be considered, but that an ordering may be placed on target locations using some prior knowledge. In this case, only a certain subset of opponent locations may result in defeat for the agent being assisted. Thus the search can be conducted in such a way as to prioritize those areas.

The search problem will now be formulated, with the appropriate search strategy then presented.

6.2.2 Problem Statement

The objective is to control a UAV to assist a human agent in a game of 1 vs. 1 capture-the-flag with limited visibility. As the focus here is on the UAV, only one phase of the capture-the-flag game will be considered, namely, that of flag return. It is assumed throughout this section that the UAV is cooperating on behalf of the attacker, although the formulations and solution strategies are symmetric and independent of

agent. The problem of computing the joint, optimal actions for the UAV and attacking agent would require a level of complexity that is infeasible, as the approach to the full game would require operating in the joint information spaces of both sides. Instead, the actions of the attacker will be left to the human agent, and the control problem will focus solely on the UAV. To set up the problem, the setup and solution to the full visibility game of capture-the-flag is briefly summarized below, with the modifications required to address limited visibility presented subsequently.

Summary of capture-the-flag with full visibility

Except for the presence of the UAV, the problem formulation is identical to that in Section 3.1. The flag return portion of capture-the-flag can be defined as a two-agent reach-avoid game. $x_a \in \mathbb{R}^2$ is the position of the attacker, and $x_d \in \mathbb{R}^2$ is the position of the defender. The positions of the two agents are limited to a domain $\Omega \subset \mathbb{R}^2$, with dynamics $\dot{x_a} = u$ and $\dot{x_d} = d$, where u and d are the velocity inputs of each agent, constrained by speed limits $v_{a,max}$ and $v_{d,max}$ according to $||u||_2 \leq v_{a,max}$ and $||d||_2 \leq v_{d,max}$. $\mathbf{x} = (x_a, x_d) \in \mathbb{R}^4$ denotes the joint state of both agents, with the joint domain denoted Ω^2.

The goal of the attacker is to reach some safe region $R \subset \Omega$ without being captured by the defender, defined as the defender coming within distance r_c of the attacker. This corresponds to a capture region $C = \{(\mathbf{x} \mid ||x_a - x_d||_2 \leq r_c\}$. The defender is constrained by the rules of the game to stay outside of R.

The solution to the game is found via the reach-avoid operator $\mathcal{RA}_T(\mathcal{T}, K)$, which gives the set of all states that can be guaranteed to be driven into set $\mathcal{T}$ by the attacker in at most time T while avoiding the undesired set K. The attacker's winning conditions are collected into a set of states R_J, representing either $x_a \in R$, $x_d \notin \Omega$, or $x_d \in R$. The defender's winning conditions are collected into a set of states C_J, representing either $\mathbf{x} \in C$ or $x_a \notin \Omega$. The attacker's winning states are thus $W_A = \mathcal{RA}_T(R_J, C_J)$, and the defender's winning states are $W_D = \Omega^2 \setminus W_A$. These solutions are described in greater detail in Section 3.1.

For a given attacker position x_a, the winning defender positions relative to x_a,

$W_D(x_a)$, can be found by projecting the 4-dimensional value function into 2 dimensions while holding x_a fixed, and similarly for the defender $W_A(x_d)$ can also be computed. If the positions of both agents are known, control strategies can be found via numerical differentiation of the value functions as described in Chapter 3.

Additions for limited visibility

The set of positions visible to the attacker at a time t are limited by line-of-sight, and is denoted $V_A(x_a, t)$. Likewise, the set of positions visible to the defender is denoted $V_D(x_d, t)$. For notational simplicity, the dependency of the visibility regions on agent position and time will be assumed and not explicitly written. Let the position of the UAV be defined as $x_q \in \mathbb{R}^2$, with dynamics $\ddot{x_q} = a$. The visibility region V_Q of the UAV is defined as a circle centered around x_q with radius r_q. The joint visibility region V_J of the attacker and the UAV is then $V_J = V_A \cup V_Q$.

Now, let P_D represent the set of possible defender positions, from the perspective of the attacker and the UAV. The attacker and UAV will be referred to jointly as the attacking side. At any time, the boundary of P_D grows no faster than $v_{d,max}$, representing how fast the defender can move, and any intersection with V_J is removed from P_D. The problem for the UAV is to search the area encompassed by P_D and locate the defender before the defender can intercept the attacker or successfully block the attacker from reaching its goal. To accomplish this, the reachable sets generated in the reachability solution to 1 vs. 1 capture-the-flag will be used to prioritize the search areas.

6.2.3 Search Strategy

The search strategy is now presented. Given an initial set P_D^0 where the defender may be, the set of all possible defender locations satisfies the following properties:

1. The boundary of P_D grows into unobserved space at a rate of $v_{d,max}$
2. The intersection $P_D \cap V_J$ is always empty

Note that unobserved in this case means unobserved by the attacking side. This set P_D may be computed and tracked using a level set representation, as presented in [80]. The visibility regions for the agents may be computed using the level set representation discussed in [81].

The search target point for the UAV is selected according to a hierarchy. The most dangerous positions for the attacker are those in the set $Z = W_D \cap V_A$, which are points where the defender has a winning input, *and* can see the attacker. From these points, the defender may follow the reachability-derived optimal input, and has a chance of successfully preventing the attacker from reaching its goal. The second set of points are points in W_D that are outside V_A. Let $\tilde{Z} = W_D \setminus V_J$ be the set of invisible points in W_D. From these points, the defender may have a winning input, but the attacker is not in sight, and thus the defender must guess at the correct input.

The priority of the UAV is to first search the set of defender positions that may reach Z without being seen. If there are no such points remaining, then the UAV should next search all positions from which the defender may reach $\tilde{Z}$. If there are no points that may reach either Z or $\tilde{Z}$, then there are no possible defender positions that may reach W_D. In this case, the UAV may search the remaining space according to distance from the UAV and the evader.

Computing the search target

The desired search target can now be found using the level set operations discussed above. Using the set of possible defender locations P_D as the initial condition, a set $\mathcal{S}_D$ can be computed on the domain $\Omega \setminus V_J$ of all points that the defender may reach before the UAV, without becoming visible to the attacker. $\mathcal{S}_D$ can be computed with the modified FMM algorithm presented in Section 4.1.3.

Recall that $Z = V_A \cap W_D$ is the set of points where the defender is in a winning configuration and both agents are visible to each other. The time-to-reach function $\psi_Z(x)$ can be computed on the domain $\Omega \setminus V_J$ to find the minimum distance of a point x to the set Z without becoming visible to the attacking side. Let $\mathcal{RA}(Z, V_J) = \{x \mid \psi_Z(x) < \infty\}$ be the set of all possible defender initial positions that can reach Z in finite time without becoming visible. If $\mathcal{RA}(Z, V_J) \cup \mathcal{S}_D \neq \emptyset$, the target x^* can be

selected as

$$x^* = \underset{x \in \mathcal{S}_D \cup \mathcal{RA}(Z, V_J)}{\arg\min} w_1(x)$$

where the cost $w_1(x)$ is defined as

$$w_1(x) = w_Z \psi_Z(x) + w_\delta \psi_\delta(x) + w_q \psi_Q(x) \tag{6.7}$$

with parameters w_Z, w_δ, w_Q being weightings on the distance to Z, to the attacker, and to the UAV, respectively.

If there are no points that can reach Z without becoming visible, then the search proceeds through all points that may reach invisible points in W_D but cannot reach visible parts of W_D without being seen. This is the set $\tilde{Z} = W_D \setminus V_J$, and the distance level set $\psi_{\tilde{Z}}(x)$ and set $\mathcal{RA}(\tilde{Z}, V_J)$ of defender positions that can reach $\tilde{Z}$ without being seen can be computed accordingly, with the target chosen as

$$x^* = \underset{x \in \mathcal{S}_D \cup \mathcal{RA}(\tilde{Z}, V_J)}{\arg\min} w_2(x)$$

where the cost $w_2(x)$ is defined as

$$w_2(x) = w_{\tilde{Z}} \psi_Z(x) + w_\delta \psi_\delta(x) + w_q \psi_Q(x) \tag{6.8}$$

with a new cost $w_{\tilde{Z}}$ on the invisible points.

Finally, if no points in P_D are in danger of reaching any part of W_D, then the search proceeds based on distance, with

$$x^* = \underset{x \in \mathcal{S}_D}{\arg\min}\, w_3(x)$$

where the cost $w_3(x)$ is defined as

$$w_3(x) = w_\delta \psi_\delta(x) + w_q \psi_Q(x). \tag{6.9}$$

6.2.4 Experimental Results

A number of experiments were carried out to evaluate the reachability-guided search algorithm using the full BEARCAT platform with UAVs. The experiments were carried out on an area of the Berkeley campus measuring approximately 250 m x 200 m, with several large buildings serving as obstacles. Each agent was equipped with an HTC Incredible smartphone reporting the agent locations and displaying reachable set information. The UAV used was an Ascending Technologies Pelican. Sensing was simulated, so that whenever the defender entered V_A both the defender and attacker received information about each other's locations, and when the defender entered V_Q the attacker was alerted to the defender position.

For these experiments, the weights used were as follows: w_q was set to 100 and all other weights set to 1. This creates a behavior in which the UAV greedily sweeps along P_D according to the best nearby points, thus reducing issues such as switching or leaving behind pockets of possible defender presence that then grow when the UAV departs.

The course of one of these experiments is illustrated in Figures 6.9a-f. The initial condition for the game is shown in Figure 6.9a, with the attacker and UAV beginning the game in the lower center of the game region and the defender on the opposite side of the large central building. The attacker and UAV visual regions V_A and V_Q are illustrated in teal and white, respectively. The defender winning region W_D is the large red region centered on the attacker, and the attacker winning region W_A is the blue region centered on the defender. The set of possible defender positions P_D, from the perspective of the attacker, is the expanding pink circle centered on the defender.

Figure 6.9b shows the game 60 seconds after beginning. The attacker has moved up to the corner of the building in an attempt to maximize visibility of the defender while still staying relatively hidden. P_D has expanded to fill up the unobserved space, and the UAV has begun its search. Note that the attacker has moved in such a way that no part of P_D can expand into the set $W_D \cap V_A$ without first passing through V_A, thus the UAV was prioritizing the area of W_D that is currently unobserved. The defender has pulled back away from the building to reduce the size of W_A, hoping to minimize the chance that the attacker will pass into W_A without being detected.

At the moment, neither agent had knowledge of the other agent's position, and both were playing cautiously and unable to commit to moving toward either side of the central building.

The UAV has just detected the defender in Figure 6.9c. As the UAV was programmed to track the defender once detection occurs, the defending agent was also aware of being found. Thus the defender began moving unpredictably in a patrolling motion, shifting to the eastern (right) side of the game domain, as seen in Figure 6.9d. The attacker had been moving eastward (toward the right) and was temporarily blocked by the motion of the defender, but after some time the defender was forced to patrol back to the west (left) since the position of the attacker was unknown, as shown in Figure 6.9e. With the UAV reporting back the position of the defender, the attacker was able to commit to moving up toward the target set from the eastern (right) side of the domain, and eventually reached the target set at the upper portion of the game domain, as shown in Figure 6.9f.

This example is prototypical of the experiments that were conducted. By using the reachability information to guide the search, the UAV was able to quickly move to the most critical area and locate the defender, giving the attacker the information advantage needed to successfully win the game.

6.3 Discussion

The work presented in this chapter demonstrates how reachability analysis may be extended beyond direct solutions to the adversarial games presented in previous chapters. Reachability analysis is a powerful tool for analysis, design, and verification of complex control problems. Using these tools, provable guarantees on safety and performance can be made for complex platforms such as the STARMAC quadrotor helicopter. In the aerobatic maneuver work, reachable set analysis allowed the design of a sequence of modes that could be guaranteed to safely transition from one to the next, arriving at a desired final state while avoiding an undesired region of the state space. The flight experiments on the STARMAC platform demonstrated this provably safe backflip maneuver and showed the validity of using reachability for

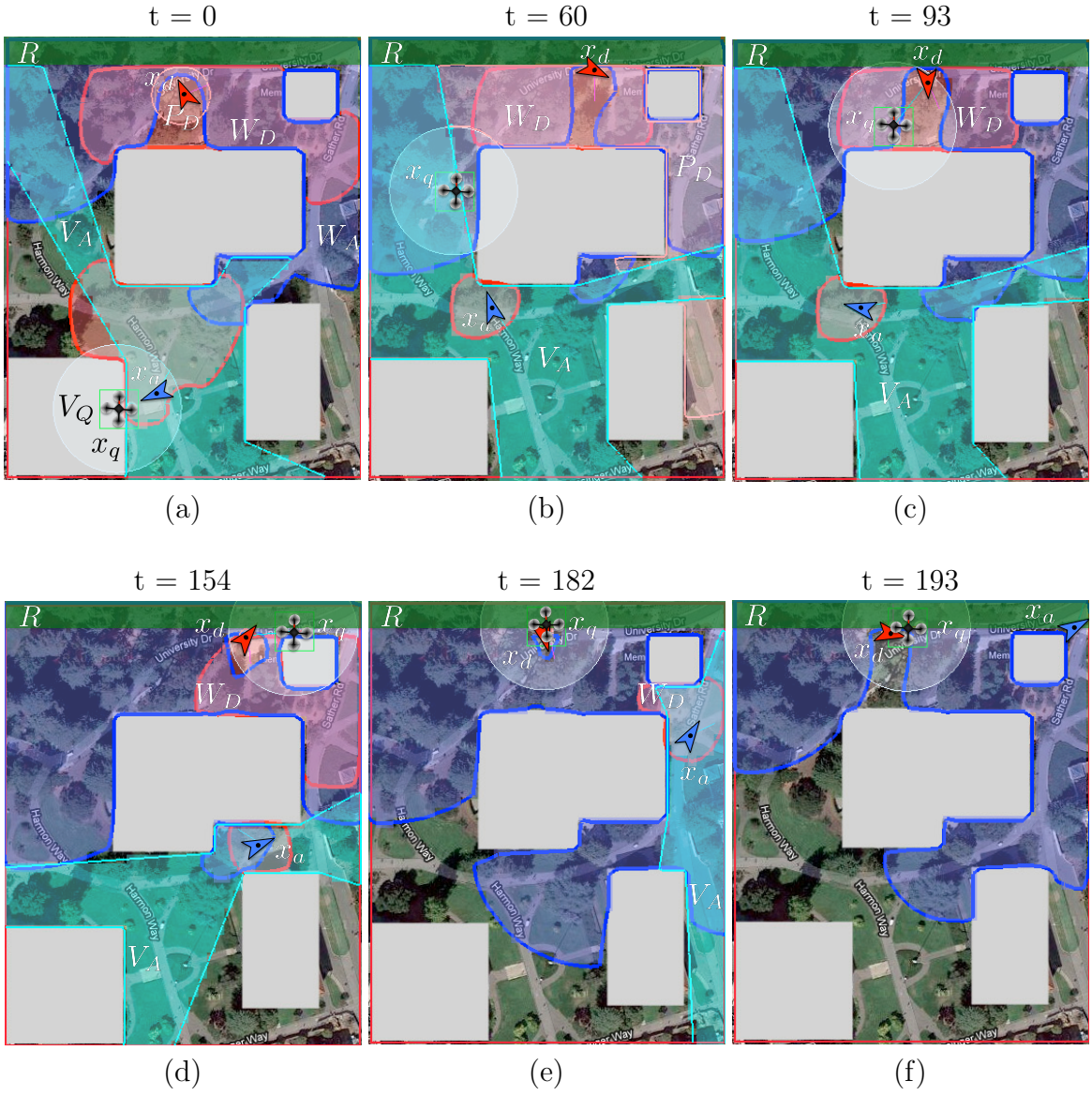


Figure 6.9: **Sequence of screenshots from the ground station laptop showing a full search game with a quadrotor UAV (map images courtesy of `maps.google.com`).**

maneuver design. Provable guarantees are valuable in the design of any system, and reachability holds great promise for further development of such guarantees for many complex systems.

In a different vein, the reachable sets computed for the capture-the-flag game are used to direct a UAV searching for a target during a more complex, limited-information version of the game. By utilizing the reachability results, the UAV is able to concentrate its search on areas that are more relevant for the agent it is supporting. The reachable sets allow the game information to be considered and included in the search formulation. Although the original reachability formulation does not explicitly include partial observability, the reachable sets nonetheless play a valuable role in simplifying the complexity of the search problem.

Chapter 7

Conclusions and Future Research Directions

This thesis addresses the development of human-friendly algorithms for control in multi-agent adversarial games. Using reachability tools, solution strategies are developed for several pursuit-evasion and reach-avoid games. In addition to the control inputs computed by the strategies, the reachable sets produced by the algorithms provide intuitive visual tools for human agents, allowing them to incorporate the strategies into their own decision-making. This chapter briefly reviews the contributions of the work presented in this thesis, and discusses directions for future work.

7.1 Summary

The contributions of this work are in a combination of theoretical developments in solving multi-agent adversarial games, and in the development of practical tools using these solutions. This research presents novel formulations and solutions for 1 vs. 1 and 1 vs. many reach-avoid games, and for a many vs. 1 pursuit-evasion game. In addition, this adversarial reachability approach is also used in UAV control, demonstrating the flexibility of these methods. These algorithmic contributions advance the state of the art in analysis of multi-agent games, but the demonstration of reachability as a tool for human agents may have the most long term practical impact.

By formulating the 1 vs. 1 capture-the-flag game as a reach-avoid game, HJI reachability can be used to compute an optimal, guaranteed solution strategy for both agents. In addition, the solution method addresses general, planar game configurations with arbitrary obstacles.

Reachability information is also used to generate solutions for a single attacking agent in a reach-avoid game against multiple defending agents. In this case, the open-loop formulation of the game is used, allowing trajectories to be generated in real time for certain initial conditions that are guaranteed to arrive at the target. By using the open-loop formulation, the problem can be solved using the HJB equation in the 2-dimensional game domain rather than in the high dimensional joint state space of all agents. This methodology is also capable of handling arbitrary planar game configurations.

The case of multiple agents coordinating against a single agent is addressed in a multi-agent cooperative pursuit problem. By jointly minimizing the evader's safe-reachable set, the pursuit strategy can be computed in real time in a decentralized fashion while still maintaining cooperation between the pursuers. In addition, a provable guarantee of capture is found for convex domains for games with equal pursuer and evader speed.

Finally, the reachability formulation is demonstrated in two UAV control scenarios, where the HJI reachability formulation is used to generate guaranteed safe maneuver sequences, and the solutions to capture-the-flag are used to guide UAV search. These results demonstrate the flexibility of the reachability formulations presented in this thesis.

In addition to the control inputs generated by these algorithms, the contextual information produced by these methods is in many ways the most useful product. The utility of these reachable sets for human agents is demonstrated through the experiments presented in this thesis. In the capture-the-flag experiments, the reachability information was useful even when the assumptions of perfect sensing and communications failed. In the safe-reachable set capture results, GPS error often resulted in slightly incorrect heading commands being displayed for the human agents. Yet these errors did not impact the ultimate results of the games, as the agents were able to

use visualizations of the safe-reachable area to correctly ascertain their areas of responsibility and correct directions of movement. In all of these cases, the automation positively supplemented human judgment, providing useful assistance when possible and degrading gracefully in the face of sensor and communications failure.

7.2 Future Research Directions

This research can be extended in several ways. First, theoretical extensions can be made to the solution of the game solution algorithms. The reach-avoid and pursuit-evasion games are similar in many ways, and in particular it would be useful if reach-avoid elements could be better incorporated into pursuit-evasion games with multiple pursuers and evaders and vice versa. For example, it is known that for any bounded planar environment and agents with equal speeds, 3 pursuers suffice to capture an evader [3, 82]. It is not yet known how to calculate efficient strategies for pursuers in such a situation, although the strategies proposed require successively partitioning the game domain and trapping the evader in smaller and smaller regions. Reach-avoid strategies may be employed to greatly decrease the capture time for such scenarios.

Another interesting direction is the development of more effective user interfaces and augmented reality tools for human agents in these games. Due to time constraints, the experiments conducted in this thesis were primarily demonstrations of feasibility rather than through explorations of the effectiveness of the tools presented. A considerable amount of research and development is still required to both improve the display of reachability information, for example via projection glasses or heads-up displays, and to properly evaluate the utility of the tools from a human factors standpoint.

The development of advanced automation tools for real time guidance of human agents is a burgeoning field, and reachability-based tools are ideal for this purpose. The tools and experimental results presented in this thesis show examples of their application to multi-agent adversarial games, but as the UAV control results demonstrate they may be applied in many other areas as well.

Bibliography

[1] C. Farivar, "Google gets license to test drive autonomous cars on Nevada roads," *Ars Technica*, May 2012, http://arstechnica.com/tech-policy/2012/05/google-gets-license-to-test-drive-autonomous-cars-on-nevada-roads/.

[2] N. Wingfield and S. Sengupta, "Drones set sights on U.S. skies," *New York Times Online*, February 2012, http://www.nytimes.com/2012/02/18/technology/drones-with-an-eye-on-the-public-cleared-to-fly.html.

[3] L. Alonso, A. Goldstein, and E. Reingold, "Lion and man: upper and lower bounds," *ORSA Journal on Computing*, vol. 4, no. 4, pp. 447–452, 1992.

[4] S. Kopparty and C. Ravishankar, "A framework for pursuit evasion games in R^n," *Information Processing Letters*, vol. 96, no. 3, pp. 114–122, 2005.

[5] S. Alexander, R. Bishop, and R. Ghrist, "Pursuit and evasion in non-convex domains of arbitrary dimensions," in *Proceedings of the Robotics: Science and Systems Conference*, Philadelphia, Pennsylvania, August 2006.

[6] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *International Journal of Robotics Research*, vol. 17, no. 7, p. 760, 1998.

[7] T. Fraichard and H. Asama, "Inevitable collision states - a step towards safer robots?" *Advanced Robotics*, vol. 18, no. 10, pp. 1001–1024, 2004.

[8] J. Van Den Berg and M. Overmars, "Planning time-minimal safe paths amidst unpredictably moving obstacles," *International Journal of Robotics Research*, vol. 27, no. 11-12, p. 1274, 2008.

[9] R. Isaacs, *Differential Games*. New York: Wiley, 1967.

[10] L. C. Evans and P. E. Souganidis, "Differential games and representation formulas for solutions of Hamilton-Jacobi-Isaacs equations," *Indiana University Mathematics Journal*, vol. 33, no. 5, pp. 773–797, 1984.

[11] T. Başar and G. Olsder, *Dynamic Noncooperative Game Theory*, 2nd ed. Philadelphia, PA: SIAM, 1999.

[12] I. Mitchell, A. Bayen, and C. Tomlin, "A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games," *IEEE Transactions on Automatic Control*, vol. 50, no. 7, pp. 947–957, 2005.

[13] M. Falcone and R. Ferretti, "Semi-Lagrangian schemes for Hamilton-Jacobi equations, discrete representation formulae and Godunov methods," *Journal of Computational Physics*, vol. 175, no. 2, pp. 559–575, 2002.

[14] J. Ding, J. Sprinkle, S. S. Sastry, and C. J. Tomlin, "Reachability calculations for automated aerial refueling," in *Proceedings of the IEEE International Conference on Decision and Control*, Cancun, Mexico, 2008, pp. 3706–3712.

[15] T. Parsons, "Pursuit-evasion in a graph," *Theory and Applications of Graphs*, pp. 426–441, 1978.

[16] M. Aigner and M. Fromme, "A game of cops and robbers," *Discrete Applied Mathematics*, vol. 8, no. 1, pp. 1–12, 1984.

[17] L. Guibas, J.-C. Latombe, S. Lavalle, D. Lin, and R. Motwani, "Visibility-based pursuit-evasion in a polygonal environment," in *Algorithms and Data Structures*, ser. Lecture Notes in Computer Science, F. Dehne, A. Rau-Chaplin, J.-R. Sack, and R. Tamassia, Eds. Springer Berlin / Heidelberg, 1997, vol. 1272, pp. 17–30.

[18] S. LaValle and J. Hinrichsen, "Visibility-based pursuit-evasion: the case of curved environments," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 2, pp. 196 –202, April 2001.

[19] B. P. Gerkey, S. Thrun, and G. Gordon, "Visibility-based pursuit-evasion with limited field of view," *International Journal of Robotics Research*, vol. 25, no. 4, pp. 299–315, 2006.

[20] J. Sprinkle, J. Eklund, H. Kim, and S. Sastry, "Encoding aerial pursuit/evasion games with fixed wing aircraft into a nonlinear model predictive tracking controller," in *Proceedings of the IEEE International Conference on Decision and Control*, Atlantics, Bahamas, December 2004.

[21] J. McGrew, J. How, L. Bush, B. Williams, and N. Roy, "Air combat strategy using approximate dynamic programming," *Proceedings of the AIAA Conference on Guidance, Navigation, and Control*, Aug 2008.

[22] M. Earl and R. D'Andrea, "A decomposition approach to multi-vehicle cooperative control," *Robotics and Autonomous Systems*, vol. 55, no. 4, pp. 276–291, 2007.

[23] G. Chasparis and J. Shamma, "Linear-programming-based multi-vehicle path planning with adversaries," *Proceedings of the IEEE American Control Conference*, pp. 1072–1077, 2005.

[24] R. L. Shaw, *Fighter Combat - Tactics and Maneuvering*. Annapolis, Maryland: Naval Institute Press, 1985.

[25] A. Coates, P. Abbeel, and A. Y. Ng, "Apprenticeship learning for helicopter control," *Commun. ACM*, vol. 52, no. 7, pp. 97–105, 2009.

[26] T. Seamster, R. Redding, J. Cannon, J. Ryder, and J. Purcell, "Cognitive task analysis of expertise in air traffic control," *International Journal of Aviation Psychology*, no. 3, 1993.

[27] J. Kahah, D. Worley, and C. Stasz, "Understanding commanders' information needs," Rand Corporation, Tech. Rep. R-3761-1-A, 2000.

[28] J. Hawkins and S. Blakeslee, *On Intelligence*. Times Books, 2004.

[29] D. Felleman and D. Essen, "Distributed hierarchical processing in the primate cerebral cortex," *Cerebral Cortex*, no. 1, 1991.

[30] M. Endsley, "Situation awareness, automation, and free flight," in *FAA/Eurocontrol Air Traffic Management R&D Seminar*, Saclay, France, June 1997.

[31] M. Oishi, I. M. Mitchell, A. M. Bayen, and C. J. Tomlin, "Invariance-preserving abstractions of hybrid systems: Application to user interface design," *IEEE Transactions on Control Systems Technology*, vol. 16, no. 2, pp. 229–244, 2008.

[32] H. Huang, J. Ding, W. Zhang, and C. Tomlin, "A differential game approach to planning in adversarial scenarios: a case study on capture-the-flag," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Shanghai, China, 2011.

[33] R. Takei, H. Huang, J. Ding, and C. Tomlin, "Time-optimal multi-stage motion planning with guaranteed collision avoidance via an open-loop game formulation," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Minneapolis, Minnesota, 2012.

[34] Z. Zhou, R. Takei, H. Huang, and C. Tomlin, "A general, open-loop formulation for reach-avoid games," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Maui, Hawaii, 2012.

[35] H. Huang, W. Zhang, J. Ding, D. Stipanović, and C. Tomlin, "Guaranteed decentralized pursuit-evasion in the plane with multiple pursuers," in *Proceedings of the IEEE International Conference on Decision and Control*, Orlando, Florida, 2011.

[36] G. M. Hoffmann, H. Huang, S. L. Waslander, and C. J. Tomlin, "Precision flight control for a multi-vehicle quadrotor helicopter testbed," *Control Engineering Practice*, Sept 2011.

[37] J. Gillula, G. M. Hoffmann, H. Huang, M. P. Vitus, and C. J. Tomlin, "Applications of hybrid reachability analysis to robotic aerial vehicles," *International Journal of Robotics Research*, Jan 2011.

[38] HTC, "HTC Droid Incredible," 2012, http://www.htc.com/us/products/droid-incredible-verizon.

[39] Google, "Android Operating System," 2012, http://www.android.com.

[40] Pantech, "UML290 4G USB Modem," 2012, http://www.pantechusa.com/phones/uml290_4g_usb_modem.

[41] Videre Design, "STH-MDCS 2 Stereo Vision Head," 2008, http://www.videredesign.com/sthmdcs2.htm.

[42] Hokuyo, "URG-04LX Laser Range Finder," 2008, http://www.hokuyo-aut.jp/products/urg/urg.htm.

[43] BackCountry Access, "Tracker DTS Digital Avalanche Beacon," 2008, http://www.bcaccess.com/bca_products/tracker/index.php.

[44] S. L. Waslander, G. Inalhan, and C. J. Tomlin, "Decentralized optimization via nash bargaining," in *Theory and Algorithms for Cooperative Systems*, D. Grundel, R. Murphey, and P. M. Pardalos, Eds. World Scientific Publishing Co., 2004, vol. 4, pp. 565–585.

[45] G. M. Hoffmann, S. L. Waslander, and C. J. Tomlin, "Mutual information methods with particle filters for mobile sensor network control," in *Proceedings of the IEEE International Conference on Decision and Control*, San Diego, CA, December 2006, pp. 1019–1024.

[46] G. M. Hoffmann and C. J. Tomlin, "Mobile sensor network control using mutual information methods and particle filters," *IEEE Transactions on Automatic Control*, vol. 55, no. 1, pp. 32–47, 2010.

[47] M. Vitus, V. Pradeep, G. M. Hoffmann, S. L. Waslander, and C. J. Tomlin, "Tunnel-MILP: Path planning with sequential convex polytopes," in *Proceedings of the AIAA Conference on Guidance, Navigation and Control*, Honolulu, Hawaii, USA, August 2008.

[48] J. Gillula, H. Huang, M. P. Vitus, and C. J. Tomlin, "Design of guaranteed safe maneuvers using reachable sets: Autonomous quadrotor aerobatics in theory and practice," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Anchorage, Alaska, May 2010.

[49] Ascending Technologies, "Pelican," 2012, http://www.asctec.de/ asctec-pelican-3/.

[50] I. Mitchell, "Application of level set methods to control and reachability problems in continuous and hybrid systems," Ph.D. dissertation, Stanford University, 2002.

[51] M. G. Crandall and P.-L. Lions, "Viscosity solutions of Hamilton-Jacobi equations," *Transactions of the American Mathematical Society*, vol. 277, no. 1, pp. 1–42, 1983.

[52] I. Mitchell, *A Toolbox of Level Set Methods*, 2009, http://people.cs.ubc.ca/ mitchell/ToolboxLS/index.html.

[53] J. Sprinkle, A. D. Ames, J. M. Eklund, I. M. Mitchell, and S. Shankar Sastry, "Online safety calculations for glide-slope recapture," *Innovations in Systems and Software Engineering*, vol. 1, pp. 157–175, 2005.

[54] M. Bardi and I. Capuzzo-Dolcetta, *Optimal control and viscosity solutions of Hamilton-Jacobi-Bellman equations*, ser. Systems & Control: Foundations & Applications. Boston, MA: Birkhäuser, 1997, with appendices by Maurizio Falcone and Pierpaolo Soravia.

[55] M. Bardi, "Some applications of viscosity solutions to optimal control and differential games," in *Viscosity solutions and applications*, ser. Lecture Notes in Math. Berlin, Germany: Springer, 1997, vol. 1660, pp. 44–97.

[56] J. A. Sethian, "Fast marching methods," *SIAM Review*, vol. 41, no. 2, pp. 199–235, 1999.

[57] ——, *Level set methods and fast marching methods*, 2nd ed., ser. Cambridge Monographs on Applied and Computational Mathematics. Cambridge, UK: Cambridge University Press, 1999, vol. 3.

[58] M. Falcone, "Fast marching methods for front propagation," November 2007, lecture notes at "Introduction to Numerical Methods for Moving Boundaries".

[59] E. Rouy and A. Tourin, "A viscosity solutions approach to shape-from-shading," *SIAM Journal of Numerical Analysis*, vol. 29, no. 3, pp. 867–884, 1992.

[60] J. A. Sethian and A. Vladimirsky, "Ordered upwind methods for static Hamilton-Jacobi equations: theory and algorithms," *SIAM Journal of Numerical Analysis*, vol. 41, no. 1, pp. 325–363, 2003.

[61] R. Takei, R. Tsai, H. Shen, and Y. Landa, "A practical path-planning algorithm for a vehicle with a constrained turning radius: a Hamilton-Jacobi approach," in *Proceedings of the IEEE American Control Conference*, July 2010.

[62] W. Cheung, "Constrained pursuit-evasion problems in the plane," Master's thesis, Faculty of Graduate Studies, Computer Science, University of British Columbia, 2005.

[63] V. Gavrilets, I. Martinos, B. Mettler, and E. Feron, "Flight test and simulation results for an autonomous aerobatic helicopter," in *Proceedings of the Digital Avionics Systems Conference*, Irvine, California, October 2002.

[64] P. Abbeel, A. Coates, M. Quigley, and A. Y. Ng, "An application of reinforcement learning to aerobatic helicopter flight," in *Advances in Neural Information Processing Systems 19*. MIT Press, 2007, pp. 1–8.

[65] S. Lupashin, A. Schollig, M. Sherback, and R. D'Andrea, "A simple learning strategy for high-speed quadrocopter multi-flips," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Anchorage, AK, May 2010.

[66] O. Purwin and R. DAndrea, "Performing aggressive maneuvers using iterative learning control," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Kobe, Japan, May 2009.

[67] E. Frazzoli, M. A. Dahleh, and E. Feron, "Maneuver-based motion planning for nonlinear systems with symmetries," *IEEE Transactions on Robotics*, vol. 21, no. 6, pp. 1077–1091, 2005.

[68] R. Burridge, A. Rizzi, and D. Koditschek, "Sequential composition of dynamically dexterous robot behaviors," *International Journal of Robotics Research*, vol. 18, no. 6, pp. 534–555, 1999.

[69] T. Lozano-Perez, M. T. Mason, and R. H. Taylor, "Automatic synthesis of fine-motion strategies for robots," *International Journal of Robotics Research*, vol. 3, no. 1, pp. 3–24, 1984.

[70] M. Zefran and J. W. Burdick, "Stabilization of systems with changing dynamics," in *Proceedings of the International Conference on Hybrid Systems: Computation and Control*, ser. Lecture Notes in Computer Science, vol. 1386. Berkeley, CA: Springer-Verlag, 1998, pp. 400–415.

[71] M. Lazar and A. Jokic, "Synthesis of trajectory-dependent control lyapunov functions by a single linear program," in *Proceedings of the International Conference on Hybrid Systems: Computation and Control*, ser. Lecture Notes in Computer Science, vol. 5469. San Francisco, CA: Springer-Verlag, 2009, pp. 237–251.

[72] M. Egerstedt, T. J. Koo, F. Hoffmann, and S. Sastry, "Path planning and flight controller scheduling for an autonomous helicopter," in *Proceedings of the International Conference on Hybrid Systems: Computation and Control*, ser. Lecture

Notes in Computer Science, vol. 1569. Berg en Dal, The Netherlands: Springer-Verlag, 1999, pp. 91–102.

[73] I. M. Mitchell, "The flexible, extensible and efficient toolbox of level set methods," *Journal of Scientific Computing*, vol. 35, no. 2-3, Jun. 2008.

[74] B. Koopman, "Search and its optimization," *The American Mathematical Monthly*, vol. 86, no. 7, pp. 527–540, 1979. [Online]. Available: http://www.jstor.org/stable/10.2307/2320580

[75] S. Benkoski, M. Monticino, and J. Weisinger, "A survey of the search theory literature," *Naval Research Logistics (NRL)*, vol. 38, no. 4, pp. 469–494, 1991.

[76] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 2, pp. 243–255, 2004.

[77] F. Bourgault, T. Furukawa, and H. Durrant-Whyte, "Optimal search for a lost target in a bayesian world," *Field and Service Robotics*, vol. 24, pp. 209–222, 2006.

[78] T. McGee and J. Hedrick, "Guaranteed strategies to search for mobile evaders in the plane," in *Proceedings of the IEEE American Control Conference*, Minneapolis, Minnesota, June 2006.

[79] D. Kingston, R. Beard, and R. Holt, "Decentralized perimeter surveillance using a team of uavs," *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1394–1404, 2008.

[80] E. Frew, "Combining area patrol, perimeter surveillance, and target tracking using ordered upwind methods," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Kobe, Japan, May 2009.

[81] Y. Tsai, L. Cheng, S. Osher, P. Burchard, and G. Sapiro, "Visibility and its dynamics in a PDE based implicit framework," *Journal of Computational Physics*, vol. 199, no. 1, pp. 260–290, 2004.

[82] Z. Zhou, J. Shewchuk, H. Huang, and C. Tomlin, "On 3-pursuer guaranteed capture in general planar domains," in *Proceedings of the IEEE International Conference on Decision and Control, submitted*, Maui, Hawaii, 2012.