A State Space Partitioning Scheme for Vehicle Control in Pursuit-Evasion Scenarios

Brian J. Goode

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Mechanical Engineering

Michael J. Roan, Chairman
Andrew J. Kurdila
Alexander Leonessa
Cory Papenfuss
Daniel J. Stilwell

October 21, 2011
Blacksburg, Virginia

Keywords: differential games, Bellman optimality, pursuit-evasion, path planning, vehicle control
Copyright 2011, Brian J. Goode

A State Space Partitioning Scheme for Vehicle Control in Pursuit-Evasion Scenarios

Brian J. Goode

ABSTRACT

Pursuit-evasion games are the subject of a variety of research initiatives seeking to provide some level of autonomy to mobile, robotic vehicles with on-board controllers. Applications of these controllers include defense topics such as unmanned aerial vehicle (UAV) and unmanned underwater vehicle (UUV) navigation for threat surveillance, assessment, or engagement. Controllers implementing pursuit-evasion algorithms are also used for improving everyday tasks such as driving in traffic when used for collision avoidance maneuvers.

Currently, pursuit-evasion tactics are incorporated into the control by solving the Hamilton-Jacobi-Isaacs (HJI) equation explicitly, simplifying the solution using approximate dynamic programming, or using a purely finite-horizon approach. Unfortunately, these methods are either subject to difficulties of long computational times or having no guarantees of succeeding in the pursuit-evasion game. This leads to more difficulties of implementing these tactics on-line in a real robotic scenario where the opposing agent may not be known before the maneuver is required.

This dissertation presents a novel method of solving the HJI equation by partitioning the state space into regions of local, finite horizon control laws. As a result, the HJI equation can be reduced to solving the Hamilton-Jacobi-Bellman equation recursively as information is received about an opposing agent. Adding complexity to the problem structure results in a decreased calculation time to allow pursuit-evasion tactics to be calculated on-board an agent during a scenario. The algorithms and implementation methods are given explicitly and illustrated with an example of two robotic vehicles in a collision avoidance maneuver.

This work was funded by ONR Contract No. N00014-06-1-0356.

To fried chicken...

Acknowledgments

To fully acknowledge all those that have contributed to my work and knowledge over the course of writing this dissertation would be a dissertation in and of itself. Here is a synopsis.

First, I would like to thank my parents, Dale and Patricia Goode, and my brother Stephen Goode. I would also like to thank my grandparents, Richard and Agnes Neach and Rita Goode for all of their support as well.

Many thanks are necessary for those directly involved with the work in this dissertation. I would like to thank Dr. Michael Roan, my advisor, for his tutelage in navigating graduate school and editing papers with me over the last three years. I would also like to thank Dr. Andrew Kurdila who was instrumental in introducing me to a variety of topics in the controls field and always showed an interest in meeting to discuss ideas. I am very grateful for Dr. Alexander Leonessa and Dr. Daniel Stilwell for serving on my committee. Thanks go to Dr. Cory Papenfuss, for not only serving on my committee, but introducing me to micro-brewing and flying (along with Brandon Dillon). Best wishes for you and Carmen.

I would also like to acknowledge Dr. Marty Johnson. I cannot think of a single educator who parallels in motivating others to learn and accomplish. His impact on my learning while at Virginia Tech is insurmountable. I extend my sincere gratitude to Dr. Mary Kasarda and Dr. Curtis Stern for their professional guidance toward becoming not just a researcher, but an educator.

Thanks to Gail Coe for all that she has done for me and the Vibration and Acoustics lab over the years.

I would also like to thank my research and design students that have contributed to this research: Phil Kwon, Tanner Willis, Doug Long, Stephen Bernero, Nicole Fink, Matt Bradcovich, Adam Smith, Kyle Damiano, Dale Arbogast, and Bryan Glock. I wish you the best as you continue your education and career.

During the summers, I had the fortune of working in Newport, RI at the Naval Undersea Warfare Center (NUWC) thanks to the ULI program run my Maria Medeiros. I would like to thank Lynn Potter, John Welch, and Dr. Adam Mirkin for their support during this time. I would also like to thank Camila Francolin, Vinesh Nishawala, Kyle DeMedeiros, and of course Wen for adding life to the cube farm. In particular, I would like to thank Lonnie

Parker for our time at URI exchanging ideas while scavenging for food. I hope the best as you finish your Ph. D, and I hope the best for you and Andrea.

Also, thanks to Dr. Don Leo, Dr. Vishnu Sundaresan, and Dr. David Hopkinson for getting me started in the research business when I first came to Virginia Tech as a freshman.

When taking the time to complete this degree, it was very easy to become too focused and engulfed in the work. This section is for those who helped me hold it down.

To "the rat pack and associated females": Trevor and Megan Anderson, David Buck and Lindsey, John and Saylem DePasquale, and Pat and Joan Ray, thank you for our time together since the very beginning. I hope the best as you and yours go through life. Mike DePasquale, thank you for your generous wisdom over the years. Jimbo Jimbabwei Jenkins, also, thank you for your wise words.

Now for my frequently visited break room, the CIMSS lab. Alex "Freedom Child" Villanueva, a dedicated researcher, exemplary Canadian, thanks for introducing me to your strange and foreign culture. I hope you have a successful and fulfilling career. You deserve it. Colin "C-Bear" Smith, thanks for ultimately introducing me to Jackie! Scott Bressers, the only man I know that can fill the shoes of... Scott Bressers. Thanks for your irreplaceable presence. Nicholas Thayer, in the lab 24-7, and about the most down to earth as they come. Thanks for the great times over the many years. I hope you and Amber have a great life together! Patrick "ze German" Heitzman, thanks for always being there when it was time for shenanigans. Colin Stewart, Dragon, Woon, and the numerous others, best of luck as you continue your studies.

In my own lab, the VAL lab, there are many talented people to whom I look up to every day. Bonsung Koo and Jeong, always provided company in the office late at night. Best wishes for you and your families. Tom-Davy Saux, thanks for being a patient drummer as I learned to play the piano with a band. Thanks to Chris Barnobi, also a night-time regular who gave me "Baby Girl", a piano and a hobby that changed my life. Thanks to Ryan Harne for the many conversations over the years. Ryan Haac, an admirable person, traveler, and inspiration. Thanks for showing how its done. Also, I would like to thank Chris, Gold and Max. I wish you all the best.

Phil Chin, your persistence and dedication is unmatched by anyone I know. I hope the best for you and your clan: Molly, Joshua, and Bryan. Thank you for the piano lessons, talks, and road trips to Memphis and Orlando. Thank you for the honest outlook on life.

Then there's Ryan Colby. My brother from another mother. Thank you for the wings, grilling, dawg, and time well spent. Same goes for Aaron Siddens. Thanks for introducing me to St. Louis, fried ravioli, and for the numerous porch conversations. For both Ryan and Aaron, thanks for providing a shower and couch during the semester when I was homeless. Aaron, along with Bireswar and Poulomi Laha, thanks for the company and not being disturbed by the piano. Eric Williams, thanks for the always positive encouragement. I wish you success as a professor and I wish the best for you and Andrea. I would also like to thank

the veteran Roanoke St. house tenants that have yet to be mentioned: Tyler, Lera, Carlos (and Kristen), Reza, and Mehdi for allowing me to have fried chicken nights at your house.

And now, I would like to thank my girlfriend, Jacqueline Speier. Besides being smart and ultra-fine, you are the sweetest most caring person I know. Thank you for all that you have done for me, and helping us grow closer together. I look forward to our future, full of life and adventure. I love you! YAIM.

Lastly, I would also like to thank Mike and his legendary meatball sub from Sycamore Deli, the baristas at Mill Mountain (where this was written), Patty from Abby's and the numerous employees at the Cellar. Thanks for providing character, sometimes a distraction, and a place to escape. Your service to the Blacksburg community is very important. Thanks to the Pabst brewing company for making Colt 45.

Although writing a dissertation is cool (once it's done), as I wrap up these final words, I can't help think that my greatest accomplishment is learning all that I have by experiencing life with everyone mentioned above. Thank you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The work appearing in this dissertation outlines and implements an algorithm for controlling a vehicle engaged in pursuit-evasion activities. Applications can be combat-oriented or peaceful. Regardless, the differential game approach for generating control tactics removes the strategy of an opposing agent as a mechanism for failing to capture or escape. Agents with this capability in their controllers bear a considerable advantage. Unfortunately, arriving at this control solution is a daunting task. Partitioning the control objectives over the state space, the main topic of this dissertation, removes a little bit of this burden by providing a fast solution technique. These solutions are used as desired paths that can be tracked using lower-level controls for actual implementation in a real system. This technique is demonstrated explicitly with a collision avoidance scenario using robotic vehicles. The methods and algorithms presented in this dissertation can be expanded in future work to account for more complex pursuit-evasion scenarios that have faster, more intricate dynamics.

1.1 Motivation

Providing guidance, decision making capabilities, and increasing levels of autonomy to vehicles is a problem in control theory that is gaining importance [4, 7, 102, 134]. Prevalent in the transportation industry, everyday technology such as cruise control for automobiles and autopilot for aircraft increase efficiency and safety by controlling monotonous tasks such as maintaining a speed or heading [63, 98]. The next generation of controllers seek to handle tasks that involve more high-level processing such as path planning [71, 119]. Developments in this area are beneficial for emerging applications that include highway vehicle control [111, 115], driver-assisted autonomous vehicles [43], and unmanned aerial or underwater vehicles (UAV and UUV, respectively) [121].

Equipping agents with autonomous controllers offers several advantages to reach these goals. One particular consequence of additional autonomy is less human-in-the-loop control. In many defense applications, this technology can add another layer of separation between people and dangerous combat situations. Current technology, such as the Predator UAV [32] offers this type of protection by processing some guidance controls on the vehicle and receiving human commands from a remote distance. Other advantages of this technology are that autonomous controllers do not display the same fatigue as a human mind having to work long hours [85]. An artificial controller is also tolerant to adverse operating conditions and can have a compact form to meet size and weight requirements. Examples include the FitPC2 for visual mapping on a mobile robot [137], and the GumStix processor for controlling an autonomous helicopter [5].

This dissertation contributes to the development of autonomous controls for unmanned pursuit-evasion missions. Here, one vehicle attempts to intercept another vehicle, while avoiding unwanted collisions. As shown in later chapters, these controls are used in appli-

cations for highway collision avoidance [16] and sensor reduction for room surveillance [1]. These problems are particularly challenging because the actions (control strategy) of one agent impact the success of another agent. As a result, the controls are found using differential game theory, where one vehicle attempts to intercept another by assuming the worst possible control actions [64]. Similar approaches using differential games are seen in defense related experiments [121] as well as modeling animal populations, business competition, and human interface devices [41, 65, 90, 131].

The task of finding a control using a differential game theoretic approach is to solve the Hamilton-Jacobi-Isaacs (HJI) equation. However, the HJI equation is rarely possible to solve analytically, and is an intense calculation when solved numerically. Some of the more prominent methods [34, 42, 73, 106] can be used, but are subject to increased computational times resulting from limited sensory information [17], the curse of dimensionality [84], reduced computational efficiency [127], and limited adaptability to learned evader dynamics [47]. Therefore, providing guidance to vehicles engaged in pursuit-evasion stands to be improved by arriving at a solution faster so that the controller a) can be implemented on-line and b) limits the amount of assumptions required for the opposing agent.

By implementing the controller on-line, it is possible to calculate the control strategy for an agent while the game is being played. The strategy can be updated quickly to account for observability issues resulting from limited sensing capabilities of the agent and uncertainties in the environment. For example, underwater autonomous vehicles (UUVs) rely primarily on acoustic sensing with sonar which gives bearing information with noise. To obtain position data, target motion analysis (TMA) algorithms must be used [94]. A controller implemented on-line can correct the strategy as more accurate information is received from the TMA algorithm. By limiting the amount of assumptions required for the opposing agent, less information needs to be stored on the on-board processor. Improving this part of the control

results in more robustness by *adapting* to the opposing agent rather than using a previously calculated solution.

1.2 Problem Statement

In this dissertation, two agents operate in a defined environment. An agent is an entity, such as a vehicle, that uses a control strategy to move about a state space which describes the environment. One agent takes the role of the pursuer who attempts to capture the other agent, an evader. It is assumed that only one agent is to receive the control and that the other agent's strategy is assumed to be unknown. A conservative estimate for the other agent's strategy is the differential game approach that assumes the worst-case among all of its possible strategies. Any assumption of a strategy that does not assume a worst-case scenario will be to the other agent's advantage [64]. However, finding the worst-case strategy is a difficult task because all of the possible controls are effectively exhausted to arrive at a solution, resulting in long computational times. This is especially a concern when the other agent's dynamic parameters are unknown and must be learned throughout the game. Therefore, the worst-case control must be quickly updated to account for parameter updates.

The control design strategy developed in this dissertation solves the problem of providing an on-line controller for certain pursuit-evasion games by addressing three sub-problems. The control design and update algorithms a) solve the infinite horizon pursuit-evasion problem, by b) recursively solving the Hamilton-Jacobi-Bellman (HJB) equation, and c) updating the control in neighborhoods of the state space (partitions) as information is learned about the evader.

1.2.1 The Infinite Horizon Solution

By solving the infinite horizon HJI equation, it is possible to determine if an agent can reach a specified target. Target reachability, as defined in this dissertation, concerns identifying where an agent can reach a set of states in the state space, called the target set, in finite time. Past work has used reachability methods to find the set of all reachable states in terms of a finite horizon solution [29]. The work presented in this dissertation uses a similar methodology, but over all possible time. The infinite horizon solution, sometimes called the Game of Kind, determines if it is possible for the trajectory to reach the target set in any finite time [64]. The difference between the two methodologies is that the finite horizon minimizes (or maximizes) an objective over a fixed finite period of time, and the infinite horizon evaluates the objective for all finite times. The problem with the finite horizon approach is that it does not guarantee that the solution will ever reach the desired target set [55]. On the other hand, the infinite horizon approach develops a solution by finding all states where there is a trajectory that ends at the target set. Despite the added benefits of knowing the location of these states, the computational process can be very time consuming due to the minimax operator and large number of action combinations.

1.2.2 Recursive Solutions using the HJB Equation

The solution to the HJI equation usually begins by propagating the control solution out from the target set. To solve the HJI equation, a value is assigned to every state in a trajectory where the agent reaches the target set and is based on a specified cost function. Solving this equation involves numerous calculations that span the entire state space using viscosity methods [42] or rule checking to avoid discontinuities in the solution [106]. Regardless of the method used, the more strategies available to the agents, the longer the solution process

takes. Therefore, one method to reduce calculation times is only finding the control solution over part of the state space. Furthermore, if the minimax operator of the HJI equation could be replaced with just a min or max operator, then the speed of the calculation would increase because less strategy combinations would need to be considered.

1.2.3 Updates to the Controller On-line

In order for a pursuit-evasion based controller to be practical, it must be able to adapt to the surrounding environment. Furthermore, it must be able to respond to different types of opposing agents as they arise. For example, an automobile chasing a motorcycle may choose different actions than the same automobile chasing a tractor-trailer. Incorporating sensory information on both the environment and the opponent into the control strategy is necessary when this information is learned throughout the game. Recalculating the entire infinite horizon solution with the HJI equation would be impractical due to the high computation times. However, breaking the solution into smaller calculation regions would allow the control to be updated without including the possible trajectories from the whole state space.

1.3 Related Research

Pursuit-evasion is a subset of differential game theory where one agent attempts to capture another agent in the state space. This problem formulation fits into a broader context of applications known as zero-sum games. Research areas such as autonomous vehicles, economics, and network control all stand to benefit from improvements to a fast solution technique for two-player zero-sum games.

1. *Autonomous Vehicle Control* - Vehicle autonomy is a major research area of control

theory applications that spans consumer market household vacuum cleaners [122] to state-of-the-art surveillance and tracking vehicles [30]. In these applications, vehicles are used to either directly engage another agent in a role as either pursuer or evader.

(a) *Pursuit Applications* - Applications where an agent is acting as a pursuer are typically involved in defense related activities. Here, the pursuit application can take the form of asset protection [20, 21], where an autonomous vehicle or multiple autonomous vehicles are used to protect an asset. Target capture, or direct pursuit-evasion has been considered in a large-scale probabilistic framework incorporating multiple types of agents such as UAVs and unmanned ground vehicles (UGVs) [70, 120, 121]. Pursuit-evasion is also cast into a distributed control framework called swarms where collective behaviors are gathered from individual actions and interactions [109]. Interference capabilities have also been introduced to the pursuit-evasion strategy by incorporating countermeasures [19]. Lastly, small scale implementations of a pursuer's strategy using approximate dynamic programming techniques are designed to learn the control law [53].

(b) *Evader Applications* - A major topic in non-defense applications is designing for collision avoidance scenarios. Since the worst type of oncoming agent occurs when the pursuing vehicle intentionally heads directly for the evading vehicle, pursuit-evasion games provide a good starting point for collision avoidance planning. One example using aircraft flight planning [117] designates a "safe" zone where the controller may operate to conduct maneuvers. Other applications include collision avoidance in a waterway where the ships are constrained to a fixed throttle turn [91]. Real-time autonomous collision avoidance maneuvers on highways are another major research area [75]. Other methods solve the game off-line and use a neural network to approximate the state-action function for real-time collision

avoidance strategies [76, 89]. Fuzzy logic [79] and potential functions [26] have been used as well.

2. *Economic Control* - The study of economic behaviors, and controlling economic markets is what gave rise to game theory [96]. By applying differential games, it is possible to determine how business evolves as a result of competition and decisions [62, 130]. This type of control theory is also used for determining pricing options for consumers [68] or production flows [13].

3. *Network Control* - In communication networks, optimizing the flow of information is critical to reduce power and congestion of transmitting information. Market based algorithms are a distributed type of control where entities compete for information handling capability [2, 68]. Further applications in network-type control are finding the best options for minimizing time spent in automobile traffic [129].

1.4 Original Contributions

This dissertation discusses and implements a method for solving two-player pursuit-evasion games to provide controls to agents. This is accomplished through calculation reductions and recursive updates. The original contributions of this dissertation are:

1. The solution to the Hamilton-Jacobi-Isaacs equation is simplified by developing a scheme that solves the Hamilton-Jacobi-Bellman equation assuming only a structure of the opposing agent's dynamics. Coupled with already existing fast-marching solution techniques, this allows for a very fast solution of a computationally difficult problem. For the scenario presented in this dissertation, an improvement of 11.6% is seen.

2. The control is solved recursively over a subset of the state space. In doing so, updates to the control reflecting new measurements of the state space or parameters of the opposing agent are easy to incorporate.

3. The temporal evolution of the control is demonstrated for the Homicidal Chauffeur with numerical simulations.

4. The control strategy is applied to a collision avoidance scenario to provide the desired path for a vehicle with more complicated dynamics. In doing so, a methodology is provided to illustrate that the control can be used in 1) a realistic scenario combined with 2) a low-level control to provide vehicle guidance.

5. The control strategy is implemented on a real robotic vehicle system to demonstrate the effectiveness of the previous contributions.

6. The control strategy is used to reduce the number of sensors in a sensor field using reachability analysis. This allows for cost and power reductions.

1.5 Organization

The main focus of this dissertation is showing how problems involving the Hamilton-Jacobi-Isaacs equation can be formed into a recursive scheme that solves the simpler Hamilton-Jacobi-Bellman equation. To illustrate this methodology, needed background information is provided in Chapter 2. Then, a finite horizon approach to minimizing a pursuit-evasion objective is given in Chapter 3. Here, the main issue of not reaching a target set using a finite horizon is discussed. As one solution to this problem, Chapter 4 presents an approximate dynamic programming solution using reinforcement learning combined with a fuzzy logic switched objective approach. With this approach, the learning function monitors the progress

of the trajectories and ensures that they do not become caught in a limit cycle or equilibrium point outside of the target set. To provide a more universal approach using the switched objective method, Chapter 5 presents an algorithm for applying this methodology to two-person pursuit-evasion games. Chapter 6 then shows a temporal evaluation of the control throughout the entire length of the game. Lastly, Chapters 7 and 8, present applications of the partition methodology. Chapter 7 shows the application of the controller to an actual robot vehicle system, and Chapter 8 shows a sensor reduction technique using the partition approach.

Chapter 2

Background

This chapter discusses the background topics consisting of static game theory, optimal control, and differential game theory as used throughout the dissertation. In the following, algorithms and theorems governing the methods in the subsequent chapters are given.

2.1 Introduction to Differential Game Theory

Game theory is a mathematical tool for decision making rooted in the work of J. von Neumann and 0. Morgenstern [96], centered on economic behavior formulated in terms of strategies. In a typical game, four components are needed: agents (or players), strategies, value, and an optimal strategy [6]. Agents are entities that choose strategies and are given a certain level of information about the state of the game. Examples of possible agents include companies, unmanned vehicles, etc. Strategies are decisions made by the agents that alter the state of the game. Essentially, they are control laws. An example of a strategy for a vehicular agent could be turn left or turn right. The value of the game measures the degree of success attributed to playing a particular strategy. This is determined from a cost function that the agent seeks to either minimize or maximize with respect to a particular

objective, e.g. maximizing profit. An optimal strategy is one that succeeds in minimizing or maximizing this value. When each agent determines an optimal strategy, the game has reached a value equilibrium. There are many types of equilibria for optimal strategies and their existence depends on the type of value function considered.

The first type of equilibrium occurs for two-player games with no available information about the other agent's strategy. When this is applied to noncooperative game theory, a gain in value for one player results in a loss of the same magnitude for the other. This is also called a zero-sum game. The optimal strategies for this game result when the Nash equilibrium is reached. The Nash equilibrium is said to occur when either agent cannot improve their value for any possible strategy played by the opposing agent [95]. However, sometimes the agents may achieve a higher value if they share information and cooperate. These are called cooperative, or non-zero sum games. In these situations, a Pareto equilibrium occurs when the agents have extremized the value, and at least one agent will do worse by deviating from the optimal strategy [44]. Lastly, the Stackelburg equilibrium occurs when the agents' choices are staggered by having each agent make alternating decisions [123].

The specific type of games considered in this dissertation are differential (dynamic) games. In these games, the strategy becomes a sequence of decisions that act as a control input into differential equations that govern the state of the game [6] (pp. 1-2). Therefore, the value function does not represent the consequence of a single decision, but a control (decision) map from the state space. Essentially, this involves choosing an optimal control law given a competing agent. As a result, elements of both static games and optimal control theory are highly prevalent in differential games. Specifically, zero-sum differential games are considered where the agents directly compete with each other. One agent seeks to minimize a cost function and the other agent seeks to maximize the same cost function. The games are also noncooperative because the agents are given no knowledge of each other's strategy. Concepts

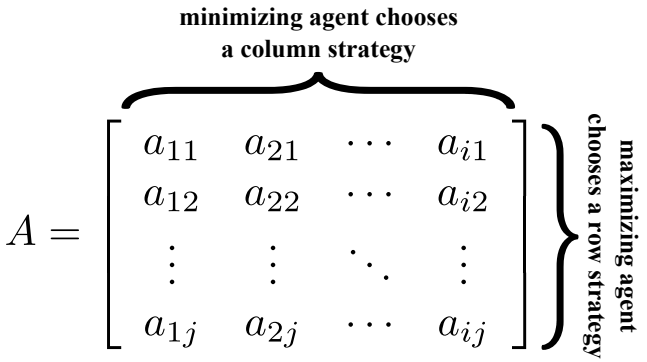


Figure 2.1: The value exchange of a zero-sum game is equal to a_{ij} from the choice of strategy i and j by the minimizing and maximizing agent respectively.

relating static game theory, optimal control theory and differential game theory as used in this work are discussed in the following sections.

2.2 Static Game Theory

Static games occur when the order of the agents' decisions is not important [6] (pp. 3). The structure of a static game can be formulated into one choice of strategy. For two-person zero-sum games this can be represented as a matrix, A, where the column position refers to the strategy choice of the minimizing agent and the row position refers to the strategy choice of the maximizing agent [6] (pp. 20). Each row-column location has a specific value that designates the amount of value exchange as a result of the chosen strategy. This is shown in Fig. 2.1.

In general, there are two types of strategies associated with solving a static game. Pure strategies occur when the agents choose to play a specific strategy pair every time the game is played. However, these solutions are not always guaranteed to exist [6] (pp. 24). The set of available strategies can then be expanded by placing a probability distribution over the set of pure strategies. These mixed strategies have the form, "play $i = 1$ 40% of the

time and $i = 2$ the other 60%". The actual control would be the result of a random number generator over the pure distributions, which would be run every time the game is played.

2.2.1 Saddle-points with Pure Strategies

For a game to have a saddle-point in pure strategies then there is a unilateral "best" move for each agent whenever the game is played. In order to define a saddle-point in pure strategies, the concept of a security level, $\underline{V}(A)$ and $\bar{V}(A)$, is introduced. The security level for an agent is the value associated with the worst possible strategy that the opposing agent can play.

Theorem 1. *([6], pp. 20-21) In every matrix game $A = \{a_{ij}\}$,*

1. *The security level of each player is unique.*
2. *There exists at least one security strategy for each player, and*
3. *The security level of P1 (the minimizer) never falls below the security level of P2 (the maximizer),*

$$\min_i a_{ij^*} = \underline{V}(A) \leq \bar{V}(A) = \max_j a_{i^*j} \tag{2.1}$$

where i^* and j^* denote security strategies for P1 and P2 respectively.

It follows that when $\underline{V}(A) = \bar{V}(A)$, then the game has a saddle-point in pure strategies. This is because the pure strategies, i^* and j^*, form a solution where neither agent can improve their value despite any decision made by the opposing player.

Theorem 2. *([6], pp. 23) Let $A = \{a_{ij}\}$ denote an $(m \times n)$ matrix game with $\bar{V}(A) = \underline{V}(A)$. Then,*

1. *A has a saddle point in pure strategies,*

2. *an ordered pair of strategies provides a saddle point for A if, and only if, the first of these is a security strategy for P1 and the second one is a security strategy for P2, and*

3. $V(A)$ *is uniquely given by* $V(A) = \bar{V}(A) = \underline{V}(A)$.

This type of solution occurs when the agents have perfect information of the states of the game, and there is a strategy that can be applied unilaterally by both agents every time the game is played. An example of where this type of solution occurs is a repeated one-step pursuit-evasion game where two agents start a specified distance apart and have a finite number of strategies to play to proceed toward their goal of either minimizing (or maximizing) the distance between them. This is shown in Fig. 2.2, where the game is initiated by having the pursuer start behind the evader.

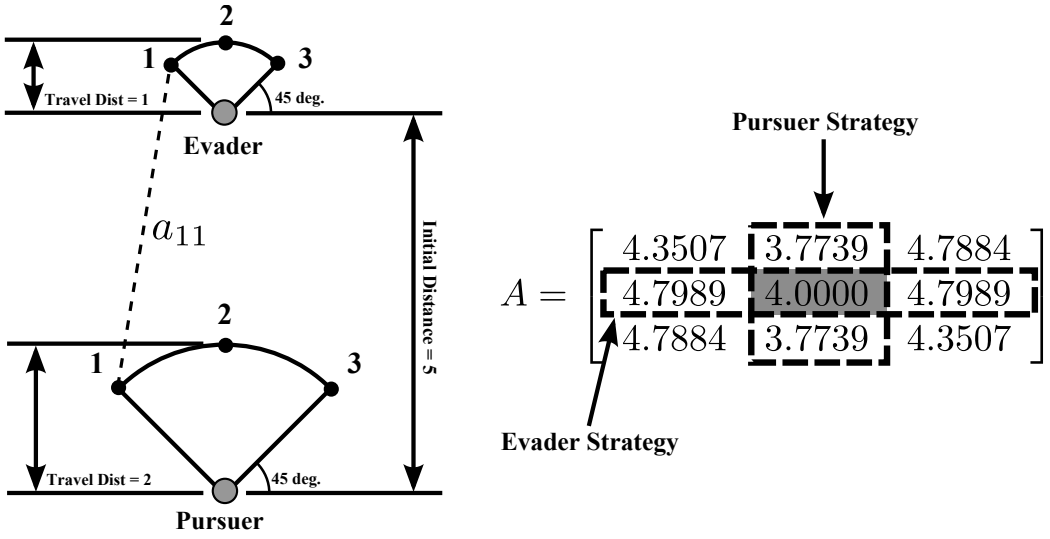


Figure 2.2: A static game of pursuit-evasion where the pursuer intends to take a step to minimize the distance between the agents and the evader takes a step to maximize this distance. The solution is a pure strategy whenever the pursuer starts behind the evader, and can be repeated until capture.

In this game, the pursuer has a higher velocity than the evader and can take a greater step. The resulting step size is 2 for the pursuer and 1 for the evader. Each agent must choose a location 1, 2, or 3, spaced in 45° intervals on the arc with a radius equal to the step size.

The agents are initially separated by a distance of 5. The game can be formulated into the matrix, A, where the pursuer's (minimizing) strategies are represented by columns and the evader's (maximizing) strategies are represented by rows. The value at a_{ij} is determined by calculating the separation distance between the agents when the pursuer plays strategy $i \in \{1, 2, 3\}$ and the evader plays strategy $j \in \{1, 2, 3\}$. Using the matrix of values, A, the pursuer's security strategy is $i = 2$, because the distance between the agents is guaranteed to be at most $\bar{V}(A) = 4$, no matter what the evader plays. Likewise, the evader's security strategy is $j = 2$, because the minimal distance between the agents is $\underline{V}(A) = 4$ no matter what the pursuer plays. In this case, the upper and lower security values coincide,

$$\bar{V}(A) = 4 = \underline{V}(A) \tag{2.2}$$

so the game has a saddle-point in pure strategies with a value, $V(A) = 4$. This indicates that both agents will always play their pure strategies, and will not perform as well if any other strategy is played. In terms of pursuit-evasion, this corresponds to the type of behavior that would be seen in a direct chase scenario with the pursuer behind the evader.

2.2.2 Games with Mixed Solutions

Not all games have a saddle-point solution with pure strategies. Continuing to use the pursuit-evasion game from the previous example, this is illustrated by changing the initial positions of the agents so that the evader starts 5 units *behind* the pursuer. However, both agents are still limited to the three travel options which is an approximation of vehicles with a finite turn radius. The game is illustrated in Fig. 2.3.

In this game, the pursuer's security value is 6.0886 and is indifferent from playing $i = 1$ or 3. Likewise, the evader's security value is 5.7507, which can be achieved with indifference of

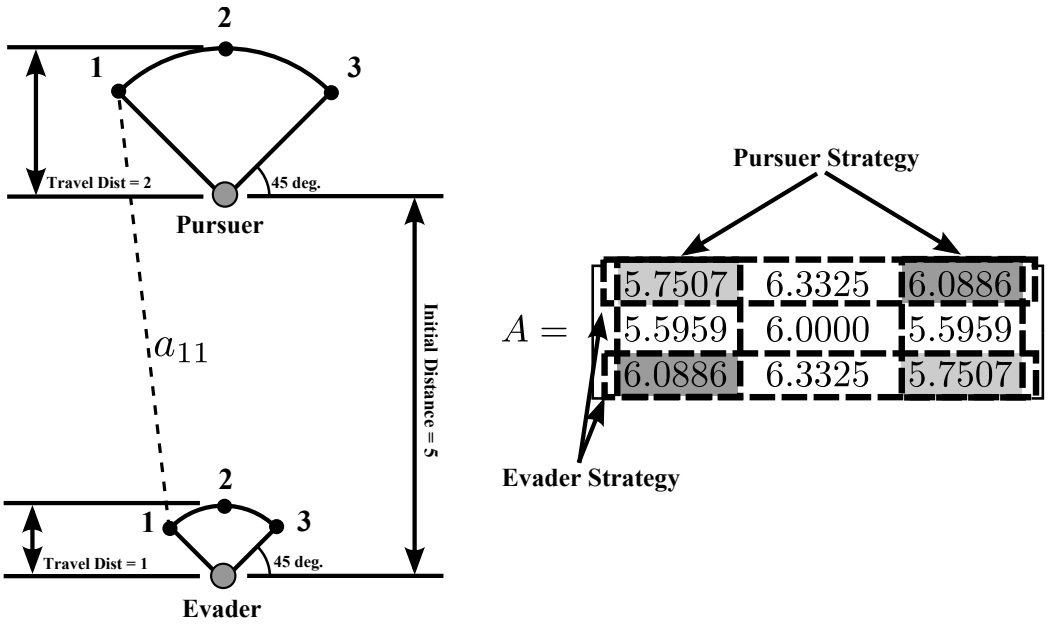


Figure 2.3: A pursuit-evasion game where the agents are not able to turn in any direction instantaneously, and the evader starts behind the pursuer.

playing $j = 1$ or 3. This type of situation results, because the pursuer is never going to just move forward. His greater velocity will always put him at a loss in this situation. Likewise, the evader will never just move forward[1], because this move will always place this agent closer to the pursuer. However, since neither agent is aware of the other agent's strategy, the agents' action to go left or right is unknown. If both agents move in opposite directions, then the pursuer's security level, $\bar{V} = 6.0886$, is achieved. If the agents move in the same direction then the evader's security level, $\underline{V} = 5.7507$ is realized. It follows then, that the optimal strategies will have some probability distribution over the pure strategies. The following theorem provides guidelines for such a result.

Theorem 3. *([6], pp. 27-28) In every matrix game, A, in which the players are allowed to use mixed strategies, the following properties hold:*

1. *The average security level of each player is unique.*

[1]This will change when considering dynamic games using longer prediction horizons.

2. *There exists at least one mixed strategy for each player.*

3. *The security levels in pure and mixed strategies satisfy the following inequalities*

$$\underline{V}(A) \leq \underline{V}_m(A) \leq \bar{V}_m(A) \leq \bar{V}(A) \tag{2.3}$$

The result of this theorem is that there is some unique value associated with each player $V_m(A)$ that is essentially a compromise of playing pure strategies. For the both the pursuer and the evader, the mixed security values, $\bar{V}(A)$ and $\underline{V}(A)$ respectively, are only an improvement on what would be achieved using pure strategies only. The following corollary defines the upper and lower mixed security values.

Corollary 1. *([6], pp. 28) In a matrix game A, the average upper and lower values in mixed strategies are given by,*

$$\bar{V}_m(A) = \min_y \max_z y' A z \tag{2.4}$$

$$\underline{V}_m(A) = \max_z \min_y y' A z \tag{2.5}$$

The upper mixed security value finds the best probability distribution, y, that the pursuer can play against a mixed strategy formed by the evader z. Similarly, for the lower mixed value, a distribution z is found to combat the pursuer's choice of y. Using mixed strategies leads to a useful result showing the equivalence of the upper and lower values.

Theorem 4. *([6], pp. 29) In any matrix game A, the average security levels coincide,*

$$\bar{V}_m(A) \min_y \max_z y' A z = \max_z \min_y y' A z = \underline{V}_m(A) \tag{2.6}$$

This theorem shows that there is a saddle-point associated with every game, A, using mixed

strategies. This is summarized in the following corollary.

Corollary 2. *([6], pp. 30-31) Let A denote an* $(m \times n)$ *matrix game. Then,*

1. *A has a saddle point in mixed strategies.*
2. *A pair of mixed strategies provides a saddle point for A if, and only if, the first of these is a mixed security strategy for P1, and the second one is a mixed security strategy for P2.*
3. $V_m(A)$ *is uniquely given by* $V_m(A) = \bar{V}_m(A) = \underline{V}_m(A)$
4. *In the case of multiple saddle points, the mixed saddle point strategies posess the ordered interchangeability property.*

Finishing the example, each agent will have a 50% probability of choosing strategy 1 or 3 because both are indifferent to these two strategies. Therefore, $y = [0.5\ 0.5]^T$ and $z = [0.5\ 0.5]^T$, and the mixed value is $V_m(A) = 5.92$. This type of play is also seen in dynamic pursuit-evasion, and will be discussed extensively in Chapter 7. However, not all games are as easy to determine the mixed strategies, as discussed in Chapter 3. The next section provides an outline for arriving at a solution to static games admitting mixed strategies.

2.2.3 Finding Mixed Solutions with Linear Programming

Finding mixed strategies for games that have multiple pure strategies can become a difficult task. To handle larger matrices, linear programming can be used to solve for both the mixed strategies and the mixed value. In order to do so, the game matrix must first be padded with a positive constant to ensure that all entries in A are positive. Then, the strategies are found with a linear programming algorithm [92](pp. 824-835) using the two sub-problems presented in the following theorem.

Theorem 5. *([6], pp. 36) Let B be an $(m \times n)$ matrix game, and A,*

$$A = B + cl_m l_n' \tag{2.7}$$

where

$$l_m = (1, 1, \ldots, 1)' \in \mathbb{R}^m \tag{2.8}$$

and the same holds for l_n. The constant, c, is chosen to make all entries positive. Introducing two linear programming (LP) problems,

1. *The primal problem:*

$$\begin{aligned} \max & \, y' l_m \\ A'y &\leq l_n \\ y &\geq 0 \end{aligned} \tag{2.9}$$

2. *The dual problem:*

$$\begin{aligned} \min & \, z' l_n \\ Az &\leq l_m \\ z &\geq 0 \end{aligned} \tag{2.10}$$

with their optimal values (if they exist) denoted by V_p and V_d respectively.

1. *Both LP problems admit a solution, and $V_p = V_d = \frac{1}{V_m(A)}$.*

2. *If (y^*, z^*) is a mixed saddle point solution of the matrix game B, then $\frac{y^*}{V_m(A)}$ solves the problem in 2.9 and $\frac{z^*}{V_m(A)}$ solves the problem in 2.10.*

3. *If $\tilde{y}^*$ is a solution of (2.9) and $\tilde{z}^*$ is a solution of (2.10), the pair $(\frac{\tilde{y}^*}{V_p}, \frac{\tilde{z}^*}{V_d})$ constitutes a mixed saddle point solution for the matrix game B. Furthermore, $V_m(B) = (\frac{1}{V_p}) - c$.*

This theorem allows for using readily available linear programming algorithms [92], (pp. 834) to numerically solve for both agent's mixed strategies. Once problems 2.9 and 2.10 are solved by obtaining $\tilde{y}^*$ and $\tilde{z}^*$, then the mixed strategy for the game can be found by multiplying by $V_m(A)$. This technique is used extensively in Chp. 3 to obtain the solutions to the static games.

2.3 Optimal Control Theory

The other major component of differential games is optimal control theory where the order of the control sequence is important as the system evolves in time. Unlike game theory with multiple agents, an optimal control for one agent is unambiguous [6], and is governed by minimizing or maximizing a cost functional. The optimal control can be found using Pontryagin's Principle [78], but can be numerically solved using dynamic programming with the Hamilton-Jacobi-Bellman (HJB) equation [15]. This equation is based on Bellman's Principle of Optimality [12] which states that an optimal trajectory must be optimal from any point, $x(t), t < t_f$, along the trajectory to a final point $x(t_f)$. Using this principle, solution techniques have been developed by finding optimal controls starting at the set of target states and working outward.

Solution techniques employing the Priniciple of Optimality can be applied in many different forms depending on the prediction horizon and the number of admissible controls [14, 15]. In this dissertation, the Principle of Optimality is applied over both a finite and infinite horizon with a discrete number of admissible controls. The infinite horizon HJB equation with discounting is applied in Chapter 4 with approximate dynamic programming techniques using learning reinforcement; however, the main thrust of this research uses the finite horizon form of the HJB equation as the final time, $t_f \to \infty$. The difference between this and the

infinite horizon HJB equation is that no discounting is applied to the cost in future states. In the subsequent section, the finite horizon HJB equation is derived where it is desired to find the optimal control from a time, t_0 to t_f. This form of the HJB equation is used in the infinite horizon graph theoretical approach presented in Chapter 5. Because a discount is not applied, specific requirements are placed on the running cost, $g(x,t)$, as noted throughout the dissertation. The main restriction is that the cost is to remain finite everywhere and equal to 0 only in the set of states where the agent should direct the trajectory.

2.3.1 The Hamilton-Jacobi-Bellman Equation

The Hamilton-Jacobi-Bellman (HJB) equation is a partial differential equation (PDE) that is satisfied by a "cost to go" or value function. The value function refers to the magnitude of the cost required to travel along the optimal trajectory from the current state, $x(t)$. The finite horizon derivation of the HJB equation is as follows (a similar derivation may be found in [15] (pp. 109-110)). Consider a cost function for a state $x \in X$ and a control law $u(\cdot)$,

$$J(x,u) = \int_t^{t_f} g(x(s),u(s))ds + g_0(x(t_f)) \tag{2.11}$$

where $g(x(t),u(t))$ is a running cost associated with each time step during the play of game. The function $g_0(x(t_f))$ is a terminal cost. Subsequently, the value function is defined as

$$V(x) = \min_{u\in\mathcal{U}} \left[\int_t^{t_f} g(x(s),u(s))ds + g_0(x(t_f)) \right] \tag{2.12}$$

where $\mathcal{U}$ is the set of admissible control functions $u(\cdot)$. This can be broken into

$$V(x) = \min_{u\in\mathcal{U}} \left[\int_t^{t+\Delta t} g(x(s),u(s))ds + \int_{t+\Delta t}^{t_f} g(x(s),u(s))ds + g_0(x(t_f)) \right] \tag{2.13}$$

and by substituting in Eqn. 2.12, can be rewritten as

$$V(x) = \min_{u \in \mathcal{U}} \left[\int_t^{t+\Delta t} g(x(s), u(s)) ds \right] + V(x(t + \Delta t)) \tag{2.14}$$

The latter term can then be approximated as a first order Taylor series

$$\begin{aligned} V(x(t)) = \min_{u \in \mathcal{U}} & \left[\int_t^{t+\Delta t} g(x(s), u(s)) ds \right] \\ & + V(x(t)) + \nabla_t V(x(t)) \Delta t + \nabla_x V(x(t)) f(x, u) \Delta t + \mathcal{O}(\Delta t^2) \end{aligned} \tag{2.15}$$

where $\mathcal{O}(\Delta t^2) \to 0$ as $\Delta t \to 0$ and contains the second order and higher terms. The term, $V(x(t))$, is canceled from both sides and the whole equation is divided by Δt resulting in

$$0 = \frac{1}{\Delta t} \min_{u \in \mathcal{U}} \left[\int_t^{t+\Delta t} g(x(s), u(s)) ds \right] + \underbrace{\nabla_t V(x(t))}_{\dot{V}(x(t))} + \nabla V_x (V(x(t)) f(x, u) + \mathcal{O}(\Delta t) \tag{2.16}$$

Taking the limit as $\Delta t \to 0$ yields the HJB equation

$$0 = \min_{u \in U} \left[g(x(t), u(t)) + \nabla_x V(x(t)) f(x(t), u(t)) \right] + \dot{V}(x(t)) \tag{2.17}$$

where the set of admissible controls has been changed to U to represent a set of single control values rather than a set control sequences. The value function and the HJB equation are related by the sufficiency theorem:

Theorem 6. *([15], pp. 111) Suppose $V(t, x)$ is a solution to the Hamilton-Jacobi-Bellman equation; that is, $V(t, x)$, is continuously differentiable in x and t, and is such that*

$$0 = \min_{u \in U} \left[g(x(t), u(t)) + \nabla_x V(x(t)) f(x(t), u(t)) \right] + \dot{V}(x(t)), \forall x, t, \tag{2.18}$$

$$V(t, x) = g_0(x), \forall x \tag{2.19}$$

Suppose also that $u^*(t,x)$ attains the minimum in Eqn. 2.18 for all t and x. Let $\{x^*(t)|t \in [0,t_f]\}$ be the state trajectory obtained from the given initial conditions $x(t_0)$ when the control trajectory $u^*(t) = u^*(t,x(t)), t \in [0,t_f]$ is used (that is $x^*(0) = x(0)$ and $\forall t \in [0,t_f], \dot{x}^*(t) = f(x^*(t), u^*(t,x^*(t)))$; we assume that this differential equation has a unique solution starting at any pair (t,x) and that the control trajectory $\{u^*(t,x^*(t))|t \in [0,t_f]\}$ is piecewise continuous as a function of t. Then V is the unique solution of the HJB equation and is equal to the optimal cost-to-go

$$V(t,x) = \min_{u \in \mathcal{U}} \left[\int_t^{t_f} g(x(s),u(s))ds + g_0(x(t_f)) \right], \forall t, x. \tag{2.20}$$

Furthermore, the control trajectory $\{u^*(t)|t \in [0,t_f]\}$ is optimal.

This theorem identifies the solution to the HJB equation as the value function, $V(t,x)$, which represents the cost-to-go with respect to a cost function, $J(x,u)$. As seen in Thm. 6, the value function and the optimal control, $u^*(t,x)$, are related through the running cost, terminal cost, and agent dynamics. The pursuit-evasion scenarios presented in the subsequent chapters all use a running cost not dependent upon u, so the optimal control law, $u^*(x)$, can be found using the value function [15](pp. 115).

$$u^*(x) = \min_{u \in U} [\Delta_x V(x(t)) f(x(t),u(t))] \tag{2.21}$$

Methods to solve the HJB equation tend to fall into two categories, policy iteration and value iteration [14, 15]. Policy iteration methods update the control until the solution converges to u^* and V, and value iteration methods recursively update the value function. The work in this dissertation uses a semi-Lagrangian approach that iterates the value function [42].

2.3.2 Numerical Solution of the HJB PDE

The solution techniques used in this dissertation are viscosity solutions that solve the HJB equation at every state to allow for discontinuities that arise in the control surface [42]. Examples of when these discontinuities occur are singular surfaces, i.e., when an agent may be indifferent between two controls. In order to solve the HJB equation numerically, we first discretize the HJB PDE given by Eqn. 2.17 in both space and time with $h = \Delta t$ and $x_i \in \bar{X} \subset X$ where $\bar{X}$ is a grid lattice.

$$V(x_i) = \min_{u \in U} \{V(x_i + hf(x_i, u))\} + hg(x_i, u) \tag{2.22}$$

In this dissertation, a running cost $g(x_i, u)$, like time-optimality is used to measure the time to reach a set of target states, $\mathcal{T} \subset X$. Furthermore, a playable area, $\Omega \subset X$ is defined where the agent may operate. Details and restrictions placed on the running cost with respect to these two sets are given in Chapter 5. To solve for the value function associated with finding a control $u(\cdot)$ to reach $\mathcal{T}$, the discrete HJB equation is evaluated at every node in the state space by solving a first-order approximation of the HJB equation, Eqn. 2.22. To initiate the algorithm, the discretized state space, $\bar{X}$, is separated into three regions [42] (pp. 249),

$$I_{\mathcal{T}} = \{i \in I : x_i \in \mathcal{T}\} \tag{2.23}$$

$$I_{out} = \{i \in I : x_i + hf(x_i, u) \notin \Omega, \forall u \in U\} \tag{2.24}$$

$$I_{in} = \{i \in I : x_i + hf(x_i, u) \in \Omega\} \tag{2.25}$$

These regions describe the set of target states, $\mathcal{T}$, the set of states where all controls lead outside of the playable area. This area, Ω, represents the set of states where there is a valid local trajectory. The scheme for solving the HJB equation is then given as

Algorithm 1. *Semi-Lagrangian Numerical Solution for the HJB PDE*

1. *INITIALIZE* $V(x_i) = V_o$
2. *UPDATE the value function (for time-optimality, see [42], pp. 249):*

$$V(x_i) = \min_{u \in U}\{V(x_i + hf(x_i, u))\} + hg(x_i, u), \qquad \textit{for } i \in I_{in} \tag{2.26}$$

$$V(x_i) = 0, \qquad \textit{for } i \in I_{\mathcal{T}} \tag{2.27}$$

$$V(x_i) = \infty, \qquad \textit{for } i \in I_{out} \tag{2.28}$$

3. *REPEAT until* $L_2(V(x_i)) < \epsilon$

This algorithm updates an initial value function, V_0, by solving an Euler step in discrete time at every discrete node, x_i, of the grid lattice, $\bar{X}$. States not located within the travel boundaries, Ω, and have no possible control to remain within Ω are assigned the highest value of ∞. In words, this indicates that it takes an infinite amount of the cost, $g(x_i, u)$, to reach the target. States within the target are assigned a value of 0, because the target has been reached. The critical part of the algorithm is the update for all the states $x_i \in \Omega \backslash \mathcal{T}$. The term, $V(x_i + hf(x_i, u))$, is estimated through interpolation of the value at the nearest nodes within a neighborhood of $x_i + hf(x_i, u)$.

The discrete HJB equation is solved recursively for the value, $V(x_i)$, at every node, x_i, until the solution converges, or the L_2 error between iterations reaches some value less than ϵ. This error is found by

$$L_2(v(x_i)) = \|V_k(x_i) - V_{k-1}(x_i)\|_2 \tag{2.29}$$

where k is the step iteration number of Algorithm 1. The initial value function, V_0, is found

by assigning the following values to the state space,

$$V_0(x_i) = \begin{cases} 0, & \text{for } x_i \in \mathcal{T} \\ \infty, & \text{otherwise} \end{cases} \tag{2.30}$$

Using this algorithm and initial value, states x_i located within a neighborhood of the target $\mathcal{T}$ will receive a value increase on the first iteration. On the next iteration, states within a larger neighborhood will receive a value increase. Essentially, the value flows outward from the target until convergence is reached. With this choice of V_0, the value $V(x_i)$ will converge monotonically to some fixed point value $V^*(x_i)$ using Algorithm 1 [42] (pp. 251).

Throughout the dissertation, this solution technique is referred to as the "classic" method, because it recursively solves every node at every iteration until convergence is reached over the entire state space. The methods in this dissertation simplify this technique to more manageable sizes. Work by Sethian [110] has produced a Fast Marching semi-Lagrangian method for finding viscosity solutions that solve the discrete HJB equation over advancing fronts, and is used extensively. An explicit algorithm for this method is given in Chapter 5.

2.4 Differential Game Theory

A two person, zero-sum, differential game describes a scenario when two agents optimize their strategy against each other by solving problems in game theory using differential equations. Topics such as saddle points and Nash equilibria are used, but applied over a length of time. Differential games also possess the components of agents, strategy, value, and an optimal strategy, except that in this formulation, a strategy is a control law. The control law then governs a set of differential equations that describe the evolution of the state

variables associated with each agent. Like the zero-sum construction in static game theory, one agent seeks to minimize the value of the game, and one agent seeks to maximize it. Similar to optimal control theory, the cost function in differential games is assumed to have the form

$$J(x,u,d) = \int_t^{t_f} g(x(s,u(s),d(s))ds + g_0(x(t_f)) \tag{2.31}$$

which has an additional variable, d, and is the opposing agent's control strategy. In differential games, the term $g_0(x(t_f))$ takes on a slightly different meaning. This boundary condition is used to describe the target set, $\mathcal{T}$. This is the set of states that one agent can guide the trajectory into and "win" the game.

2.4.1 Types of Differential Games

Many different examples of differential games can be found in the literature and vary depending on the level of information available to the agents, the dynamics of the agents, and the type of target set. Games with perfect information assume that the agents know everything about the state of the system. Likewise, games with limited or no information on the state of the game have imperfect information.

Classic examples of differential games are:

1. Homicidal Chauffeur [64]-

   The Homicidal Chauffeur game is a pursuit-evasion game where a car (the pursuer) chases a slower, but more agile person (evader). The pursuer wins if it is able to guide the state of the game so that the relative distance between the two agents is less than a prescribed magnitude. The agents have perfect information of each other's state. The main challenge of this game is to determine the singular surface in the state space

where the pursuer must temporarily turn away from the evader before turning about to proceed in capture with a trajectory called the "swerve".

2. Acoustic Homicidal Chauffeur [105]-

   The Acoustic Homicidal Chauffeur game is similar to the ordinary Homicidal Chauffeur except that the pursuer's perception of the evader's location depends on the sound output of the evader. This is captured in the dynamics by assuming that the evader moves more slowly when close to the pursuer. The main result of this game, shown by [23], is that holes with infinite value can appear in the value function and designate where the evader can escape.

3. Game of Two Cars [87]-

   The Game of Two Cars was also developed by Isaacs and has the same motivation of the Homicidal Chauffeur. In this game, however, the evader is not given the freedom to instantaneously choose a direction. Rather it is also confined to the constraint of having a turn radius. The solution to the game of two cars has appeared in work such as [87, 104]. The additional constraint results in more discontinuities appearing in the surface of the value function, and the game becomes harder to solve.

4. Princess and the Monster [45, 48]-

   All of the games above are deterministic, because they are solved assuming perfect information (the Acoustic Homicidal Chauffeur makes assumptions on the dynamics to account for the lack of information). However, in this game neither the pursuer (the monster) nor the evader (the princess) have knowledge of the state of the other agent. The solution of this game is one where there is a probability distribution over the actions of the agents. This is similar to the mixed strategy solutions seen in static games. The mixed strategy is needed to account for the lack of knowledge of the

other agent's position. Therefore, the goal becomes trying to increase (or decrease) the probability of running into the other agent.

2.4.2 Principles of Differential Games

Similar to the HJB equation in optimal control theory, the Hamilton-Jacobi-Isaacs (HJI) equation relates the value function and optimal controls of a differential game to a specified cost. To derive the HJI equation, we assume a cost given by Eqn. 2.31. The optimal cost-to-go for a finite time horizon is given by

$$\bar{V}(x,t) = \min_{u\in\mathcal{U}} \max_{d\in\mathcal{D}} \left[\int_t^{t_f} g(x(s,u(s),d(s))ds + g_0(x(t_f)) \right], \text{ (upper value)} \tag{2.32}$$

$$\underline{V}(x,t) = \max_{d\in\mathcal{D}} \min_{u\in\mathcal{U}} \left[\int_t^{t_f} g(x(s,u(s),d(s))ds + g_0(x(t_f)) \right], \text{ (lower value)} \tag{2.33}$$

where $\mathcal{U}$ and $\mathcal{D}$ are the sets of admissible control strategies for the minimizer and maximizer, respectively, and the min and max functions assume that the control exists in $\mathcal{U}$ and $\mathcal{D}$. The value function is currently identified by the order of the minimax operator. In this work on pursuit-evasion, the values coincide so that $\bar{V} = \underline{V} = V$ using the minimax assumption.

Assumption 1. *Minimax Assumption ([64], pp. 35):*

$$\begin{aligned} \min_{u\in\mathcal{U}} \max_{d\in\mathcal{D}} \left[\int_t^{t_f} g(x(s,u(s),d(s))ds + g_0(x(t_f)) \right] = \\ \max_{d\in\mathcal{D}} \min_{u\in\mathcal{U}} \left[\int_t^{t_f} g(x(s,u(s),d(s))ds + g_0(x(t_f)) \right] \end{aligned} \tag{2.34}$$

This assumption is satisfied for the pursuit-evasion scenarios in this dissertation because the dynamics, $f(x,u,d)$ and the terminal cost are separable and functionally independent of the strategies, $u(\cdot)$ and $d(\cdot)$ [64]. Conditions are also placed on the set of available control values.

Assumption 2. *([64], pp. 40-42) The set of available control values U and D are assumed to be both (1) convex and (2) closed.*

Convexity of the available controls ensures that there is a unique solution. The assumption of a closed set ensures that the limit is included [64].

Using these assumptions and replacing the min operator with the minimax operator in the HJB derivation of Section 2.3.1, the HJI equation is given as,

$$0 = \min_{u \in \mathcal{U}} \max_{d \in \mathcal{D}} \left[g(x(t), u(t), d(t)) + \nabla_x V(x(t)) f(x(t), u(t), d(t)) \right] + \dot{V}(x(t), t) \tag{2.35}$$

This equation is solved similarly to the methods presented for the HJB equation. However, the minimax operator adds an additional level of difficulty that must be addressed. Solution techniques are now discussed that provide the optimal control,

$$u^*(x) = \arg\min_{u \in U} \max_{d \in D} [\Delta_x V(x(t)) f(x(t), u(t), d(t))] \tag{2.36}$$

assuming a worst-case opponent, $d^*(x)$, (assuming the Nash equilibrium exists)

$$d^*(x) = \arg\max_{d \in D} \min_{u \in U} [\Delta_x V(x(t)) f(x(t), u(t), d(t))] \tag{2.37}$$

over the infinite horizon as $t_f \to \infty$.

2.4.3 Solution Techniques for Differential Games

Solutions that implement Bellman optimality by solving the Hamilton-Jacobi-Bellman (HJB) or Hamilton-Jacobi-Isaacs (HJI) equations over the infinite horizon are able to identify sets of initial states where it is known that trajectories will ultimately terminate in the set

of target states, $\mathcal{T}$. This is possible, because methods for solving these equations (such as [34, 42, 64, 73]) begin calculating the solution at the set of target states and expanding outward. If the resulting value of a state is finite, $V(x) < \infty$, then it will eventually terminate on the target set. Solutions that provide this information are known as infinite horizon solutions, and solve the HJI equation as $t_f \to \infty$.

Despite knowing the set of states that are capable of traversing to the target set, the main disadvantage of solving the HJI equation is the number of computations required to reach a solution. Unlike finite horizon methods which do not optimize from the target set, $\mathcal{T}$, the very first state which is farthest from the target, takes the longest to calculate. Solutions such as the Fast Marching Semi-Lagrangian (FMSL) methods [34] seek to alleviate this burden by reducing the set of states involved in the calculations to those where the value has not yet converged. Methods using extremal aiming techniques have a strictly front-based propagation making them faster for short distances, but experience difficulties correcting for discontinuities that develop as the front propagates as seen in [106].

This section outlines three different types of solutions for differential games. The first of which is a finite horizon one-step Nash equilibrium solution that repeatedly solves the static matrix game. The second is the direct solution to the HJI equation, Eq. 2.35, over the infinite horizon. Lastly, approximate dynamic programming techniques are given that solve a discounted infinite horizon HJI equation to improve the calculation efficiency of finding the solution.

One-Step Nash Equilibrium

In terms of pursuit-evasion differential games, finite horizon objectives differ from infinite horizon objectives because they do not initiate the optimization calculation at the target

set. Rather, the minimization starts at the current state of the agent and optimizes over a finite amount of time. This is a receding horizon approach where the agent repeats the optimization repeatedly throughout the game.

One method that has had success in finding controls for probabilistic differential games using a one-step Nash approach is proposed in [61]. The states and actions of the agents are discretized into finite lattices, and the game is solved similarly to Example 1. Using a nested information structure, the evading agent is aware of all of the pursuer's states, but the pursuer must use observations to gain information about the evader. The game proceeds by having the pursuer(s) minimize the distance to the evader by making a decision one step at a time. This one-step Nash approach is very attractive for its quick computational aspects and incorporating observations, but as shown in [55], even with perfect observations assigned to both agents, the one step look ahead horizon is not enough to successfully guide either agent to their desired target set when constrained by certain dynamic qualities. In [56] the authors give another example using the Homicidal Chauffeur game to show when this type of behavior can emerge with short-horizon objectives. In both papers, when the cost function fails to direct the agent to make the proper decisions outside of the prediction horizon, the trajectories either stagnate or form a limit cycle in the state space outside of the target set. When such a condition exists, the agents would actually perform better by temporarily deviating from the Nash equilibrium.

Solving the HJI equation Directly

Solving the HJI equation directly involves finding a control law and value function pair that satisfies Eqn. 2.35. For games like the Homicidal Chauffeur, the HJI equation can be solved by integrating the value function and dynamics in reverse time. This gives an analytical solution throughout most of the state space as shown by Isaacs [64]. The reverse integration

begins at the target and extends into the state space. As singular surfaces are encountered, the process is repeated with trajectories starting on these surfaces. Unfortunately, with multiple singular surfaces and complicated dynamics, the solution may not even be possible to obtain. Therefore, numerical methods have been developed to address these issues.

When solving the HJI equation numerically, the dynamic equations are discretized in both time and space. Schemes that approximate the value function, $V(x)$, include projecting the trajectories in reverse time from a level-set and propagating outward from the target as seen in [106]. However, these types of schemes require careful attention to discontinuities appearing in the front propagation. The method used as a comparison throughout this dissertation is the semi-Lagrangian scheme given earlier for optimal control solutions to the HJB equation [42]. Much of the procedure will remain the same except the min operator is replaced with the minimax operator.

The discrete HJI equation with time step $h = \Delta t$ and grid lattice $x_i \in \bar{X} \subset X$ is given as

$$V(x_i) = \max_{d \in D} \min_{u \in U} \{V(x_i + hf(x_i, u, d))\} + hg(x_i, u, d) \tag{2.38}$$

Just as with the HJB solution, a running cost, $g(x_i, u, d)$, like time-optimality is used to measure how long it takes to reach a set of target states, $\mathcal{T} \subset \Omega \subset X$. Once again, Ω is the admissible playing area. In this dissertation, time-optimality is used where $g(x_i, u, d) = 1$ and the target set, $\mathcal{T}$, is a ball around the pursuer. To handle infinite values, the Kružkov transformation, $w(x_i) = 1 - e^{-\lambda V(x_i)}$ can be substituted into 2.38 so that the HJI equation becomes,

$$w(x_i) = \max_{d \in D} \min_{u \in U} \{e^{-h} w(x_i + hf(x_i, u, d))\} + 1 - e^{-h} \tag{2.39}$$

This substitution provides a one-to-one transformation from $V(x_i)$ to $w(x_i)$ which has a range $[0, 1]$. Proceeding in a similar manner to the HJB solution, the algorithm is initiated

by separating the discretized state space, $\bar{X}$, into three[2] regions [42] (pp. 252),

$$I_{\mathcal{T}} = \{i \in I : x_i \in \mathcal{T}\} \tag{2.40}$$

$$I_{out} = \{i \in I : x_i \notin \Omega\} \tag{2.41}$$

$$I_{in} = \{i \in I : x_i + hf(x_i, u, d) \in \Omega \backslash \mathcal{T} \text{ for any } u \in U, d \in D\} \tag{2.42}$$

These regions designate where the value will be interpolated, and where it has a static value due to a state being in the target set or outside of the playable area. The algorithm for solving the HJI equation is given as,

Algorithm 2. *Semi-Lagrangian Numerical Solution for the HJI PDE*

1. *INITIALIZE* $w(x_i) = w_o$

2. *UPDATE the value function ([42], pp. 252):*

$$w(x_i) = \max_{d \in D} \min_{u \in U} \{e^{-h} w(x_i + hf(x_i, u, d))\} + 1 - e^{-h}, \qquad \text{for } i \in I_{in} \tag{2.43}$$

$$w(x_i) = 0, \qquad \text{for } i \in I_{\mathcal{T}} \tag{2.44}$$

$$w(x_i) = 1, \qquad \text{for } i \in I_{out} \tag{2.45}$$

3. *REPEAT until* $L_2(w(x_i)) < \epsilon$

The algorithm is repeated until the temporal error, $L_2(w(x_i))$ becomes less than some con-

[2]The source defines four regions to designate playable areas for the pursuer and evader separately. The work in this dissertation does not make this distinction by assuming that both agents can travel in the same region, Ω.

stant ϵ. The grid is initiated as before with w_0 given as

$$w_0 = \begin{cases} 0, & \textit{if } x_i \in \mathcal{T} \\ 1, & \textit{otherwise} \end{cases} \tag{2.46}$$

For a convergence result, the sequence w_n^δ is produced by Algorithm 2 with each iteration, n. Furthermore, $\mathcal{T}_\delta = \{x : d(x, \mathcal{T}) \leq \delta\}$ and is a neighborhood of states around the target set. Lastly, the true lower value of the game, $\underline{V}(x)$ after undergoing the Kružkov transformation is given by

$$w^* = 1 - e^{-\lambda \underline{V}(x)} \tag{2.47}$$

Using this notation, the scheme given in Algorithm 2 converges temporally to the true lower solution as stated in the following theorem.

Theorem 7. *([42], pp. 253) For all x there exists the limit*

$$\bar{w}(x) = \lim_{\substack{\delta \to 0^+ \\ n \to +\infty \\ n \geq n(\delta)}} w_n^\delta(x) \tag{2.48}$$

and it coincides with the lower value w^* of the game with target $\mathcal{T}$, i.e., $\bar{w} = w^*$. Convergence is uniform on every compact set where w^* is continuous.

This provides a temporal convergence result that approaches the value of the game given a target, $\mathcal{T}$. However, the convergence may not happen quickly as discussed in Chapter 5. As the resolution of the state space discretization becomes finer and more complex, it becomes subject to the curse of dimensionality. For example, [34] reports a six-fold increase in the calculation time for a four-fold increase in the number of nodes. To address this problem, current reduction methods for solving the HJI equation are found in the field of approximate dynamic programming.

2.5 Approximate Dynamic Programming

Due to numerical issues such as the curse of dimensionality, solving the HJB or HJI equation directly can become highly impractical with fine resolution grids and complex dynamics. Therefore, the field of approximate dynamic programming has produced a set of practical algorithms for indirectly solving the value function. In particular, these algorithms seek to solve the infinite horizon discounted HJB equation,

$$V(x) = \min_u \{g(x,u) + \gamma V(x)\}, \qquad \text{(continuous)} \tag{2.49}$$

$$V(x(k)) = \min_u \{g(x(k+1),u) + \gamma V(x(k+1))\}, \qquad \text{(discrete)} \tag{2.50}$$

where $\gamma \in [0,1]$ is a constant discount factor. With approximate dynamic programming, the opposing agent's control $d(\cdot)$, is not included because these schemes learn and update the value function, $V(x)$, based on $d(\cdot)$ recursively. Therefore, it is not possible to act against a worst-case scenario with these methods unless these opposing actions are known in advance.

There are four tasks of developing an approximate dynamic programming algorithm [127]:

1. What functional structure will be used to approximate the value?
2. How will the functional structure be updated?
3. How will the control, $u(x)$, be updated?
4. How to coordinate and manage the updates?

Numerous algorithms have been created that address each of these points. They can be broken down into three categories: (1) value approximation, (2) using alternative equations, and (3) hybrid designs. Value function approximations seek to use alternate functional forms

of the value function, $V(x) \approx V(x, W)$, where W is some set of adjustable parameters. For example, $V(x, W)$ can be implemented as a neural network [33]. Other forms can be Taylor series, wavelet or fuzzy logic approximators [127]. The alternate equation approach looks at using other equations other than the HJB or HJI equation. One common approach in differential games is Q-learning where a recursion equation is used to solve for a state-action function, $Q(x, u)$, rather than the value function $V(x)$ [127]. The remainder of this section serves as an overview of research in both Q-learning and value function approximation using fuzzy logic, which is used extensively in Chapter 4.

2.5.1 Q-Learning

In a Q-learning scheme, an alternate equation is used to associate state-action pairs by solving a recursion equation derived from the infinite horizon discounted HJB equation in discrete form, with time step k, given as [127],

$$V(x(k)) = \max_{u(k)} \{r(x, u) + \gamma V(x(k+1))\} \tag{2.51}$$

where $\gamma \in [0, 1]$ is a discount. Here, instead of minimizing a cost, $g(x, t)$, the agent tries to maximize a reward, $r(x)$, by playing an action u at state x. By defining Q as

$$Q(x, u) = \{r(x, u) + \gamma V(x(k+1))\} \tag{2.52}$$

and substituting Eqn. 2.52 into Eqn. 2.51, the discrete HJB equation can be written as

$$V(x) = \max_{u(k)} Q(x(k), u(k)) \tag{2.53}$$

Then, substituting Eqn. 2.53 into Eqn. 2.52 results in the Q-learning recursive equation,

$$Q(x,u) = \{r(x(k+1),u(k+1)) + \gamma \max_{u(k+1)} Q(x(k+1),u(k+1))\} \tag{2.54}$$

Once trained, the Q-learning method can be implemented as a lookup table [127]. However, this approach also suffers from the need to increase computational resources as the state space becomes larger and more complex. Therefore, some other approaches have implemented neural networks to approximate $Q(x,u)$ [35]. More recently, Q-learning has been combined with a fuzzy inference system to provide capabilities for on-line learning as agents repeatedly interact in pursuit-evasion scenarios [37, 38]. With these techniques, supervised learning is not required so an agent can learn a control law in real time, but many iterations are still required for the solution to converge. Multi-agent problems also use Q-learning in a hierarchical approach, where it is used to govern the pursuit aspects of the agents' motion [81, 82].

2.5.2 Adaptive Fuzzy Reinforcement Learning

Adaptive fuzzy reinforcement learning has been used in pursuit-evasion schemes to learn optimal strategies through repeated plays of the game. A current implementation of an adaptive fuzzy controller applied to pursuit-evasion scenarios is given by Givigi et al. [52, 53] in solving the Game of Two Cars [64, 87]. There are three main steps to the fuzzy control scheme: 1.) predict the value function, 2.) control the vehicle, and 3.) update the value function approximation parameters. In this case, the parameters are consequents of the fuzzy logic approximator. The block diagram of the adaptive fuzzy approach is shown in Fig. 2.4.

The scheme works by approximating Eqn. 2.49 using fuzzy logic approximators. In the

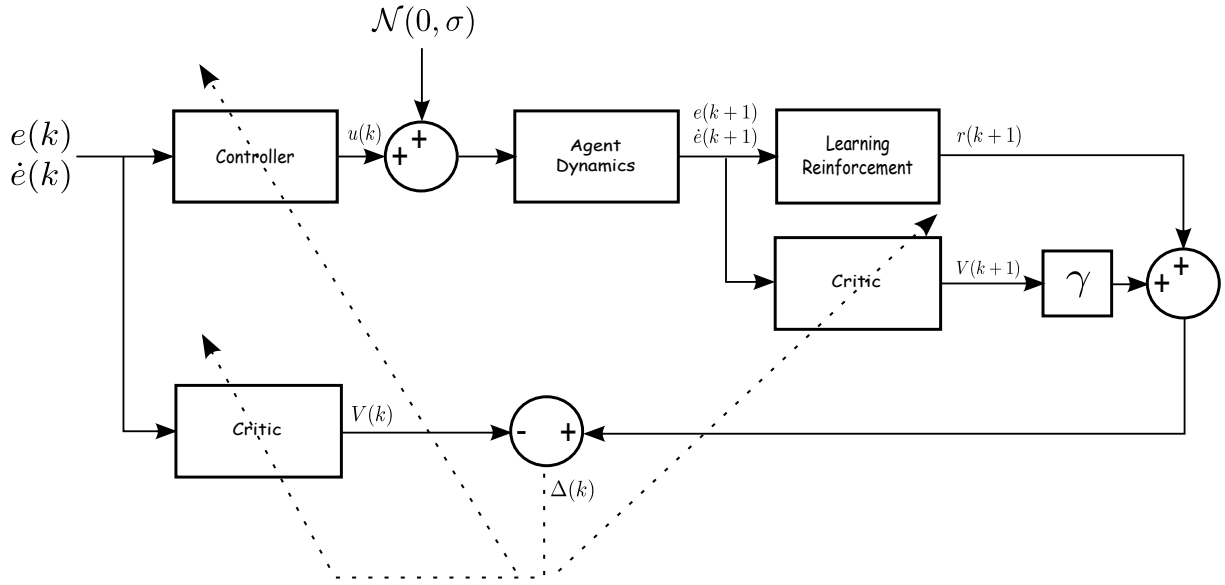


Figure 2.4: Block diagram for the adaptive fuzzy reinforcement learning scheme for approximate dynamic programming.

block diagram, the critic acts as a fuzzy approximator of the value function. Using the state error measurements as an input, it outputs the associated value for both the predicted, V_{k+1}, and measured, V_k, values. The choice of control is evaluated using learning reinforcement which assigns a penalty or reward for the error predicted as a result of the control. This function, $r(k+1)$, which may also be created using fuzzy logic, is intended to provide a running cost similar to $g(x,t)$ based on the error of the system from the desired state. The discount variable is γ, and the noise, $\mathcal{N}(0,\sigma)$, is assumed to be added to the control with standard deviation σ. The controller is also a fuzzy logic function that produces a control value based on the error. All of the consequents of the fuzzy logic critics and controllers are updated until the temporal difference reaches 0, and Eqn. 2.49 is satisfied. Using the error term as an input guides the agent to a desired state configuration, such as maintaining the same heading as the opposing agent.

An example of learning occurring using the methods in [53] is given in Fig. 2.5 where many iterations are required for the solution to converge for the pursuer in the Game of Two Cars.

The individual trajectories are shown where the lighter colors are the early attempts to catch the evader and the darker colors occur after training. The advantage of this scheme is that

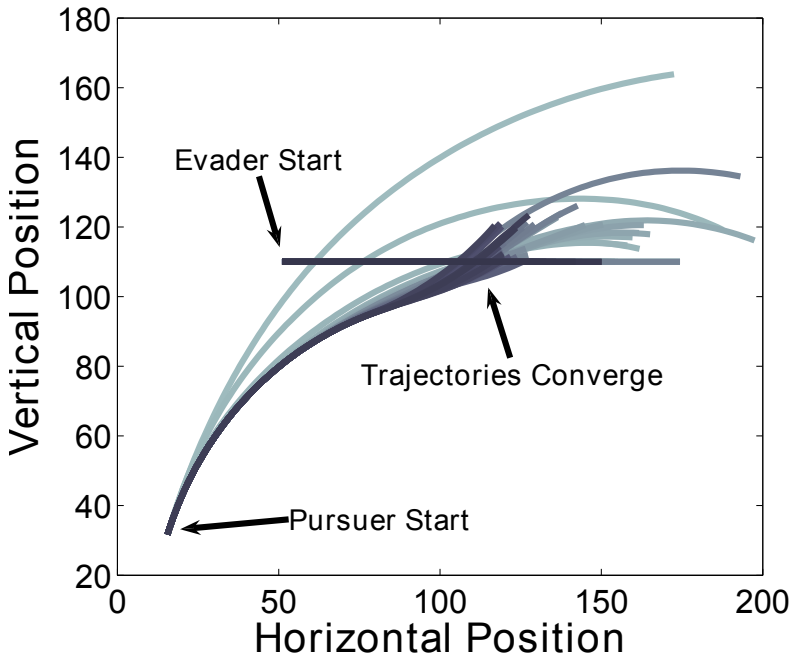


Figure 2.5: Convergence of the optimal strategy using an adaptive fuzzy framework.

little information is needed about either agent to arrive at the optimal control solution using the infinite horizon, discounted HJB equation. The problem, however, is that the game must be played repeatedly, as shown in Fig. 2.5, for the solution to be reached. Training can be conducted off-line, but more information of the opposing agent will be required. This dissertation seeks to find methods that solve the HJI equation quickly without needing training before the solution can be used.

2.6 Conclusions

In this chapter, the ideas and solution methods leading to controls based on a differential game theoretic framework are discussed. Differential games are a combination of both static

game theory and optimal controls. Current solution techniques range from using a repeated static game solution for a more probabilistic approach to solving a combinatorial problem of all the possible controls. Advances have been made, particularly in the area of dynamic programming, toward simplifying the control solution. However, the problem of providing a control law that is fast to calculate while maintaining target reachability guarantees is still a topic of ongoing research. The following chapters discuss further progress toward this end by combining finite horizon methods with the HJB equation to develop a partitioning scheme to derive control laws for an agent involved in two-player pursuit evasion. Namely, the convergence results of the HJB solutions presented in this chapter are exploited over regions of the state space to provide a faster control law realization.

Chapter 3

Finite Horizon Minimization

This chapter focuses on applying a control that minimizes the distance between the pursuer and evader with a finite horizon control law. The evader is rewarded for actions that increase the distance between the agents and is punished for actions that increase its noise output. The main result from this work is the outcome that results in a limit cycle trajectory that never reaches the target. This reinforces the notion that an infinite horizon is needed to guarantee that the target set is reached.

©2010AACC. Reprinted with permission, from Brian Goode, Andrew Kurdila, Michael Roan, "Pursuit-evasion with Acoustic Sensing using One Step Nash Equilibria", in IEEE American Control Conference, June 2010.

3.1 Introduction

Pursuit-evasion in the context of differential games is rooted in the work of Rufus Isaacs in the 1960s, who solved the classic Homicidal Chauffeur Game [64]. In this scenario, a faster car has the intention to catch his slower, but more agile human counterpart in the minimal amount of time. In his model, perfect information and simplified dynamics lead to an easily obtainable solution. However, complex dynamics and sensing limitations have led

researchers to consider more advanced solution techniques.

3.1.1 Evolution of Pursuit-Evasion Games

Following the Homicidal Chauffeur, several other games of interest have emerged. Both Isaacs and Merz [64, 87] studied the Game of Two Cars where the Evader is no longer able to turn on point, but is also given a limited turning radius. Later, Patsko and Turova gave a numerical solution technique for the Acoustic Homicidal Chauffeur, where the Evaders speed is a function of the distance to the Pursuer [105]. Here, the idea is to see what happens when an Evader slows down to avoid detection, but the assumption of perfect information is still present and the Evader has no direct control over his sound output.

Very recent work has emerged from Brooks whose model incorporates acceleration into the dynamics of the vehicles from the Game of Two Cars [19]. The vehicles obtain state knowledge with an imperfect information observation model and give the Evader the option to use countermeasures to disrupt the Pursuer's sensing ability. The solution method of this paper solves for the Nash equilibrium of a zero sum game at each time step over a one step horizon. A one step horizon means that the controls for both the Pursuer and Evader are locally optimal with respect to the Nash equilibrium, but there are no guarantees for optimality over the whole horizon. One very interesting result to come from this work is that the Pursuer is better off assuming a cost function that rewards actions based on perfect information.

In this paper, we use the vehicle dynamics of [19], i.e. both agents have acceleration and a turning radius and incorporate them in an acoustic sensing model for the Pursuer's observations. Our specific contributions include throttling the acoustic sound injected into the sensing model as a function of Evader acceleration and speed. We separate the two sources of sound for the Evader because sound generated from propulsion is often a function

of acceleration, and noise from fluid flow such as wind is a function of speed. Thus, we seek behaviors of the Evader that reduce his noise output and detectability, yet still allow him to escape. Analogously, the ability to throttle the sound output can be thought of as a continuous countermeasure.

3.1.2 Acoustic One Step Nash Equilibrium Solution

To model the joint behavior of the agents, we study the one step Nash equilibrium solution proposed by Hespanha [61]. This work provides an elegant solution to the case where the uncertainty in the agents observations creates a non-zero-sum game from a cost structure that would otherwise be zero-sum with perfect information. The formulation provides a solution technique for the case of nested information structures. The non-zero-sum game formed from the cost functions of both agents becomes a zero-sum matrix game at each time step with this information assumption.

Our implementation of this nested information stochastic model assumes that the Evader is aware of the Pursuer's states. We allow this assumption because we model a Pursuer that is dynamically superior and not dependent upon stealth to achieve capture.

3.2 Problem Formulation

We now present the dynamic equations governing the motion of the vehicles. This section begins by introducing the differential equations of motion, and then proceeds to describe how these are used to form the transition probabilities from one state to the next.

3.2.1 Continuous Dynamics

The motion of each vehicle is represented by four states: horizontal position, y_1, vertical position, y_2, speed, y_3, and absolute bearing, y_4. It is understood that these states evolve in time following

$$\dot{y}_1(t) = y_3 \sin(y_4), \tag{3.1}$$

$$\dot{y}_2(t) = y_3 \cos(y_4), \tag{3.2}$$

$$\dot{y}_3(t) = -cy_3 + \alpha u_1, \tag{3.3}$$

$$\dot{y}_4(t) = \frac{y_3}{R} u_2, \tag{3.4}$$

where these equations are subject to noise. The coefficient, c, describes dissipation effects, α is the maximum possible acceleration, R is the minimal turning radius associated with the vehicle, and the control inputs u_1,u_2, take values

$$\begin{aligned} u_1 &\in [0, 1] \\ u_2 &\in [-1, 1] \end{aligned} \tag{3.5}$$

In a physical sense, the dissipation coefficient may represent the drag or internal friction of the vehicle. It serves to limit the maximum obtainable speed. Both the acceleration and turning radius coefficients are bounds on the possible motion, meaning the vehicle can obtain no higher performance than these coefficients. For ease of notation, we will define the evolution operator

$$Y_{i,u}(y_0, t, C) : \mathbb{R}^4 \times \mathbb{R}^+ \times \mathbb{R}^3 \to \mathbb{R}^4, \tag{3.6}$$

to represent the solution of (1)-(4) for the states at time, t, given the initial state, y_0, and performance coefficients, C, for agent, i, and control vector, u. There are many well known methods to approximate this operator. We chose the forward Euler method for this work.

3.2.2 Discretized State Space

Next, we proceed to establish a finite and discrete state space on which we construct probability transitions from one state to the next for one time step. First, we define for each dynamic state an integer set

$$\begin{aligned} x_1 &\in \{1, 2, 3, , n_{x_1}\}, \\ x_2 &\in \{1, 2, 3, , n_{x_2}\}, \\ x_3 &\in \{1, 2, 3, , n_{x_3}\}, \\ x_4 &\in \{1, 2, 3, , n_{x_4}\}, \end{aligned} \tag{3.7}$$

where n_{x_1} is the number of nodes in the discrete set for x_1, and likewise for all the other states. We can now define the vector of the vehicle states,

$$\mathbf{x}_i := \{x_1, x_2, x_3, x_4\}, \tag{3.8}$$

that defines a position on the 4-dimensional discrete lattice that encodes a particular horizontal position, vertical position, speed, and bearing for vehicle, i. For every possible state combination of x_i, there is a corresponding integer on the set

$$s_i \in S := \{1, 2, 3, \ldots, n_s\}, \tag{3.9}$$

where $n_s = \prod_{p=1}^{4} n_{x_p}$. Each agent transitions from the state at time, t, to the next state at time, $t+1$, incremented by Δt, with probability

$$\begin{aligned} &Pr_i(s_i(t+1) = \\ &\quad s_i' s_i(t) = s_i, u(t) = u) = p_i(s_i, s_i', u), \end{aligned} \tag{3.10}$$

which is connected to the differential equations by

$$p_i(s_i, s_i', u) = \mathcal{O}_i\left(s_i, s_i', Y_{i,u}(y_0, \Delta t, C)\right), \tag{3.11}$$

where the function, $\mathcal{O}_i$, is a function that transforms s_i, into its unique counterpart on $\mathbb{R}^4$ to form y_0 and assigns a probability to the trajectory reaching $y(\Delta t)$ that corresponds to s_i'. All of the equations presented in this section apply to the evolution of an individual agent.

3.2.3 Definition of the Pursuer and Evader

We now express the formulation for two agents, the Pursuer and the Evader, consisting of "teams" of individual agents. This allows us to use two player notation for the remainder of the paper, yet still remain applicable to any number of agents. In this sense, we define r as the set of all i agents that are classified as a Pursuer. Likewise, we define v to correspond to all agents classified as an Evader. We may now define the control sets for the Pursuer team as

$$u := \{u_1, u_2, \ldots, u_{n_r}\}, \tag{3.12}$$

where n_r is the number of pursuing agents and u is composed of the individual controls for each vehicle in the set, r. Similarly, the control, d, can be formed for the Evader team consisting of n_v pursuing agents.

The state transition probability of the entire agent force is

$$p(s, s', u, d) := \\ Pr(s(t+1) = s's(t) = s, u(t) = u, d(t) = d), \tag{3.13}$$

given

$$p(s, s', u, d) = p(s_p, s'_p, u)p(s_e, s'_e, d), \tag{3.14}$$

where s_p and s_e are collections of the integer states for the Pursuer and Evader teams respectively. Using the conditional independence of all the agents we may then define for the Pursuer

$$p(s_p, s'_p, u) = \prod_{\forall i \in r} p_i(s_i, s'_i, u_i), \tag{3.15}$$

where $p(s_e, s'_e, d)$ for the Evader is realized in the same way.

3.2.4 Sensing and Detection Model

The Pursuer determines the Evader's position based on the sound output from the Evader. This sensing ability is compromised when the Evader is quiet or the Pursuer is loud. We model the Evader sound output as a function of both acceleration and speed. We assume that the Pursuer is capable of filtering out its propulsive noise from its measurements, so only the exogenous noises are considered in the Pursuer sound output. Consequently, we

model sound output as having the functional form for vehicle, i.

$$S_i(t) = W_i(x_{i,3}(t)) + E_i(d_{i,1}(t)), \tag{3.16}$$

where $E_i : d_{i,1} \to \mathbb{R}$ is the sound output due to propulsion and $W_i : y_{i,3} \to \mathbb{R}$ is sound output due to vehicle speed. These functions can either be measured or estimated. For $i \in r, E_i(\cdot) = 0$.

Because we are interested in the sound present at the Pursuer, we must model the transmission loss that occurs between the Pursuer and Evader. This is a functional,

$$TL : S_i(t) \times EV(x) \times x \nrightarrow S'(x,t), \tag{3.17}$$

that maps the sound output of a specified agent in an environment described spatially by $EV(x)$ for some position, $x \in \mathbb{R}^2$, in the environment. This transmission loss may be measured, found using numerical modeling, i.e [101], or assumed to follow $\frac{1}{d^2}$ intensity decay in a homogeneous medium with spherical propagation [77].

$$P_{TP} = \min\{1, C_D d^{-\gamma}\}, \tag{3.18}$$

where C_D is a constant, d is the euclidean distance between between the individual Pursuer and Evader. The value $\gamma = 2$ corresponds to inverse squared transmission loss in a homogeneous medium. However, because C_D is a constant, this occurs for a constant sound level of the Evader. Because our formulation models varying sound levels, we capture this physics by replacing d with:

$$d_{EFF} = \sqrt{\frac{d^2}{P_{OUT}} P_{REF}}, \tag{3.19}$$

where the term P_{OUT} is the normalized sound output,

$$\frac{S_i(t)}{\max\{S_i\}}, i \in v \tag{3.20}$$

of the Evader. P_{REF} is the reference sound intensity. Because P_{OUT} is bounded on the interval, $(0, 1]$, P_{REF} is also bounded on $(0, 1]$. Essentially, d_{EFF} converts a change in the sound level to an equivalent change in the distance variable in the true probability calculation. The same effect may be achieved by calculating C_D for various sound intensities, but as a matter of convenience, the effective distance will be used in our game termination criteria.

The Pursuer generates noise with increased motion, so the noise surrounding his vehicle reduces his ability to decide if he is, in fact, detecting an Evader. We model the probability of a false positive as a function of the Pursuer's speed,

$$P_{FP} = \min\{1, C_{FP} + N(x_{i,3})\}, \tag{3.21}$$

where C_{FP} is a constant representing the false positive rate when the Pursuer speed is 0. It is the false positive rate caused by non-Pursuer generated noise in the environment. The term $N(x_{i,3})$ represents the Pursuer speed with an increase in the false positive rate. We define the sensing region of the Pursuer as a function of P_{FP}:

$$R_{SENSE} = \left(\frac{C_R}{P_{FP}}\right)^{\frac{1}{\gamma}}, \tag{3.22}$$

where the Evader is detected if located within the ball of this radius. C_R is a constant used to vary the effect of the false positive on the Pursuer's detectability.

3.3 Game Formulation

In this section, we present how the dynamic and sensing models are incorporated in to a differential game framework. We begin by discussing the game over states and observations. Then we show how these are formed into cost functions and solved for the Nash equilibrium at every time step.

3.3.1 Game Over Status

The game is terminated when either the Pursuer or Evader wins. For this paper we say that the Pursuer wins when all Evaders have been captured. The Evader wins when all Evaders have escaped the sensing ranges of the Pursuers. We can now define two sets

$$\begin{aligned} s_{OVER,P} &\subset S | d_{EFF} < Target \\ s_{OVER,E} &\subset S | d_{EFF} > R_{SENSE} \end{aligned} \tag{3.23}$$

To make the game well-posed, we must append to the previously derived transition probabilities, the condition that if $s \in s_{OVER,P} \cup s_{OVER,E}$:

$$p(s, s', u, d) = \begin{cases} 1, if\, s' \in s_{OVER,P} \cup s_{OVER,E} \\ 0, otherwise \end{cases} \tag{3.24}$$

so that it is impossible for a state to exit a game over state.

3.3.2 Observations

The Pursuer and Evader make observations of each other's position(s) at every time step. These observations are realized using the detection model. We assume a nested information structure where the Pursuer must estimate the states of the game and the Evader has access to the Pursuer's knowledge. In this case the Pursuer is not capable of using stealth as a tactic. The agents do not maintain a history, so for the Pursuer we have the observation set

$$Y := \{s_p, e\}, \tag{3.25}$$

where $e \in S$, is the Pursuer's estimate of the Evaders state. We give the nested information structure of the Evader as

$$Z := \{s_e, Y\}, \tag{3.26}$$

and note that each agent has perfect knowledge of their states.

3.3.3 Cost Functions

Following [61], we are able to transform our non-zero sum game into an equivalent zero-sum game because we have nested information structures. The cost of the game is given as

$$J(p,q) = \mathcal{W}^T \times \begin{bmatrix} \sum_{s' \in s_{OVER,P}} p(s,s',u,d) \\ \sum_{s' \notin s_{OVER,E}} p(s,s',u,d) \\ \sum_{v,r} P_{TP}(u_v, x_{v,3}, d_{v,r}) \\ \sum_v 1 - P_{FP}(x_{r,3}) \end{bmatrix}, \tag{3.27}$$

where $\mathcal{W} \in \mathbb{R}^4$, is a weighting vector for the costs and (p, q) are pure policy distributions for the controls (u, d) respectively. In this cost structure, the Pursuer seeks to maximize the probability of winning, the probability that the Evader loses, the probability of detecting the Evader being a true positive and the probability of not detecting the Evader being a true negative. Contrarily, the Evader seeks to minimize this cost. There exist many standard linear programming algorithms to find the solution to this zero-sum game.

3.3.4 Game Evolution with One Step Nash Equilibrium

To simulate the game, we concern ourselves to finding the mixed strategies p^*, q^* over which the pure policies for the controls u, d should be played. A pure policy is a distribution over the control set where the probability of playing a particular element of the set is 1. We consider only a one step horizon, in which we only look ahead to the time, $t + 1$. The game evolves with the following algorithm:

1. SET initial conditions, performance coefficients
2. GAME OVER? PROCEED, if no. STOP, if yes.
3. COMPUTE Value matrix for all combinations, p, q.
4. COMPUTE p^*, q^* using linear programming [6].
5. EVALUATE p^*, q^* to realize controls u, d.
6. EVOLVE states with controls, u, d.
7. REPEAT, go to step 2.

3.4 Example with results

In the following example, we show how the model is able to capture the physics of acoustic sensing by solving the one step Nash equilibrium solution at every time step. In particular, we show by changing the parameters describing the vehicle's performance, the outcome of the game will result in an Evader escape, Pursuer capture, or neither. The first two are an expected result, but the latter case is special in that if neither agent wins, yet, both still agree on the Nash equilibrium, we begin to question their rationality. It is conceivable that either agent might see gains by deviating from the Nash equilibrium and attempting another strategy to secure a win.

3.4.1 Game Parameters

To illustrate the functionality of the physics behind the model, we consider a simple case with one Pursuer and one Evader. Because our focus is on the manipulation of the sound field through acceleration and speed, we form a game where the Pursuer and Evader chase each other along a straight line ($x_{0(1,4)} = x_{0(2,4)}$) with no ability to turn ($u_2 = 0, d_2 = 0$). Although seemingly over restrictive, there are many pursuit games that converge to this type of play at the last phase of the game.

We assign to the sound output of the Evader a linear increase in the sound output based on vehicle acceleration and speed. These equations are

$$W_i(x_{i,3}(t)) = \epsilon x_{i,3}, \tag{3.28}$$

$$E_i(d_{i,1}(t)) = \eta d_{i,1}, \tag{3.29}$$

where in this game we set both coefficients, ϵ and η to unity. We can now assign the sound

output reference as unity which corresponds to the maximum sound output by the Evader. From the sound output, we can construct a mapping of true distance, vehicle speed, and acceleration to the effective distance. This is visualized in Fig. 3.1 where the red color map corresponds to the pure strategy of no acceleration and the blue color map corresponds to the pure strategy of full acceleration. The darker the colors in the maps represent less vehicle speed. Immediately, we can see how the Evader could cease acceleration, and even while moving at a decent speed, enact a very quick and substantial increase in the effective distance.

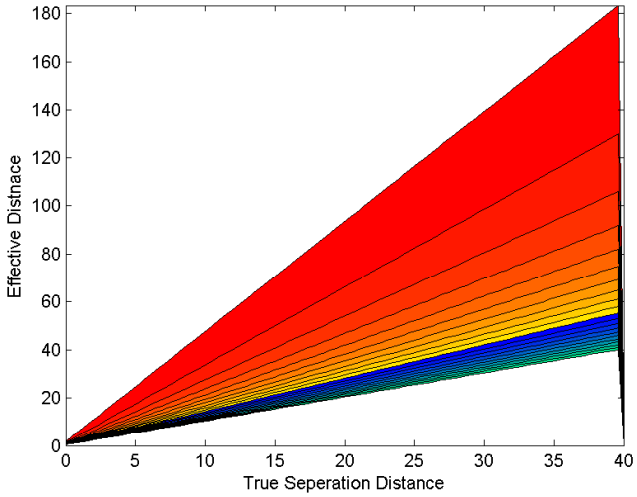


Figure 3.1: The effective distance as a function of true distance, acceleration, and vehicle speed.

For sensing, we assign $C_D = 50$ and $C_R = 810$. In this example, these values ensure that the Evader is sufficiently far away to escape. For the probability of a false positive we assign the ambient noise characteristic as $C_{FP} = 0.1$ and encode the Pursuer speed contribution as $\beta = 0.9$.

For the purposes of illustrating the behaviors associated with the vehicles, we restrict the vehicle observations to the case where

$$P(Y,s) = \begin{cases} 1 \; if \, d_{EFF} < R_{SENSE} \text{ and } Y = s \\ 0 \; otherwise \end{cases} \tag{3.30}$$

$$P(Z,s) = \begin{cases} 1 \; if \, Z = s \\ 0 \; otherwise \end{cases} \tag{3.31}$$

This observation formulation states that while the Evader is in the bounds of R_{SENSE}, he will be noticed by the Pursuer. However, the cost function will still cause the agents to act such that there is a minimal (maximal) chance that the Evader will escape outside the sensing region.

For every game, we start the agents at an initial distance of 20 and a speed of 0.35. We take as pure strategies, p and q such that the controls u, d take values in the set $\{0, 1\}$. The simulation continues until the Pursuer wins, Evader wins, or it is apparent that the game has entered into a draw.

3.4.2 Results

In our analysis, we ran an array of simulations where the Evader acceleration parameter took values between 0 and 4. The Pursuer acceleration parameter remained at a constant value of 1.5. Both dissipation coefficients were held constant at 1. The cost weighting matrix was set to $\mathcal{W} = \begin{bmatrix} 10 & 10 & 1 & 1 \end{bmatrix}^T$, which given the sound field parameters and agent dynamics corresponds to a "close in and detect" strategy by the Pursuer. This means that the Pursuer gains more value by decreasing the distance to a suspected target before slowing down to ensure that it is not a false positive. Likewise, the weighting matrix results in an "accelerate

and coast" strategy on the Evader, who gains more value by attempting to flee the Pursuer by increasing the true distance, and then coasting to increase the effective distance with a reduced sound output.

As can be predicted, when α_e is sufficiently lower than α_p, the Evader is captured as shown in Fig. 3.2A. However, in Fig. 3.2B we see an interesting trend in the velocities that shows signs of a possible cycle forming. The reason for this is apparent in Fig. 3.2C where we see that the Pursuer first closes in on the Evader by maximizing the probability of a true positive and then decreases speed to increase the probability of a true negative. The Evader, after an initial increase in speed, slows down to avoid detection, but then begins to speed up as the Pursuer slows. The Evader's effort is thwarted, because even at a reduced speed, the Pursuer is still able to overtake the Evader.

As α_e is increased, the distance between the Pursuer and Evader in Fig. 3.2D does not approach a capture state. Rather, it oscillates about a mean value. We can see where the control signals form a periodic pattern that coincides with the rate of change in distance. The sudden deviations in the control signal are a byproduct of our control configuration where we give the agents only full acceleration or none at all. Fig. 3.2E shows where the periodic nature of the controls forms a cyclic speed pattern. This differs from the previous case, because the Evader now has enough acceleration to consistently increase the separation distance every time the Pursuer slows to increase its detection ability. Therefore, the Pursuer must increase speed to prevent the Evader from escaping too far, and the cycle repeats.

Apart from the limit cycle formed by the vehicle speeds, we also see in Fig. 3.2F the individual probability costs obtain a pattern in time as well. The Pursuer tries repeatedly, and unsuccessfully, to increase P_{TN}. This is important to note because it implies that for this case, the Nash equilibrium is unable to converge to a solution, be it a cycle or single value, that guarantees a win for either agent. A better alternative may be for the Pursuer to learn

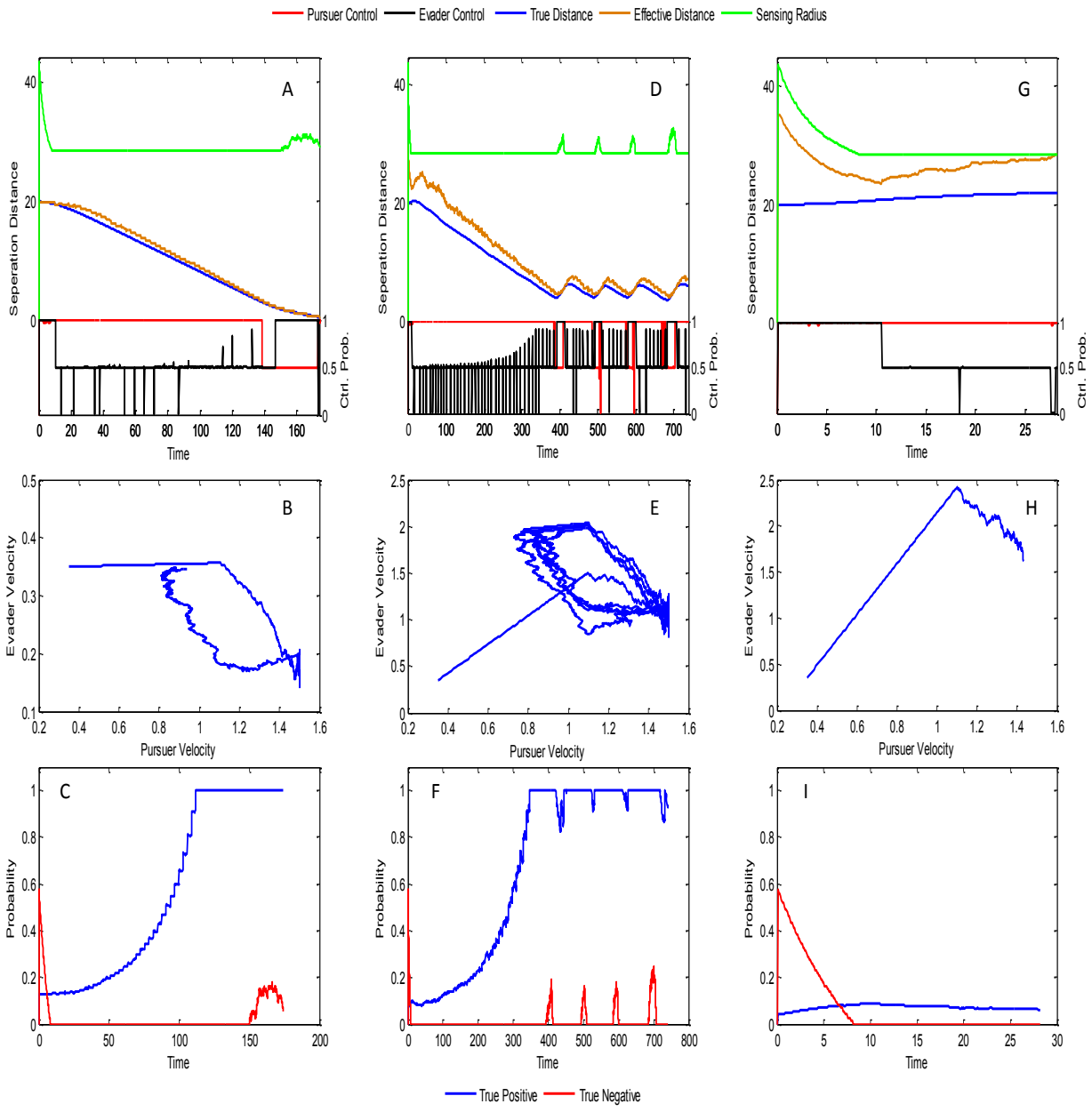


Figure 3.2: Simulation results as α_e is increased. Left column: $\alpha_e = 0.36$. Middle column: $\alpha_e = 2.12$. Right column: $\alpha_e = 3.52$.

the behavior of the Evader, and exploit it by temporarily ignoring P_{TN} and advance heavily upon its suspected target. We may also consider a situation where learning on the part of an agent can lead to a self adjustment to the weights in the matrix, $\mathcal{W}$, in a non-zero-sum application.

The last case we observe is shown in Fig. 3.2G, where α_e is large enough to out run the Pursuer. Interestingly, the Evader does not utilize his full acceleration for the full extent of the game. We see in Fig. 3.2H, that the speed phase diagram begins in the same fashion as the previous cases where, after an initial attempt to flee the Pursuer at full speed, the Evader slows down to corrupt the Pursuers probability of a true positive. This is evident in Fig. 3.2I, where the time track of the P_{TP} never climbs to 1 as in the previous cases. The effect of this trend to slow down is seen in Fig. 3.2G where the effective distance initially decreases despite an increase in the true distance because of increased sound output. By slowing down, the Evader obtains a greater increase in the effective distance per unit time than in true distance. Thus, he is able to escape the Pursuers sensing by forcing the effective distance over the sensing threshold.

3.5 Discussion and Conclusion

We are able to capture the dynamics of two vehicles moving in an environment where the Evader minimizes detection by using stealthy tactics to manipulate an acoustic field. We currently simulate the scenario over a one time step horizon using a zero-sum Nash equilibrium game. When the weighting matrix is such that the Pursuer engages in a "close in and detect" strategy and the Evader engages in an "accelerate and coast" strategy, a small change in the speed dynamics of the agents can result in a different outcome of the game. Particularly, we note that when the speeds of the Pursuer and Evader are close, a limit cycle

forms in the controls (speeds) that does not result in a win for either agent.

This is interesting for a number of reasons. First, these findings support observations that the Nash equilibrium of this game may not be in the best interest of either agent. For example, if caught in a limit cycle and the Pursuer took a chance and ignored detectability for an instant, the game would end in capture. However, this also opens a question as to the role of the cost function assigned to each agent. It may be argued that the cost function assigned to the agents in this game, although in spirit represents the agent's desired goals, is responsible for not leading to the proper termination. In this case, both agents are thought as taking the best actions with the Nash equilibrium, but were in effect ignorant due to the reward they were given. If true, then the zero sum nature of this game may not manifest as a direct distance or sensing correlation, but a much more complex relationship partitioned by the state space.

The result of this chapter is that another layer of complexity is needed to guarantee that satisfying the finite horizon objective functions allows the agent to reach the target set. One step toward this solution is given in the following chapter where multiple finite horizon objectives are controlled using an adaptive fuzzy controller to succeed in target reachability.

Chapter 4

Adaptive Fuzzy Solution

The work in this chapter seeks to combine the use of finite horizon controls with an infinite horizon guarantee of the pursuer reaching its target, the evader. Approximate dynamic programming is used to select the proper finite horizon control to use. In particular, an adaptive fuzzy logic framework with reinforcement learning is used to learn the movement of the opposing agent for a specific pursuit-evasion scenario.

©2010IEEE. Reprinted, with permission, from Brian Goode, Andrew Kurdila, Michael Roan, "Adaptive Fuzzy Control of Switched Objective Functions in Pursuit-Evasion Scenarios", in IEEE Conference on Decision and Control, December 2010.

4.1 Introduction

Pursuit-evasion scenarios are zero-sum systems in which one party, the pursuer, seeks actions that guide the state of the system to a predefined target set that designates a "win" for that agent. Likewise, the evader seeks to steer the system to its own target set. In general, this can be implemented as a tag-chase scenario where the pursuer's target set is determined by entering some proximity around the evader. However, alternative target sets may incorporate

stealth tactics [105].

When evaluating these scenarios, the solution consists of finding the control strategy for each agent such that an objective function is extremized. Many solution techniques exist for obtaining these control strategies, but there is no guarantee that the objective function used in the solution will perform well in a given scenario. In [55], we see a simple example where an agent is pursuing another along a straight line using acoustic sensing as a means of gathering information. The authors propose an objective that combines minimizing the distance between the players and the probability of a false positive detection with a weighted summation. However, due to the objective function structure, the pursuing agent with superior speed does not always succeed. Under certain weights, the dynamics will actually enter into a limit cycle.

Thus, we seek to the find a controller that will adapt to a potential adversary using adaptive learning schemes while making few assumptions on the dynamic capabilities of the opposing agent. A current implementation of adaptive fuzzy controllers applied to pursuit-evasion scenarios is given in [53]. This work provides a very good foundation on which the controller of one agent can be adapted using methods from [118]. The scenario presented is a direct pursuit-evasion scenario of the game of two cars where one car has a tighter turning radius and the other is faster [87]. The objective of the game is to minimize the time to capture. The authors present an adaptive fuzzy scheme, that over several iterations converges to the time optimal solution using reinforcement learning. However, limitations are placed on the bearing the two agents must maintain throughout the game. One of the reasons for this is that the time-optimal solution is approximated by simplified rewards of separation distance and the time derivative of separation distance. These assumptions work well when the pursuing agent is able to make a turn and pursue the evader directly; however, if the initial conditions or exploration noise in the control signal place the states such that the

pursuing agent must first increase the distance between the evader before closing in the correct "swerve" action will actually be penalized by the adaptive system.

Rather than the fuzzy controller choosing actions specifically, the adaptive scheme in this paper builds on the previous work by restructuring the reinforcement learning to adapt to the best strategies which correlate to choosing certain objective functions. The set of possible objective functions are based around the agent's capabilities which can include speed, stealth, etc. The role of the fuzzy controller becomes deciding how the objective partitions are to be assigned throughout the state space. The strategies emerge as potential solutions to situations when one agent's capabilities are compromised by the opposing agent's actions. This paper proceeds by giving an overview of switched objective functions and proceeds to give the necessary background needed to construct adaptive fuzzy controllers from reinforcement learning techniques. Then, we provide the construction of the adaptive fuzzy controller for the strategy switching function. We show that the adaptive fuzzy scheme converges to approximately the same solution as Alg. 3 in Sec. 4.2.1. Finally, we give an example of the controller using the Homicidal Chauffeur scenario.

4.2 Background

The controller architecture in this paper is the synthesis of switched objective systems and adaptive fuzzy controllers. Both topics are briefly reviewed in this section.

4.2.1 Switched Objective Function Systems

We seek a structured partition of the state space where a forward propagating reward is assigned to each partition to achieve the same result as a backward propagating objective

by utilizing an intelligent switching function. Agent i chooses a control, $u_i^* \in U_i$, given the other agents' ($I \neq i$) controls, $\tilde{u}_i \in \tilde{U}_i$, and a distribution, b_i, over $\tilde{U}_i$ following

$$u_i^* = \underset{u_i^* \in U_i}{\arg\min} \left[\sum_{\tilde{u}_i \in \tilde{U}_i} b_i(x, t, u_i, \tilde{u}_i) V_i(x, t, u_i, \tilde{u}_i) \right] \tag{4.1}$$

Here, we give $V_i(x, t, u_i, \tilde{u}_i)$ the switched objective form

$$V_i(x, t, u_i, \tilde{u}_i) = \begin{cases} R_0(\mathcal{F}(x, t, u_i, \tilde{u}_i)), \textit{if } x \in P_0 \\ R_1(\mathcal{F}(x, t, u_i, \tilde{u}_i)), \textit{if } x \in P_1 \\ R_2(\mathcal{F}(x, t, u_i, \tilde{u}_i)), \textit{if } x \in P_2 \\ \vdots \\ R_Q(\mathcal{F}(x, t, u_i, \tilde{u}_i)), \textit{if } x \in P_Q \end{cases} \tag{4.2}$$

where objective q is defined with the form $R_q(x) := X \to \mathbb{R}$, $P_q \subset X$, and $\mathcal{F}$ is a temporal evolution operator

$$\mathcal{F} := X \times T \times U_i \times \tilde{U}_i \to X \tag{4.3}$$

that predicts the next state in the system given the current state, time, and control actions of all agents.

We now address the task of developing the Q switched objectives, R_q, and partitions, P_q, that guide the agent. The only restriction we place on the set of R_q is that there must exist one primary objective that guides the agent to its target set, $\mathcal{T}_i$.

Definition 1. *Let the primary objective function be a strictly minimizing reward,* $R_q(x) : X \to \mathbb{R}$*, where* $R_q(x)$ *satisfies:*

1. $R_q(x) > 0, \forall x \notin \mathcal{T}_i$,

2. $R_q(x) = 0, \forall x \in \mathcal{T}_i$, *and*

3. $R_q(x)$ *increases with* $\inf \|x - \mathcal{T}_i\|_2$.

The first two criteria give the objective Lyapunov-like qualities [69]. The third criterion is present because the agent uses $R_q(x)$ as a measure of how close the current state is to $\mathcal{T}_i$. The significance of this particular type of reward metric is that it forms the basis for how well the agent is succeeding in traversing to the target state. In most cases, this reward may be the only metric needed by the agent to make decisions. However, finite horizons of forward objectives may not be possible for an agent to apply unilaterally throughout the entire state space. Therefore, partitioning and multiple rewards are appended. To complete the construction, we use the following algorithm from [56].

Algorithm 3. *Constructing the switched reward system proceeds as follows:*

1. *Set* $\Psi^q = \mathcal{T}_i$, $\tilde{X} = X$ *and* $q = 1$.

2. *Select a reward metric,* R_q, *to be applied to* $\tilde{X}$. *If* $q = 1$, R_q *must satisfy Def. 1. For* $q > 1$, *the reward can be any appropriate reward metric.*

3. *Identify the states,* Ψ^q, *where the agent is able to navigate to* Ψ^{q-1} *using* R_q.

4. *Form the set* $\Psi^{q+1} = \Psi^q \cup \Psi^{q-1}$.

5. *Set* $\tilde{X} = x \notin \Psi^{q+1}$ *and* $q = q + 1$.

6. *Repeat with a new reward until* $\Psi^{q+1} = \Psi^q$ *or* $\tilde{X} = \emptyset$.

Essentially, this construction algorithm finds the largest possible set of initial states, Ψ^q for the agent that will result in the state trajectory terminating in $\mathcal{T}_i$. Because it begins by finding the states that lead to the target set given the primary objective, R_1, then all

subsequent partitions that lead to partition P_1 also terminate at the target set. Due to this algorithm requiring previous knowledge of the opponent(s), we will now present how this can be implemented on-line using an adaptive reinforcement learning algorithm.

4.2.2 Adaptive Fuzzy Controller with Reinforcement Learning

Reinforcement learning occurs when an agent adapts its strategy in response to stimuli present in the environment. Those outcomes which are seen as a positive influence given the predefined reinforcement metric are encouraged in future iterations of play [47]. This method is particularly applicable to pursuit-evasion because nothing is assumed about the dynamics of the opponent. Rather, the derived control is learned purely from the output of the reinforcement function.

Likewise, a fuzzy controller is useful because if a priori knowledge of the system is known, it may be encoded into the controller using linguistics [124]. As more knowledge of the dynamics is learned, the individual rules may be adjusted on-line. The RL-fuzzy controllers from [53, 118] are of the form shown in Fig. 4.1. There are four major parts to the reinforcement system. The control is a fuzzy logic controller that takes the error input and converts it to a control signal (in our formulation this occurs over two blocks). The reinforcement block is the reward given to the outcome of the actions taken by the agent, and the critic predicts the expected value of the action in future iterations. The history of actions considered is determined by a discount factor. The plant represents the actual dynamics of the system which does not need to be known by the controller or critic. In general, the system works by comparing the expected reward at the current step to the predicted reward at the future step and is known as a temporal difference. If there is a difference between the predicted and realized rewards, both the controller and critic are updated to reflect the changes. In

order for there to be a stable convergence of the system to the appropriate control signals, the critic must adapt at a faster rate than the controller.

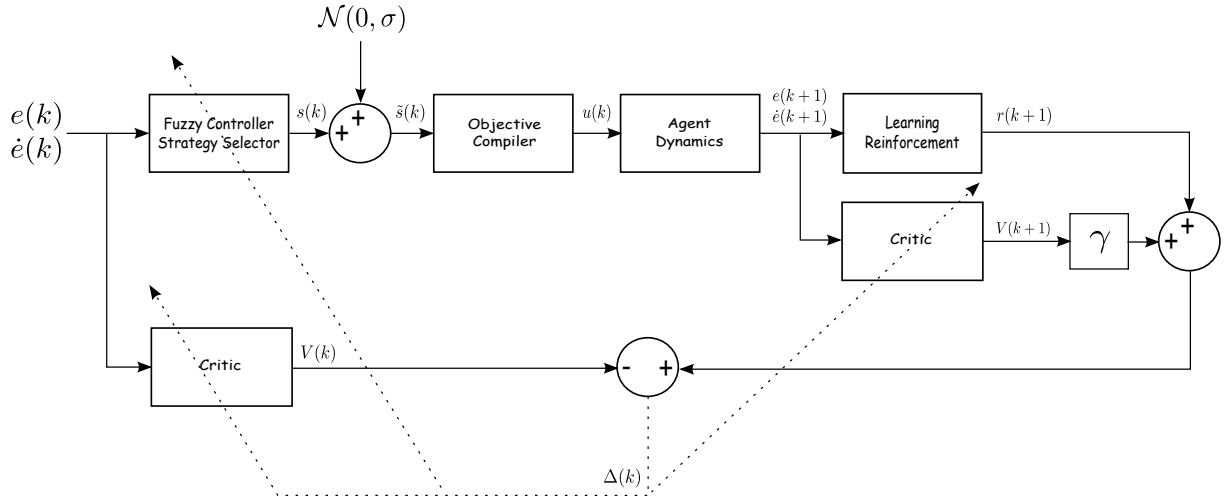


Figure 4.1: Strategy reinforcement scheme for a single agent controller.

4.3 Controller Architecture

The controller presented in this paper seeks to satisfy two objectives. First, starting from an initial partition of one objective function applied unilaterally throughout the state space it is desired for the agent implementing this controller to learn when to create the necessary switch of rewards in one round of play.

The block diagram of the controller is given in Fig. 4.1. Similar to [53, 118], we implement the same style of delayed reinforcement for actions. However, unlike the previous work, we make the reasonable assumption that we know the possible objectives available to the agent, and the fuzzy controller becomes responsible for choosing which objective function to use. We further deviate by removing the objective function from the reinforcement reward and replace this with rewards for movement through increased movement through the state space.

4.3.1 Agent Dynamics and Objective Compiler

The controller and agent dynamics are what guide the overall motion of the system trajectory. We assume the plant has underlying dynamics following Eqn. 4.3, where the agent trajectories are evolved independent of each other's state. The plant takes as an input a control, $u_i \in U_i$, from agent i and the other agents in the system play the control (vector) $\tilde{u}_i \in \tilde{U}_i$. The output of the plant block is a new error which is then propagated to the critic and learning reinforcement block. The state of the system, X, is stored internally to the plant.

The objective compiler, in terms of our formulation, converts the strategy designator, $\tilde{s}$, to a control value that is used in the actual dynamic equations in the plant. The compiler takes the incoming strategy designator and with a mapping $\mathcal{W} : S \to \mathbb{S}^Q$ where $\mathbb{S}^Q$ is a Q-dimensional simplex and Q is the number of strategies considered by the strategy selector. This mapping is simply a weighting function applied to each of the strategies, similar to the form in [55]. The weighting vector is used to find a control $u_i(k)$ by

$$u_i(k) = \mathcal{W}(\tilde{s}(k))^T \mathcal{U}_i(k) \tag{4.4}$$

where $\mathcal{U}_i(k) := [u_i^1(k) u_i^2(k) \dots u_i^Q(k)]^T$ is a vector of possible control actions where each element represents a control action calculated using Eqn. 4.1 for each of the Q objective functions.

4.3.2 Strategy Selector

The strategy selector is responsible for choosing which of the strategies in the controller is played. The structure of the controller is a fuzzy controller using TS-rules. The antecedents

of the controller are held constant whereas the consequents of the fuzzy controller are adapted based on the temporal difference, Δ, error signal from the critic and learning reinforcement blocks. An error signal and the derivative of the error signal are given as inputs giving the controller a PD-type nature. The output of the controller is given by

$$s(k) = \frac{\sum_{l=1}^{M} \left(\left(\prod_{j=1}^{n} \mu^{F_j^l}(x_j(k)) \right) \cdot \chi_l \right)}{\sum_{l=1}^{M} \left(\prod_{j=1}^{n} \mu^{F_j^l}(x_j(k)) \right)} \tag{4.5}$$

where M is the number of fuzzy rules, n is the dimensionality of the input (2, in this case), $\mu^{F_j^l}$ is the membership function for fuzzy rule l and χ is the consequent assigned to membership function l. The p^{th} consequent is adapted using

$$\chi_p(k+1) = \chi_p(k) + \beta\Delta \left[\frac{\tilde{s}(k) - s(k)}{\sigma} \right] \frac{\partial s(k)}{\partial \chi_p} \tag{4.6}$$

where

$$\frac{\partial s(k)}{\partial \chi_p} = \frac{\prod_{j=1}^{n} \mu^{F_j^p}(x_j(k))}{\sum_{l=1}^{M} \left(\prod_{j=1}^{n} \mu^{F_j^l}(x_j(k)) \right)}, \tag{4.7}$$

β is a constant that controls the rate of adaptation, and σ is the variance of the noise input. A zero-mean Gaussian noise input, $\mathcal{N}(0, \sigma)$, is added to the control signal, $s(k)$, to promote exploration for the adaptation to take place.

4.3.3 Critic

The critic, appearing twice in the model, is responsible for estimating the reinforcement learning reward a given error signal will generate given the current state of the consequents

of the fuzzy strategy selector. The critic works in much the same way as the strategy selector with the same type of input, the same rules with constant antecedents, and adaptive consequents. In order to achieve convergence, the critic must have an update value greater than the fuzzy selector. Furthermore, when initializing a series of games, the consequents of the critic are always set to 0. In this way, the critic consequents will be adapted to the correct value before the controller, and any deficiencies in the controller brought out by exploration due to the noise injection will be adapted into the strategy selector consequents. Similar to the fuzzy controller, the value output of the critic is given by TS rules. The output equation for this fuzzy block is

$$V(k) = \frac{\sum_{l=1}^{M} \left(\left(\prod_{j=1}^{n} \mu^{F_j^l}(x_j(k)) \right) \cdot \zeta_l \right)}{\sum_{l=1}^{M} \left(\prod_{j=1}^{n} \mu^{F_j^l}(x_j(k)) \right)} \tag{4.8}$$

where ζ_l is the consequent for rule l. Furthermore, the consequent adaptation occurs in the same form as the controller with

$$\zeta_p(k+1) = \zeta_p(k) + \alpha \Delta \frac{\partial V(k)}{\partial \zeta_p} \tag{4.9}$$

where

$$\frac{\partial V(k)}{\partial \zeta_p} = \frac{\prod_{j=1}^{n} \mu^{F_j^p}(x_j(k))}{\sum_{l=1}^{M} \left(\prod_{j=1}^{n} \mu^{F_j^l}(x_j(k)) \right)}, \tag{4.10}$$

α is a constant that controls the rate of adaptation of the critic.

4.3.4 Learning Reinforcement

The learning reinforcement block in this model operates differently than the previously discussed models. In the previous models, the learning reinforcement block employs the objective function for modifying the actions taken by the agent in the fuzzy controller. However, since this model does not incorporate the use of direct control adjustment in the fuzzy controller, the learning reinforcement does not monitor the performance of an objective function such as separation distance. Rather, the learning reinforcement in this block rewards motion throughout the state space and penalizes stagnation. The type of motion that is rewarded does not necessarily have to contribute positively towards reaching the underlying objective of the agent. This case will be evident in the forthcoming example.

The reason for this type of learning reinforcement is that we seek to encourage the agent to move actively throughout the state space in order to enable faster adaptation. Another advantage is that by not choosing to directly reward a player for tending toward a desired goal, the learning reinforcement is less likely to cause the agent to become trapped in a state and may be applied at all states or error states.

The learning reinforcement tool in this scheme is a static fuzzy controller that maps the error signals to a value in $[0, 1]$. Once again, using TS rules, the output of the learning reinforcement block is determined by

$$r(k+1) = \frac{\sum_{l=1}^{M} \left(\left(\prod_{j=1}^{n} \nu^{F_j^l}(x_j(k)) \right) \cdot \xi_l \right)}{\sum_{l=1}^{M} \left(\prod_{j=1}^{n} \nu^{F_j^l}(x_j(k)) \right)} \tag{4.11}$$

where ν are the fuzzy rules for the reinforcement block. Unlike the critic and the controller, the fuzzy rules for the reinforcement block can have a completely different structure.

4.3.5 Adaptive Partitioning of the State Space

The main objective of this adaptive fuzzy learning scheme is to be able to partition the state space into regions where a specified short horizon objective function is played. In order to accomplish this, we seek to find areas in the state space that given an objective following Def. 1 or one of the auxiliary functions produced by Alg. 3 stagnate under the current objective function. By doing so, the learning algorithm may then try a different objective in these states. Stagnation as termed in this paper refers to

Definition 2. *Stagnation states for objective R_q that form a set, $\Gamma_q \subset X$, are those where*

$$e \neq 0$$
$$\dot{e} = 0$$

and the operator $\Gamma^*(x)$ assigns to a state x the highest objective level, q^*, where $\Gamma_{q^*} = \emptyset$.

We now show that the adaptive fuzzy controller strategy selector is bounded on an interval that is guaranteed to cause motion throughout the state space given that the objective functions are ordered as described in Condition 1 of Thm. 8. First we define two mappings:

Definition 3. *Let $s^*(x) : X \to S$ give the strategy control value that corresponds to playing the pure objective R_{q^*}.*

Definition 4. *Let $\mathcal{E}$ be a mapping $\mathcal{E} : X \times T \to E \times \dot{E}$ that calculates the error of the current state.*

The set of objective functions available to the agent are given as $\mathcal{R} := \{R_1, R_2, R_3, \ldots, R_Q\}$ where Q is the number of objectives that may be played by the agent. We now show that the output of the strategy selector fuzzy controller is bounded.

Theorem 8. *If an agent with an adaptive fuzzy controller with reinforcement learning has the structure of Fig. 4.1 and the critic, controller, and learning reinforcement have TS-rules using Eqns. 4.8, 4.5, and 4.11 respectively with update equations 4.6 and 4.9 and if*

1. *there exist a set of objective functions, $\mathcal{R}$ where R_1 satisfies Def. 1 and as q increases, $\Gamma_{q-1} \subset \Gamma_q$ for all $q > 1$,*
2. *$\alpha >> \beta$ such that the critic converges before the strategy selector,*
3. *$\sigma > 0$, and*
4. *the strategy controller is initiated such that $s_0(\mathcal{E}(x)) < s^*(x)$ for all x*

then the trajectory of the strategy selection values is bounded such that, $\underline{s}(\mathcal{E}(x)) < s_k(\mathcal{E}(x)) < s^*(x)$ as $k \to \infty$, where $\underline{s} >= s_0$ is a constant.

Proof. We begin the proof by considering the lower bound $\underline{s}(\mathcal{E}(x))$. Assigning $\underline{s}(\mathcal{E}(x)) = \sup\{S|x \in \Gamma^*(x)\}$ provides a lower bound which correlates to the highest level objective function that causes state stagnation. From Condition 1, the set $\{S|x \in \Gamma^*(x)\}$ is a continuous interval. When $s_k < \underline{s}$ the state is a stagnation state. Following condition 2, as $s_0 \to 0$, $r_{k+1} \to 0$ and $V_{k+1} - V_k \to 0$. Therefore, if $x \in \Gamma_q$ for some objective, R_q, then the temporal difference reduces to $\Delta_k = r_{k+1}$. Because the reinforcement learning is bounded on $[0, 1]$, then the exploration noise guaranteed by Condition 3 will cause $\Delta(k) > 0$ if $\tilde{s}_k - s_k > 0$ and $\Delta_k = 0$ otherwise. The controller update, Eqn. 4.6 will subsequently result in $s_{k+1} >= s_k$ for all values, where $s_{k+1} > s_k$ occurs when $\tilde{s}_k$ nears the appropriate objective for the region. This shows that for a given partition of the state space, the update for s_k must increase to $s_k > s_0$.

As $s_k \to s^*(x)$, then $V_{k+1} - V_k \to -1$ and $r_{k+1} \to 1$. Substituting into the temporal difference,

$\Delta \to 0$, then we must have that $s_k < s^*(x)$ as $k \to \infty$ if Condition 4 is met. Therefore, the output of the adaptive strategy selector is bounded in the infinite horizon. □

In this proof, we have shown that the adaptive fuzzy controller will force the strategy to output some value that results in motion throughout the state space. This results because the output of the strategy is bounded between a level that is just above stagnation and one that is just below the strategy level that corresponds to the correct objective. Where the controller finally settles depends on the choice of variance for the exploration noise and the objective structure. Qualitatively, there appears to be a compromise between risky learning with an increased σ and the ability of the strategy to converge closer to the correct value. However, this remains an open question that will be investigated in future work.

4.4 Illustrative Example

We present an implementation of the adaptive fuzzy controller by considering the dynamics of the Homicidal Chauffeur game. In this scenario, a faster pursuer is trying to capture a slower but more agile evader. In the classic game, time-optimality, a backward propagating reward function, is used by both agents to determine the optimal actions. This exercise will show how an agent can be equipped with a switched reward to achieve similar results when optimality of an action is considered over a one step ahead horizon. The example shows two scenarios. In both scenarios, the evader is equipped with a time-optimal objective and the pursuer is equipped with the adaptive fuzzy controller. In the first example, the exploration noise of the pursuer is switched off so learning does not take place and the only variation in the strategy selection is due to noise. In the second example, we see how the learning algorithm results in the pursuer adapting to both the evader's dynamics and objective function to secure a win.

4.4.1 Homicidal Chauffeur Dynamics

Throughout the three scenarios, each of the agents move on the plane with the following dynamics:

$$\dot{x}_1 = v_p \sin(x_3) \tag{4.12}$$

$$\dot{x}_2 = v_p \cos(x_3) \tag{4.13}$$

$$\dot{x}_3 = \frac{v_p}{r} u_1 \tag{4.14}$$

$$\dot{x}_4 = v_e \sin(u_2) \tag{4.15}$$

$$\dot{x}_5 = v_e \cos(u_2) \tag{4.16}$$

where x_1 and x_4 are the horizontal positions of the pursuer and evader. Likewise, x_2 and x_5 are the vertical positions of the respective agents. x_3 represents the bearing of the Pursuer. The variables, v_p and v_e, are speed parameters assigned to the pursuer and evader respectively. The variable, r, is the pursuer turn radius parameter ($r = 0$ for the evader). Finally, a_1 and a_2 are the steering controls provided to the the pursuer and evader. The action set, $A := A_1 \times A_2$ is decomposed as

$$a_1 \in A_1 = [-1, 1] \tag{4.17}$$

$$a_2 \in A_2 = [0, 2\pi) \tag{4.18}$$

The bounds of the action set represent the pursuer making the sharpest possible turn to the right or left as well as the evader choosing its instantaneous direction. The game terminates when the evader enters into a ball around the pursuer of radius, l.

$$\mathcal{T} = \{x \in X, \sqrt{x^2 + y^2} < l\}$$

The parameters for this scenario are $v_p = 2$, $v_e = 1$, $r = 1$, and $l = 0.15$. Perfect information is assumed to be available to both agents.

4.4.2 Implementation of the Adaptive Fuzzy Controller

In this section, we discuss how the fuzzy controller described in Sec. 4.3 applies to solving the Homicidal Chauffeur problem. In this scenario, the pursuer is cognizant of the opposing agent's location, but is not familiar with any of the parameters that govern the evader's motion through the state space. We know from [56, 64] that given the capabilities of the agents, the pursuer can adopt one of two strategies: minimize ($s_k = 0$) or maximize ($s_k = 1$) the distance to the evader. If the pursuer is aware of the evader's velocity, then it would be trivial to partition the states associated with separate objective functions for the agent to use. However, in this case, we will use the adaptive fuzzy controller to implement the switch. Essentially, this results in an automatic partitioning of the state space for selecting the appropriate objective function.

The strategy selector, critic, and learning reinforcement take as error inputs the bearing angle, ϕ and the angle rate, $\dot{\phi}$. For both the strategy selector and critic, the consequents are initialized to 0 and the triangular membership functions for the angle are given as

$$\mu^{F_1^l}(\phi) = \max\left\{0, \left(1 - \left|\frac{n_a(\phi - a_l)}{2\pi}\right|\right), \left(1 - \left|\frac{n_a(\phi - a_l - 2\pi)}{2\pi}\right|\right)\right\} \tag{4.19}$$

where n_a is the number of elements in the vector of membership centers, a_l. Values in a_l are

required to be equidistant on the interval $[0, 2\pi)$. Likewise for the angle rate,

$$\mu^{F_2^l}(\dot{\phi}) = \max\left\{0, \left(1 - \left|\frac{n_b(\dot{\phi} - b_l)}{2\pi}\right|\right)\right\} \tag{4.20}$$

where n_b is the number of elements in the vector of equidistant membership centers, b_l, on the interval $[\min\dot{\phi}, \max\dot{\phi}]$. In this model, the adaptation of the critic and strategy selector occurs at a rate of $\alpha = 0.5$ and $\beta = 0.1$ respectively. The scenario produced in this paper uses $n_a = 16$ and $n_b = 10$ on the interval $[-5, 5]$. The learning reinforcement block uses the fuzzy membership functions from Fig. 4.2. The consequents associated with the learning block are given in Table 4.1. The consequents are assigned such that when evaluated using the TS-rules from Eqn. 4.11, motion through the state space is encouraged and stagnation is heavily penalized.

Table 4.1: Consequents for learning reinforcement.

	Angle Rate				
Angle	1	0.5	0	0.5	1
	0.25	0.25	1	0.25	0.25
	1	0.5	0	0.5	1

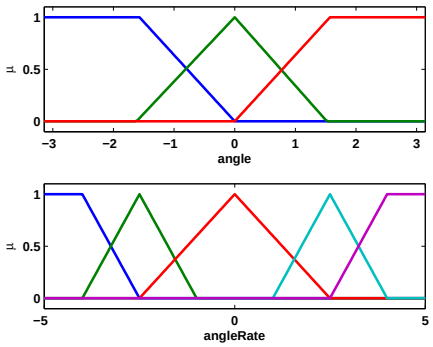


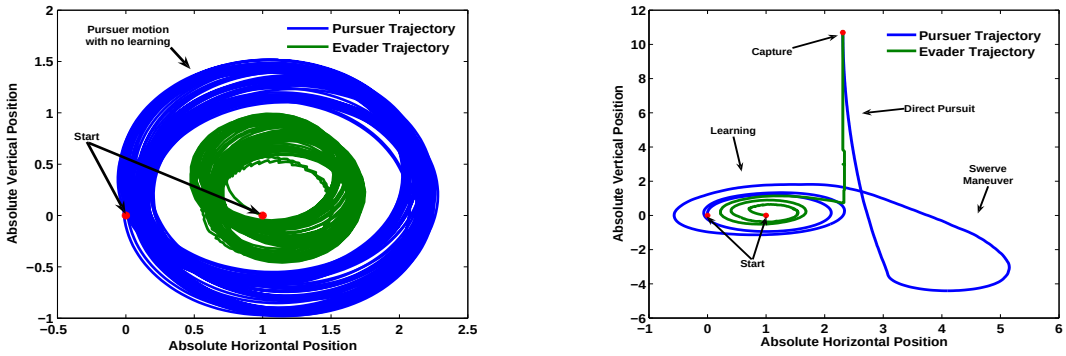
Figure 4.2: The membership functions associated with the learning reinforcement block promote movement through the state space.

4.4.3 Results

Two different versions of the Homicidal Chauffeur problem are considered. In both, we equip the evader with the time-optimal solution and the pursuer with the adaptive fuzzy controller. In the first play of the game, however, the pursuer is not given a controller update, rendering the adaptive controller void except for the noise input. This scenario is then compared to a second iteration of the game where the pursuer's learning is enabled.

Fig. 4.3a shows the result of the first scenario. The pursuer starts from an initial point at the origin and the evader begins at the point $x = 1, y = 0$. In this scenario, the evader uses the time-optimal solution to take actions to place its location behind and to the side of the pursuer, as close as it is able. Because the controller of the pursuer is initialized to minimize the distance to the evader, the resulting trajectories amount to nothing more than repeated attempts to capture the evader who is safely located within the pursuer's turning radius. The only effect of the noise due to exploration is the slight translation of the cyclic motion on the plane. Due to the relative speeds of the two vehicles, the evader is able to easily compensate for this exploration noise as is evident in the similar appearance of both trajectories.

In the second round of play, seen in Fig. 4.3b, the effect of the learning becomes evident. Starting from the same initial position, the pursuer's controller is updated to reflect the positive reinforcement from the exploration noise. Exhibiting a rough spiral effect, the learning regime reinforces actions supporting the alternative objective function when no progress is made to closing in on the evader. This is evident in both the increased diameter of the cyclic trajectories and in the control plot of Fig 4.4. The strategy of the pursuer begins by using the minimizing strategy and turning full right ($u = 1$), but then starts to reinforce more assertively as u decreases more with time. Finally, when the pursuer is able



(a) (b)

Figure 4.3: (a) The pursuer engages only in adding noise to the control signal without learning. (b) The controller update is activated to enable learning to take place.

to break free of the limit cycle, play returns to the minimizing strategy which has not yet received any type of reinforcement for the current error signals. The minimizing strategy is now successful and the usual direct pursuit takes place until capture.

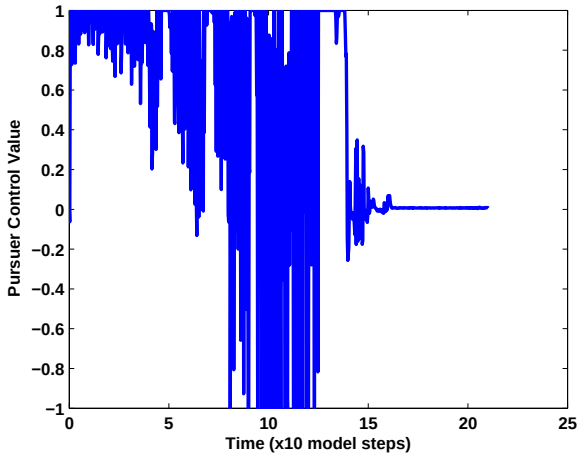


Figure 4.4: Pursuer control value. As learning takes place, the strategy selection signal tends to support the opposite strategy until direct pursuit takes place.

The second case is particularly interesting because of the way in which the pursuer learns to partition the state space into separate rewards. Because of the adaptation, the fuzzy controller makes a good switching function because it does not require any knowledge of

the counter agent's dynamic capabilities. This is in contrast to [56], where the switching function is set to trigger at a particular value. The difference is very apparent in the way that the switched controller using Alg. 3 switches when a threshold is reached, undergoes the regime and switches back very abruptly as compared to the gradual adaptation of the controller in this paper.

4.5 Discussion and Conclusions

The main contribution of this work is that it builds upon existing adaptive fuzzy and learning reinforcement theories by combining the controller update with known qualitative dynamics of the agent for whom the objectives are designed. This allows for expert knowledge on the capabilities of an agent to be included into the controller to induce switched objectives. However, since the dynamics and capabilities of an adversary are not always known, it is not always clear as to when each strategy should be played. Therefore, the adaptive nature of this system allows for a game to take place, while the agent learns how to partition the state space into regions where certain strategies should be played.

The problem with this scheme is that there is no guarantee that the scenario will fit the format of the controller to enable the switched adaptive fuzzy controller to select the proper control. In the next chapter, a graph theoretic criterion is given to identify such problems. Further a recursive scheme will be developed to update the controller based on learned information of both the environment and opposing agents.

Chapter 5

Graph Theoretic Partitioning Algorithm

The ideas of finite horizon minimization and the organization of these finite horizon objectives are now combined to create an infinite horizon scheme for solving the HJI equation. A criterion, consisting of properties for a graph adjacency matrix, governs an algorithm used to solve the HJI equation by recursively solving a simpler HJB equation. In this chapter an algorithm and control scheme are explicitly given to accomplish this task. Results are also given to illustrate the benefits of the approach.

©2011ASME. Reprint allowed through copyright authority (see Appendix), from Brian Goode, Andrew Kurdila, Michael Roan, "A Partitioning Scheme for a Switched Feedback Control Law in Two Agent Pursuit-Evasion Scenarios", in ASME Journal of Dynamic Systems, Measurement, and Control, accepted 2011.

5.1 Introduction

This work presents a new design for a switched feedback control law that enables a pursuing agent to intercept an evading agent with no prior knowledge of the evader's dynamics.

Although pursuit-evasion scenarios are the focus of this article [19, 49, 51, 53, 86, 99, 112], this controller applies to a variety of applications. The control law seeks to guide the pursuer's trajectory, subject to disturbance, from its initial state to a target state in finite time. Capture states are defined as all the initial states from where the pursuer's control law will guide the trajectory to the target. A successful realization of a control law is one that guides the pursuer to the set of target states from a maximum number of initial states by increasing the number of capture states in the capture set.

Issues such as limited sensory information [17], the curse of dimensionality [84], computational efficiency [127], and adaptability to learned evader dynamics [47] impede progress toward achieving such a control law. Maximizing the capture set of an agent generally requires more calculation time. This is not conducive to forming fast decisions or updating the control law to reflect current observations of the evader's dynamics. However, the methods presented in this work achieve high computational efficiency by dividing the control law into partitions covering the state space. As a result, many unnecessary calculations are eliminated from the control update process.

Control laws may be characterized by the length of the evaluation horizon. Infinite horizon solutions evaluate the performance of a control law over an entire trajectory, whereas finite horizon solutions only consider a portion of the trajectory. Many classic control law designs [34, 42, 64, 73] solve the infinite horizon Hamilton-Jacobi-Bellman (HJB) and Hamilton-Jacobi-Isaacs (HJI) equations [22]. The solution to these equations is a value function from which the optimal control for an agent is derived. All possible open-loop controls are effectively exhausted during the calculation process, maximizing the number of capture states. However, infinite horizon methods are especially susceptible to extended computation times for reasons cited above. Solution techniques such as Fast Marching Semi-Lagrangian (FMSL) methods [34] seek to increase time efficiency by only updating those states which have not

converged to a value. Extremal aiming techniques have a fast front-based propagation that is good for short distances to the target, but longer distances suffer from extra computation time by having to identify discontinuities in the value surface [106]. For any of these methods to be useful, perfect information about the evader should be known a priori. Although not unreasonable for specific scenarios, adapting these control techniques to games where information about the evader is learned during play is difficult.

Finite horizon control laws are generally faster solution techniques that adapt better to learned information. These solutions appear in more complicated problems [19], and in probabilistic systems with information uncertainty [27, 40, 54, 61]. However, a finite horizon control law can cause undesired fixed points or form a limit cycle in states outside of the target set on the plane. Even with perfect observations assigned to both agents, there is no guarantee that the pursuer will catch the evader with a finite horizon control [55]. To rectify this, work in the field of approximate dynamic programming increases calculation efficiency by combining simple finite horizon control laws with the increased capture capability of infinite horizon controls. Methods such as temporal differences and Q-learning have been used in conjunction with fuzzy logic and neural networks [127] where critics predict either the value function or the state-action function (Q). One example using an adaptive fuzzy temporal difference framework is in [53]. However, these methods may still be subject to finite horizon errors [56].

The control law presented in this work combines the finite and infinite horizon solutions through a digraph. If the digraph satisfies certain criteria, the pursuer will intercept the evader [57]. This is similar to work seen in [135] where the graph becomes the solution to the game, but we do not consider the optimality of the graph in the game cost [36]. In our case, the digraph represents partitions in the state space where specific control laws are applied and the pursuer switches accordingly. Therefore, the control law inside a partition

only needs to satisfy the finite horizon goal of traversing to specific partitions.

Traversing partitions resembles the problem guiding agents through an environment with physical obstacles [93, 132]. An agent must both identify object boundaries to maneuver around and reach the target in finite time. Pursuit-evasion scenarios without physical objects have virtual boundaries [60] that are functions of the evader's ability to disrupt the pursuer's desired trajectory. The difficulty is that the evader's strategy must be estimated from measurements of the evader's dynamics. We assume that parameters governing the evader's dynamics can be measured, and estimate the evader's strategy by assuming a worst-case scenario. We calculate the evader's strategy using FMSL methods, but simplify the minimax problem of pursuit-evasion to an optimal control problem of updating the virtual partition boundaries as measurements are received.

5.2 Problem Formulation

The system consists of a pursuing agent assumed to have perfect information about its own dynamics and an evading agent treated as a disturbance to the pursuer's trajectory. Parameters governing the evader's motion are not known until the game begins at $t = 0^+$. The state, x, is a member of the compact state space $X \subset \mathbb{R}^n$. The pursuer control, u, is a finite set $U \subset \mathbb{R}$, and the evader's control, d, is a finite set $D \subset \mathbb{R}$. A solution of the system is a trajectory, $w = (x, u, d)$, where $w \in \mathcal{W}$. The set, $\mathcal{W}$, is the set of all valid trajectories that satisfy the agent dynamics

$$\dot{x}^t = \mathbf{f}(x^t, u^t, d^t) = \mathbf{f}_P(x^t, u^t) + \mathbf{f}_E(x^t, d^t) \tag{5.1}$$

for all time, $t \in \mathbb{R}^+$, from some initial state, x^{t_0}, where $\mathbf{f}_P(x^t, u^t)$ and $\mathbf{f}_E(x^t, d^t)$ are the dynamic contributions of the pursuer and evader, respectively. The form of the evolution equations implies the dynamics of each agent are exogenous and separable. State evolutions for fixed functions, $u(\cdot)$, and disturbance, $d(\cdot)$, are found using

$$x^{t_f} = S(t_f, t_0; u(\cdot), d(\cdot))x^{t_0} \tag{5.2}$$

where S is the state transition operator that finds the value of x^{t_f} from an initial state x^{t_0}.

5.2.1 Control Guidance to the Target

The pursuer's goal is to generate a trajectory in $\mathcal{W}$ such that $x^{t>t^*} \in \mathcal{T}$ for some $t_0 < t^* < \infty$ where $\mathcal{T} \subset X$. $\mathcal{T}$ is termed the target set and is positively invariant. The pursuer's control law, $u = \mathcal{M}(x)$, should drive the trajectory to reach $\mathcal{T}$ in finite time. The game can terminate when the trajectory has entered the target set, a stable equilibrium point, or limit cycle. The set of initial states from which the trajectories terminate in $\mathcal{T}$ will be called the capture set.

5.2.2 Capture Set

The kinematic capture set, $\mathcal{C}_K \subset X$, is the set of initial states from which there exists a valid trajectory to the target set with some choice of open-loop control $u(\cdot)$ given a disturbance $d(\cdot)$. Namely,

$$\mathcal{C}_K(d(\cdot)) := \\ \left\{x^{t_0} \in X | \exists u(\cdot) \; s.t. \; (x, u, d) \in \mathcal{W}, x^{t>t^*} \in \mathcal{T}, t_0 < t^* < \infty\right\},$$

where x^t is found using the state transition operator of Eq. (5.2). This set designates those states from where it is possible for the pursuer to traverse to the target set given the governing dynamics of the agents in Eq. (5.2) and the sets of agent controls, U and D. The kinematic capture set is not limited by the pursuer's choice of strategy.

The set of states that can be driven to the target set with feedback law, $u = \mathcal{M}(x)$, given a disturbance, $d(\cdot)$, are

$$\mathcal{C}_F(d(\cdot), \mathcal{M}(x)) := \\ \left\{ x^{t_0} \in X | (x, u, d) \in \mathcal{W}, u = \mathcal{M}(x), x^{t>t^*} \in \mathcal{T}, t_0 < t^* < \infty \right\}$$

where x^t is found using the operator of Eq. (5.2). Similar to the kinematic capture set, this feedback capture set is a function of the evader's input, $d(\cdot)$. Certain properties of the capture sets are used to develop the switched control law.

Remark 1. *$\mathcal{C}_F(d, \mathcal{M}(x)) \subseteq \mathcal{C}_K(d)$ for any feedback control, $\mathcal{M}(x)$.*

This follows from the fact that the feedback control, $\mathcal{M}(x)$, is a particular control law chosen from the set of all possible controls. The kinematic capture set is the largest capture set that a feedback control may produce, and is used as an upper bound for the feedback capture set. Consider a fixed evading strategy, $d(\cdot)$, then the feedback control law which produces the largest capture set is

$$\mathcal{M}^*(x) = \arg\min_{u \in U} \nabla V^*(x) \mathbf{f}(x, u, d) \tag{5.3}$$

where $V^*(x)$ is the value function when the disturbance, d, is assumed to play according to a worst-case scenario.

Definition 5. *The value $V^*(x)$ is the solution to the partial differential equation (Dynamic*

Programming Principle)

$$0 = \min_{u \in U} \max_{d \in D} \left[g(x,u,d) + \nabla V^*(x) \mathbf{f}(x,u,d) \right] \tag{5.4}$$

subject to the boundary condition

$$V^*(x) = 0, \ \forall x \in \mathcal{T} \tag{5.5}$$

and the constraints

$$g(x,u,d) = \begin{cases} 0, \ \forall x \in \mathcal{T} \\ (0, \infty), \ otherwise \end{cases} \tag{5.6}$$

that finds the control functions $u(\cdot)$ and $d(\cdot)$ that minimize/maximize the cost $C(u,d) = \int_0^\infty g(x^t, u^t, d^t)dt$.

The value, $V^*(x)$, must satisfy three criteria. First, the optimal control takes values in a finite control set. Second, the value is finite inside the target set. Lastly, a finite running cost ensures the value remains finite if the state is in the capture set. A least upper bound on the feedback capture set can be established for a worst-case scenario disturbance, d, by identifying all states with $V^*(x) < \infty$.

Proposition 1. *If $V^*(x)$ satisfies Def. 5, then $\mathcal{C}_F(d, M^*(x)) = \mathcal{C}_K(d)$ where $M^*(x)$ is the control defined in Eq. (5.3).*

Proof. Assume for some initial state, x^{t_0}, $V^*(x^{t_0}) = \infty$ using the closed loop control $\mathcal{M}^*(x^t)$, and there exists an open loop control $u(\cdot)$ such that $x^{t_0} \in \mathcal{C}_K(d)$. Using Def. 5, if $x^{t_0} \in \mathcal{C}_K(d)$, then there is a constant, $C = \int_{t_0}^\infty g(x,u,d)dt < \infty$ where $g(x,u,d)$ is integrated along the trajectory, $q = (x,u,d)$. However, if $C < \infty$, Def. 5 is cannot be satisfied because a minimizing control does exist. Therefore, when Def. 5 holds, $\mathcal{C}_K(d) \subseteq \mathcal{C}_F(d, M^*(x^t))$.

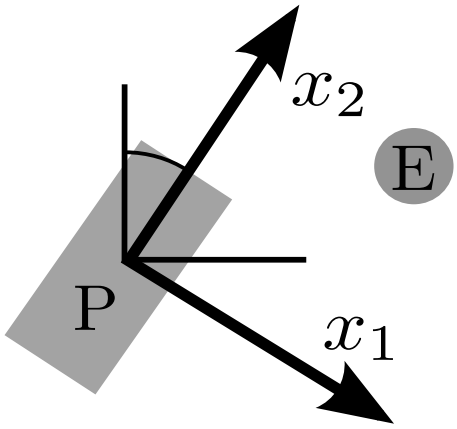


Figure 5.1: The reduced coordinates, x, are fixed with respect to the pursuer's forward velocity.

Incorporating Rmk. 1, then $\mathcal{C}_F(d, M^*(x^t)) = \mathcal{C}_K(d)$ holds. □

States in the open-loop capture set, $\mathcal{C}_K(d)$, are identified when $V^*(x) < \infty$. If the evader deviates from the strategy, the pursuer will have an increased capture set and vice versa.

Example 1. *The relationship between finite and infinite horizon controls in terms of capture set properties is illustrated using the Homicidal Chauffeur game [64]. The terms of Eq. (5.1) are given as*

$$\mathbf{f}_P(x^t, u^t) = \left\{ \begin{array}{c} -\frac{v_p x_2}{r} u \\ \frac{v_p x_1}{r} u - v_p \end{array} \right\} \tag{5.7}$$

for the pursuer. The parameter, r, is the turn radius, v_p is the speed, and the control $u \subset U = [-1, 1]$. Likewise,

$$\mathbf{f}_E(x^t, d^t) = \left\{ \begin{array}{c} v_e \sin(d) \\ v_e \cos(d) \end{array} \right\} \tag{5.8}$$

governs the dynamics for the evader where v_e is its speed and $d \in D = [0, 2\pi)$. The states, x_1 and x_2, used in these equations are a fixed reference frame with respect to the pursuer's forward velocity as shown in Fig. 5.1. From [64] it is known that when the inequality with

$\gamma = \frac{v_e}{v_p}$,

$$\frac{l}{r} > \sqrt{1-\gamma^2} + \sin^{-1}\gamma - 1 \tag{5.9}$$

is satisfied, the pursuer can intercept the evader within a ball of radius, l, from any initial state. By setting the parameters to $v_p = 2$, $v_e = 1$, and $r = 1$, it follows that $\mathcal{C}_K(d) = X$. The form of feedback control law for the pursuer is

$$\mathcal{M}(x) = \arg\min_{u \in U} \nabla V(x)\mathbf{f}(x,u,d) \tag{5.10}$$

where $V(x)$ is any function defined over the state space. The evader control playing the worst case is

$$d = \arg\max_{\bar{d} \in D} \nabla V^*(x)\mathbf{f}(x,u,\bar{d}) \tag{5.11}$$

Two types of objectives, $V(x)$, will be assigned to the pursuer. One will satisfy Eq. (5.3) for $\mathcal{M}^*(x)$, and one will not.

CASE 1. The classic solution is calculated using time-optimality with $g(x,u,d) = 1$ as the running cost. This satisfies Def. 5 so that $\mathcal{M}(x) = \mathcal{M}^*(x)$. The value, $V^*(x)$, was calculated using FMSL methods. For our solution, a 180×180 node grid was used to approximate the state space. The calculation was initialized by assigning the value of the target to 0, and iteratively solving Eq. (5.4) at each node in the grid. Details of the calculation are given in [34], and an algorithm for completing the calculation is given in Sec. 5.4.2. The result is shown in Fig. 5.2. One striking feature of this surface is the discontinuity near the pursuer target set where the trajectory is temporarily guided away from the target to compensate for the pursuer's dynamics. Prop. 1 may be verified by treating $V^*(x)$ as a Lyapunov function and taking the derivative along any trajectory. Since it is a time optimal function it can be shown that $\dot{V}^*(x) \leq -1$ everywhere, which guarantees reaching the target in finite time.

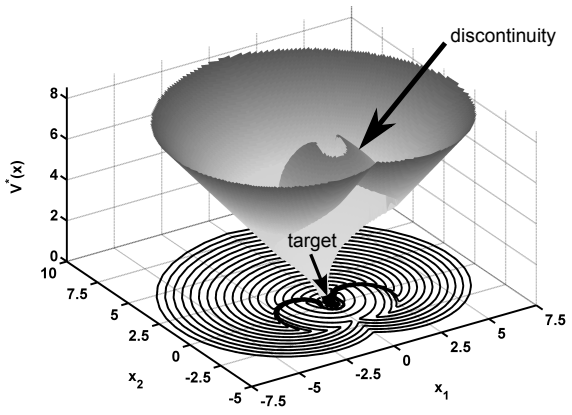


Figure 5.2: $V^*(x)$ for the time-optimal homicidal chauffeur.

CASE 2. A finite horizon objective of minimizing the distance to the evader, $V(x) = \inf_{y \in \mathcal{T}} \|x - y\|$*, is applied to the pursuer. Fig. 5.3 shows the phase portrait of the trajectories at points in the state space. States in the region enclosed by the dashed line are not capable of reaching the target, because the evader can keep the trajectory within the semi-permeable surface. Instead, trajectories from these states are attracted to the equilibrium point, reducing the capture set and illustrating Rmk. 1.*

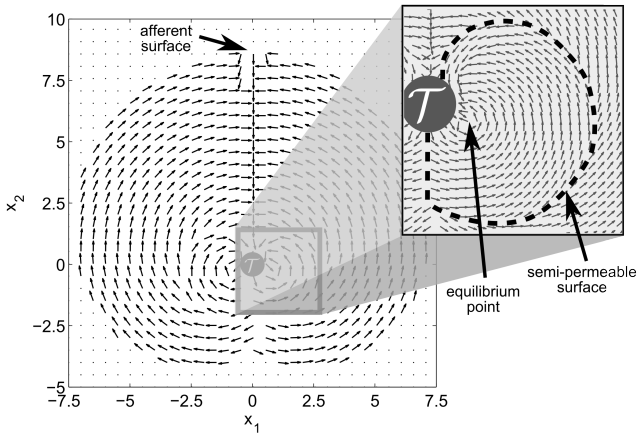


Figure 5.3: The phase plot shows states where trajectories are prevented from entering $\mathcal{T}$ due to the finite horizon objective.

This example illustrates that simple feedback controls can produce trajectories that do not lead to the target, despite having the ability to succeed otherwise. The choice of control is the limiting factor affecting the pursuer's capture set, because time optimality over the infinite horizon provided adequate guidance at the expense of calculation time. However, the finite horizon control did not give the pursuer the appropriate look-ahead horizon to first remove the evader from its turn radius. Labeling the turn radius as a partition, the pursuer needs only to remove the evader from this region before moving toward the evader. It is on this principle that the switched feedback controller is formed.

5.3 Properties of the Switched Feedback Controller

This section introduces the switched feedback controller for the pursuer. Properties of the state partitioning scheme are explored and conditions for admitting a partition into the capture set are given using a graph theoretical approach.

5.3.1 Control Form

The control law is given the switched form

$$\mathcal{M}(x) = \mathbf{m}^T \mathbf{I} \tag{5.12}$$

where $\mathbf{m}$ is a vector of q feedback controls

$$\mathbf{m} = \begin{bmatrix} m_1(x) & m_2(x) & \cdots & m_q(x) \end{bmatrix}^T \tag{5.13}$$

and membership of a control to a particular set of states is governed by

$$\mathbf{I} = \begin{bmatrix} \mathbf{1}(x \in \sigma_1) & \mathbf{1}(x \in \sigma_2) & \cdots & \mathbf{1}(x \in \sigma_q) \end{bmatrix}^T \tag{5.14}$$

where $\mathbf{1}()$ is the indicator function and $\sigma_1 \ldots \sigma_q$ are partitions of states.

5.3.2 State Partitions

The state partitions are a cover, Σ, of the state space, X,

$$\Sigma := \{\sigma_0, \sigma_1, \sigma_2, \ldots, \sigma_q\}$$

where q corresponds to the number of partitions in the state space. Each partition is assigned a feedback control, $m_q(x)$, except for σ_0. The subscript of the partition label will be referred to as the degree. Because the entire state space should be included in the partition, the cover must satisfy

$$X = \bigcup_{k=0}^{q} \sigma_k \tag{5.15}$$

where partitions have open borders with higher degree partitions and closed borders with lower degree partitions. The rules governing the assignment of states to partitions are a state can only belong to one partition so any intersection $\sigma_i \cap \sigma_j, i \neq j$ has 0 measure and $\sigma_0 = \mathcal{T}$ is closed.

5.3.3 Properties of the Adjacency Matrix

[1] The controller may be represented by a digraph where the partitions, Σ, are nodes, and edges occur if the trajectory transitions to another partition,

$$\exists(w \in \mathcal{W}, \bar{t}) \; s.t. \; x^t \in \sigma_k, x^{\bar{t}} \in \sigma_{j \neq k}, t < \bar{t} < \infty.$$

Edges represent transitions across partitions, and transitions can be coupled along a trajectory to form a path. A loop edge occurs when the pursuer wants to transition, but fails and remains in the same partition as seen in Example 1. The digraph may be represented as a square adjacency matrix, A

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1q} \\ a_{21} & a_{22} & \cdots & a_{2q} \\ \vdots & \vdots & \ddots & \vdots \\ a_{q1} & a_{q2} & \cdots & a_{qq} \end{bmatrix} \tag{5.16}$$

populated according to

$$a_{ij} = \begin{cases} 0, \text{ if there is no edge from node } i \text{ to } j \\ 1, \text{ if there is an edge from node } i \text{ to } j \end{cases} \tag{5.17}$$

One useful property is that, A^q, contains a count of how many paths of length q exist from node i to j [3].

5.3.4 Inclusion of a Partition in the Capture Set

Trajectories starting in $\mathcal{C}_F(d(\cdot), \mathcal{M}(x))$ satisfy:

[1] Please see Appendix B for updated version of Sections 5.3.3 and 5.3.4.

(1) $(x, u, d) \in \mathcal{W}$ with $u = \mathcal{M}(x)$.

(2) Termination in the target set in finite time.

If an edge exists between two nodes, then the trajectory between the partitions is valid. The problem of determining whether or not states in a partition are a member of the capture set is now cast in terms of the adjacency matrix.

Theorem 9. *Assume a switched feedback law, $\mathcal{M}(x)$, is given to an agent and the opposing agent uses a control, $d(\cdot)$. Set $\bar{A} = \sum_{k=1}^{q} A^k$. If*

(1) A has all 0 eigenvalues,

(2) the entry, $\bar{a}_{i1} > 0$ for $i > 1$, and

(3) for all entries, $\bar{a}_{j1} = 0, j > 1$, then, $\bar{a}_{ij} = 0, \forall i$.

then partition, $\sigma_i \in \mathcal{C}_F(d(\cdot), \mathcal{M}(x))$.

Proof. A matrix is nilpotent if and only if it has all 0 eigenvalues (for proof, see [107]), and there is an integer, $k \in \{1, \ldots, q\}$ such that $A^k = \mathbf{0}$. For higher powers, $A^j = A^k A^{j-k}, j > k$ such that $A^j = \mathbf{0}, \forall j > k$. Since A^k determines how many paths of length k exist between two partitions, zero eigenvalues guarantee that all paths from any node to another node are finite. It follows that all trajectories will reside in a single partition after a finite amount of time, because there are no cycles or loops in which the trajectory can be caught.

To show that the final partition can only be the target set, consider a finite path of the digraph that terminates in any partition $\sigma_{j\neq 0}$. Termination in a partition implies positive invariance because it never leaves that set of states upon entry. There are two modes by which this will occur: (1) the trajectory started in σ_j or (2) the trajectory traversed into

σ_j from another partition. If (1) is true, then $\bar{a}_{i1} = 0, i = j$ which contradicts condition 2. Likewise, if (2) is true, then $\bar{a}_{j1} = 0$ for some $j > 1, i \neq j$ and $\bar{a}_{ij} \geq 1$, which contradicts condition 3. Therefore, if the conditions 1-3 hold, then the trajectory is guaranteed to terminate in σ_0. □

This theorem provides sufficient conditions to show that a given partition is in the capture set. These conditions are easy to calculate using the flowchart in Fig. 5.4, but are restrictive in that they require all trajectories to traverse a finite number of partitions. This is important for cases where the evader's actions are not immediately known to avoid a situation where a cyclic response could form even though the current partition may lead to the target. However, an advantage to this method is that it is scalable in terms of the size of the adjacency matrix and can evaluate a variety of different scenarios that emerge.

5.3.5 Impact of Evader Disturbance on the Capture Set

The evader is modeled as a disturbance to the dynamics of the pursuer, and its effects are analyzed in terms of Thm. 10. If there is no disturbance, $\mathbf{f}_E(x, d) = 0$, then it is straightforward to develop control law parameters $\mathcal{P} = \{\Sigma, \mathbf{m}\}$, that include the most states in the capture set (see Sec. 5.4.1). However, an added disturbance can have the following impact on the digraph of the controller [57]:

(1) *Stagnation*, the evader is able to turn a partition σ_k into an invariant set.

(2) *Cyclic Partition Response*, the evader generates a cyclic path on the digraph.

(3) *Traversal to an Invariant Partition*, the evader steers the trajectory away from the target.

(4) *Aid in Desired Path*, the evader helps reach the target.

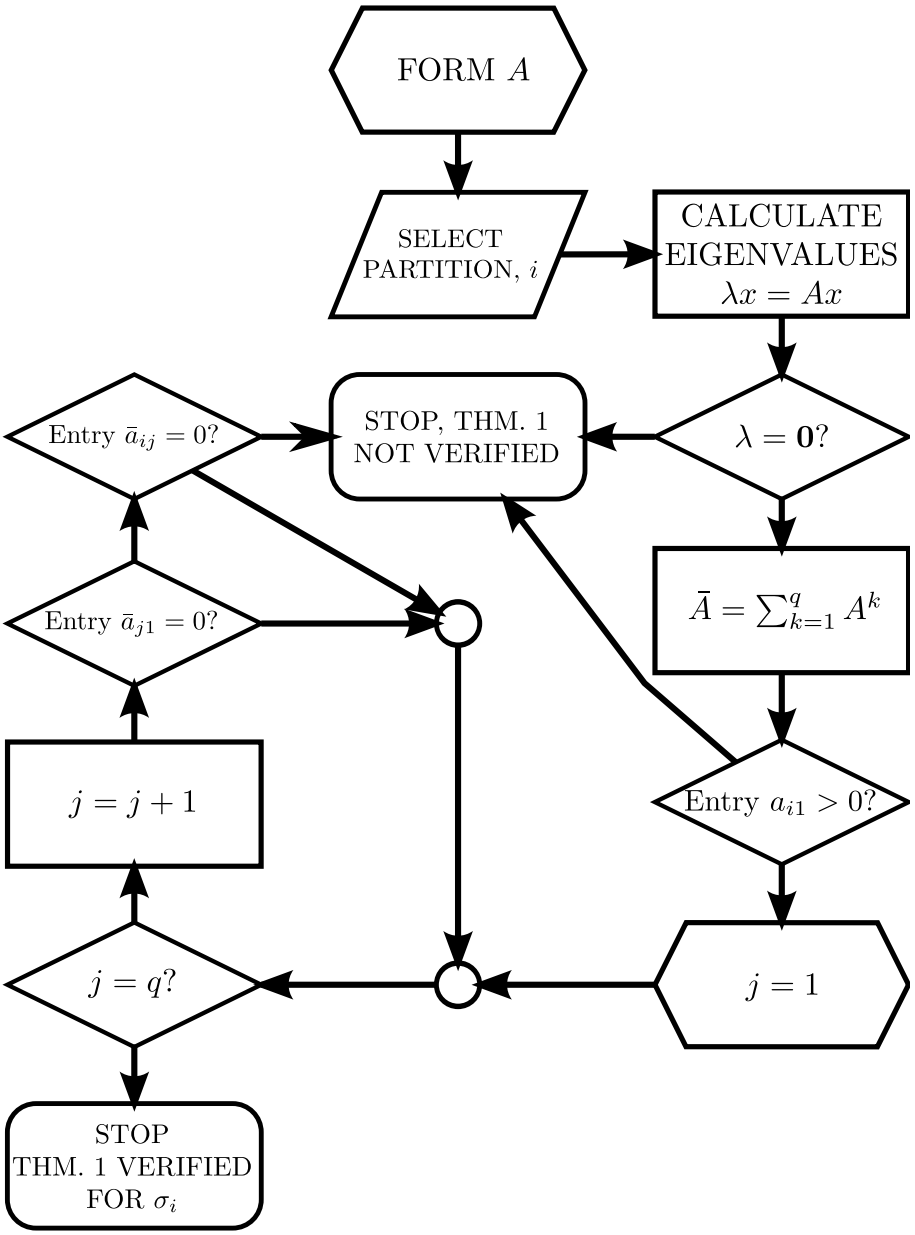


Figure 5.4: Flowchart for verifying Thm. 10.

Response 4 is trivial, but Responses 1-3 occur when the evader tries to prevent the trajectory from reaching the target. Consequently, the size of the pursuer's capture set is reduced because the disturbance causes a trajectory to enter an invariant set outside of the target set. To prevent this, the conditions of Thm. 10 can be quickly evaluated for specific partitions. Then, the switched controller is updated using the methods in the subsequent section.

5.4 Construction of the Switched Controller

Construction of the switched control law consists of two parts: (1) an initial switched control law formed off-line, before the game starts and (2) on-line updates to the control law parameters. The authors considered a variety of approaches such as neural networks [28], fuzzy logic [38, 56], and potential fields [72] to use for developing the partition controls. An objective-based approach was chosen that integrates intuitively with many artificial intelligence applications. The pursuer is assumed to have a known set of capabilities. These will vary depending upon the game, but examples include top speed, turn radius, noise output level, etc. A finite and countable set of objectives, $\mathcal{V}$, termed a library is then formed to extremize these capabilities. For example, an unmanned aerial vehicle may maximize altitude, minimize noise, or maximize speed to name a few.

Figure 5.5 shows the block diagram for the system with control updates. The output of the observer/estimator is a new state estimate at each time step, n. The controller is updated with a new set of partitions, Σ, and feedback controls, $\mathbf{m}(x)$, at every time, j, where $n \geq j$. In this article, the effects of measurement errors and time delays are assumed to be negligible, but are addressed in [59].

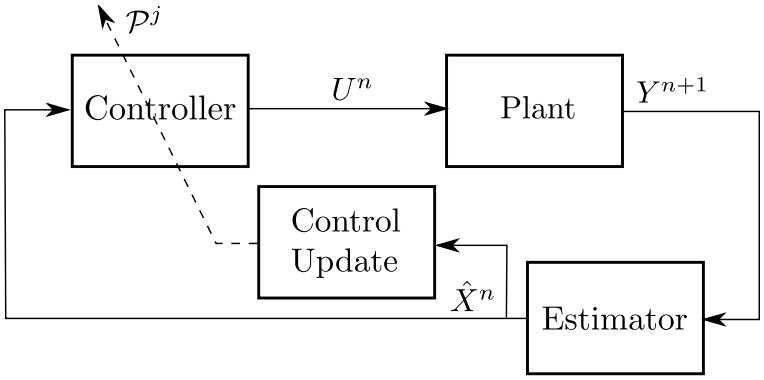


Figure 5.5: The block diagram of the switched control system.

5.4.1 Control Law with no Disturbance

The first step in constructing the switched feedback controller is to define an initial parameter set, $\mathcal{P}_0 = (\Sigma_0, \mathbf{m}(x))$, by assuming that the evader's dynamics are zero, $\mathbf{f}_E(x, d) = 0$. This is calculated before the game begins and serves as a starting point for the pursuer as measurements are received. Using an objective based approach, the control assigned to the partitions is in the form of either

$$\begin{gathered} m_q(x) = \arg\min_{u \in U} v_q(x, u, d) \\ \text{or} \\ m_q(x) = \arg\max_{u \in U} v_q(x, u, d) \end{gathered} \tag{5.18}$$

where $v_q(x, u, d) \in \mathcal{V}$. The last assumption is that the state space may be reasonably approximated as a discrete grid, $\bar{X} \subset X$. Using this and the objective library, $\mathcal{V}$, the following algorithm is used to form an initial partition and control law assignment of the state space. The algorithm starts with a single partition, $\Sigma = \sigma_0 = \mathcal{T} \subset \bar{X} \subset X$.

Algorithm 4. *Constructing the switched objective system proceeds as follows:*

(1) INITIALIZE $\sigma_0 = \mathcal{T}$, $\hat{X} = \bar{X} \backslash \sigma_0$, $\mathbf{f}_E(x, d) = 0$, *and* $q = 0$.

(2) SET $q = q + 1$.

(3) SELECT a function $v_q(x, u, d) \in \mathcal{V}$ *to assign to the control* $m_q(x)$ *over the domain,* $\hat{X}$. *If* $q = 1$, $v_q(x, u, d) = 0, \forall x \in \sigma_0$ *and must be positive definite and increasing with* $\|x\|_2$ *for* $x \notin \sigma_0$.

(4) CALCULATE the reachability of the states by integrating the trajectories from the initial states $\hat{X}$ *using the operator in Eq. (8.2)*[2].

(5) SET $\sigma_q := \left\{ x^{t_0} \in \hat{X} | \exists x^{t^*} \in \bigcup_{p=0}^{q-1} \sigma_p, t_0 < t^* < \infty \right\}$

(6) SET $\hat{X} = \bar{X} \setminus \bigcup_{p=0}^{q} \sigma_p$.

(7) REPEAT from step 2 until $\hat{X} = \emptyset$ *or* $\sigma_q = \{\emptyset | \ \forall v_q(x, u, d) \in \mathcal{V}\}$.

(8) SET $\sigma_q = \hat{X}$ *if* $\sigma_q = \emptyset$ *in step 7.*

The post-condition is $\Sigma = \bigcup_{i=0}^{q} \sigma_i = \bar{X} \subset X$. Upon termination of the algorithm, a set of partitions, Σ_0, and control vector $\mathbf{m}_0(x)$ are found for the initial parameter set, P_0. This algorithm also produces the largest capture set for a given objective library, $\mathcal{V}$. All of the states that are not grouped into a partition that leads to the target set are grouped into σ_q, which is an invariant set. In terms of the adjacency matrix the diagonal value for this partition is still 0, because the agent has no control to steer the trajectory from the set.

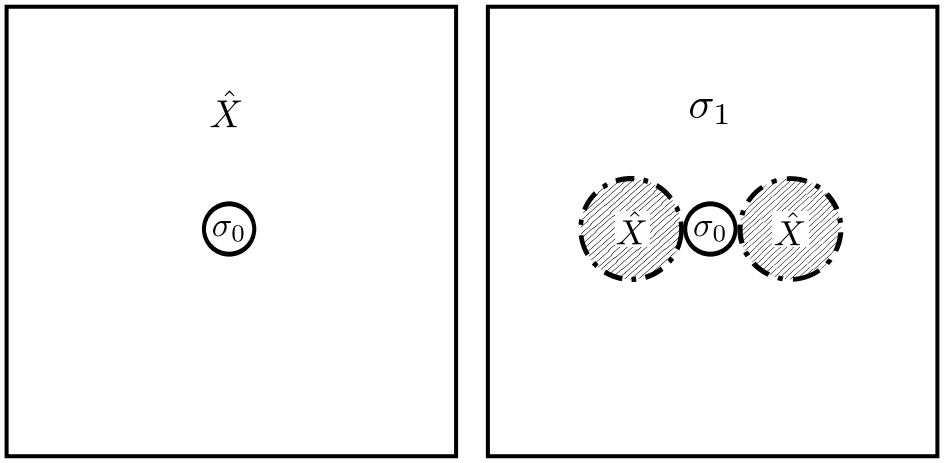
Proposition 2. *The output partition set of Alg. 4,* Σ*, and control vector,* $\mathbf{m}$ *result in a digraph with a strictly lower triangular adjacency matrix.*

Proof. Suppose a trajectory starting from σ_k, travels to a partition $\sigma_{j \geq k}$. Then from step 5, the state would not be included in σ_k, and is a contradiction. □

[2]There are several methods of integrating the vector field to determine find the state trajectories. For simple planar cases, the authors use a visual inspection.

Strictly lower triangular matrices are all nilpotent, and since no trajectory travels into the invariant set, σ_q, Thm. 10 is satisfied. Therefore, partitions $\sigma_{k<q}$ are included in the feedback capture set.

Example 2. *Alg. 4 is used to establish an initial control law pursuer in the Homicidal Chauffeur. Figure 5.6a shows the initialization step for $\hat{X}$ and the target set. The library of*



(a) Initialization of the algorithm.

(b) The shaded region is assigned a new control.

Figure 5.6: Construction of the initial control partitions.

available control laws is given as

$$\mathcal{V} = \left\{ \begin{matrix} \min_{u \in U} \|x^{t+\Delta t}\| \\ \max_{u \in U} \|x^{t+\Delta t}\| \end{matrix} \right\} \tag{5.19}$$

where Δt is the look-ahead horizon. First, the control law in $\hat{X}$ is found by setting the initial control in these states to $m_1(x) = \min_{u \in U} \|x^{t+\Delta t}\|$ and finding all of the states with trajectories leading to σ_0. The result is shown in Fig. 5.6b where σ_1, outside of the turn radius, is the set that leads to the target. The remainder of the state space is labeled $\hat{X}$, and a second objective from the library is added, $m_2(x) = \max_{u \in U} \|x^{t+\Delta t}\|$. To satisfy Thm.

10, this objective must guide the trajectories to $\sigma_1 \cup \sigma_0$. Applying the objective now enables trajectories in $\hat{X}$ to traverse into partition σ_1 to the rear of the agent. After setting $\hat{X}$ to the latest partition, σ_2, the algorithm is completed because the entire state space is included in the capture set. The initial partitions are

$$\Sigma_0 = \begin{Bmatrix} \sigma_0 \\ \sigma_1 \\ \sigma_2 \end{Bmatrix} = \begin{Bmatrix} \mathcal{T} \\ X \backslash \{\sigma_0 \cup \sigma_2\} \\ \{x \in X, \sqrt{(x_1 \pm r)^2 + x_2^2} < r - l \end{Bmatrix} \tag{5.20}$$

with the following switched control vector

$$\mathbf{m}(x) = \begin{bmatrix} \min_{u \in U} \|x^{t+\Delta t}\| \\ \max_{u \in U} \|x^{t+\Delta t}\| \end{bmatrix}. \tag{5.21}$$

5.4.2 Preventing Traversal to Undesired Partitions

With the initial partition scheme in place we now ensure that Thm. 10 holds for the off-diagonal terms of the adjacency matrix for the current measurement of the evader dynamics. First, $\mathcal{B}_k$ is defined as the set of partitions that share a border with partition σ_k, and $\mathcal{R}_k$ is the set of partitions, $\sigma_j \in \mathcal{B}_k$, where Thm. 10 would not be satisfied if $a_{jk} = 1$. This grouping of states is where the pursuer does not want the evader to guide the trajectories. It is reasonable to assume the evader's unknown policy acts according to a worst-case scenario, because if it does not subscribe to this policy, it is only beneficial to the pursuer. Similar to Prop. 1, Bellman optimality is used to identify states in σ_k where the evader can guide the trajectory to $\mathcal{R}_k$. The switched control law with parameter set P_j is given to the pursuer so Eq. (5.4) is reduced to an optimal control problem as follows.

Proposition 3. *Assume an invariant set $\mathcal{D}$, a control law $u(x) = \mathcal{M}(x)$ and the value,*

$V(x)$, is the solution to the partial differential equation (Dynamic Programming Principle)

$$0 = \min_{d \in D} \left[g(x,u,d) + \nabla V(x) \mathbf{f}(x,u,d) \right] \tag{5.22}$$

subject to the boundary condition

$$V(x) = 0, \ \forall x \in \mathcal{D} \tag{5.23}$$

and the constraints

$$g(x,u,d) = \begin{cases} 0, \ \forall x \in \mathcal{D} \\ (0,\infty), \ otherwise \end{cases} \tag{5.24}$$

The evader is able to guide the state to the set $\mathcal{D}$ from an initial state $x_0 \in X$ when $V(x) < \infty$.

The proof follows from Prop. 1 because the running cost, $g(x,u,d)$, is finite and accrues at every point on the trajectory that is not in $\mathcal{D}$. In practice, we set $\mathcal{R}_k = \mathcal{D}$. Since $\mathcal{D}$ is invariant, if the trajectory never reaches, then $V(x)$ will tend toward infinity. To solve Eq. (5.22), a version of the FMSL method [34], is summarized with some modest changes to the algorithm where noted. The advantage to this method is that it terminates when the level sets of $V(x)$ converge, and minimal time is spent calculating the function at states where it is not possible to traverse into $\mathcal{D}$. However, in order to implement the FMSL method, Eq. (5.22) must be discretized in time and space with $h = \Delta t$ and $x_i \in \bar{X} \subset X$ respectively.

$$\begin{aligned} V(x_i) = \min_{d \in D} \{ & V(x_i + h f(x_i, \mathcal{M}(x), d)) \} \\ & + h g(x_i, \mathcal{M}(x), d) \end{aligned} \tag{5.25}$$

Time-optimality,

$$g(x_i, \mathcal{M}(x), d) = \begin{cases} 0, if\, x_i \in \mathcal{R}_k \\ 1, otherwise \end{cases} \tag{5.26}$$

is chosen as the running cost. Substituting in for $g(x_i, \mathcal{M}(x), d)$ and rearranging yields

$$V(x_i) = \min_{d \in D}\{V(\underbrace{x_i + hf(x_i, \mathcal{M}(x), d)}_{\eta})\} + h \tag{5.27}$$

$$-e^{-\lambda V(x_i)} = \min_{d \in D}\{-e^{-\lambda V(\eta)}\}e^{-\lambda h} \tag{5.28}$$

$$-e^{-\lambda V(x_i)} = e^{-\lambda h} + \min_{d \in D}\{-e^{-\lambda V(\eta)}\}e^{-\lambda h} - e^{-\lambda h} \tag{5.29}$$

$$-e^{-\lambda V(x_i)} = e^{-\lambda h} \min_{d \in D}\{1 - e^{-\lambda V(\eta)}\} - e^{-\lambda h} \tag{5.30}$$

$$1 - e^{-\lambda V(x_i)} = e^{-\lambda h} \min_{d \in D}\{1 - e^{-\lambda V(\eta)}\} + 1 - e^{-\lambda h} \tag{5.31}$$

Using the Kružkov transformation, $w(x_i) = 1 - e^{-\lambda V(x_i)}$, the discrete dynamic programming problem becomes

$$w(x_i) = \min_{d \in D}\{e^{-\lambda h} w(x_i + h\mathbf{f}(x_i, \mathcal{M}(x), d))\} + 1 - e^{-\lambda h} \tag{5.32}$$

where $\lambda \in (0, 1]$. The original value may be found by reversing the transformation with, $V(x) = -\frac{ln(1-w(x))}{\lambda}$. The Kružkov transformation and the scaling factor, λ, act to bound $w(x) \in (0, 1]$ because $V(x)$ can have an infinite value.

The following algorithm presents the methods by which the solution to Eq. (5.32) is found. The conditions, $\mathcal{R}_k = \mathcal{D}$ and $|\mathcal{D}| > 0$ must be satisfied at the beginning of the algorithm.

Algorithm 5. *Solution method for Eq. (5.32):*

(1) Nodes of the set $\mathcal{D}$ are placed in the set, A and $w(x_i) = 0, \forall x_i \in \mathcal{D}$. For all other nodes,

$x_i \notin \mathcal{D}$, $w(x_i) = 1$.

(2) All neighbors of the nodes in A are placed in N.

(3) $w(x_i)$ is calculated for all $x_i \in N$ using Eq. (5.32).

(4) The node in N with the lowest value, $w(x_i)$, is removed from N and placed in A. Neighbors of this node are then added to N if they are reachable (have dynamics that can reach A).

(5) Repeat until N is empty.

Upon conclusion, $|N = \emptyset|$ and $\mathcal{D} \subseteq \mathcal{A}$. By solving Eq. (5.32), the set $\{x_i \in X | V(x_i) < \infty\}$ (or $\{x_i \in X | w(x_i) < 1\}$) identifies states where trajectories enter $\mathcal{D}$. Figure 5.7 provides an illustration of this algorithm. The white circles represent the accepted set, A. The black circles are the set of neighboring nodes, N, and the gray circles are unassigned. The term $w(x_i + h\mathbf{f}(x_i, \mathcal{M}(x), d))$, is approximated by interpolating the value of $w(x_i)$ from neighboring nodes. The arrows represent the vectogram initiating from "A", where nodes I, II, III will be used in the interpolation. The choice of h determines the magnitude of $x_i + h\mathbf{f}(x_i, \mathcal{M}(x), d)$ which should be small enough to contain the vectogram within the neighboring nodes to achieve convergence. If "A" is moved from the set N to A because it has the smallest value, then the only neighbor labeled "N" not included in N or A will be added to N. Interpolating the value at each of the nodes in N repeats until no nodes are in the neighboring set.

The advantage to this method is that if fewer states are used in the calculation before $N = \emptyset$, the faster it completes. If the agent parameters are unknown, it is not possible to provide an exact completion time. However, it has an upper bound of the completion time when all states are involved in the computation, which in most cases describes the traditional FMSL calculation used to find the solution to Eq. (5.4).

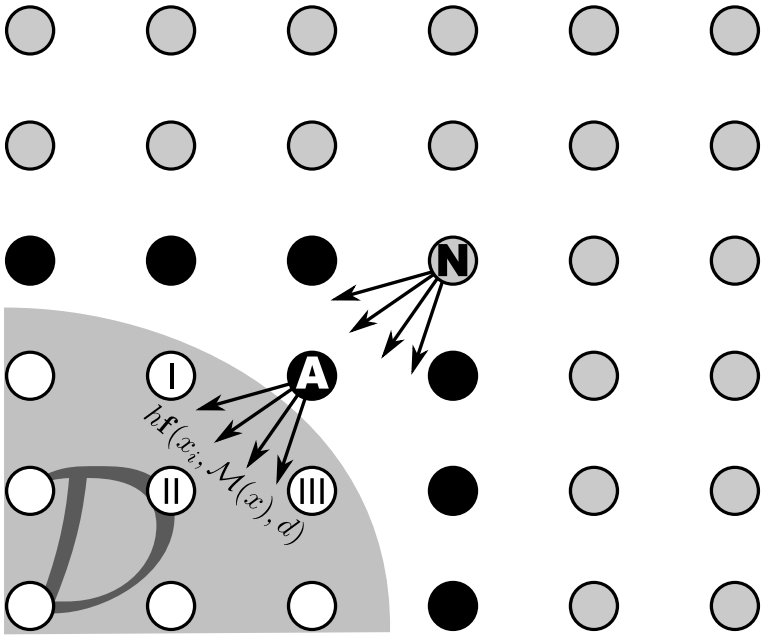


Figure 5.7: Illustration of Alg. 5 on the plane.

Example 3. *In this example, we use Alg. 5 to identify those partitions that need to have a different control due to an evader disturbance on the trajectory. The parameters given by Eqs. (5.20) and (5.21) from Example 1 are used as the initial control mapping and are outlined in Fig. 5.8. The adjacency matrix of the graph representing this scenario is*

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

which satisfies Thm. 10.

Since σ_0 is invariant by definition, it follows that the first row will be 0 regardless of the evader's actions. Therefore, only two terms on the off-diagonal could potentially cause Thm. 10 not to hold: a_{31} and a_{23}. The transition, $a_{31} = 1$, is just a shorter path to the target set. A non-zero value at a_{23} is the only transition that violates the criteria, because a cyclic path

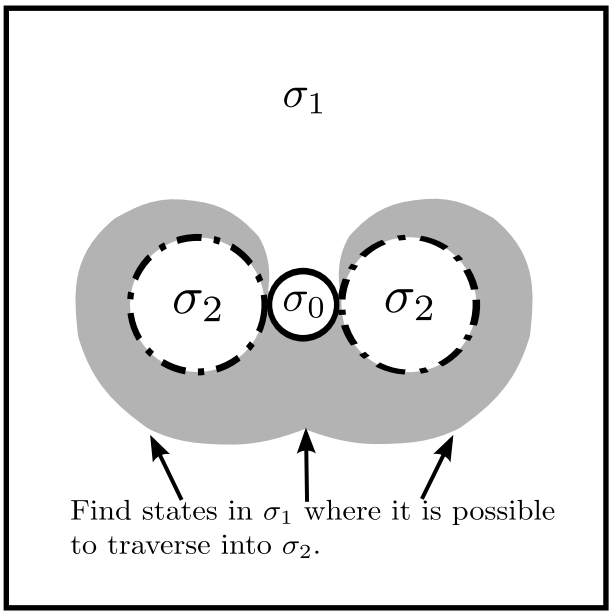


Figure 5.8: Initial partitioning scheme of the homicidal chauffeur. The shaded region is the assumed region where the evader can traverse to an undesired partition.

can form between σ_1 and σ_2. In this case, the evader seeks to make σ_2 an invariant set by always remaining within the turning radius of the pursuer.

To prevent this, we set $k = 1$, $\mathcal{R}_k = \sigma_2 = \mathcal{D}$, and use Alg. 5. The states where $V(x) < \infty$ are identified in Fig. 5.8 by the shaded region in σ_1. In this example, the control update needs only to reassign states in the shaded region to σ_2, for Thm. 10 to be satisfied again. The advantage of this method is that the only states used in the FMSL calculations in the shaded region, unlike traditional methods which would require finding the value starting at the target set.

5.4.3 Determining the Invariance of a Set

Previously, states were identified where trajectories led to invariant sets outside of the target set. Now a partition, σ_k, is analyzed for invariance by ensuring there are no nonzero terms

on the diagonal of the adjacency matrix. There are many tools in nonlinear control theory to determine if such a condition occurs [69]. Two methods are presented: Dulac's criterion for planar dynamics and an optimal control problem solved using FMSL methods for more complex cases. For planar pursuit-evasion, an invariant set is caused by either a fixed point or limit cycle. Fixed points are identified by setting $\mathbf{f}(x, u, d) = 0$ and finding disturbance values that can cause trajectory a fixed point in σ_k. Likewise, an absence of limit cycles can be verified using Dulac's criterion.

Dulac's Criterion [114]: If the following conditions hold,

(1) $\dot{x} = \mathbf{f}(x, u, d)$ is a continuously differentiable vector field on a simply connected subset of $\sigma_k \subseteq X \subseteq \mathbb{R}^2$.

(2) $z(x)$ is a continuously differentiable function such that $\nabla \cdot (z\dot{x})$ has one sign throughout X,

then there are no closed orbits lying entirely in σ_k.

The result of these operations is that a bound on the disturbance, $\mathbf{f}_E(x, d)\|_2 \leq C$, may be found before the game begins. As the control law is updated on-line, the invariance of a partition is quickly evaluated with threshold, C.

Higher dimensional problems and those with very complicated dynamics require a different approach. The optimal control problem presented in Prop. 3 can be adapted to these situations. Using $\mathcal{B}_k$, we define $\mathcal{S}_k$ as the set of partitions $\sigma_j \in \mathcal{B}_k$ where Thm. 10 is satisfied if $a_{jk} = 1$. In this form, $\mathcal{S}_k$ represents the set of partitions where a trajectory from σ_k could travel and still reach the target. The methods from the previous section are carried out by

setting $\mathcal{S}_k = \mathcal{D}$ and Eq. (5.32), is replaced by

$$w(x_i) = \max_{d \in D} \{e^{-\lambda h} w(x_i + h\mathbf{f}(x_i, \mathcal{M}(x), d))\} + 1 - e^{-\lambda h}. \tag{5.33}$$

The evader now seeks to maximize $w(x_i)$ to find all the possible controls that will prevent traversal into $\mathcal{D}$. The task now becomes identifying all those states where $w(x_i) = 1$ (or $V(x_i) = \infty$) as those where the current partition is invariant. Unlike Sec. 5.4.2, it is desired to have as many states as possible satisfying $V(x_i) < \infty$, because trajectories will leave the partition in finite time. However, this has some negative implications on the calculation time. More nodes involved in the calculation mean Alg. 5 takes longer to terminate. In the worst case scenario, the trajectory is temporarily caught in the partition before a new control from $\mathcal{V}$ can be applied. In the best case scenario, the current $\mathcal{M}(x)$ will guide the trajectory to the next set before the update calculation concludes.

Example 4. *In this example, partitions σ_1 and σ_2 of the Homicidal Chauffeur game are analyzed for invariance. Here, σ_2 is evaluated using Dulac's criterion and σ_1 will be evaluated using the FMSL method.*

Dulac's Criterion for a Parameter Bound: *To find a parameter bound using Dulac's criterion, a function, $z(x)$, must be identified such that $\nabla \cdot (z\dot{x})$ is valid. Then, the set of states where this criterion holds is identified. However, the bounds of σ_2 cannot be identified when the opponent's dynamics have not yet been observed. The only information available is that $m_2(x)$ is the pursuer control in σ_2. Therefore, this control is evaluated over the entire state space to find the evader parameter and state combinations that result in no fixed points or limit cycles.*

From the previous examples, the control in σ_2 maximizes the distance to the evader and can

be expressed as

$$u = \begin{cases} 1, \ if\, x_1 \leq 0 \\ -1, \ otherwise \end{cases} \tag{5.34}$$

We consider two regions in the state space: $x_1 \leq 0$ *and* $x_1 > 0$. *When* $x_1 \leq 0$, *Eqs. (5.1), (5.7), and (5.8) are*

$$\dot{x}_1 = -\frac{v_p x_2}{r} + v_e \sin(d) \tag{5.35}$$

$$\dot{x}_2 = -\frac{v_p |x_1|}{r} - v_p + v_e \cos(d) \tag{5.36}$$

because $u = 1$. *There are no fixed points in this region due to* $\dot{x}_2$ *when* $v_p \neq v_e$. *To determine if any limit cycles exist, we choose* $z(x) = x_2$. *Substituting into* $\nabla \cdot (z\dot{x})$,

$$\nabla \cdot (z\dot{x}) = \frac{\partial}{\partial x_1} z\dot{x}_1 + \frac{\partial}{\partial x_2} z\dot{x}_2 \tag{5.37}$$

$$= -\frac{v_p |x_1|}{r} - v_p + v_e \cos(d) \tag{5.38}$$

$$\leq -v_p + v_e \tag{5.39}$$

Therefore, we may conclude that $\nabla \cdot (z\dot{x})$ *maintains the same sign if* $v_p > v_e$, *and applies to all states in the left half-plane. A similar argument using* $z(x) = x_2$ *holds for the right-half plane where* $u = -1$. *No matter the shape of* σ_2, *the trajectory will leave the partition as long as* $\sigma_2 \subset \mathbb{R}^2$ *and* $v_p > v_e$.

FMSL Method for Determining Invariance: *Sometimes it is not possible to establish bounds on the evader's parameters before the start of the game due to complex dynamics. In this second part of the example, steps are given to determine if a partition is invariant while on-line. The methods are similar to the FMSL procedure of the previous section, but the direction of the calculation is reversed. The evader now tries maximizing the time to reach*

the set $\mathcal{D}$.

The first step is to find the set $\mathcal{S}_k = \mathcal{D}$ when the state is located in σ_1. The only partition where the trajectory can travel and still satisfy Thm. 10 is the target set, σ_0. Therefore, we set $\mathcal{S}_k = \sigma_0 = \mathcal{D}$ and use Alg. 5 to solve Eqn. 5.33. When the algorithm terminates, the same boundary noted by the shaded region in Fig. 5.8 emerges, except that all the states with $V(x) < \infty$ will be in σ_1. These are all the states that are capable of reaching σ_0 despite any action taken by the evader. Any states in σ_1 where $V(x) = \infty$ are invariant and the control must be updated so the trajectory can exit.

Two methods were presented for determining whether or not a set is invariant. The first method is performed off-line using Dulac's criterion and provides a threshold on the evader's dynamics to determine if the pursuer can steer the trajectory out of σ_2. The second is performed on-line and can handle more complicated dynamics by reversing the direction of the FMSL solution of Sec. 5.4.2.

5.4.4 Control Update Algorithm

The steps outlined in Sections 5.4.1-5.4.3 are captured in the flowchart in Figure 5.9 used to update the control block. A single update consists of adapting the partitions, Σ, and control law, $\mathcal{M}(x)$, to estimates of the evader dynamics to form a sequence of parameter updates, $\{\mathcal{P}_0, \mathcal{P}_1, \ldots, \mathcal{P}_j\}$. The algorithm initializes a switched feedback control law and set of partitions. Then, the game begins and evader measurements are taken. The focus becomes determining whether the evading agent can find a path from the current partition so the adjacency matrix of the partition graph no longer satisfies Thm. 10. The current partition is given by the metric,

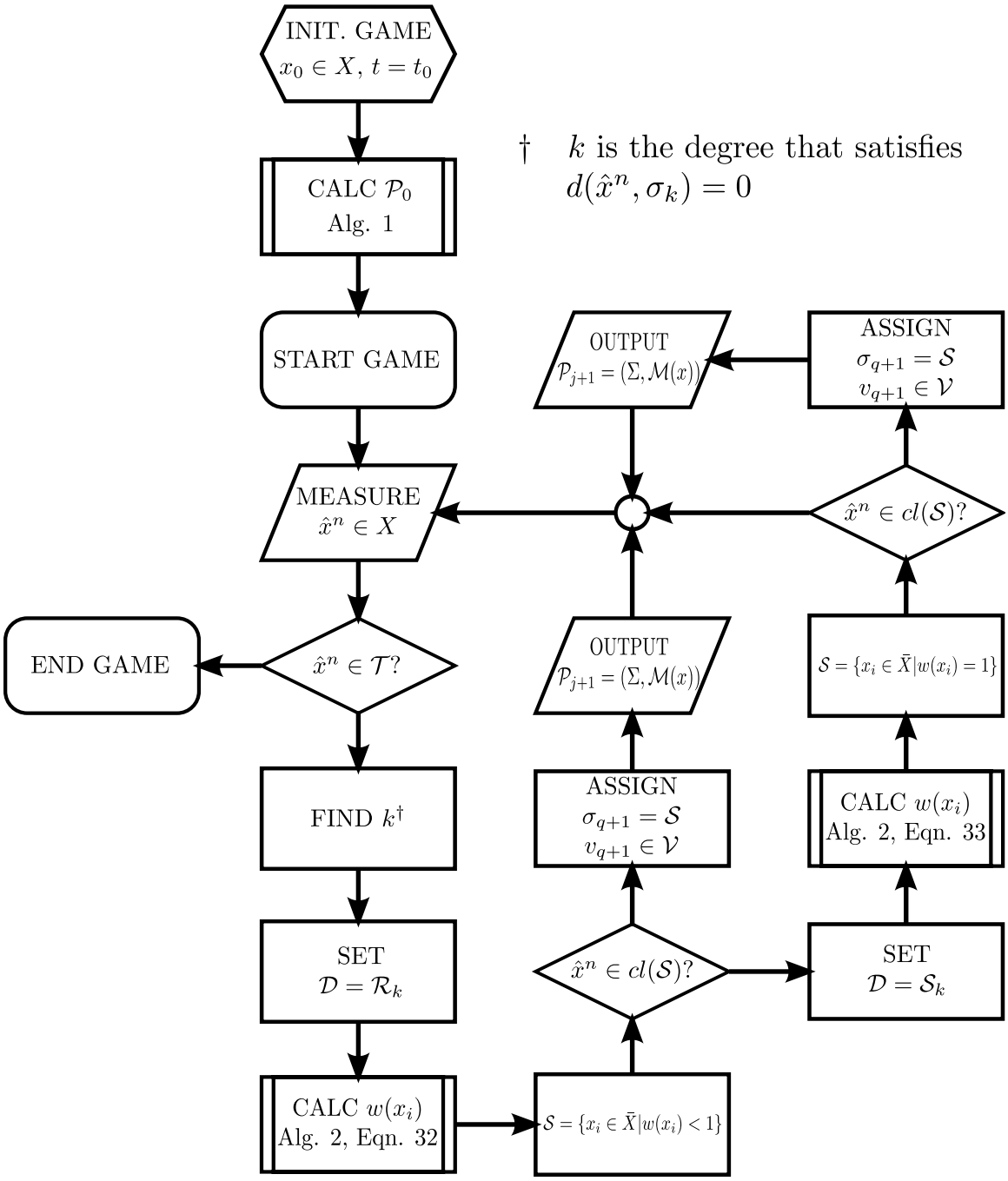


Figure 5.9: Flowchart for control updates.

$$d(x, \sigma_k) = \begin{cases} 0, \text{ if } x \in \sigma_k \\ \inf_{y \in \sigma_k} \|x - y\|, \ otherwise \end{cases} \tag{5.40}$$

Once it is known the evader cannot steer the trajectory to an undesired partition, calculations are performed to determine if the current partition contains any invariant sets. The updates are iterative in that a different control law from the library of controls is used until another estimate is received or the state has traversed to a different partition. If the control library is exhausted, and the trajectory has not been able to properly traverse to the next partition, then this amounts to a loss for the pursuer. In terms of improvement, either the dynamics of the agents do not permit a win, or the control library needs to be updated to be competitive for the evader.

5.5 Implementation Results

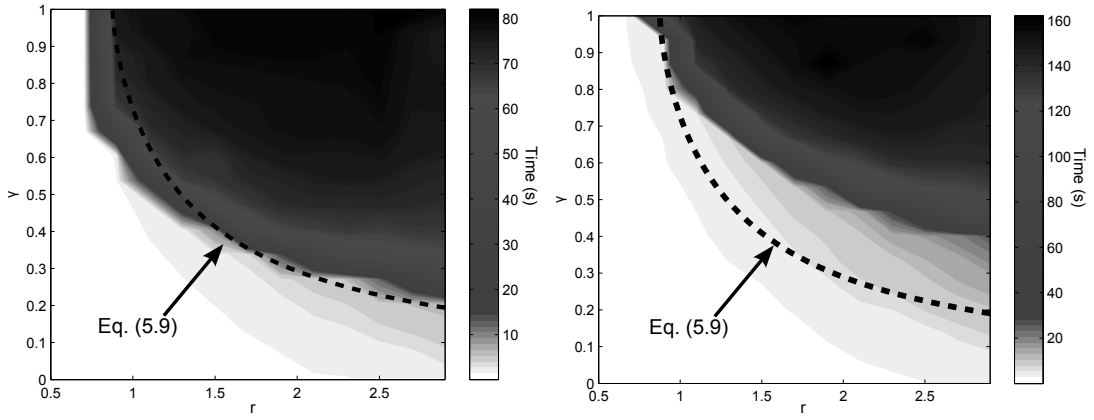
Successful interception of the evader is largely dependent upon the time efficiency and accuracy of calculating the control parameters, $\mathcal{P}$. The performance of a single update is presented in terms of these metrics using the Homicidal Chauffeur from the previous examples. By restricting $v_p > v_e$, the switched control update of Sec. 5.4.4 simplifies to determining the boundary between σ_1 and σ_2. This boundary is a function of the pursuer turn radius and both agents' speeds, and it is calculated using the update methods of Sec. 5.4.2. Similar results are seen for the FMSL method of calculating invariance in Sec. 5.4.3.

5.5.1 Effect of Vehicle Parameters and Grid Resolution on Calculation Time

Simulations with multiple trials were conducted to show the effect of the turn radius, r, and speed ratio, γ, on calculation time. In all cases, γ is evaluated on the interval $[0, 1]$, and the turning radius, r, is evaluated on the interval $[0.5, 3]$. The parameter bounds were chosen so Eq. (5.9) is valid.

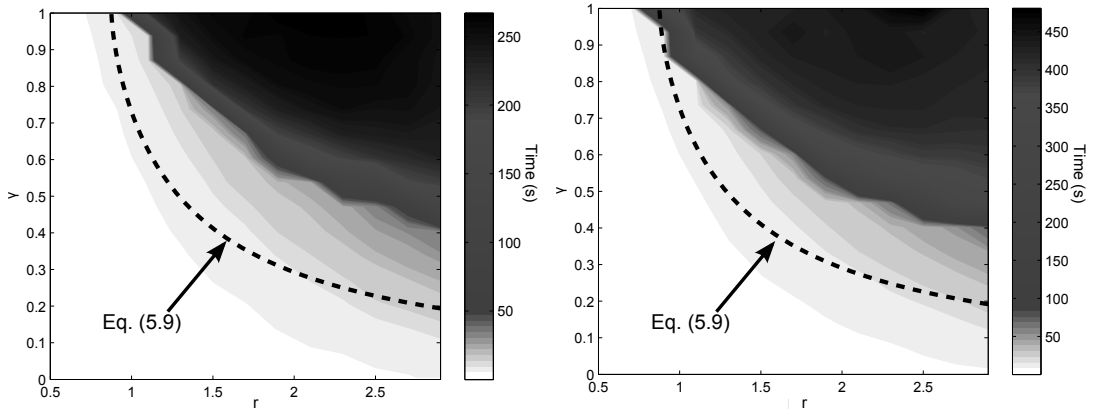
Figure 5.10 shows the computation time results for different grids computed on a FitPC 2 with 1.1 GHz Atom processor and 1 GB of DDR RAM. Eq. (5.9) is represented by the dashed curve. The results show two regions emerging in all cases Fig. 5.10a-5.10e. The darker area above the curve corresponds to the parameter space where the evader escapes, because it can enter into the pursuer's turn radius. This results in all of the grid nodes becoming involved in the FMSL calculation, increasing the calculation time. This time is greatly reduced below the curve, because the evader cannot escape in this region. Fewer states are involved because there are a small number of states where the evader can steer the trajectory into the pursuer's turning radius. As a result, the parameter update occurs much faster.

Between the light and dark regions is a gray transition that exhibits a sharp increase in the calculation time from the parameters corresponding to evader escape. As the grid sizes are changed, the transition band appears to remain rather constant. The exception to this is the course 41 x 41 grid size shown in Fig. 5.10a. Here, the transition band appears to coincide nicely with the dashed line. This is merely an artifact of the discrete grid, because the pursuer's target set is represented by a single node, allowing the evader to enter pursuer's turn radius more easily. For all grid sizes, artifacts of the discrete grid produce longer calculation times in the light gray region below the curve. This occurs, because the



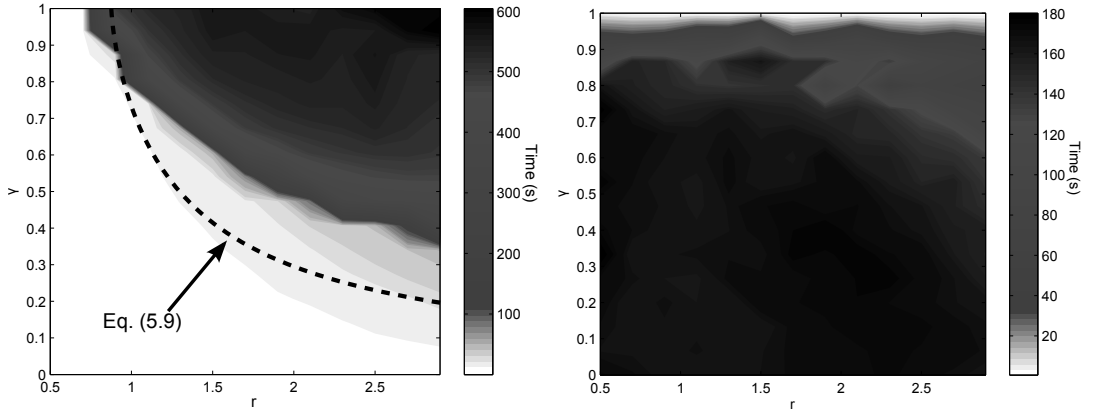
(a) 41×41 grid

(b) 51×51 grid



(c) 61×61 grid

(d) 71×71 grid



(e) 81×81 grid

(f) traditional FMSL, 51×51 grid

Figure 5.10: The time (s) is plotted as a function of the speed ratio, γ, and turning radius, r.

set of nodes considered to have trajectories leading into σ_2 is larger than the true value. As the grid size increases, the calculation time becomes more sensitive to these effects, causing larger increases relative to the grid size.

No matter the grid size, the results show that the efficiency of the control update increases when the parameters result in evader capture. To compare to traditional FMSL methods (solving Eq. (5.4)), the calculation time of a 51×51 grid is given in Fig. 5.10f. Since all of the nodes are used throughout most of the parameter space, the calculation times are high (similar results for the Tag-Chase game are in [34]). The traditional method exhibits short calculation times when the evader is well capable of escaping; however, the region of interest is when the pursuer can actually capture the evader. Comparing the results of Fig. 5.10b to Fig. 5.10f, an 11.6% improvement in calculation time averaged over the entire parameter space is observed by implementing the switched control.

5.5.2 Effect of Grid Resolution Accuracy

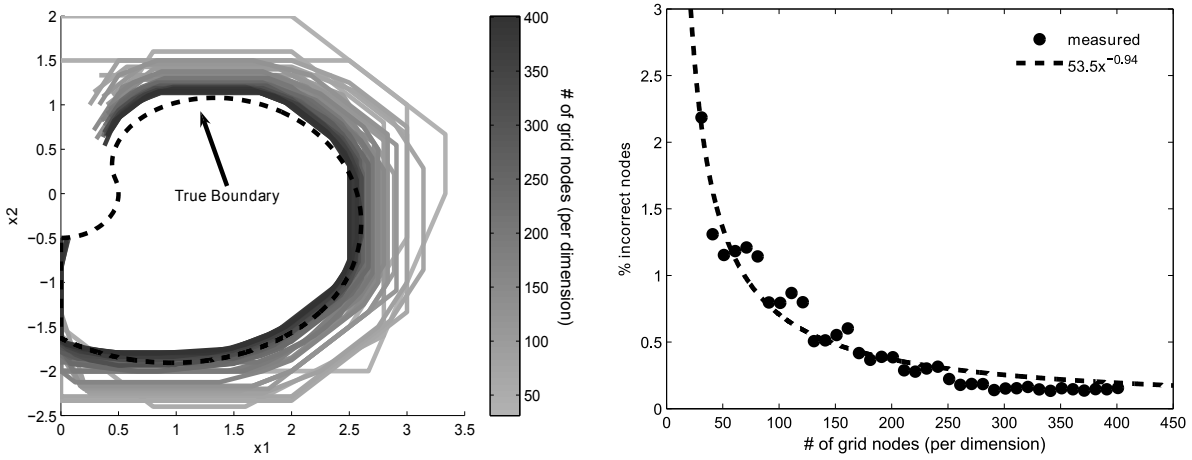
This section presents the effect of grid resolution on the accuracy of calculating the σ_2 boundary. Figure 5.11 shows the results from an array of trials performed on square grids where each dimension was incremented between 31-401 nodes in 10 node intervals. All other game parameters were held constant with $v_p = 3$, $v_e = 1$, and $r = 1$. These were chosen to provide a scenario where the game parameters enable evader capture so the boundary of σ_2 does not encompass the entire state space.

Figure 5.11a shows how a grid that is low in resolution, shown by lighter gray outlines, produces a rougher estimate of the true boundary (see [64]) than the higher resolution grids. One other advantage of the switched control method is that the value function serves as a logical operation to identify states with partitions, where traditional methods rely on the

convergence of the value for the actual control outputs. Therefore, the error for the switched control can be calculated by

$$\text{error } \% = \frac{\text{\# of incorrect nodes}}{\text{\# of total nodes}} \times 100 \tag{5.41}$$

and is proportional to the area in the state space incorrectly assigned to a partition. As the resolution increases, the grid better approximates both the shape and size of the true boundary. However, the results exhibit decreasing returns to scale for the larger grids, as shown in Fig. 5.11b. Depending on the application, it may not be worth the computational effort of larger grid sizes to obtain a better resolution of the partition boundary. In this example, the approximations tend to predict a larger σ_2 than what is actually present, giving a margin of safety ensuring the trajectory permeates the true boundary before switching to the σ_1 feedback law.



(a) Enlarged view of boundary differences.

(b) Error of different grid sizes.

Figure 5.11: The effect of the accuracy of the solution as the grid size is increased.

5.6 Discussion and Conclusions

In this paper, a new switched control law has been developed. Initial parameters for the control are found off-line and updated using estimates of the evader's dynamics. The updates are governed by Thm. 10, which provides a graph theoretical criterion for classifying the target reachability of trajectories. Advantages of this control design are threefold: (1) perfect information of the evader is not required before the start of the game, (2) conditions where the evader is not capable of escape result in decreased calculation time compared to traditional methods, and (3) minimal error in the accuracy of partition assignments is exhibited. The controller is able to achieve these results by simplifying the minimax problem of pursuit-evasion to localized optimal control problems.

The authors extend this work by analyzing the temporal evolution of the control law parameters with noisy measurements [59], and applying the switched control to a collision avoidance scenario for highway vehicles. Difficulties arise with more complicated pursuit-evasion games because of increased dimensionality and computational requirements. With collision avoidance, the number of dimensions increases, because the state space must represent both the relative position of the vehicles to each other and their absolute position relative to the road. To solve this problem, the objective of a vehicle is split into (1) remaining centered in the lane and (2) avoiding oncoming traffic. The switched control law is used to satisfy each objective, and then the state space is partitioned to determine where each control law should be played.

Chapter 6

Temporal Evolution of the Partitions

The methods from the previous chapter are now applied over the entire length of the Homicidal Chauffeur game. Here, the implementation of the controller is discussed in depth and the methods are given explicitly. Simulations are conducted to give both a qualitative and quantitative analysis of the control performance with respect to calculation time and target reachability compared to more traditional Semi-Lagrangian methods.

©2011IASTED. Reprinted, with permission, from Brian Goode, Andrew Kurdila, Michael Roan, "Performance of a Switching Controller for Pursuit-Evasion Scenarios with Noisy Measurements", in IASTED Control Applications Conference, June 2011.

6.1 Introduction

A topic of interest in control of autonomous agents are pursuit-evasion scenarios where a pursuer, with prescribed dynamic capabilities, is to capture an evader in finite time [19,49,51]. In these types of games, there are three classes of states: target, capture, and initial states. Target states define when an agent has "won", usually by entering a proximity around the other agent. Capture states are where the pursuer's strategy can guide the trajectory to

the target states depending on the strategy (control law) assigned to both agents. Lastly, initial states are where the game starts. These typically will cover the entire state space. A successful control law guides the pursuer to the target from a maximum number of initial states in the capture set.

Classic literature [64] refers to determining if the agent has the dynamic capability to reach the target set as the "Game of Kind". Similarly, finding the optimal control for the pursuer to reach the target is called the "Game of Degree" and optimizes over an infinite horizon. Solving the "Game of Degree" for a particular scenario yields a value function that the pursuer uses to generate a control. There are numerous solution methods available [34, 42, 88, 106]. These methods have very computationally expensive calculation times, but yield a solution that is guaranteed to reach the target in finite time if the pursuer has the dynamic capability. However, perfect information about the evader is required to arrive at a solution. Alternate methods exist where objectives are minimized/maximized over finite prediction horizons [27, 61]. These are generally much faster at arriving at a control solution, but do not guarantee that the agent will reach the target [55]. Approximate dynamic programming methods such as reinforcement and Q-learning seek to use finite approximations to infinite horizons, but are still subject to the same limitations as other finite horizon methods [53, 56, 127].

Previously, the authors have developed a switched control method for evaluating the "Game of Kind". Rather than determining whether the pursuer has the dynamic capability to capture the evader explicitly, we consider the pursuer's ability to drive the trajectory to partitions in the state space where specific control laws are assigned. Using a graph theoretic approach [57], the authors have shown that it is possible to quickly determine if the pursuer is capable of reaching the target given a partition-objective mapping, which forms the entire control law. This technique is used to update an initial mapping during the game using the

fast marching (FMSL) method given in [34]. The advantage of using partitions is that the computation time decreases to facilitate fast updates as measurements are gathered.

6.2 Background

The system consists of a pursuing agent assumed to have perfect information about its own dynamics and an evading agent treated as a disturbance to the pursuer's trajectory. Parameters governing the evader's motion are not known until the game begins at $t = 0^+$. The state, $\mathbf{x}$, is a member of the compact state space $X \subset \mathbb{R}^n$. The pursuer control, u, is a finite set $U \subset \mathbb{N}$, and the evader's control, d, is a finite set $D \subset \mathbb{N}$. It is assumed that both u and d are mappings to compact sets in $\mathbb{R}$. A solution of the system is a trajectory, $w = (\mathbf{x}, u, d)$, where $w \in \mathcal{W}$ and each trajectory in $\mathcal{W}$ must satisfy

$$\dot{\mathbf{x}}^t = \mathbf{f}(\mathbf{x}^t, u^t, d^t) = \mathbf{f}_P(\mathbf{x}^t, u^t) + \mathbf{f}_E(\mathbf{x}^t, d^t) \tag{6.1}$$

for all time, $t \in \mathbb{R}^+$, from some initial state, $\mathbf{x}^{t_0}$, where $\mathbf{f}_P(\mathbf{x}^t, u^t)$ and $\mathbf{f}_E(\mathbf{x}^t, d^t)$ are the dynamic contributions of the pursuer and evader, respectively. The form of the evolution equation implies the dynamics of each agent are exogenous and separable. State evolutions for fixed functions, $u(\cdot)$, and disturbance, $d(\cdot)$, are found using

$$\mathbf{x}^{t_f} = S(t_f, t_0; u(\cdot), d(\cdot))\mathbf{x}^{t_0} \tag{6.2}$$

where S is the state transition operator that finds the value of $\mathbf{x}^{t_f}$ from an initial state $\mathbf{x}^{t_0}$ given the control functions $u(\cdot)$ and $d(\cdot)$. In pursuit-evasion, the pursuer's goal is to generate a trajectory in $\mathcal{W}$ such that $\mathbf{x}^{t>t^*} \in \mathcal{T}$ for some $t_0 < t^* < \infty$ where $\mathcal{T} \subset X$. $\mathcal{T}$ is termed the target set and is positively invariant. The game terminates when the trajectory has entered

the target set, a stable equilibrium point, or limit cycle.

6.2.1 Control Law

The techniques described in the subsequent sections generate a control law for the pursuer and has the feedback form

$$u = \mathcal{M}(\mathbf{x}) \tag{6.3}$$

The function, $\mathcal{M}(\mathbf{x})$, should produce a trajectory that drives initial states to reach the target, $\mathcal{T}$, in finite time. The set of initial states that satisfy this condition is the capture set. Under the switched control law formulation, Eq. (6.3) is given the switched form

$$\mathcal{M}(\mathbf{x}) = \mathbf{m}^T\mathbf{I} \tag{6.4}$$

where $\mathbf{m}$ is a vector of q feedback controls

$$\mathbf{m}(\mathbf{x}) = \begin{bmatrix} m_1(\mathbf{x}) & m_2(\mathbf{x}) & \cdots & m_q(\mathbf{x}) \end{bmatrix}^T \tag{6.5}$$

and membership of a control to a particular set of states is governed by

$$\mathbf{I} = \begin{bmatrix} \mathbf{1}(\mathbf{x} \in \sigma_1) & \mathbf{1}(\mathbf{x} \in \sigma_2) & \cdots & \mathbf{1}(\mathbf{x} \in \sigma_q) \end{bmatrix}^T \tag{6.6}$$

where $\mathbf{1}()$ is the indicator function and $\sigma_1 \ldots \sigma_q$ are partitions of states.

6.2.2 State Partitions

The state partitions are a cover, Σ, of the state space, X, that are identified by a set of symbols

$$\Sigma := \{\sigma_0, \sigma_1, \sigma_2, \ldots, \sigma_q\}$$

where q corresponds to the number of partitions in the state space. Each partition has an associated feedback control, $m_{1,\ldots,q}(\mathbf{x})$, except for σ_0. The subscript of the partition label will be referred to as the degree. Because the entire state space should be included in the partition, the cover must satisfy

$$X = \bigcup_{k=0}^{q} \sigma_k \tag{6.7}$$

where partitions have open borders with higher degree partitions and closed borders with lower degree partitions. The only rules placed on the assignment of states to partitions is that a state can only belong to one partition and the target, $\sigma_0 = \mathcal{T}$, is closed.

6.2.3 Problem Statement

The pursuer is given a library of objectives, $\mathcal{V} := \{m_1(\mathbf{x}), m_2(\mathbf{x}), \ldots, \}$ from which to construct $\mathbf{m}(\mathbf{x})$. If the pursuer initially has no knowledge of the evader's dynamic capabilities, find a tuple, $\mathcal{P} := \{\Sigma, \mathbf{m}(\mathbf{x})\}$, such that the control law, $u = \mathcal{M}(\mathbf{x})$, enables the pursuer to drive the state to the capture set,

$$\mathcal{T} = \sqrt{x_1^2 + x_2^2} \leq l \tag{6.8}$$

in finite time. $\mathcal{P}$ is to be found on-line using noisy measurements of the evader's dynamics.

6.3 System Structure

The Homicidal Chauffeur game considered is shown in the block diagram in Fig. 6.1. The plant represents the evolution equation, given by Eqn. 6.2 and outputs the true states, $\mathbf{x}$. It operates continuously, and the evader controller and pursuer estimator sample the outputs at model steps $1 \ldots n$ with time interval, $dT = t_f - t_0$ between each step. The controllers for each player are independent of one another, and we assume that the effects of the simultaneous play of controls are negligible when compared to a true scenario where the agent inputs would occur at different times (or irregularly). Representing the worst-case scenario for the pursuer, the evader receives the true states with no noise. The evader also has perfect information of the pursuer's dynamics to develop a time-optimal control law before the start of the game. However, the pursuer receives measurements of the states subject to Gaussian

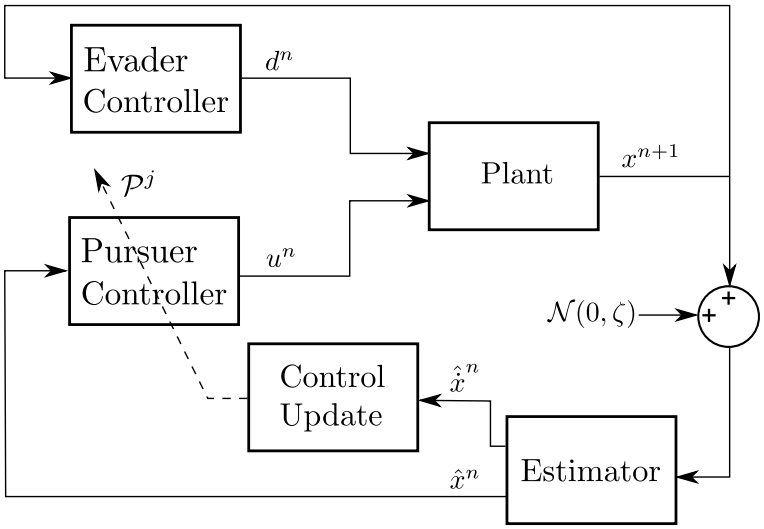


Figure 6.1: Block diagram with perfect information available to the evader and noisy measurements given to the pursuer.

noise, $\mathcal{N}(0, \zeta)$. These measurements are inputs to a Kalman filter that estimates both the position and velocity of the evader. The position is used in the pursuer's control law, which is the switched control law of Eqns. 6.3 and 6.4. The control law parameters, $\mathcal{P}$, are updated

at time steps, j using the control update. The pursuer starts with no information of the evader's dynamics, so updates are needed as fast as possible to reflect speed measurements of the evader.

6.3.1 Plant Dynamics

The plant model uses the reduced dynamic formulation of the Homicidal Chauffeur [64], where the coordinate system is oriented with the pursuer's forward direction. Eqn. 6.1 is

$$f_P(\mathbf{x}(t), u(t)) = \left\{ \begin{array}{c} -\frac{v_p x_2}{r} u \\ \frac{v_p x_1}{r} u - v_p \end{array} \right\} \tag{6.9}$$

for the pursuer, and

$$f_E(\mathbf{x}(t), d(t)) = \left\{ \begin{array}{c} v_e \sin(d) \\ v_e \cos(d) \end{array} \right\} \tag{6.10}$$

for the evader. The pursuer control input, u, represents a steering value in $[-1, 1]$. Likewise, the evader control input, d, represents a disturbance to the pursuer's intended trajectory, and selects values in $[0, 2\pi)$. The pursuer has both a speed, v_p, and turning radius. r, parameter. The evader has only a speed parameter, v_e. Eqn. 6.2 was implemented using Eqn. 6.1 with Eqns. 6.9 and 6.10 using a numerical Runge-Kutta ODE45 continuous solver. The output $\mathbf{x}^{n+1}$ was achieved by sampling at specific time intervals, dT.

6.3.2 State Estimator

The pursuer's estimator is a discrete Kalman filter [92] that receives position measurements of the evader. It outputs both the evader's estimated position and velocity. State transitions

for the filter are approximated as,

$$\begin{bmatrix} \underline{\hat{\mathrm{x}}}^n \\ \underline{\hat{\dot{\mathrm{x}}}}^n \end{bmatrix} = A \begin{bmatrix} \hat{\mathbf{x}}^{n-1} \\ \hat{\dot{\mathbf{x}}}^{n-1} \end{bmatrix} \tag{6.11}$$

where $\hat{\mathbf{x}}^n$ is the estimate of the evader's position and $\hat{\dot{\mathbf{x}}}^n$ is the estimate of the evader's velocity components used to estimate the evader's speed, $\hat{v}_e$. The state transition matrix, A, is given as

$$A = \begin{bmatrix} 1 & 0 & dT & 0 \\ 0 & 1 & 0 & dT \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.12}$$

The update for the state covariance, P^n is given by

$$\underline{\mathrm{P}}^n = AP^{n-1}A^T + Q \tag{6.13}$$

where Q is the process noise. Held constant throughout the game, the process covariance is estimated to be

$$Q = \gamma \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.14}$$

where γ is a constant scalar. The equation used to find the Kalman gain is

$$K^n = \underline{\mathrm{P}}^n H^T (H \underline{\mathrm{P}}^n H^T + R)^{-1} \tag{6.15}$$

where, H is a matrix that transforms the state vector into measurement states. In this formulation, H takes the form,

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{6.16}$$

because the pursuer is assumed to have the capability to measure the position states of the evader. R is a measurement noise covariance matrix, that in this model, takes the form

$$R = \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{6.17}$$

where λ is a constant scalar. The original state estimate of Eqn. 6.11 is then updated using

$$\begin{bmatrix} \hat{\mathbf{x}}^n \\ \hat{\dot{\mathbf{x}}}^n \end{bmatrix} = \begin{bmatrix} \underline{\hat{\mathrm{x}}}^n \\ \underline{\hat{\dot{\mathrm{x}}}}^n \end{bmatrix} + K^n \left(z^n - H \begin{bmatrix} \underline{\hat{\mathrm{x}}}^n \\ \underline{\hat{\dot{\mathrm{x}}}}^n \end{bmatrix} \right) \tag{6.18}$$

where z^n is the position measurement of the evader with noise added. It is these states that the estimator block outputs to the pursuer control and control update blocks. Likewise, the state covariance matrix is updated using

$$P^n = (I - K^n H)\, \underline{\mathrm{P}}^n \tag{6.19}$$

where I is a 4×4 identity matrix.

6.3.3 Evader Control

The evading agent is assumed to have perfect information of the states and uses the value function, V^*, found by solving the Hamilton-Jacobi-Isaacs (HJI) equation for the Homicidal

Chauffeur game (see [34, 88, 106]). The equations producing the control are given by

$$d(\mathbf{x}) = \arg\max_{\bar{d}\in D} \min_{\bar{u}\in U} \left[\nabla V^*(\mathbf{x})\mathbf{f}(\mathbf{x}, \bar{u}, \bar{d})\right] \tag{6.20}$$

Here, the evader's control solves the minimax time-optimal gradient problem using value function, $V^*(\mathbf{x})$. Essentially, the control value, d, maximizes the gradient assuming a worst-case scenario for the pursuer's control. The value function given to the evader is calculated in this work using the FMSL method outlined in [34]. Using this method, the solution to the value function is propagated outward from the target set, taking into account all agent dynamics, clearly giving the evader an advantage[1]. However, this provides a worst-case scenario for the pursuer's control outlined in the following section. Even with fast marching methods, this value function approach cannot practically be used in a realistic application, because it is a very expensive calculation in terms of time and memory (curse of dimensionality, approximation resolution, etc.). For example, if perfect information is not available to the pursuer, then this value function has to be calculated every time an updated measurement is received, which would be inefficient.

6.3.4 Pursuer Control and Update

In this game, the pursuer receives the switched control law. The pursuer controller is a function, Eqn. 6.3, that takes the state as an input and feeds a control, u, into the plant. The control update is responsible for updating the switching parameters in Eqn. 6.4. The rest of this section details how the control update is achieved. We only consider cases where $v_e < v_p$, otherwise the evader would always escape despite any possible control.

[1] If the pursuer were to choose a control that minimizes the gradient, then the result would be the solution to the Homicidal Chauffeur with perfect information.

The measurements received by the pursuer are defined as

$$\hat{v}_e := \|\hat{\mathbf{f}}_E(\mathbf{x}(t), d(t))\|_2 \tag{6.21}$$

where $\hat{\mathbf{f}}_E(\mathbf{x}(t), d(t)) = \hat{\dot{\mathbf{x}}} - \mathbf{f}_P(\mathbf{x}(t), u(t))$. The latter term is assumed to be known by the pursuer. The update block operates independently of the plant by sampling the current velocity prediction, $\hat{\dot{X}}$, at time step, n, and outputting a new tuple, $\mathcal{P} := \{\Sigma, \mathbf{m}(\mathbf{x})\}$, that becomes the control law. Therefore, over the course of the game there will be a sequence of updates, $\mathcal{P}_0, \mathcal{P}_1, \ldots, \mathcal{P}_j$ that result from updates to $\hat{v}_e$. The update specific to this problem will be given below.

The controller is given an initial control law, $\mathcal{P}_0$, before the start of the game using the objective library $\mathcal{V}$. In the Homicidal Chauffeur scenario, the pursuer can either (1) decrease the distance to the evader or (2) increase the distance to the evader. This is represented as

$$\mathcal{V} = \left\{ \begin{matrix} \min_{u \in U} \|\mathbf{x}^{t+\Delta t}\| \\ \max_{u \in U} \|\mathbf{x}^{t+\Delta t}\| \end{matrix} \right\} \tag{6.22}$$

The objectives in this library minimize and maximize the distance, respectively, to the evader. The objective is evaluated over some finite horizon, $\Delta t < \infty$, which is not dependent on the system step time, dT.

Because no information is known by the pursuing agent, we initially assume that the evader dynamics have no disturbance effect on the pursuer. Therefore, we assume $\hat{v}_e = 0$, and the initial control mapping only takes into account that the pursuer must first increase the distance to the evader until the agent is out of the turning radius. Therefore, $\mathcal{P}_0 =$

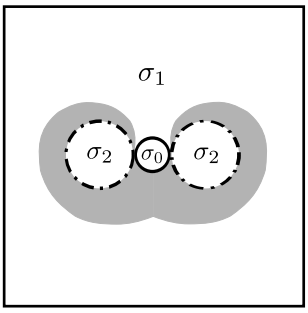


Figure 6.2: The initial partitions are represented by the dashed lines, and the gray region depicts the possible states around the initial boundary that may need to be included in σ_2.

$\{\Sigma_0, \mathbf{m}(\mathbf{x})\}$ takes the form of

$$\Sigma_0 = \left\{ \begin{array}{l} \sigma_0 = \mathcal{T}, \\ \sigma_1 = X \backslash \{\sigma_0 \cup \sigma_2\}, \\ \sigma_2 = \mathbf{x} \in X, \sqrt{(x_1 \pm r)^2 + x_2^2} < r - l \end{array} \right\}$$

and

$$\mathbf{m}(\mathbf{x}) = \begin{bmatrix} \min_{u \in U} \|\mathbf{x}^{t+\Delta t}\| \\ \max_{u \in U} \|\mathbf{x}^{t+\Delta t}\| \end{bmatrix}$$

Figure 6.2 shows what this looks like when applied to the state space. The dashed lines in the figure represent the initial control law configuration, which is based around the turning radius of the pursuer. The solid line represents the target of the pursuer. However, this is not a static configuration, because it will be updated according to measurement inputs, $\hat{v}_e$. Therefore, the control update must identify those states, such as the ones shaded in gray, that will need to be changed in the updated configurations, $\mathcal{P}_{j>0}$, so that the target is still reachable by the pursuer.

Once the initial mapping is set, the problem becomes ensuring that the pursuer can still reach the target even though the evader has a measured speed. This means ensuring the

following criteria[2]:

1. All trajectories starting in σ_2, leave σ_2.
2. All trajectories starting in σ_1, leave σ_1.
3. All trajectories starting in σ_1, do not enter σ_2.

Criteria 1 and 2 are satisfied as long as $v_p > v_e$. However, criterion 3 is dependent on the turning radius as well. As each measurement of the evader speed is received, criterion 3 can be evaluated by solving the HJB equation posed in Proposition 4 [57]. The benefit of this proposition is it simplifies a minimax problem into an optimal control problem, reducing the number of control combinations to test. Unlike the evader's control, the update only has to take place inside a partition. For a fast solution to the optimal control problem, the authors use the fast marching formulation from [34]. However, this does require a grid approximation of the state space, so the resolution of the mesh must be fine enough to approximate the curves of the partition boundaries.

Proposition 4. *Assume an invariant set $\mathcal{D}$, a control law $u(\mathbf{x}) = \mathcal{M}(\mathbf{x})$ and the value, $V(\mathbf{x})$, is the solution to the partial differential equation (Dynamic Programming Principle).*

$$0 = \min_{d \in D} \left[g(\mathbf{x}, u, d) + \nabla V(\mathbf{x}) \mathbf{f}(\mathbf{x}, u, d) \right] \tag{6.23}$$

subject to the boundary condition

$$V(\mathbf{x}) = 0, \ \forall \mathbf{x} \in \mathcal{D} \tag{6.24}$$

[2]A graph theoretical justification is given for these criteria in [57]

and the constraints

$$g(\mathbf{x}, u, d) = \begin{cases} 0, \ \forall \mathbf{x} \in \mathcal{D} \\ (0, \infty), \ otherwise \end{cases} \tag{6.25}$$

The evader is able to guide the state to the set $\mathcal{D}$ *from an initial state* $\mathbf{x}^{t_0} \in X$ *when* $V(\mathbf{x}) < \infty$.

To evaluate criterion 3, first set $\mathcal{D} = \sigma_2$. Then, using Proposition 4, all the states where $V(\mathbf{x}) < \infty$ indicate where the evader may guide the trajectory into the turning radius before the pursuer can reach the target set, σ_0. For the Homicidal Chauffeur scenario, this is resolved by placing all the states where $V(\mathbf{x}) < \infty$ into σ_2. Thus, as the evader's speed estimate increases, the partition, σ_2, will grow to include states shown in the gray region of Fig. 6.2.

6.4 Experimental Setup

To simulate the agents' movements in a controlled environment, two computers were used. On one computer, a Simulink model was created to model the movement of the robots through a planar environment using the evolution equation, Eqn. 6.1. This model simulated the entire system with an ODE45 continuous time solver except for the update equation for the pursuer. The update calculations were performed on a fitPC2, 1Mhz Intel Atom processor with 1GB RAM because it simulates the processing time as it would be experienced on a real autonomous platform. Also, separating the calculations across computers ensures that the environment model is not interrupted or slowed to allow the the pursuer computations to finish. Otherwise, this would result in an unfair advantage for the pursuer. The models on both computers communicate using UDP over a wireless network. The environment model sends an estimate, $\hat{v}_e$, and the control update sends $\mathcal{P}$. For the environment model, it was

imperative that the simulation time match wall-clock time. Otherwise, the environment model would evolve much faster than the pursuer update calculation, which would not be realistic. To solve this problem, the "Set Pace" block in the aerospace blockset was used to regulate the environment model.

The vehicle parameters chosen for this study are: $r = 1$, $v_p = 2$, and $v_e = 1$. In the update calculations, the state space was bounded on the interval $[-10, 10]$ for each state x_1, x_2. The state space was approximated by a 51×51 node grid for the fast marching calculation. The model time step, dT, is $0.1s$. The control updates were set to run as fast as possible, with a $0.1s$ pause in between each update. The realized control update time depends on the measurement value, $\hat{v}_e$, where a larger value causes a longer calculation time. However, previous work [57] has shown that the update time for this scenario is on the order of $1s$. The variance of the white noise for the evader's position measurement was chosen to be $\zeta = 0.3$. The process noise coefficient, γ, for the Kalman filter was set to 10. The measurement noise coefficient, λ, was set to 40. The radius of the capture set, $\mathcal{T}$, defined by Eqn. 6.8 is chosen to be $l = 0.75$.

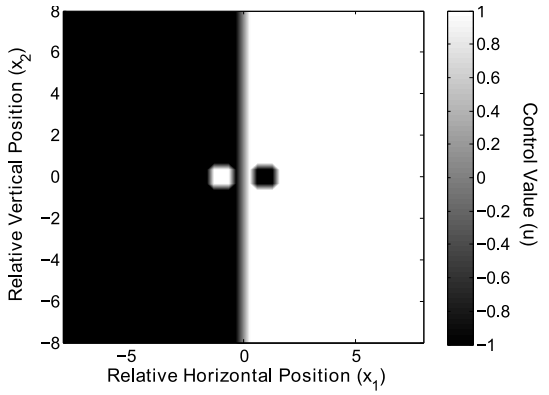
6.5 Results

In this section, results are given first as a qualitative example of the control update for one set of initial conditions as verification that the controller is working as expected. Then, a quantitative analysis is given comparing the times-to-capture of the scenario implementing the switched objective with perfect information to the scenario with perfect information available to both agents.

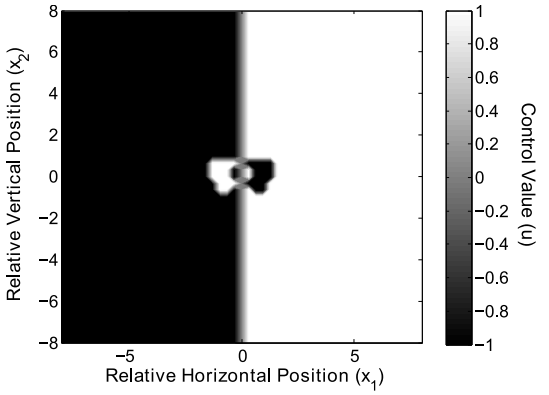
6.5.1 Trajectory Analysis

Here, we show the impact of the switched controller on the trajectory of the state evolution and the construction of the control law with the update block. The initial state chosen for this is $\mathbf{x}^{t_0} = (2, 0)$, where it is possible for the evader to traverse to the pursuer's target radius if proper actions are not taken. Figure 6.3 shows how the control law changes with the estimated speed of the evader. The initial control law shown in Fig. 6.3a shows the initial control mapping in terms of $u(\mathbf{x})$, where the evader's speed is assumed to be 0. Then, the first speed measurement, $\hat{v}_e = 0.0018$, processed by the control update shows in Fig. 6.3b that even with such a small value, there are some states around the turn radius where the evader can steer the trajectory into the turn radius. This is shown by the enlarged area of the states representing partition σ_2. By the next measurement, the Kalman filter has approached the approximate value of v_e and σ_2 takes the shape of the discontinuity commonly seen in the value function of the Homicidal Chauffeur (see [106]) seen in Fig. 6.3c. With this update, the pursuer can successfully prevent the evader from stalling capture by entering the turning radius. Once the transition has been made from σ_2 to σ_1 and the measured speed of the evader is correct, then the pursuer will succeed in capture. However, the control updates will continue to be calculated to ensure that the measured speed of the evader does not change unexpectedly. Once reached, the measured speed of the evader hovers around the true speed of 1. Figures 6.3d and 6.3e show the control law produced for further measurements.

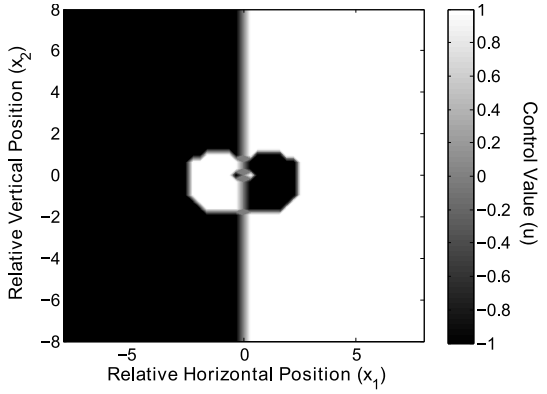
Figure 6.4 shows a sample trajectory produced by the switched control starting from $\mathbf{x}^{t_0} = (2, 0)$. The relative coordinate system used in Eqn. 6.1, is shown in Fig. 6.4a. The same trajectory displayed in absolute coordinates in Fig. 6.4b, show how the pursuer and evader would move if the reference plane was fixed at every time step. Beginning with Fig. 6.4a, the trajectory starts at $\mathbf{x}^{t_0}$ (A). Given the control scheme, $\mathcal{P}_0$, this point is located within σ_1, so the pursuer steers toward the evader to minimize the distance, seen by the solid line in



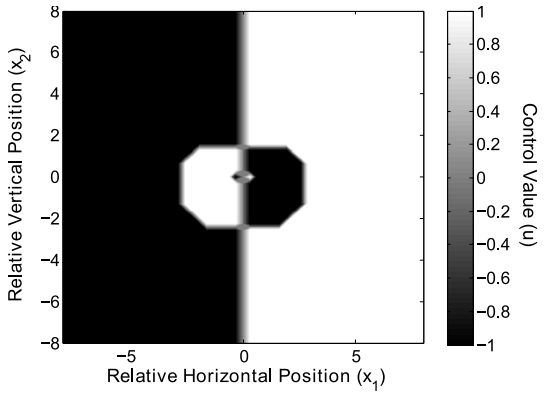
(a) $\mathcal{P}_0, \hat{v}_e = 0.0000$



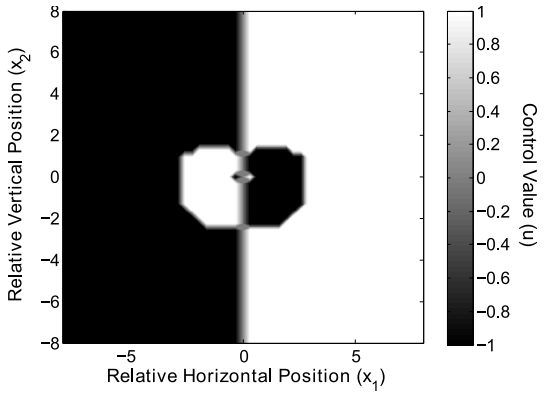
(b) $\mathcal{P}_1, \hat{v}_e = 0.0018$



(c) $\mathcal{P}_2, \hat{v}_e = 0.9212$



(d) $\mathcal{P}_3, \hat{v}_e = 1.3221$

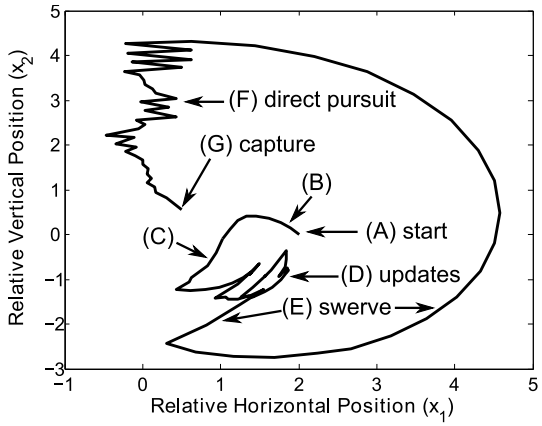


(e) $\mathcal{P}_4, \hat{v}_e = 1.3599$

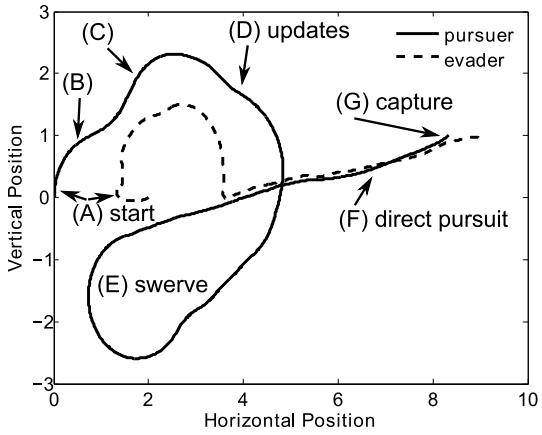
Figure 6.3: An example of the control updates starting from initial position, $\mathbf{x}^{t_0} = (2, 0)$, which is located within the pursuer turn radius.

Fig. 6.4b (B). However, the evader is able to guide the trajectory into the pursuer's turning radius, σ_2. Here, the pursuer then executes the maximizing strategy by steering away from the evader (C). Because the control law has not yet converged to the correct partition scheme, the trajectory exits into σ_1 prematurely, and the evader is able to once again steer into the turning radius (D). It is only when the control has converged to the correct scheme in Fig. 6.3c that the pursuer is able to successfully execute the "swerve" maneuver (E) [64]. Here, σ_2 has grown large enough such that the pursuer increases the distance to the evader such that it can turn around before the evader has the chance to re-enter σ_1. The chase concludes with a direct pursuit to capture (F,G), which appears noisy in the relative coordinates. This is an artifact of subtle changes in course by the pursuer to account for measurement noise in the evader's position.

The main qualitative result of implementing the switched control law is that the trajectory can become caught in a partition until updates reflect an accurate measurement of the evader's dynamics. This is evident in Fig. 6.4 where the pursuer is unable to make progress toward the capture set because the trajectory keeps re-entering σ_2 (D). Therefore, there is an additional time delay in this region until the control law is able to converge appropriately. Trajectories initiating in σ_1 where the evader cannot enter into σ_2 will be unaffected because the control law will use the correct controls from the initial control law. However, if the estimator outputs a larger estimate of the evader's speed than what is real, the effect will also be an additional time delay to reaching the capture set. This is because the pursuer will be forced to increase the distance more than what is necessary, and longer calculation times will appear in the control update block.



(a) Trajectory in Relative Coordinates



(b) Trajectory in Absolute Coordinates

Figure 6.4: The trajectories in (a) reduced coordinates and (b) absolute coordinates show how the update delay affects the trajectory of the pursuing agent.

6.5.2 Time-to-Capture Analysis

In this section we quantify the overall impact of the switched objective control and compare it to the case with perfect information where both agents use the minimax solution for determining their control law. To accomplish this, the value functions showing the time-to-capture as a function of state can be compared. Three cases are considered with different levels of information:

(i) The pursuer has perfect information, $(v_e, u(\mathbf{x}))$.

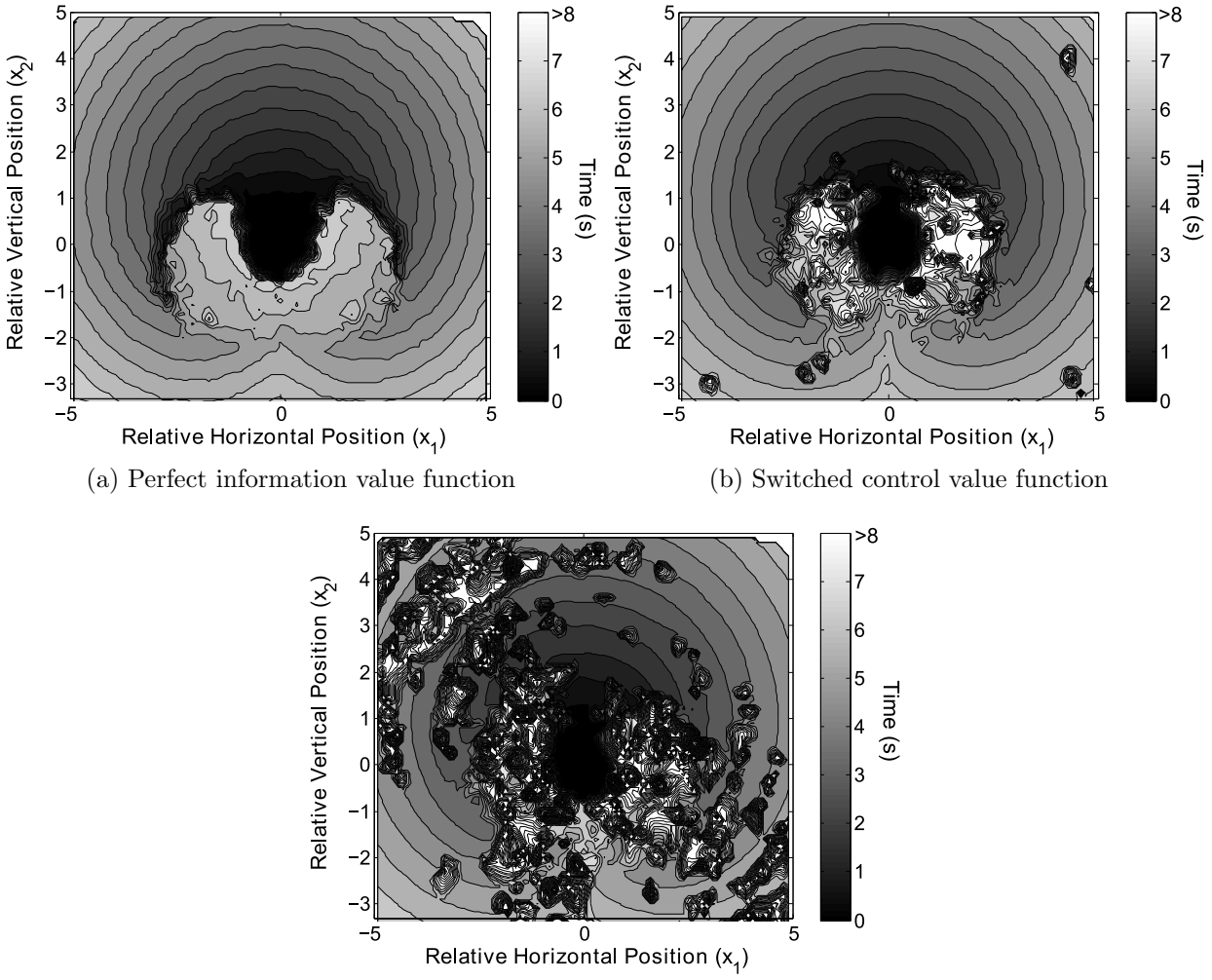
(ii) The pursuer has the evader's position for the control law, but must estimate the evader's speed parameter, $(\hat{v}_e, u(\mathbf{x}))$.

(iii) The pursuer must estimate the evader's position for the control law and estimate the evader's speed parameter, $(\hat{v}_e, u(\hat{\mathbf{x}}))$.

The value function for each case was created with a 2500 iteration Monte-Carlo experiment where the initial position, $\mathbf{x}^{t_0}$ is subject to a random uniform distribution. We then measured the time it took for capture to occur, if at all. In the following simulations, we show two main results: (1) The pursuer is able to capture the evader from the same initial states with the switched controller using imperfect information as the optimal controller using perfect information. (2) The time to capture increases from the optimal control to the switched control, but is centered around the region corresponding to the pursuer's turn radius.

The first comparison between cases (i) and (ii) is shown in Fig. 6.5. This comparison shows the amount of time increase attributed only to the calculations of the control update. Figure 6.5a shows case (i), which takes the form of the normal Homicidal Chauffeur value function. The calculation time increases as the state moves farther from the target, and the discontinuous region also displays higher times. Comparing to case (ii), illustrated in Fig. 6.5b, the effect of the controller on the calculation time is seen to be more prevalent near the turn radius of the pursuer. This effect is shown by the increased area of white in this region. Increased times are predominately located in this area because the pursuer cannot exit this partition until the measurement of the evader's speed provides a correct value to the control update. On the contrary, the region outside of the turn radius exhibits similar time-to-capture values, because the controller does not need to be fully updated in order for the trajectory to leave this region. In Fig. 6.5b, scattered points of increased time are seen throughout the state space. This is an artifact of the discretization of the control, which

due to measurement noise, can sometimes cause the wrong control value to appear when the trajectory is close to the target. As a result, the evader is able to narrowly escape, leading to another round of chase before it is subsequently captured. This particular source of error in the capture time is attributed to a choice of grid resolution.



(a) Perfect information value function

(b) Switched control value function

(c) Switched control with estimates

Figure 6.5: The value function of the Homicidal Chauffeur is plotted with time-to-capture as a function initial state.

The L^1 error of the capture times between cases (i) and (ii) is 25.6%. This error, largely seen in the region corresponding to the turn radius, is expected to decrease as the evader's

speed decreases and increase otherwise. This is because, σ_2 varies with v_e. It is also expected that the capture error increase and decrease with the turning radius, r. The more agile the pursuer or less agile the evader, the better performance of the switched control.

We now consider case (iii) where the pursuer must also use estimates of the evader's position to input into the control law. Figure 6.5c corresponds to the time-to-capture value function produced by the system depicted in Fig. 6.1. This gives a realistic version of the game and the resulting times-to-capture. Qualitatively, it has the same features as cases (i) and (ii) with the exception that there appear to be a lot of points with very high capture times. These occur because of noise in the output of the Kalman filter. In areas away from the target this is not as critical, and the chase proceeds as normal with more noise in the control input. Near the target, a repeated miscalculation on the pursuer can enable the evader to temporarily escape. This case shows that despite noise appearing in the input to the control law and the control update, the pursuer is still able to capture the evader in finite time from the same set of initial states as case (i). However, by adding in noise of the evader's position, the pursuer may have to complete successive passes before capture.

6.6 Conclusions

In this work, we have presented a switched control law applied to the Homicidal Chauffeur game. The evader receives perfect information and the pursuer must estimate the evader's relative position and speed. The pursuer is given no knowledge of the evader before the start of the game, and must develop a control law while the game is played to capture the evader. In order to determine the effectiveness of the switched control updates, a Monte Carlo experiment with 2500 iterations was used to approximate the time-to-capture value function for three cases of varying degrees of information. Considering only the impact of the

switched control law, the time-to-capture in this scenario exhibits a 25.6% increase averaged across all of the initial conditions. This is mostly due to the inability of the trajectory to exit the turning radius of the pursuer until the controller has been updated with the proper control law. When the pursuer must estimate all evader parameters, the results show that the number of initial states with trajectories that lead to capture are the same as the case with perfect information. However, imperfect information may lead to multiple passes at the evader before capture is achieved.

Chapter 7

Application 1: Nonholonomic Vehicle Path Planning

The first application discussed in terms of the graph theoretical control law is a collision avoidance problem for two vehicles. Here the control law is calculated on the vehicle using a hierarchical approach. An approximation of the dynamics is given by assuming the Homicidal Chauffeur model and using this as a high-level path planning control law. Then, a low-level control law is used to guide the actual vehicle, with more complicated dynamics along the trajectory. Methods of implementation are given explicitly, and results are given showing the regions of the state space where success is likely and where it is not. Improvements to this control law formulation are centered primarily on improving the measurement system.

Submitted to Journal of Intelligent and Robotic Systems, 2011.

7.1 Introduction

One problem in the field of robotics and autonomous control is to design collision avoidance systems for unmanned vehicles. Developments in this area are beneficial for applications that include highway vehicle control [111,115], driver-assisted autonomous vehicles [43], and

unmanned aerial (or underwater) vehicles [121]. The particular scenario that we address is designing a controller to prevent vehicular collision while traveling at speed. An example of such a threat is a vehicle that enters into the immediate path of oncoming traffic by making a left turn. As communication and infrastructure improves in the areas of traffic monitoring [66], GPS [18], and instrumented road technology [67], an autonomous controller becomes a viable option to prevent these types of collisions.

Some prior approaches to the collision avoidance problem have included coding human actions into fuzzy logic [79], creating potential fields for control [26], and using differential game-theory [89]. In the latter approach, a worst-case scenario is assumed. Using terminology from pursuit-evasion games, the agent seeking to avoid collisions is the evader. The other agent is the pursuer, who actively tries to cause a collision. This worst-case assumption adds robustness to the control, because the pursuer's strategy is removed from consideration. Regardless of the pursuer's actions, the outcome of the scenario depends on the dynamic parameters of the agents, e.g. speed ratio, turn radius, etc. If the pursuer does not play the worst-case control, then the evader will perform no worse with respect to avoiding a collision. A past approach [89] has used a neural network to provide a real-time control based on the Stackleburg equilibrium of the Game of Two Cars [87]. The neural network is trained off-line on a cluster of processors. Our approach is similar, in that we also consider collision avoidance as a zero-sum maneuver where a gain for one agent is a loss for the other with respect to a defined cost function such as time, separation distance, etc. This involves finding a saddle-point solution for the controls of both agents, such that if either agent deviates from this control they will perform worse with respect to the cost [108](pp. 380-381). However, we improve the versatility of implementing the control by solving the Hamilton-Jacobi-Isaacs (HJI) equation on-line, as the scenario unfolds. This allows for control solutions to be developed for different types of pursuers as they are encountered. For

example, an evader might react differently for a small car compared to a large truck.

The HJI equation is not practical to solve analytically because it is difficult to find a solution. Numerical methods [34, 42, 73, 106] can be used, but are subject to increased computational times with limited sensory information [17], the curse of dimensionality [84], computational efficiency [127], and adaptability to learned evader dynamics [47]. However, partitioning algorithms [58] can generate a solution to the HJI equation quickly on-line with limited sensory information [59]. This method for solving the HJI equation works by partitioning the state space and assigning local feedback controls to each partition. An update algorithm is used to alter the partition and feedback control structure based on a graph theoretic criterion [58]. In doing so, infinite horizon calculations that normally receive support over the entire state space are now contained within subsets of the state space, and the direction of the control updates proceed in the direction that the agent is traveling.

In order to implement the controller on-line, path planning is used to guide the evader past the pursuer and back into its normal travel path. This collision avoidance scenario is decomposed into two parts: those that are zero-sum (governed by the HJI equation) and those that are not. The zero-sum parts include the pursuit-evasion portion of the control strategy to guarantee collision avoidance. The nonzero-sum parts ensure the control does not violate travel constraints such as remaining in the road, channel, etc. The zero-sum elements of the game are solved using a model of the dynamics to find the controls using the partitioning algorithm. There are several choices of games from which to model the dynamics of the agents such as Game of Two Cars discussed previously or the Lifeline game [64](pp. 255-260). Here, the Homicidal Chauffeur is used in this work because it is a classic differential game that has simple dynamics, a reasonable approximation of the agents' motion, and can be quickly solved using the graph theoretic control algorithm. A similar approach was used for building stochastic games [103] and two-target games [51]. Once the game is solved using

the graph-theoretic control, the resulting partition-feedback control structure is combined with the nonzero-sum elements using a receding horizon control [83] scheme. Although not entirely representative of realistic dynamics, this pursuit-evasion model achieves high-level path planning, while a low-level tracking control keeps the actual vehicle with more complex dynamics on the generated path.

This paper proceeds by first giving an overview of the problem and high-level solution techniques. In particular, the zero-sum part of the problem is simplified by reducing the HJI equation to the simpler Hamilton-Jacobi-Bellman (HJB) equation using graph theoretic techniques [58]. Next, we show how a receding horizon control captures both the constraints of the game and the zero-sum pursuit-evasion control model. Then, a test platform that implements the controller is described in full. Finally, we conclude by showing trials performed on laboratory robotic vehicles using indoor GPS and on-board computers for strategic decisions. This shows that this control structure can be implemented on a simple processing system with noisy measurements and remain effective. Future work by the authors will apply this methodology to more quickly evolving systems using dedicated hardware and higher resolution measurement techniques.

7.2 Theoretical Formulation of the High-Level Controller

This section discusses the high-level controller for path planning. The construction of the high-level controller is detailed by presenting the individual components of the problem. The evader will be kept within the boundaries of the game by using conventional optimal control, and the collision avoidance maneuver will be calculated by solving the HJI equation

associated Homicidal Chauffeur game, which approximates the dynamics of the agents. The path generated by this control will then be sent to a low-level controller to be tracked by the agent with more complicated dynamics.

7.2.1 Agent Dynamics

The dynamics of the agents are separated into absolute dynamics and relative dynamics. The absolute dynamics are formulated with respect to the playing area of the game and represents the permissible states in which the agents may travel. The relative dynamics are formulated with respect to the evader's position relative to the pursuer. This set of dynamics is used to reduce the number of calculations needed to solve the Homicidal Chauffeur game.

Absolute Coordinates

The agents' dynamics are approximated by the well-known Homicidal Chauffeur differential game [64]. The evader with coordinate system, $X \subset \mathbb{R}^2$, tries to avoid a collision with a pursuer. A state, $x = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T \in X$, corresponds to the absolute position of the evader in X where the origin is fixed with respect to a domain $\Omega \subset X$. Here, Ω represents the playable area of the game such as a road, channel, etc. The evader chooses a control $d \in D$. The convex set, $D \subseteq [0, 2\pi)$ represents the instantaneous direction component of the velocity with fixed speed, v_e. The resulting dynamic equations for the evader are

$$\dot{x} = f_E(d(t)) = \begin{Bmatrix} v_e \sin(d(t)) \\ v_e \cos(d(t)) \end{Bmatrix}, \tag{7.1}$$

which represent that this agent is able to turn instantly in any direction available in D. Likewise, the pursuer has a coordinate system $Y \subset \mathbb{R}^3$. The state $y = \begin{bmatrix} y_1 & y_2 & y_3 \end{bmatrix}^T \in Y$

represents the absolute position of the vehicle (y_1, y_2) with an origin also fixed with respect to Ω. The appended state, y_3, represents the angular heading of the agent. The orientation of the heading is fixed with respect to y_1 and y_2. The pursuer is given a control $u \in U$. The convex set $U \subseteq [-1, 1]$ corresponds to the rate of turn of the pursuer to the left (negative) or right (positive). The minimum turn radius is governed by the parameter, r_p, and the speed of the pursuer is the parameter, v_p. The resulting dynamic formulation is

$$\dot{y} = f_P(y(t), u(t)) = \begin{Bmatrix} v_p \sin(y_3(t)) \\ v_p \cos(y_3(t)) \\ \frac{v_p u(t)}{r_p} \end{Bmatrix} \tag{7.2}$$

which corresponds to a typical car with fixed turning radius. However, the turning radius can also be a function of the vehicle speed, $r_p(v_p)$, in some applications. The dynamics of the agents are exogenous, in that the trajectories, $x(\cdot)$ and $y(\cdot)$, of both agents are completely independent of each other's control input.

The absolute coordinates are used to reference the agents' position in the playable domain Ω, and will be used in the nonzero aspects of the control. Defined in the absolute coordinate system is the target set, $\mathcal{T}_a \subset \Omega$. This is the set of states that the evader tries to reach in finite time, and the game concludes in favor of the evader when this set is reached. Because Ω represents the acceptable travel area, the game is over in favor of the pursuer if the agent crosses the boundary, $d\Omega$. It is assumed that the evader has perfect knowledge of this domain before the start of the game. In practice, this would be measured and estimated as $\bar{\Omega}$ (e.g. measuring road boundaries). There are a variety of techniques involving LiDAR and image processing making progress toward this end [11, 46, 50, 97, 116, 125], and methods such as those cast in a river navigation problem [132] show how the domain can be updated with measurements. However for this work, we assume perfect information of the domain, $\bar{\Omega} = \Omega$.

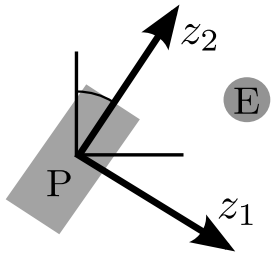


Figure 7.1: Orientation of the reduced coordinate system, Z.

Relative Coordinates

A relative coordinate system is now given that represents the position of the evader relative to the pursuer. This coordinate system is used to reduce the number of dynamic equations involved in finding the saddle-point solution for the zero-sum part of the problem. This decreases the complexity of the problem and allows for a faster solution which is necessary for a real-time implementation. The third coordinate system $Z \subset \mathbb{R}^2$ is formed by combining the absolute dynamics into a reference frame that moves with the pursuer. The derivation of the dynamics for the Homicidal Chauffeur game can be found is given by Isaacs [64](pp. 28-30), where the origin of Z is the pursuer's absolute position and the positive vertical axis, z_2, is fixed to the forward direction of the pursuer's velocity as shown in Fig. 7.1. To account for the pursuer's heading, the evader's control is given by the transformation, $\tilde{d} = d - y_3$. The evader now chooses $\tilde{d} \in \tilde{D}$ from the convex set, $\tilde{D} \subseteq [0, 2\pi)$, and the pursuer's control remains the same as before. The relative dynamic equations for the Homicidal Chauffeur game are given by

$$\dot{z} = \tilde{f}(z(t), u(t), \tilde{d}(t)) = \left\{ \begin{array}{c} -\frac{v_p z_2(t)}{r} u(t) + v_e \sin(\tilde{d}(t)) \\ \frac{v_p z_1(t)}{r} u(t) - v_p + v_e \cos(\tilde{d}(t)) \end{array} \right\} \tag{7.3}$$

where the parameters, v_p, v_e and r retain their original meanings.

To interface the relative and absolute coordinate systems, a transformation, $\mathcal{Y}$ is given so the

positions of the pursuer and evader are mapped into the relative coordinate system, Z. The inverse of this function, $\mathcal{Y}_E^{-1}$, transforms the motion of the evader in relative coordinates to absolute coordinates in X. Likewise, $\mathcal{Y}_P^{-1}$ uses the relative motion of the pursuer and transforms it to absolute coordinates in Y. These functions are [88],

$$\mathcal{Y}(x,y) = \left\{ \begin{array}{c} (x_1 - y_1)\cos(y_3) - (x_2 - y_2)\sin(y_3) \\ (x_1 - y_1)\sin(y_3) + (x_2 - y_2)\cos(y_3) \end{array} \right\} \tag{7.4}$$

$$\mathcal{Y}_P^{-1}(y(t_0), u(\cdot)) = \left\{ \begin{array}{c} y_1(t_0) + \int_{t_0}^{t} \sin(y_3(\xi))d\xi \\ y_2(t_0) + \int_{t_0}^{t} \cos(y_3(\xi))d\xi \\ y_3(t_0) + \int_{t_0}^{t} u(\xi)d\xi \end{array} \right\} \tag{7.5}$$

$$\mathcal{Y}_E^{-1}(x(t_0), y_3(\cdot), \tilde{d}(\cdot)) = \left\{ \begin{array}{c} x_1(t_0) + \int_{t_0}^{t} \sin(\tilde{d}(\xi) + y_3(\xi))d\xi \\ x_2(t_0) + \int_{t_0}^{t} \cos(\tilde{d}(\xi) + y_3(\xi))d\xi \end{array} \right\} \tag{7.6}$$

The transformation of the trajectories from the absolute coordinate systems to the relative coordinate system, $\mathcal{Y}(x,y)$, does not require knowledge of the control strategy and can be performed on-line as the trajectories evolve. The inverse transformation is more involved, because it requires the entire control trajectory from the reduced system and the set of initial conditions in the absolute coordinate system to perform the integration.

7.2.2 Time-Optimal Control Generation

In this work, a successful collision avoidance maneuver is one that (1) guides the evader to a predefined target, $\mathcal{T}_a$, when the pursuer is not a threat and (2) avoid the pursuer within the bounds of the playing area, Ω. These two problems are presented in the following sections. The controls used to generate the paths for the maneuver are based on time-optimality, because every possible control trajectory available to the agent is evaluated.

This is beneficial because the set of states from which the evader can successfully reach $\mathcal{T}_a$ is known to be those where the time-optimal value function is finite, and the time-optimal control maximizes the number of states in this set [58]. The approach of this work is evaluate the two tasks sequentially by first avoiding the pursuer and then returning to $\mathcal{T}_a$. If there is no pursuer present, then the agent proceeds to just reach $\mathcal{T}_a$. Due to its sequential structure, the overall control law for the maneuver is composed of time-optimal control laws for each problem. The following four sections describe how these are constructed. Namely, four time-optimal value function and control law pairs will be evaluated:

1. **Time-optimal guidance to the steady state target set, $\mathcal{T}_{ss}$:** The steady state target set, $\mathcal{T}_{ss} \subset X$, is a "desired" path such as a road or waterway, where the evader should travel when no pursuer is present. It also has the property $\mathcal{T}_{ss} \supset \mathcal{T}_a$, so that $\mathcal{T}_a$ is a member of the steady state target set. Here, the evader uses the control law, $d^*_{ss}(x)$, to steer the trajectory to $\mathcal{T}_{ss}$ in time equal to the time-optimal value function, $V_{ss}(x)$.

2. **Homicidal Chauffeur Model for Collision Avoidance:** The evader seeks to *avoid* entering a target set, $\mathcal{T}_P \subset Z$, which is a neighborhood of states around the pursuer that indicate capture. Therefore, the control law $d^*_{hc}(z)$ steers the trajectory away from $\mathcal{T}_P$ in time equal to $V_{hc}(z)$, a time-optimal value function that the evader desires to equal infinity.

3. **Partitioned Solution to the Homicidal Chauffeur Model:** The problem above is reformulated using state partitions to find a set of states, $\mathcal{T}_E \subset Z$, the evader tries to enter and is guaranteed to avoid a collision with the pursuer. Intended to have the same effect as $d^*_{hc}(z)$, this problem of collision avoidance builds a control law, $d^*_{-hc}(z)$, which steers the trajectory to $\mathcal{T}_E$ in time $V_{-hc}(z)$.

4. **Control Law for the Entire Collision Avoidance Maneuver:** Lastly, a time-dependent control law, $d^*(x,t)$, is formed from $d^*_{ss}(x)$ and $d^*_{-hc}(z)$ to complete the entire collision avoidance maneuver ending in the target set, $\mathcal{T}_a$, in time equal to $V_c(x,t)$, a time-dependent function. Here the target set $\mathcal{T}_a$ is defined as the set of states in $\mathcal{T}_{ss}$ that are not reachable by the pursuer.

The choice of coordinate systems for the two tasks, X and Z, is a result of simplifications by reducing the number of dynamic equations, particularly when solving the Homicidal Chauffeur game. The states can always be related through the transformations in Eqns. 7.4, 7.5, and 7.6. The construction of these control laws and value functions are given explicitly along with associated properties.

Time-Optimal Control with no Oncoming Vehicle

For this part of the collision avoidance scenario, we consider the evader's motion in the absolute coordinate system, X. This control guides the evader in the absence of a pursuer where it is desired to follow the steady state path or return to it. Since it is not desirable for the agent to deviate from this path for any length of time, we use a time-optimal control to guide the evader from any initial point $x(t_0) \in \Omega$ to a steady state target $\mathcal{T}_{ss} \subset \Omega$ which is a set of states within the playing area of the state space. For this part of the scenario, we assume that the choice of controls given to the evader is limited to $D \subset [d_{min}, d_{max}] \subset [0, 2\pi)$. The bounding parameters for the evader's direction limit the lateral motion that the agent has with speed, v_e. This control set is intended to achieve subtle changes to the evader's trajectory as expected in passenger vehicles.

In general, we seek to solve a nonlinear optimal control problem with running cost, $g(x)$,

and cost functional

$$J(x(t_0), d(\cdot)) = \int_{t_0}^{\infty} g(x(t))dt \tag{7.7}$$

where time-optimality of a control is achieved by setting $g(x) = 1, \forall x \notin \mathcal{T}_{ss}$. The control strategy is solved before the start of the scenario because it is a static solution, but can also be solved on-line if updates to $\bar{\Omega}$ are needed [132]. The problem is defined as follows:

Definition 6. *Assume an invariant target set $\mathcal{T}_{ss}$, a control law $d(x)$ and the value, $V_{ss}(x)$, is the solution to the partial differential equation (PDE) subject to the dynamics of Eqn. 7.1.*

$$0 = \min_{d \in D} \left[g(x) + \nabla V_{ss}(x) f_E(d) \right] \tag{7.8}$$

subject to the boundary condition

$$V_{ss}(x) = \begin{cases} 0, \ \forall x \in \mathcal{T}_{ss} \\ \\ \infty, \forall x \in d\Omega \end{cases} \tag{7.9}$$

and running cost

$$g(x) = \begin{cases} 0, \forall x \in \mathcal{T}_{ss} \\ \\ 1, otherwise \end{cases} . \tag{7.10}$$

The time-optimal control law is

$$d^*_{ss}(x) = \arg\min_{d \in D} \nabla V_{ss}(x) f_E(d), \tag{7.11}$$

and will guide the agent from state, x, to $\mathcal{T}_{ss}$ in minimal time given by $V_{ss}(x)$.

This problem definition results from the control law - value function duality given by [15] (pp. 111). Specific to this application, the boundary condition represents that no time is

required to reach the target set if already located inside it. Infinite time is needed to reach the target set if at the boundary, because the vehicle is assumed to be disabled. Once the value function $V_{ss}(x)$ is calculated, a receding horizon controller [83] can be used to solve for $d^*_{ss}(x)$ as the agent completes the evasive maneuver. This control guides the evader back to the steady state path using a control law that forms the time-optimal value function, V_{ss}. However, by establishing the control set, D, and sacrificing time optimality by adjusting the running cost, $g(x)$, the value function needs to ensure that the evader does not veer toward the pursuer in an attempt to return back to the path. A simple solution is to maintain the same heading until the pursuer has completely passed the evader.

Time-Optimal Collision Avoidance Maneuver: The Homicidal Chauffeur

When a pursuer is present, the evader assumes it will actively try to cause a collision in minimal time. This scenario is approximated by the Homicidal Chauffeur, where the evader attempts to flee the pursuer by choosing controls that increase the time until collision. Likewise, the pursuer chooses controls that decrease this time. It has been shown in [64] that a unique minimax saddle-point exists in the choice of controls. If the resulting capture time is infinite, then a collision will not occur.

Remark 2. *If the pursuer does not actively try to cause a collision by playing sub-optimally with respect to the time-optimal minimax solution to the Homicidal Chauffeur game, then the evader's worst-case control will only improve with respect to increasing the time until collision [64].*

Agent trajectories in the Homicidal Chauffeur game evolve in the reduced coordinate system, Z. The dynamics follow Eqn. 7.3, and the starting point of a trajectory, $z(t_0)$ is found using the transformation $z(t_0) = \mathcal{Y}(x(t_0), y(t_0))$. The set of available controls is

$\tilde{D} \subset [\tilde{d}_{min}, \tilde{d}_{max}] \subset [0, 2\pi)$, where the bounding parameters represent the maximum safe swerve maneuver that can possibly be achieved by the vehicle. Although instantaneous changes of direction are unrealistic for most vehicles, the actual controls needed to accomplish these avoidance maneuvers are not very drastic and can be approximated by $\tilde{D}$. The solution to this game is a value function that can be found by solving the minimax HJI equation. Here a set of states designated as $\mathcal{T}_P \subset Z$ is a ball of radius l centered around the pursuer. The output of this PDE is the value function $V_{hc}(z)$ that represents the time to reach $\mathcal{T}_P$ from each state $z \in Z$. If $V_{hc}(z) = \infty$, then despite any possible control strategy by the pursuer, there exists a strategy, $d^*_{hc}(z)$, with which the evader can escape. The solution to this game is defined as follows.

Definition 7. *The running cost $V_{hc}(z)$ is the solution to the PDE (Dynamic Programming Principle) that finds the control functions $u(\cdot)$ and $\tilde{d}(\cdot)$ to minimize and maximize the cost $J(z, u, \tilde{d}) = \int_0^\infty g(z(t), u(t), \tilde{d}(t))dt$, where $g(z(t), u(t), \tilde{d}(t))$ is a time-optimal running cost subject the to dynamics of Eqn. 7.3. Assume that the pursuer tries to steer the reduced state, z, to an invariant target set, $\mathcal{T}_P$, then the optimal control solution to the Isaacs equation*

$$0 = \min_{u \in U} \max_{\tilde{d} \in \tilde{D}} \left[g(z, u, \tilde{d}) + \nabla V_{hc}(z) \tilde{f}(z, u, \tilde{d}) \right] \tag{7.12}$$

subject to the boundary condition

$$V_{hc}(z) = 0, \ \forall z \in \mathcal{T}_P \tag{7.13}$$

and running cost

$$g(z, u, \tilde{d}) = \begin{cases} 0, \ \forall z \in \mathcal{T}_P \\ 1, \ otherwise \end{cases} \tag{7.14}$$

is the control

$$d^*_{hc}(z) = \arg\max_{\tilde{d}\in\tilde{D}} \min_{u\in U} \nabla V_{hc}(z)\tilde{f}(z,u,\tilde{d}) \tag{7.15}$$

that maximizes the time for the trajectory to reach $\mathcal{T}_P$. For the pursuer,

$$u^*_{hc}(z) = \arg\min_{u\in U} \max_{\tilde{d}\in\tilde{D}} \nabla V_{hc}(z)\tilde{f}(z,u,\tilde{d}) \tag{7.16}$$

minimizes the time to capture.

There are multiple methods to solving the HJI equation [34,42,64,73,88,106]; however, these are susceptible to long computational times. By using time-optimality defined in Eqn. 7.14, these methods find the maximum number of states where the evader can escape. However, this classic formulation can be simplified so that this task may be accomplished on-line.

Collision Avoidance Maneuver: Solving using Partitions

Solving the Homicidal Chauffeur problem explicitly as a saddle-point problem using conventional methods is highly susceptible to long calculation times. Therefore, it is beneficial to simplify the problem by removing one of the agent's controls from HJI equation and reduce the problem to solving the simpler HJB equation. This new form of the Homicidal Chauffeur game can then be solved on-line using the partitioning methods in [58]. Using this methodology, there are two possible ways that the evader can avoid a collision: (1) the evader is faster than the pursuer, ruling out a chase, and (2) the evader is able to enter into the pursuer's turn radius. With collision avoidance, (1) is not of concern because the pursuer cannot turn around fast enough to catch the evader in a direct chase. Therefore, we focus on the case when the evader can cause (2). Using this approach, the state space can be partitioned according to Fig. 7.2 where partition $\mathcal{T}_P$ is the set of states where a collision

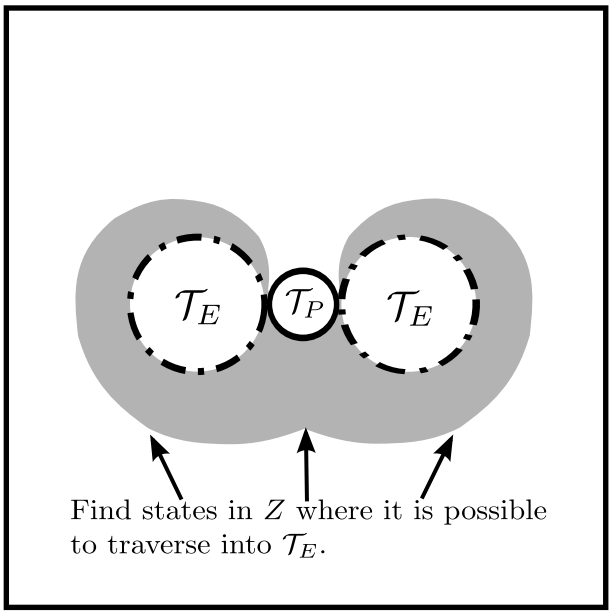


Figure 7.2: A partitioning scheme where the collision area is set as $\mathcal{T}_P$ and the pursuer's turning radii are set as $\mathcal{T}_E$. Using the HJB equation, it is possible to identify which states enter $\mathcal{T}_E$.

occurs and is called the pursuer's target set. Likewise, $\mathcal{T}_E$ is the set of states where a collision cannot occur inside the pursuer's turn radius.

The solution to the Homicidal Chauffeur can then be calculated in inverse fashion by finding all of the states $z \in Z$ where the evader can traverse to the target, $\mathcal{T}_E$, if the pursuer plays a control law, $u^*_{hc}(z)$. These states are represented by the gray region in Fig. 7.2. For this particular application, the problem simplifies to finding these states by assigning the worst-case control for the pursuer, $u^*_{hc}(z) = \text{sgn}(z_1)$ [64]. Here the state space is partitioned into two regions where the pursuer plays a specific control. Because $u^*_{hc}(z)$ is assumed to be known, this simplified version of the game can be solved using the HJB equation by removing the pursuer's control. The HJB representation of the Homicidal Chauffeur model is given by

Proposition 5. ***([58])*** *Assume an invariant set $\mathcal{T}_E$, and invariant set $\mathcal{T}_P$ where $\mathcal{T}_P \cap \mathcal{T}_E = \emptyset$, a pursuer control law, $u^*_{hc}(z)$, an evader control law $d_{-hc}(z)$ and the value, $V_{-hc}(z)$, is the*

solution to the PDE

$$0 = \min_{\tilde{d}\in\tilde{D}} \left[g(z,u,\tilde{d}) + \nabla V_{-hc}(z)\tilde{f}(z,u,\tilde{d}) \right] \tag{7.17}$$

subject to the boundary condition

$$V_{-hc}(z) = \begin{cases} 0, & \forall z \in \mathcal{T}_E \\ \infty, & \forall z \in \mathcal{T}_{\mathcal{P}} \end{cases} \tag{7.18}$$

and the running cost

$$g(z,u,\tilde{d}) = \begin{cases} 0, \ \forall z \in \mathcal{T}_E \\ 1, \ otherwise \end{cases} \tag{7.19}$$

The evader is able to guide the state to the set $\mathcal{T}_E$ from an initial state $z(t_0) \in Z$ when $V_{-hc}(z) < \infty$ when the optimal control

$$d^*_{-hc}(z) = \arg\min_{\tilde{d}\in\tilde{D}} \nabla V_{-hc}(z)\tilde{f}(z,u,\tilde{d}) \tag{7.20}$$

is used.

Proof. Using the HJB sufficiency theorem given in, [15](pg. 111), we know that $V_{-hc}(z)$ is a unique solution to Eqn. 7.17 if $d^*_{-hc}(z)$ attains a unique minimum in Eqn. 7.17 for all t, z. Because the set of controls, $\tilde{D}$, produces a convex set of trajectories, $\tilde{f}(z,u,\tilde{d})$, from every state z for any u, there is a minimum, $\tilde{d}^*$. Using the same theorem, $V_{-hc}(z)$, is the cost-to-go starting from a particular state, z, when $d^*_{-hc}(z)$ is played. Then, from [58], because $g(z,u,\tilde{d})$ is finite everywhere and 0 only on $\mathcal{T}_E$, if the trajectory never reaches $\mathcal{T}_E$, then $V_{-hc}(z)$ will tend toward infinity. As a result, when $V_{-hc}(z) < \infty$, then it is possible for the evader to reach $\mathcal{T}_E$. □

When $\mathcal{T}_E$ is equal to the pursuer's turning radius, $V_{-hc}(z)$ gives the time required for the

evader to escape from the pursuer. Therefore, if the evader chooses a control law $d^*_{-hc}(z)$, and if $V_{-hc}(z) < \infty$, the evader will be able to traverse to the pursuer's turning radius, and avoid a collision. Minimizing $V_{-hc}(z)$ in the evader's zero-sum part of the control allows the minimax solution to the HJI equation to reduce to the Bellman equation [58] by partitioning the state space and placing a known pursuer control in the partitions. This results in major calculation reductions so the control solution can be found on-line using a processor located on the agent.

Control for the Collision Avoidance Maneuver and Target Reachability

A control law is now given for the entire collision avoidance maneuver by avoiding the pursuer and returning to steady state. We first define an augmented value function, $V_c(x,t)$, which represents the time to capture of guiding the agent from an initial starting point, $x(t_0) \in \Omega$, to $\mathcal{T}_a \subset \Omega$. The target set, $\mathcal{T}_a := \{x \in \mathcal{T}_{ss} | y(t) \neq x, \forall t\}$ is the set of states in the steady state target set (evader's desired path) that are not reachable by the pursuer. For example, if the evader reaches $\mathcal{T}_{ss}$ after $\mathcal{T}_E$, then the it has reached $\mathcal{T}_a$. To incorporate this, the evasion maneuver occurs sequentially by entering the pursuer's turn radius, $\mathcal{T}_E$ at time t^*, and then returning to the steady state path, $\mathcal{T}_{ss}$, in separate maneuvers. The change in operating modes occurs at time t^*, and results in a time-dependent value function to reach $\mathcal{T}_a$,

$$V_c(x,t) = \begin{cases} V_{-hc}(z(x(t),y(t))) + V_{ss}(x(t^*)), & t \leq t^* \\ V_{ss}(x(t)), & t > t^* \end{cases} \tag{7.21}$$

where t^* is the first time the trajectory, $z(t)$ enters $\mathcal{T}_E$. We note that the coordinate systems can be related using the transformation of Eqn. 7.4, $z(t) = \mathcal{Y}(x(t),y(t)) = z(x(t),y(t))$. It follows that $x(t^*) = \mathcal{Y}_E^{-1}(x(t_0), y_3(\cdot), \tilde{d}(\cdot))$ where t_0 is the start of the game. Therefore, the

term, $x(t^*)$, is dependent upon the control choices $u(\cdot)$ and $\tilde{d}(\cdot)$ before t^*. The effect of these controls on the value function is the value associated with reaching $\mathcal{T}_a$ after the avoiding the pursuer. For example, a pursuer and evader control pair that results in successful escape from the pursuer, but lands the evader outside of the playable area, Ω, will result in an infinite time to capture because this corresponds to an evader loss when the function switches to minimizing $V_{ss}(x)$. Therefore, a conservative estimate for the maneuver completion time assumes a worst-case scenario for the pursuer's actions. Remark 2 shows that the maximum time for this to occur is when the pursuer plays its saddle-point minimum, u^*_{hc}, to force the most deviation from the steady state path.

Remark 3. *$\bar{V}_c(x,t)$ is the augmented value function generated when the pursuer plays the control $u^*_{hc}(x)$ and satisfies,*

$$V_c(x,t) \leq \bar{V}_c(x,t) \tag{7.22}$$

for any evader control, $d(\cdot)$, and pursuer control, $u(\cdot)$.

The control law for the entire collision avoidance scenario assumes the worst-case scenario where the pursuer plays u^*_{hc}. The final control is given as,

Proposition 6. *Assume the pursuer has a control strategy, $u(\cdot)$, that produces a trajectory, $y(\cdot)$ and Ω^* is a connected subset of Ω containing $\mathcal{T}_a$ and satisfying $\bar{V}_c(x,t) < \infty$. If the evader uses the control,*

$$d^*(x,t) = \begin{cases} d^*_{-hc}(z(x(t),y(t)), & \text{if } t \leq t^* \\ d^*_{ss}(x(t)), & \text{otherwise} \end{cases} \tag{7.23}$$

then the target, $\mathcal{T}_a$ is reachable from every initial point $x(t_0) = \Omega^*$.

Proof. If $\bar{V}_c(x,t) < \infty$, then both $V_{-hc}(z) < \infty$ when $t < t^*$ and $V_{ss}(x(t^*)) < \infty$. Proposition

5 shows that if $V_{-hc} < \infty$ then the time-optimal control $d^*_{-hc}(z)$ leads to the target $\mathcal{T}_E$ in some finite time, t^*. Likewise, Definition 6 shows that the time-optimal control, $d^*_{ss}(x)$, leads the agent to $\mathcal{T}_{ss}$ in finite time, $(t < \infty)$ when $V_{ss}(x) < \infty$. Then, the sequential use of the controls by playing $d^*_{-hc}(z(x(t), y(t)))$ and $d^*_{ss}(x(t))$ will keep the trajectory within Ω^* because $\bar{V}_c(x,t)$ decreases with $V_{-hc}(z(x,y))$ and $V_{ss}(x)$ when playing their respective controls so that $\bar{V}_c(x,t) < \infty$ remains true. Further, because $\mathcal{T}_E$ is required to be reached before $\mathcal{T}_{ss}$, then the agent reaches $\mathcal{T}_a$ in finite time. This holds for any choice of control by the pursuer because $V_c(x,t) \leq \bar{V}_c(x,t)$. □

We note that there are cases where the solution obtained by using the control $d^*(x,t)$ is sub-optimal with respect to the time-optimal game of reaching $\mathcal{T}_a$ due to reaching $\mathcal{T}_E$ and $\mathcal{T}_{ss}$ sequentially. However, time-optimality is preserved in each of these separate tasks. The control, $d^*(x,t)$, serves as the basis for producing a path that the vehicle can follow in order to conduct the evasive maneuver. If the dynamics of the actual vehicle are capable of following this path, and the initial state is in Ω^*, then $\mathcal{T}_a$ is reachable. In practice, Ω^* is not trivial to determine and may not be known unless every trajectory is evaluated because of the boundary constraints. Therefore, the controller used for the more complicated dynamics will take into account the term, $V_{ss}(x(t^*))$ to ensure that the avoidance maneuver remains in Ω^*. This will be accomplished by using a receding horizon control that predicts the future state of both agents and ensures that the trajectory satisfies the boundary constraints.

7.3 Experimental Setup

To demonstrate the control, a roadway scenario is given where a pursuer makes a left turn across a roadway into the immediate path of an oncoming evader. The pursuer does not actively try to intercept the evader, but the evader's control makes this assumption to prevent

a collision, if possible. The goal is to form a control so that a vehicle with more complicated dynamics can conduct maneuvers to follow the path generated by the high-level control with approximate dynamics. The robotic vehicles used to implement this controller consist of an IRobot Roomba 560 robot and an IRobot Create. Although the robots are capable of zero turn radius maneuvers, they were given a fixed speed and finite turn radius. The pursuer's dynamic equations remain the same as before, and the evader's actual dynamics are now given as

$$\dot{x} = f(x(t), d(t)) = \begin{Bmatrix} v_e \sin(x_3(t)) \\ v_e \cos(x_3(t)) \\ \frac{v_e d(t)}{r_e} \end{Bmatrix} \tag{7.24}$$

Each robot is given a separate Fit PC2 1GHz processor (1Gb DDR RAM) to process their control solutions. A diagram of the system interactions is given in Fig. 7.3. Distance measurements for each robot are obtained from the Cricket Indoor GPS system [113]. This system works by placing beacons that transmit both an electromagnetic and ultrasonic pulse on the ceiling of a room. Each robot is given a listener module to measure the distance to each transmitting beacon within range for state estimation. These state estimates are then passed to a path generator (high-level controller) that calculates the Homicidal Chauffeur game for the vehicle parameters. Then, a vehicle (low-level) controller ensures that the path is followed. In the following sections, the measurement and estimation equations are given in terms of the evader coordinates, X, but a similar approach can be applied to the pursuer with Y coordinates.

7.3.1 GPS Measurement

The room chosen for the tests has area dimensions $3.66m \times 5.79m$ $(12ft \times 19ft)$. Five ceiling mounted beacons, $\{b_1, b_2, b_3, b_4, b_5\}$, were given a constant power supply to ensure

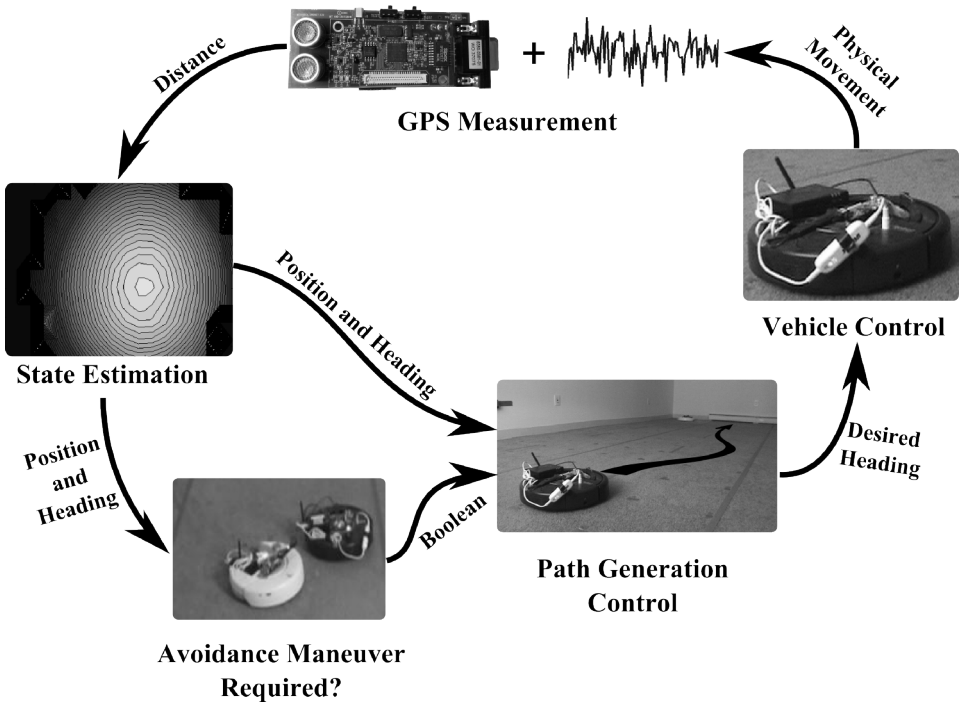


Figure 7.3: Diagram showing the interactions of the system components.

consistent measurements. The location for each beacon is shown in Fig. 7.4. The room was divided into equal spaced grids, shown by the cross-hairs, to calibrate the indoor GPS and account for imperfections in the room construction. The measurements output by a listener of the indoor GPS system is noted by Ψ. Likewise, the estimated value of Ψ at a particular state x is given by the empirical distance map, $h(x, v)$, where v is assumed to be Gaussian measurement noise. Each robot was placed at every node, and several distances were measured from the beacons within range of the listener to create an empirical distance map, $h_i(x)$ for beacon, b_i, as seen in Fig. 7.4. Due to radial symmetry, the inverse h_i^{-1}, is not one-to-one. Therefore, at least three beacons need to have overlapping sensor ranges for the state to be observable. Estimates of the measurement value at a particular state are given by,

$$h(x, 0) = \sum_{i=1}^{5} \mathbf{1}_{\Psi}(b_i)(h_i(x)) \tag{7.25}$$

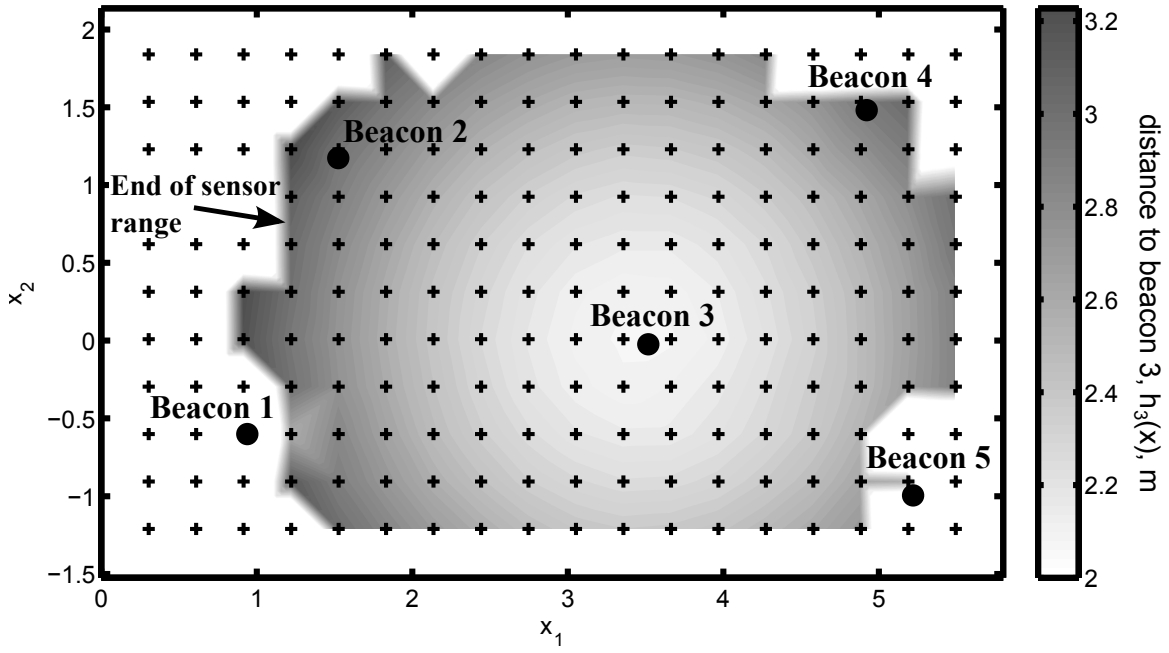


Figure 7.4: The sensor distance profile for beacon 3. The sensing region for this beacon nearly covers the entire floor area. Position is given in meters.

where $\mathbf{1}_{\Psi}()$ is the indicator function equal to 1 if the actual measurement, Ψ, is from beacon, b_i. Because $h_i(x)$ is formed from multiple averaged values at a particular location x, we assume that $v \to 0$.

7.3.2 State Estimation

State estimation for both agents is accomplished using an extended Kalman filter [92, 126] that receives distance measurements Ψ_n from the indoor GPS system. Likewise, these measurements are estimated using the empirical mapping, $h(x, 0)$ given by Eqn. 7.25. The output of the filter for both agents is full state information containing position and heading at time step, n. The time difference between measurements n and $n-1$ is dT. The state transition function for the evader with control d_n with Gaussian process noise, $w = \begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix}$, for

the filter is,

$$f(\hat{x}_{n-1}, d_n, w_{n-1}) = \hat{x}_{n-1} + \begin{bmatrix} (v_e + w_{1,n-1})\sin(\hat{x}_{3,n-1}) \\ (v_e + w_{2,n-1})\cos(\hat{x}_{3,n-1}) \\ (\frac{v_e}{r_e} + w_{3,n-1})d_n \end{bmatrix} dT \tag{7.26}$$

and reflects the dynamics given by Eqn. 7.24 with process noise due to uncertainty of the speed and turning radius parameters. The filter first approximates the state transition by assuming no measurement noise,

$$\underline{\hat{x}}_n = f(\hat{x}_{n-1}, d_{n-1}, 0) \tag{7.27}$$

where $\hat{x}_n$ is the estimate of the evader's position at time step n. The update for the state covariance, P_n is given by

$$\underline{P}_n = A_n P_{n-1} A_n^T + W_n Q_{n-1} W_n^T \tag{7.28}$$

where A is the Jacobian of the transition equation given as

$$A_{[i,j]} = \frac{\partial f_{[i]}}{\partial x_{[j]}}(\hat{x}_{n-1}, d_{n-1}, 0) \tag{7.29}$$

$$= \begin{bmatrix} 1 & 0 & v_e\cos(\hat{x}_{3,n-1})dT \\ 0 & 1 & -v_e\sin(\hat{x}_{3,n-1})dT \\ 0 & 0 & 1 \end{bmatrix} \tag{7.30}$$

and W is the process noise Jacobian,

$$W_{[i,j]} = \frac{\partial f_{[i]}}{\partial w_{[j]}}(\hat{x}_{n-1}, d_{n-1}, 0) \tag{7.31}$$

$$= \begin{bmatrix} \sin(\hat{x}_{3,n-1}) & 0 & 0 \\ 0 & \cos(\hat{x}_{3,n-1}) & 0 \\ 0 & 0 & d_{n-1} \end{bmatrix} dT. \tag{7.32}$$

Held constant throughout the game, the process covariance, Q, is estimated to be

$$Q = q \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 10 \end{bmatrix} \tag{7.33}$$

where q is a constant scalar. The equation used to find the Kalman gain is

$$K_n = \underline{P}_n H_n^T (H_n \underline{P}_n H_n^T + V_n R_n V_n^T)^{-1} \tag{7.34}$$

where, H_n is a matrix that transforms the state vector into measurement states. It takes the form,

$$H_{k,[i,j]} = \frac{\partial h_{[i]}}{\partial x_{[j]}}(\underline{\hat{x}}_n, 0) \tag{7.35}$$

$$= \begin{bmatrix} \frac{\partial h}{\partial x_1}(\underline{\hat{x}}_n) \\ \frac{\partial h}{\partial x_2}(\underline{\hat{x}}_n) \\ 0 \end{bmatrix} \tag{7.36}$$

where the gradients, $\frac{\partial h_{[i]}}{\partial x_{[j]}}$ are numerically approximated. $V = 1$ is the Jacobian of the measurement noise and R is a measurement noise covariance matrix that takes the form

$$R = \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{7.37}$$

where λ is a constant scalar. The original state estimate of Eqn. 7.27 is then updated using

$$\hat{x}_n = \underline{\hat{x}}_n + K_n \left(\Psi_n - h(\underline{\hat{x}}_n, 0) \right) \tag{7.38}$$

where Ψ_n is the position measurement of the evader with noise added. The updated states are the output of the estimator. Lastly, the state covariance matrix is updated using

$$P_n = (I - K_n H) \, \underline{P}_n \tag{7.39}$$

where I is a 3×3 identity matrix.

We assume that the agents do not regularly exchange state information with each other. When a fault is detected, such as maneuvering too far from the steady state course, an error is broadcast with the agent's current location. The other agent assigns the broadcasting agent the role of pursuer and plans for evasive action.

7.3.3 Evader Path Generation

The evader path generation algorithm is a high-level control responsible for providing a feasible path, $\mathcal{S}$, for the robot to follow. The path, $\mathcal{S}(s) = \begin{bmatrix} x_1(s) & x_2(s) \end{bmatrix}^T, s \in [0, 1]$ is a parameterized curve with start at $s = 0$ and end at $s = 1$. The path method was chosen so that a sequence of waypoints, $\{\mathcal{S}(s_0), \mathcal{S}(s_1), \ldots, \mathcal{S}(s_n)\}$, can be given to guide the low-level

vehicle control. As a result, the high-level path can be computed less often. A feasible path in this work is any path with the properties that it: (1) starts at the evader's last estimated position and (2) is capable of being traversed given the evader's dynamic capabilities. In this example, the radius of curvature at any point along the path must be greater than or equal to the pursuer's turning radius.

The control that finds the *approximated* path assuming Homicidal Chauffeur dynamics is given by Eqn. 7.23, and finds the steepest descent along the value function, $V_c(x,t)$. However, the actual robot dynamics are given by Eqn. 7.24, where the evader's control now takes the form $d \in \mathcal{D} = [-1,1]$ with the addition of a turning radius, r_e. To find a control that follows the approximated path Eqn. 7.23 and satisfies the criteria to be a feasible path by taking into account the more complicated dynamics, a receding horizon controller is used to minimize a new function,

$$d^* = \arg\min_{d(\cdot)}\{\mathcal{V}\} \tag{7.40}$$

where the cost, $\mathcal{V}$, is defined as

$$\mathcal{V} = \mathbf{w}^T \begin{bmatrix} V_{-hc}(x(t_f)) \\ v_e \sin(x_3(t))\frac{\partial}{\partial x_1}V_{-hc}(x(t)) + v_e\cos(x_3(t))\frac{\partial}{\partial x_2}V_{-hc}(x(t)) \\ e^{((x_2(t)+\zeta_1)*\gamma)} + e^{((-x_2(t)+\zeta_2)*\gamma)} \end{bmatrix} \tag{7.41}$$

with some weighting vector, $\mathbf{w}$, over the interval $t \in [t_0, t_f]$. The control of Eqn. 7.23 is captured in this new function because the first two rows of Eqn. 7.41 guide the evader to avoid the pursuer. Then, the third row contains exponential terms that keep the vehicle within Ω, and can be adjusted with terms ζ_1, ζ_2, and γ. This term keeps the value of $V_{ss}(x(t^*))$ finite. The calculation of $V_{-hc}(x)$ can be computed quickly using the state partition methods in [58]. Once the control is found, the path, $\mathcal{S}(s)$ is generated by integrating Eqn. 7.24 and

using the variable substitution $s = \frac{t-t_0}{t_f-t_0}$,

$$\begin{bmatrix} \mathcal{S}(s) \\ x_3(s) \end{bmatrix} = \begin{bmatrix} x_1(s) \\ x_2(s) \\ x_3(s) \end{bmatrix} = \begin{Bmatrix} v_e \Delta t \int_0^s \sin(x_3(s\Delta t + t_0))ds \\ v_e \Delta t \int_0^s \cos(x_3(s\Delta t + t_0))ds \\ \Delta t \int_0^s \frac{v_e d(s\Delta t + t_0)}{r_e} ds \end{Bmatrix}, s \in [0,1] \tag{7.42}$$

where $\Delta t = t_f - t_0$. The time is normalized so the path has support, $s \in [0,1]$, where in the time domain, $\mathcal{S}(0) = x(t_0)$ and $\mathcal{S}(1) = x(t_f)$.

7.3.4 Low-Level Control

The low-level controller tracks the path, $\mathcal{S}(s)$ with a vehicle control, d. There are many options for controlling a nonholonomic vehicle. Path generation methods [9, 133] generate an optimal path in between waypoints from the high-level control. However, we use a sliding mode tracking algorithm [8] due to its simplicity. For this control to work, we assume that the high-level control has assigned some path, $\mathcal{S}(s) = [x_1(s), x_2(s)]^T$ where $s \in [0,1]$. The path has a radius of curvature associated with each point, $\mathcal{R}(s)$, where we assume that only the sign is known, and it has a lower bound equal to the turning radius of the vehicle, $\min_{s\in[0,1]} \mathcal{R}(s) \geq r_e$.

The evader's control law introduced by Balluchi et. al. [8] is

$$d = \text{sgn}(\mathcal{R}(t))\text{sgn}(\sigma)\frac{v_e}{r_e} \tag{7.43}$$

where the radius of curvature at time, t is given by

$$\mathcal{R}(t) = \mathcal{R}(\hat{s}), \hat{s} = \underset{s\in[0,1]}{\arg\min} \{\|X(t) - \mathcal{S}(s)\|\} \tag{7.44}$$

and corresponds to the radius of curvature of the path at the point that is the shortest distance from the vehicle. The operator $\| \ \|$ finds the Euclidean distance between the position states. Lastly, σ is the sliding variable given by

$$\sigma(\tilde{y}, \tilde{\theta}) = -\frac{\tilde{y}}{r_e} - \text{sgn}(\tilde{\theta})(1 - \cos(\tilde{\theta})) \tag{7.45}$$

where the variable $\tilde{y}$ is the perpendicular offset distance from the nearest point of the path, and $\tilde{\theta}$ is the difference in bearing between the vehicle and nearest point of the path as shown in Fig. 7.5. The variable, σ, can be thought of as a measure as to how well the vehicle is tracking the path. To guarantee that a nearest point on the path $\mathcal{S}(\hat{s}), \hat{s} = \arg\min_{s\in[0,1]} \{\|X(t) - \mathcal{S}(s)\|\}$ exists, we assume that the vehicle has an initial state and a trajectory that satisfies $\tilde{y}(t) < r_e, t \in [t_0, t_f]$. This is not a hard assumption to make, because the high-level control places the path at the last known location of the agent.

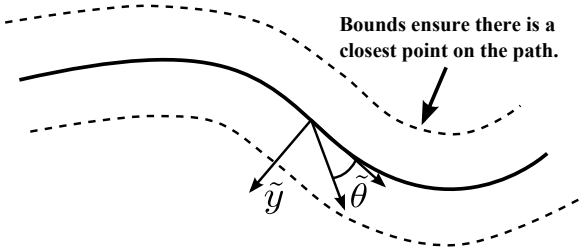


Figure 7.5: The coordinates used as inputs to the sliding mode controller.

The sliding manifold along $\sigma = 0$ governs the direction of the agent's turn to account for nonholonomic effects of the vehicle kinematics. With the control law given by Eqn. 7.43, the vehicle will turn either to the left or right with the largest possible turning speed, $\dot{x}_3 = \frac{v_e}{r_e}$. To reduce the effects of chatter, the control is modified by adding a saturation function and removing the signum function around σ,

$$d = \text{sat}\left\{\alpha \text{sgn}(\mathcal{R}(t))\sigma\frac{v_e}{r_e}\right\} \tag{7.46}$$

This is the control signal that is sent to the robot. The parameter, α, is used to control the rate at which the magnitude of σ causes the vehicle to turn. By setting α very low, slight deviations from the path will result in slight turns by the vehicle. Otherwise, when α is high, then a slight deviation could cause the vehicle to turn as fast as possible, similar to Eqn. 7.43. It is important to have the robot execute slight turns for slight deviations to ensure that the noise from the state estimator does not cause excess chatter in the robot's motion. With this control structure, the robot is able to quickly calculate the steering control, d that will allow the robot to follow a path given by the high-level control.

7.4 Experimental Results

The control system was evaluated using a left turn scenario using the experimental setup described above. Two aspects of the control were analyzed by comparing (1) the resulting region, Ω^*, from which the agent succeeds in reaching $\mathcal{T}_a$ for both the approximate dynamics and the actual dynamics through Monte Carlo simulation and (2) the evasion maneuvers produced in simulation and those produced using the robots. In the simulations, both numerical and on the robots, the evader was given a constant speed of $v_e = 0.038\frac{m}{s}$ and a turning radius $r_e = 1m$. Likewise, the pursuer was given a constant speed of $v_p = 0.038\frac{m}{s}$ and a turning radius $r_p = 3m$. The evader was given the collision avoidance controller and the pursuer simulated making a left turn into on-coming traffic by setting $u = -1$. A collision was said to occur if the trajectory enters $\mathcal{T}_P$, a ball of radius $l = .33m$ around the pursuer as shown in Fig. 7.6. This value corresponds to the diameter of the robots. The steady state target, $\mathcal{T}_{ss}$ was the center of the evader's lane, and the final target after an evasive maneuver was $\mathcal{T}_a$ on the right boundary of $\mathcal{T}_{ss}$. The pursuer's turning radius was $\mathcal{T}_E$. In both the robotic and computer simulations, the pursuer started at location,

$y(t_0) = [5.35m, 1.576m, -\frac{\pi}{2}]^T$. The boundary has lower bound $x_2 = 0$ and upper bound $x_2 = 2.14$. The boundary also has a left bound $x_1 = 0$ and right bound $x_1 = 5.8$. The pursuer's lane is the top of the region, and the evader's lane is the lower part of the region as shown in Fig. 7.6. To keep the evader within the boundaries of its lane we set, $\zeta_1 = 1.1$, $\zeta_2 = 0.1$, and $\gamma = 6$.

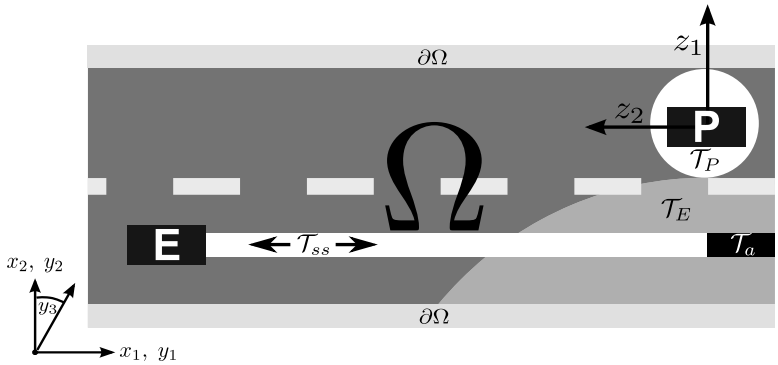


Figure 7.6: An illustration of the left-turn scenario with boundaries, $d\Omega$, and target sets.

7.4.1 Qualitative Analysis of the Receding Horizon Approximation

The time to capture of the approximate dynamics serve as the benchmark result for the receding horizon control minimizing, $\mathcal{V}$. Figure 7.7 shows the value function associated with the zero-sum, pursuit-evasion part of the game, $V_{-hc}(\mathcal{Y}(x, y(t_0)))$ in the absolute coordinate system at the pursuer's initial position. There are no states with an infinite time to capture, which allows the evader to escape from the pursuer from anywhere in the state space. Taking the boundaries into consideration, the evader with Homicidal Chauffeur dynamics can almost always avoid a pursuer making a left turn. The evader, which has the same speed and better turning capabilities, could just move with the pursuer until there is a clear path to $\mathcal{T}_a$. This is possible from almost all initial states except those in or around a small neighborhood of

$\mathcal{T}_P$. However, those states are unlikely starting locations for a collision avoidance scenario. The ideal control for the actual vehicle dynamics would have the same characteristics for Ω^*.

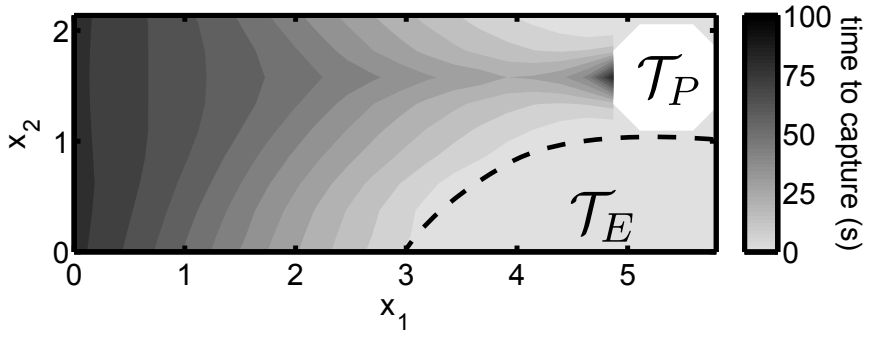


Figure 7.7: The value function, $V_{-hc}(\mathcal{Y}(x, y(t_0)))$, displayed in absolute coordinates, X. Position is given in meters.

To show Ω^* for the actual dynamics, a Monte Carlo simulation was performed with the initial states chosen from Ω using a uniform distribution. For these simulations, the evader takes evasive action from the start of the game to simulate what happens for different relative positions of the agents. The result is shown in Fig. 7.8 which shows the time to capture for the complete maneuver from $x(t_0)$ to $\mathcal{T}_a$ for regions labeled A and C. The remaining regions, D and B, have a time to capture from $x(t_0)$ to $\mathcal{T}_{hc}$.

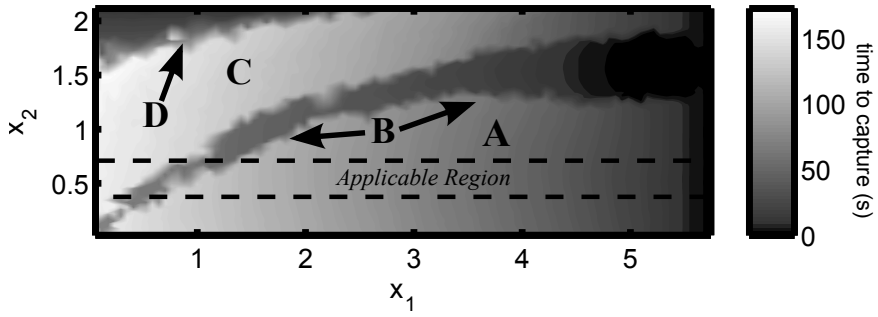


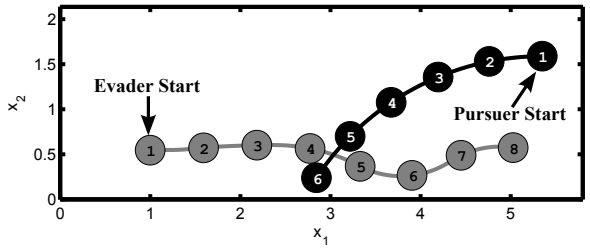
Figure 7.8: Time to completion of the game using the receding horizon controller with realistic dynamics. Positions are given in meters.

The four regions appearing in Fig. 7.8 mark where the evader is able to succeed in the evasive

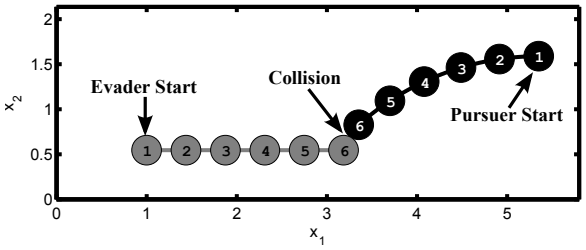
maneuver. In region A, the evader recognizes the pursuer is going swerve at a point when the pursuer's left turning radius is the best possible point of entry. Therefore, as shown by the resulting trajectory in Fig. 7.9a, the evader swerves below the pursuer before returning to its steady state path in the center of its lane. As x_1 increases in region A, the swerve maneuver becomes less necessary, because the evader is closer to the pursuer's turn radius. However, if the maneuver is not performed in states closer to the border between regions A and B, collision is eminent. This is shown in Fig. 7.9b where the pursuer and evader have the same initial states as Fig. 7.9a.

The behavior of the trajectories in region C is similar to region A, except that the pursuer's right turning radius is closer. This causes the agent to swerve above the pursuer before returning to steady state as shown in Fig. 7.9c. In region B, the evader is not able to succeed in escaping from the pursuer. This occurs because the Homicidal Chauffeur approximate dynamics normally steer the evader directly away from the pursuer. However, this action is not possible with the actual dynamics, so region B becomes a dispersal surface where it is equally beneficial to swerve above or below the pursuer. The evader is then guided to head straight for the pursuer, and deviate at the last minute. Given the turning radius of the realistic dynamics, this is not possible from close range as shown in Fig. 7.9d. Lastly, region D marks the states where the pursuer is unable to complete the evasive maneuver because of the boundary limitations of the road.

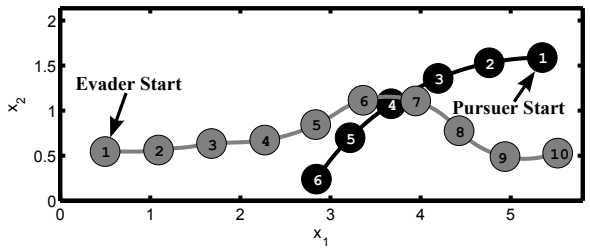
With the receding horizon control, 78% of the state space will result in a successful evasive maneuver. This means that regions A and C composing Ω^* cover 78% of the state space. The remaining 22% is a result of the either the inability of the evader to avoid capture, or the initial state is within the pursuer's target set, $\mathcal{T}_P$. However, the evader will most likely start the collision avoidance maneuver while traveling along its steady state path, $\mathcal{T}_{ss}$. Therefore, the dashed lines define the applicable region which is most likely to represent the outcome



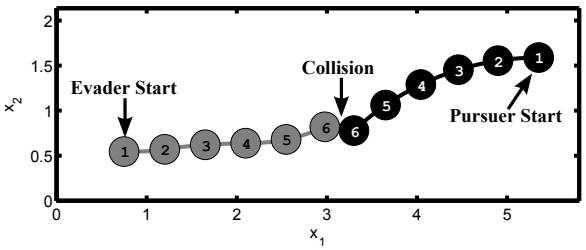
(a) The evader is able to avoid a collision by swerving below the pursuer.



(b) If the control were not applied, a crash would have resulted from the lack of control.



(c) If the control were not applied, a crash would have resulted from the lack of control.



(d) The evader is limited by its turn radius to avoid the pursuer using the approximate dynamic model.

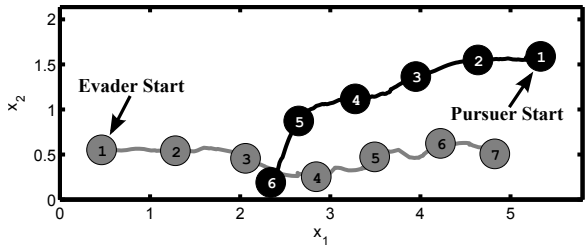
Figure 7.9: Output from numerical simulations showing the behaviors of three regions of the state space. Positions are given in meters.

of the game. In this region, a successful maneuver is highly likely.

7.4.2 Application to a Real System

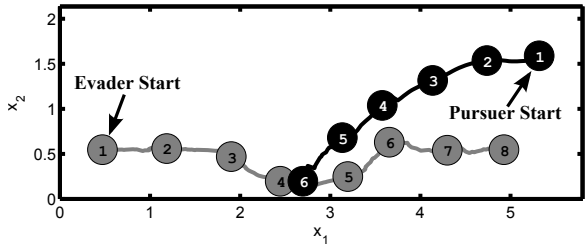
The left turn scenario was implemented on a simple robotic system to show that the control can be computed on-line, and achieve similar results to the simulations. The evader and pursuer started from the same initial positions, $x(t_0) = [0.46m, 0.53m, \frac{\pi}{2}]^T$, $y(t_0) = [5.35m, 1.58m, -\frac{\pi}{2}]^T$. The pursuer travels straight, and starts the left turn at a random position on the horizontal axis, y_1. Randomly assigning a turn for the pursuer requires that the control law and high-level path be generated on-line once the pursuer deviation from the steady state trajectory is broadcast.

Given the initial positions of both of the agents and since the applicable region from Fig.

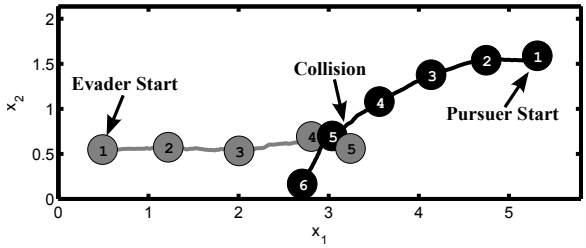


(a) This shows the trajectory of one of the trials from the real robot data.

(b) If the agent's real and approximated dynamics are close enough, a collision can be avoided.



(c) This shows the trajectory of one of the trials from the real robot data.



(d) This shows a trajectory where the maneuver is close.

Figure 7.10: Actual results from robot trajectories with noisy measurements from the Cricket indoor GPS system. Positions are given in meters.

7.8 consists mostly of region A, we expect the evader to take turns that swerve below the pursuer. However, measurement error and error in the actual movement of the robots from what was desired can lead to behavior seen in other regions. In the first example shown in Fig. 7.10a, the pursuer's turn does not require a major evasive action. The control still guides the evader to shift right in the lane, heading for the nearest entry into the pursuer's turn radius showing region A behavior. This occurs due to the worst-case scenario assumption. The exaggerated response is a mild consequence on paper, but it would be unpleasant in a passenger vehicle. Depending on the actual application, this type of response can be corrected by adding an extra term in the receding horizon objective, $\mathcal{V}$, that rewards less movement when the maneuver starts from a safer position.

In cases where the pursuer makes a turn closer to the evader, more aggressive evasive maneuvers are needed with region A behavior. A video capture of this scenario is shown in

Fig. 7.10b and the trajectory is shown in Fig. 7.10c. The pursuer comes close to the evader, but the evader plans a trajectory where a collision cannot occur despite any action by the pursuer. Furthermore, the evader is still able to return to the steady state path because it has not crossed $d\Omega$. Lastly, trajectories seen in Fig. 7.10d result when the pursuer makes a very compact left turn as soon as possible. Here, the evader reacts to the pursuer's turn using region B behavior by waiting until the last minute to try to enter the turn radius which causes a collision. Although not a desirable consequence, it is important to understand that this dispersal surface can occur in a variety of scenarios where two control options have an equivalent impact on the outcome of the game.

As with any system, the performance of the controller degrades with measurement noise and other inaccuracies. The most common cause of failure (either a collision or the evader leaving $d\Omega$) was measurement error caused by crossing the border of the GPS beacon detection fields. Unfortunately, this condition is unavoidable due to the range of the beacon transmission. This caused a jump in the expected distance from the beacon which causes the state estimator to become very inaccurate. The resulting error was reduced by thresholding the measurements to check for valid inputs, but was not eliminated completely. Furthermore, the heading state estimate for each of the agents was also a major source of error in our tests because it is not directly measured. A timeout issue with the Matlab 2006b serial commands would also arise frequently. Apart from what was predicted using the numerical simulations, few failures were a result of incorrect guidance from the control law. By increasing the measurement accuracy, this control can be scaled to more complicated systems in faster paced scenarios.

7.5 Conclusions

In this paper, methods that solve the HJI equation by reducing it to an HJB formulation were applied to a collision avoidance scenario. This simplified form enables us to extract zero-sum controls on-line, which adds versatility to the controller against different opposing agents. Nonzero-sum objectives that incorporate constraints into the problem were included using a receding horizon controller. These methods were then successfully implemented on a simple robotic system used to simulate collision avoidance when a vehicle makes a left turn into oncoming traffic. Despite noise and measurement error, the control law performed as expected, in agreement with the numerical simulations. Even though this work only considers a left turn scenario, the methods shown here can be applied to a variety of cases, such as air traffic control or autonomous navigation, where a controller needs to decide on an avoidance maneuver on-line. The advantage of this approach is that the worst-case assumption removes the pursuer's strategy from the control, leaving only the agent's dynamic limitations to govern the success of the maneuver.

Future research initiatives on the control will include reducing the size of the region in the state space where the evader does not succeed by finding these singular surfaces and treating them as part of the partitioned control approach used for solving the approximate dynamics. In doing so, more complicated dynamics can be considered in the approximate dynamic model allowing for more planning to increase Ω^* and calculate a control for more quickly evolving dynamics. Furthermore, cases with multiple pursuers or evaders will be considered to handle scenarios with vehicular traffic.

Chapter 8

Application 2: Sensor Networks

In this second application, the control law, particularly the reachability set analysis of the reduced HJB formulation are used to determine areas of the state space where an agent may traverse. This chapter shows how the causal relationship between sets of states allows the number of needed sensors to be reduced in order to make inferences of an agent's target set. Results are given for a crowded hallway scenario where an agent has a choice between two exits and it is critical to know which exit will be used.

Copyright ©2011 Society of Photo Optical Instrumentation Engineers. One print or electronic copy may be made for personal use only. Systematic electronic or print reproduction and distribution, duplication of any material in this paper for a fee or for commercial purposes, or modification of the content of the paper are prohibited.

Brian J. Goode, Philip A. Chin and Michael J. Roan, "A sensor reduction technique using Bellman optimal estimates of target agent dynamics", Proc. SPIE 8050, 805018 (2011).[1]

DOI abstract link: http://dx.doi.org/10.1117/12.884108

[1]Brian Goode's contribution are the Bellman calculations, and Phil Chin's contribution are the probabilistic experiments.

8.1 Introduction

One topic of interest in distributed sensing is sensor placement to monitor targets in a surveillance field [31, 39, 80]. By strategically placing sensors according to single or multiple objectives [128], the total number of sensors in a distributed network can be reduced for more efficient data processing and decreased number of material resources. There are multiple approaches to solving the geometric coverage [100] sensor placement problem such as combinatorial optimization [25], Voronoi diagrams [80], and genetic algorithms [10]. Others consider a data-driven combinatorial optimization approach to maximize mutual information [74] between sensor locations. However, in this paper, the authors propose a sensor reduction technique using a dynamic trajectory data-driven model to reduce the coverage area of the surveillance field, while still maintaining the ability to determine outcomes in the uncovered area. By approximating the dynamics of agents in the surveillance field, certain activities can be identified without having to cover the entire area.

A classic example of this is the Zermelo navigation problem in optimal control theory [136]. In this problem, a swimmer (agent) is trying to reach the shore in optimal time. However, the current (environment) in the river (surveillance field) provides a constant sweep downstream. Using viability theory, Cardaliaguet, Quincampoix, and Saint-Pierre [24] formed a new version of the problem where there is a waterfall downstream of the river. The problem becomes determining whether or not the swimmer will be able to make it to shore before reaching the waterfall. In their analysis, they use the dynamics, and not the trajectories of the water current and agent to find where in the river it is possible for the swimmer to reach shore to safety. This forms a barrier in the river where, if located upstream of the barrier, the agent will be able to make it to the shore to safety. However, if located downstream, the swimmer reaches the waterfall first. In terms of surveillance, if one wanted to know using sensors that the swimmer is making progress toward the shore, the sensor coverage

would only need to span the river upstream of the barrier. Otherwise, it is known where the swimmer's trajectory will terminate.

Finding these barriers is known as the "Game of Kind" in the classic optimal control literature [64]. Equivalently, one could use a time optimal value function to find barriers in the state space [57]. If the time optimal value for a state is infinite, then the agent will not be able to traverse to the target. Contrarily, if the value is finite at a state, then it is possible for the agent to reach the target. There are several methods for calculating the value function [14] using Bellman optimal dynamic programming techniques. These infinite horizon solution techniques begin calculating the value at the target and expand out into the state space (surveillance field). The fast marching (FMSL) method [34] uses this to its advantage by ceasing the calculation as soon as there are no more states that will reach the target. The fast marching algorithm is based on the semi-Lagrangian solution method that converges over the entire state space [42] and is used in this work.

This paper proceeds by giving an overview of target capture sets [57]. We show how the formulation can be applied to a surveillance field with environment and agent dynamics to reduce the coverage area needed in a surveillance field to determine target reachability. An algorithm adapted from Falcone [34] is given to determine the capture sets and eliminate sensors from a sensor grid using the Bellman optimal dynamic programming principle. The formulation and algorithm are then applied to a crowded hallway where it is desired to know if an agent will exit from the main exit following the crowd or divert to an alternate exit as soon as possible. The result of the optimal control calculations are applied to remove sensors in the surveillance field where it is not possible for an agent to exit through the alternate exit due to the motion of the crowd. The joint probabilities of the trajectories are then shown to become more distinct as the sensors are eliminated, leading to an increase in the ability to detect the diverting agent.

8.2 Preliminaries

The problem consists of three distinct entities: (1) a surveillance field, (2) an environment, and (3) agents. The surveillance field is a bounded region where the sensors are to monitor activity. The environment is a set of dynamics that are either known or estimated in the surveillance field. These are assumed to be associated with "common" movement in the surveillance field. Lastly, agents are the entities under surveillance. In terms of application, these can include people, cars, etc. Agents have a set of dynamics that act as a disturbance on the environment dynamics. Agents that are not detectable will have a minimal disturbance on the environment, and agents that are the target of surveillance by engaging in anomalous behavior will have a larger disturbance impact on the environment dynamics.

The surveillance field consists of a planar state space to which the trajectories of the agents are projected. A state, x, is a member of the compact state space $X \subset \mathbb{R}^2$. The agents' dynamics are a function of the control, d, and consists of a finite set $D \subset \mathbb{N}$. It is assumed that d is a mapping to a compact set in $\mathbb{R}$. A solution of the system is a trajectory, $\psi = (x, d)$, where $\psi \in \Psi$ and each trajectory in Ψ must satisfy

$$\dot{x}^t = \mathbf{f}(x^t, d^t) = \mathbf{f}_E(x^t) + \mathbf{f}_A(x^t, d^t) \tag{8.1}$$

for all time, $t \in \mathbb{R}^+$, from some initial state, x^{t_0}, where $\mathbf{f}_E(x^t)$ and $\mathbf{f}_A(x^t, d^t)$ are the dynamic contributions of the environment and agent respectively. The form of the evolution equations implies the dynamics of the two entities are exogenous and separable. State evolutions given the disturbance function, $d(\cdot)$, are found using

$$x^{t_f} = S(t_f, t_0; d(\cdot))x^{t_0} \tag{8.2}$$

where S is the state transition operator that finds the value of x^{t_f} from an initial state x^{t_0} given the control function $d(\cdot)$. An agent seeks to generate a trajectory in Ψ such that $x^{t>t^*} \in \mathcal{T}$ for some $t_0 < t^* < \infty$ where $\mathcal{T} \subset X$. $\mathcal{T}$ is termed the target set and is positively invariant. In summary, these equations represent the physical limitations that govern agents' motion through the surveillance field given both the agents' and environmental dynamics.

8.2.1 Capture Set

The kinematic capture set, $\mathcal{C}(\mathcal{T})$, is the set of initial states, x^{t_0}, from which there exists a valid trajectory to the target set, $\mathcal{T}$ with some choice of open-loop control, $d(\cdot)$. Namely,

$$\mathcal{C}(\mathcal{T}) := \left\{ x^{t_0} \in X | \exists d(\cdot) \; s.t. \; (x, d) \in \Psi, x^{t>t^*} \in \mathcal{T}, t_0 < t^* < \infty \right\}, \tag{8.3}$$

where x^t is found using the state transition operator of Eqn. 8.2. This set designates those states from where it is possible for the agent to traverse to the target set given the governing dynamics of Eqn. 8.2 and the set of agent controls, D.

Calculating the capture set, $\mathcal{C}(\mathcal{T})$ for a given target, $\mathcal{T}$, is not trivial, because all the possible controls, $d(\cdot)$ for the agent must be considered. In order to calculate this set, we use Bellman optimality to determine from which initial states it is possible for an agent to traverse to the target set. This method is advantageous for this purpose because it considers the trajectory over the infinite time horizon. Therefore, we now present an optimal control problem using the Hamilton-Jacobi-Bellman equation (HJB) that allows us to accomplish this task.

8.2.2 Finding the Capture Set

Here, properties of the infinite horizon solution to the HJB equations are used to establish the initial states where the agent is capable of reaching a target set, $\mathcal{T}$. In solving the HJB equation, the result is a value function, $V(x)$ that represents the "cost to go" of a state in the state space. By choosing an appropriate running cost, $g(x, d)$, then the value function will reveal which states actually reach the target. In the following, the cost to go is strictly finite throughout the state space, and 0 in the target set. Therefore, the value function will have finite value for all states that reach $\mathcal{T}$.

Proposition 7. *Assume an invariant set $\mathcal{T}$, a control law $d(\cdot)$ and the value, $V(x)$, is the solution to the partial differential equation (Dynamic Programming Principle)*

$$0 = \min_{d \in D} \left[g(x, d) + \nabla V(x) \mathbf{f}(x, d) \right] \tag{8.4}$$

subject to the boundary condition

$$V(x) = 0, \ \forall x \in \mathcal{T} \tag{8.5}$$

and the constraints

$$g(x, d) = \begin{cases} 0, \ \forall x \in \mathcal{T} \\ (0, \infty), \ otherwise \end{cases} \tag{8.6}$$

The agent is able to guide the state to the set, $\mathcal{T}$, from an initial state $x_0 \in X$ when $V(x) < \infty$.

Proof. The proof follows from the running cost, $g(x, d)$, being finite and accruing at every point on the trajectory that is not in $\mathcal{T}$. The function $g(x, d)$ is defined to have 0 value on $\mathcal{T}$. Since $\mathcal{T}$ is assumed invariant, if the trajectory never reaches, then $V(x)$ will tend toward

infinity. □

From this result, the definition of the capture set may be restated as,

$$\mathcal{C}(\mathcal{T}) := \{x^{t_0} \in X | V(x) < \infty\} \tag{8.7}$$

where $V(x)$ is the solution to Proposition 7. In this work, a fast marching semi-Lagrangian (FMSL) method is used to find the solution to the optimal control problem in Proposition 7. The FMSL method presented in the subsequent section is based on of the results of [34], which will be summarized with some modest changes to the algorithm where noted.

8.2.3 Algorithm for Calculating the Capture Set

The advantage of the Fast Marching algorithm is that it terminates when the level set of $V(x)$ ceases to propagate, and minimal time is spent calculating the function at states where it is not possible to traverse into $\mathcal{T}$. In this work, we seek to solve Eqn. 8.4. In order to implement the FMSL method, both the state space and time have to be discretized. The discretized equation becomes

$$V(x_i) = \min_{d \in D}\{V(x_i + hf(x_i, d))\} + hg(x_i, d) \tag{8.8}$$

This is the discrete dynamic programming problem. In choosing a value for $g(x_i, d)$ it is convenient to use time optimality as a running cost.

$$g(x_i, d) = \begin{cases} 0, if\, x_i \in \mathcal{T} \\ 1, otherwise \end{cases} \tag{8.9}$$

Substituting in for $g(x_i, d)$, and utilizing the Kružkov transformation $w(x_i) = 1 - e^{-\lambda h}$, the dynamic programming problem becomes

$$w(x_i) = \min_{d \in D} \{e^{-\lambda h} w(x_i + h\mathbf{f}(x_i, d))\} + 1 - e^{-\lambda h} \tag{8.10}$$

where $h = \Delta t$, and $\lambda \in (0, 1)$. The original value (time) may be found by reversing the transformation with, $V(x) = \frac{-ln(1-w(x))}{\lambda}$. The Kružkov transformation and the scaling factor, λ, are used to bound $w(x) \in (0, 1]$ because $V(x)$ can have an infinite value at some states. By using this change of variables $w(x) = 1$ corresponds to $V(x) = \infty$.

The term $w(x_i + h\mathbf{f}(x_i, d))$, is approximated by interpolating the value of $w(x_i)$ at the neighboring nodes. This is shown in Fig. 8.1, where the arrows represent the vectogram initiating from the node labeled "A". Therefore, in this case, the nodes labeled with the numerals I, II, III will be used in the interpolation. The choice of time step h determines the magnitude of the 1^{st} order approximation, $x_i + h\mathbf{f}(x_i, d)$. For convergence, h should be chosen such that the vectogram of possible trajectories is contained within the neighboring nodes. The following algorithm presents the methods by which the solution to Eqn. 8.10 is found.

Algorithm 6. *Solution method for Eqn. 8.10:*

1. *Nodes of the set $\mathcal{T}$ are placed in the set, A and $w(x_i) = 0, \forall x_i \in \mathcal{T}$. For all other nodes, $x_i \notin \mathcal{T}$, $w(x_i) = 1$.*

2. *All neighbors of the nodes in A are placed in N.*

3. *$w(x_i)$ is calculated for all $x_i \in N$ using Eqn. 8.10.*

4. *The node in N with the lowest value, $w(x_i)$, is removed from N and placed in A. Neighbors of this node are then added to N if they are reachable (have dynamics that*

can reach A).

5. *Repeat until N is empty.*

Figure 8.1 provides an illustration of the different components of this algorithm. The white circles represent the accepted set, A. The black circles are the set of neighboring nodes, N and the gray circles are the set of nodes that have not been considered. As an example

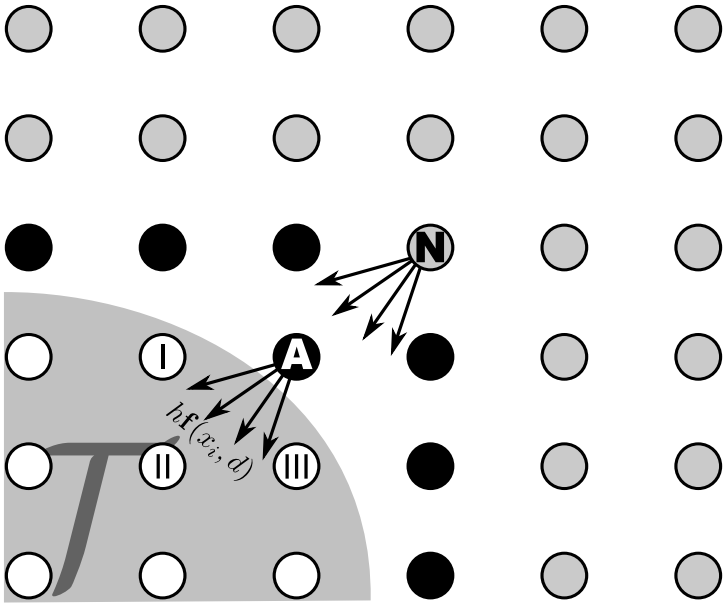


Figure 8.1: Illustration of Alg. 6 on the plane.

of a transition, consider the node labeled "A". It is closest to $\mathcal{T}$, which means that it will have the minimal value. Therefore, it is moved from the set N to A. Consequently, the node labeled "N" will be added to the set N, because it is the only neighbor of the newly accepted node that is not already included. Following the steps in the algorithm, the process of interpolating the value at each of the nodes in N, repeats until there are no more nodes that are in the neighboring set.

By solving Eqn. 8.10, it is possible to find the set of states, $\{x_i \in X | V(x_i) < \infty\}$ from which trajectories will traverse into $\mathcal{T}$ given an agent's control set, D. The set may be equivalently

written as $\{x_i \in X | w(x_i) < 1\}$. Using this formulation, the capture set can be approximated by

$$\tilde{\mathcal{C}}(\mathcal{T}) = \{x_i \in X | w(x_i) < 1\} \tag{8.11}$$

The quality of the approximation is dependent on the quality of the resolution of the state space.

8.3 Dynamics of Hallway Surveillance

In the previous section, the mathematical formulation given can be used to determine from which states in the surveillance field it is possible for an agent to reach a specified target. In this section, we apply this mathematical formulation to a surveillance scenario where the surveillance field consists of a hallway filled with moving people. The dynamics of the environment model have the crowd move from one end of the hallway to a main exit, E_1. We are interested in detecting if an agent moves toward an alternate exit, E_2. An agent in this scenario is capable of moving in any direction, but is also subject to the environmental dynamics of the crowd that keeps the agents moving in the horizontal direction. We assume that there are proximity sensors that indicate whether or not an agent has entered the square region surrounding the sensor or not. In the following, each of the equations describing the entities are presented.

8.3.1 Surveillance Field

The surveillance field for this work is a planar state space, $X \subset \mathbb{R}^2$. Agent trajectories evolve in this space. To simulate a rectangular hallway, the horizontal state is bounded such that $x_1 \in [x_{1,min}, x_{1,max}]$, and the vertical state is bounded by $x_2 \in [x_{2,min}, x_{2,max}]$. The main exit

is defined to be

$$E_1 := \{x \in X | x_1 = x_{1,max}\} \tag{8.12}$$

and the second exit is defined in the state space as

$$E_2 := \{x \in X | x_1 \in [e_{2,min}, e_{2,max}], x_2 = x_{2,min}\}. \tag{8.13}$$

The set of all initial states, where it is assumed that all trajectories start, is the entrance of the hallway such that

$$X_0 = \{x \in X | x_1 = x_{1,min}\}. \tag{8.14}$$

To determine the exact initial condition, $x_0 \in X_0$, of an individual agent trajectory, the initial vertical state, $x_{0,2}$ must be specified. This is determined following

$$x_{0,2} \sim N\left(0, \sqrt{\frac{1}{2} x_{2,max}}\right), \tag{8.15}$$

and all agents are assumed to enter into the surveillance field at random initial times. All agents in the surveillance field have dynamics subject to Eqn. 8.1, which has both an environmental and individual agent component. The next sections outline the specific form of this evolution equation in terms of the environment and agent dynamics.

8.3.2 Environmental Dynamics Model

The environmental dynamics approximate the flow of a crowd of people through the hallway (surveillance field). As a simple model, it is assumed that people move horizontally through the hallway with an average speed, $v_p > 0$, which is a constant parameter. To account for local randomness in the trajectories, the vertical state is subject to a random input. The following model was used to approximate the crowd motion.

$$\mathbf{f}_E(x) = \begin{Bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{Bmatrix} = \begin{Bmatrix} v_p \\ A_1 \sin(R_1) + A_2 \sin(\omega x + 2\pi R_2) \end{Bmatrix} \tag{8.16}$$

where R_1 and $R_2 \sim N(0,1)$. Parameters R_1 and R_2 approximate a Gaussian random noise and phase input, respectively, to the vertical direction movement.

8.3.3 Agent Model

Agents in the hallway have two choices. They can either follow the crowd dynamics and exit from E_1, or attempt to exit the hallway from E_2. To capture this, the agent makes a decision using its control, $d \in D$, where $D = \{0, 1\}$. For the agent attempting to exit from E_2, two types of models were used to test the surveillance system. A difficult trajectory to detect is approximated by the agent Type I dynamics:

$$\mathbf{f}_A(x, d) = \begin{Bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{Bmatrix} = d \begin{Bmatrix} 0 \\ -c \end{Bmatrix} \tag{8.17}$$

where $c > 0$ and is a constant. The agent Type I dynamics show that the agent attempting to exit from E_2 does so with a constant downward velocity. Agent Type II has the following set of dynamics:

$$\mathbf{f}_A(x, d) = \begin{Bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{Bmatrix} = d \begin{Bmatrix} 0 \\ -\gamma v_p e^{\gamma x_1} \end{Bmatrix} \tag{8.18}$$

where γ is some positive constant. The Type II dynamics model the evader as becoming more and more aggressive with attempting to flee as E_1 draws near. In this formulation the control, d, acts to turn the agent dynamics, $\mathbf{f}_A(x, d)$, on or off. Therefore, the agent has the choice to either follow the crowd or attempt to exit. In this work, we assume that this choice

is made by the agent prior to entering the hallway, and that d remains constant once it is chosen.

8.4 Sensor Model

The surveillance field is monitored by a rectangular grid of sensors, S, with i sensors on the horizontal direction and j sensors in the vertical direction. s_{ij} denotes the sensor at location (i, j), and there exists a mapping from (i, j) to $\mathbb{R}^2$. The proximity sensor data is generated assuming no intensity or measure of closeness to the center of proximity sensor. If an agent trajectory moves into the proximity region of a sensor, then a hit is recorded at the earliest time this occurs. Each sensor records the total number of hits recorded over K time intervals,

$$H_{s_{ij}} = \sum_{k=1}^{K} h_{s_{ij}}(k) \tag{8.19}$$

where $h_{s_{ij}}(k)$ is the number of hits at sensor s_{ij} in the k^{th} time interval. The time intervals occur at regular time steps, dT. For this work, we take the time step to be $dT = \frac{\Delta x_{1,sensor}}{v_p}$ which corresponds to the time it takes an agent subscribing to the crowd dynamics to move from one column of sensors in the grid to the next.

8.4.1 One-Step Probability of Traversing Two Sensors

The goal is to calculate the probability of a joint path being traveled. A comparison of the joint probability of the trajectories of agents choosing to be a part of the crowd using the environment dynamics to the joint probability of the alternate trajectories will show that the alternate paths are less probable. Trajectories from each proximity sensor s_{ij} can travel to any sensor above, below, and to the right. The set that identifies these neighboring

sensors is S'_{ij}. This arrangement is consistent with the one-way direction of travel from the environment dynamics in the hallway. Two different methods were used to calculate the probability of a transition from a sensor, s_{ij} to its neighboring sensors, S'_{ij}. The first is the classic Conditional Probability method, and the second is a Conditional Probability Ratio method. In both methods, all possible paths are considered by including all of the neighboring sensors in the conditional probability calculation.

Conditional Probability: The conditional probability that $H_{s'_{ij}}$ alarms occur at a neighboring sensor, $s'_{ij} \in S'_{ij}$ given $H_{s_{ij}}$ alarms at sensor s_{ij} is

$$\begin{aligned} P\left(H_{s'_{ij}}|H_{s_{ij}}\right) &= \frac{\#\text{ of outcomes in } H_{s_{ij}} \text{ and } H_{s'_{ij}}}{\#\text{ of outcomes in } H_{s_{ij}}} \\ &= \frac{\min[\{H_{s_{ij}}, H_{s'_{ij}}\}]}{\sum_{S'_{ij}} \min[\{H_{s_{ij}}, H_{s'_{ij}}\}]} \end{aligned}$$

Conditional Probability Ratio Method: This is an alternative to the conditional probability method. First, a ratio between the number of alarms is defined for a sensor and its neighbors using,

$$J = \begin{cases} \frac{H_{s_{ij}}}{H_{s'_{ij}}}, & \text{if } H_{s_{ij}} \leq H_{s'_{ij}} \\ \\ \frac{H_{s'_{ij}}}{H_{s_{ij}}}, & \text{if } H_{s_{ij}} > H_{s'_{ij}} \end{cases} \tag{8.20}$$

Then, the probability that $H_{s'_{ij}}$ alarms occur at a neighboring sensor, $s'_{ij} \in S'_{ij}$ given $H_{s_{ij}}$ alarms at sensor s_{ij} is

$$P\left(H_{s'_{ij}}|H_{s_{ij}}\right) = \frac{J}{\sum_{S'_{ij}} J} \tag{8.21}$$

Both methods shown above are practical to calculate the probability of a sensor transition. We use the ratio method in this work, because the latter ratio method more accurately weights the transition probabilities from the sensor data.

8.4.2 Joint Probability of Traversing Paths

Since we are monitoring trajectories in the surveillance field, the joint probability of traversing multiple sensor proximity regions is needed. To accomplish this, we define a path in the sensor grid as

$$\mathcal{S} := \left\{ s_{ij}^1, s_{ij}^2, \ldots, s_{ij}^N \right\} \tag{8.22}$$

where $\mathcal{S}$ is a specific sequence of N sensors, s_{ij}, that are traversed by an agent trajectory, ψ. We note the position in the sequence using $\mathcal{S}_1 = s_{ij}^1, \mathcal{S}_2 = s_{ij}^2$, and so on. To be a valid sequence the following criterion must hold

$$\mathcal{S}_n \in S'_{\mathcal{S}_{n-1}}, \ \forall n \in \{2, \ldots, N\}. \tag{8.23}$$

The rest of this formulation assumes that $\mathcal{S}$ is a valid sequence. Otherwise, the transition probability is 0. We begin by considering the probability of a specific one-step transition when $N = 2$, with the path beginning at an initial sensor, $\mathcal{S}_1$. The joint probability of transitioning between two adjacent sensors is

$$P(H_{\mathcal{S}_1}, H_{\mathcal{S}_2}) = P(H_{\mathcal{S}_2} | H_{\mathcal{S}_1}) P(H_{\mathcal{S}_1}). \tag{8.24}$$

It follows that the joint probability of traversing three sensors ($N = 3$) is

$$P(H_{\mathcal{S}_1}, H_{\mathcal{S}_2}, H_{\mathcal{S}_3}) = P(H_{\mathcal{S}_3} | H_{\mathcal{S}_2}, H_{\mathcal{S}_1}) P(H_{\mathcal{S}_2}, H_{\mathcal{S}_1}). \tag{8.25}$$

After applying the Markov property on the RHS, this simplifies to

$$P(H_{\mathcal{S}_1}, H_{\mathcal{S}_2}, H_{\mathcal{S}_3}) = P(H_{\mathcal{S}_3} | H_{\mathcal{S}_2}) P(H_{\mathcal{S}_2} | H_{\mathcal{S}_1}) P(H_{\mathcal{S}_1}). \tag{8.26}$$

The joint probability of traversing a sequence, $\mathcal{S}$, of N sensors is given by

$$P(\mathcal{S}) = P(H_{\mathcal{S}_1}, \ldots, H_{\mathcal{S}_N}) = P(H_{\mathcal{S}_N}|H_{\mathcal{S}_{N-1}})P(H_{\mathcal{S}_{N-1}})|H_{\mathcal{S}_{N-2}}) \ldots P(H_{\mathcal{S}_2}|H_{\mathcal{S}_1})P(H_{\mathcal{S}_1}) \quad (8.27)$$

$$= \prod_{n=1}^{N-1} P(H_{\mathcal{S}_{n+1}}|H_{\mathcal{S}_n})P(H_{\mathcal{S}_1}) \quad (8.28)$$

8.5 Sensor Reduction Using Bellman Optimality

Having developed the models for the environment, agents, and sensors, we show how the dynamics of the environment and agents can be integrated to eliminate sensors. To do this we use the optimal control problem outlined in Sec. 8.2. Specifically, we will use Proposition 7 to estimate the capture set $\tilde{\mathcal{C}}(\mathcal{T})$. Table 8.1 shows all the parameters used in the subsequent example calculations. The parameters in Table 8.1 are applied to the

Table 8.1: Model parameters used for sensor reduction calculations.

Surveillance Field	**Environment Dynamics**	**Agent Dynamics**	**Sensor Model**
$x_{1,min} =$ 0	$v_p =$ 2	$\gamma =$ 0.122	$\Delta x_{1,sensor} =$ 2
$x_{1,max} =$ 50	$A_1 =$ 0.5	$c =$ 0.8	$\Delta x_{2,sensor} =$ 2
$x_{2,min} =$ -10	$A_2 =$ 0.1571		
$x_{2,max} =$ 10	$\omega =$ 0.1571		
$e_{2,min} =$ 25			
$e_{2,max} =$ 30			

surveillance and sensor model that is illustrated in Fig. 8.2. The sensor grid, S, presently covers the entire surveillance field, X. In the following calculations, we will determine the sensors the set of sensors, $\{S \cap \tilde{\mathcal{C}}(\mathcal{T})\}$, needed to monitor E_2. In particular, we want to determine from which states in the state space it is possible for an agent to succeed in exiting through exit, E_2. Therefore, we set $\mathcal{T} = E_2$, and proceed with calculating $V(x)$ in Proposition 7 using Alg. 6. However, in order to use Alg. 6, the state space must be discretized into a grid lattice. We have chosen for this work to discretize the states with a 0.5

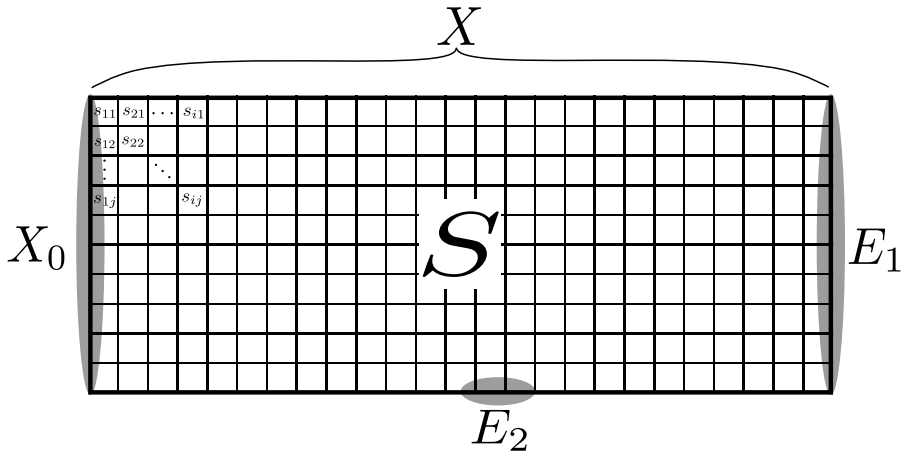


Figure 8.2: This is the layout of the surveillance field. The entrance is X_0, and the two exits are E_1 and E_2. Agents entering the surveillance field will be monitored by the sensor grid, S.

node spacing in both directions of the lattice. Lastly, $\lambda = 0.1$ was chosen as the bounding rate for the change of variables from $V(x)$ to $w(x_i)$.

8.5.1 Determining the Capture Set

We first consider the dynamics associated with agent Type I. To approximate the dynamics of the agent, we ignore the noise term of the crowd. So at any point in time, the agent has the option of following the crowd or trying to move down the crowd toward the exit. The capture set, $\tilde{\mathcal{C}}(\mathcal{T})$ for this scenario is shown in Fig. 8.3. The gray region in the figure shows all the initial states where it is possible for the agent to reach E_2 in finite time. This is the capture set. The darker regions of the capture set show the states where it takes a smaller amount of time to reach E_2, and the lighter states show longer times to reach the alternate exit. Also shown are two extreme trajectories emanating from the initial point $x_0 = (0, 10)$. The trajectory terminating on the left most region of E_2 is the case where the agent enters the surveillance field and immediately heads for the exit. Here, $d = 1$ beginning at $t = t_0$. The other trajectory shows the case where the agent can follow the crowd dynamics until

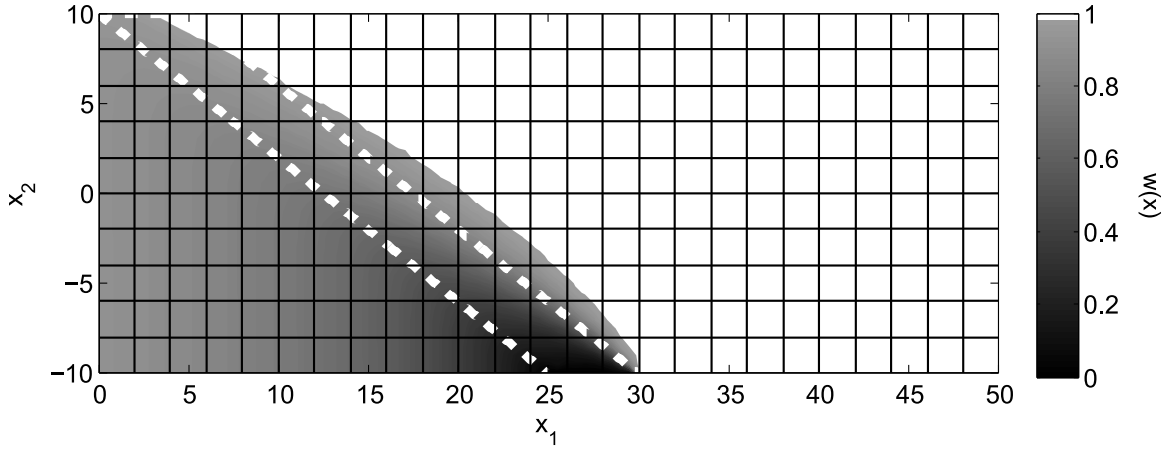


Figure 8.3: $w(x)$ is plotted over the state space, assuming a Type I agent. The trajectories show two of the extreme routes that could be taken by an agent from $x_0 = (0, 10)$. The gray region depicts the capture set, $\tilde{\mathcal{C}}(\mathcal{T})$

$x_1 = 5$, must try to head for the exit. If the agent were to follow the crowd for any longer, it would be forced to exit from E_1 due to the crowd input of the dynamics. This trajectory can be interpreted as the right barrier of the true capture set, $\mathcal{C}(\mathcal{T})$, where the surveillance field boundaries serve as the other boundaries. As long as the agent does not pass this path, then it is possible to reach the exit by playing the proper control value, d. We see that the estimated capture set, $\tilde{\mathcal{C}}(\mathcal{T})$, for this example follows this trajectory. The error seen by the curvature of the set, is a result of numerical error in the calculation and the discretization of the grid. Therefore, to adequately monitor the surveillance field for a Type I agent seeking to exit from E_2, the only sensors needed are those whose proximities are covered, even if just partially, by the capture set. In this scenario, over half of the sensors can be eliminated, because it is known that if the agent is not within the capture set, it must exit from E_1.

In Fig. 8.4, the same calculations were performed except the Type II agent dynamics are shown. In comparing the two cases, we see that the gray capture set is much larger. This is because the speed of the agent toward exiting the crowd is assumed to increase exponentially with x_1. Therefore, there are more states as x_1 increases from which the agent can reach

the target. Once again two extreme trajectories are shown that initiate at $x_0 = (0, 10)$.

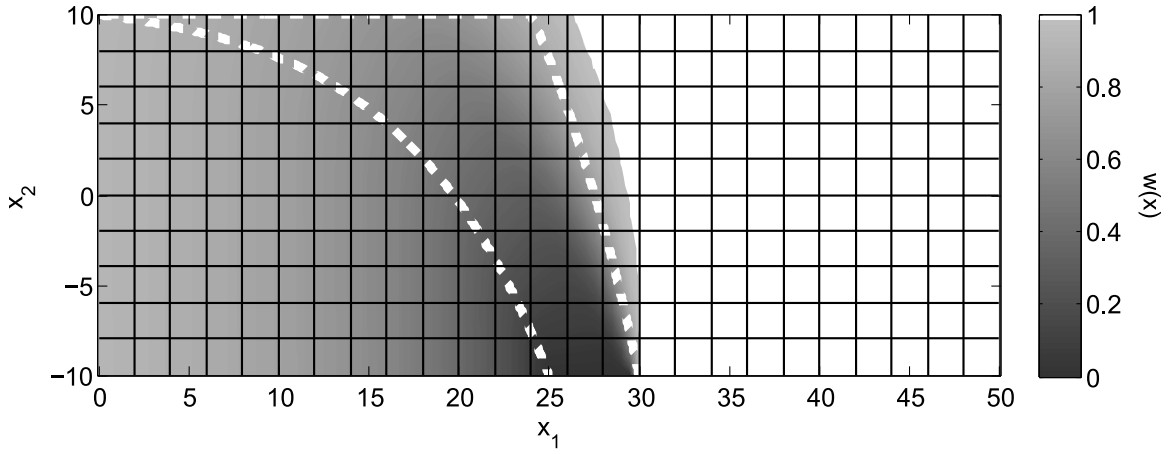


Figure 8.4: $w(x)$ is plotted over the state space, assuming a Type II agent. The trajectories show two of the extreme routes that could be taken by an agent from $x_0 = (0, 10)$. The gray region depicts the capture set, $\tilde{\mathcal{C}}(\mathcal{T})$

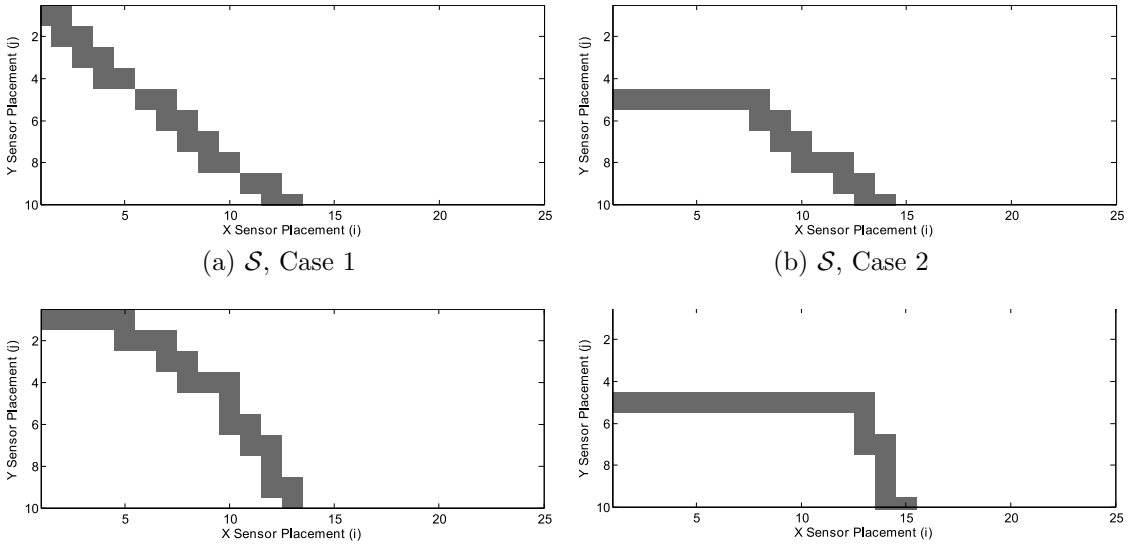
The first trajectory which terminates on the left side of the exit, E_2, occurs when $d = 1$ for the entire time the agent enters the surveillance field. Likewise, the other trajectory that terminates on the far side of the exit, occurs when the agent plays $d = 0$ for as long as possible. This trajectory also acts as a barrier where the agent is guaranteed to reach the target if located on the left side of the path it produces. We also note how the estimated capture set approximates this trajectory reasonably well in that every sensor the trajectory crosses through is also a member of the capture set.

Although these examples are simple in nature, they show that the calculation of the estimated capture set reasonably approximates the barrier trajectory. As more complex dynamics are considered, the barrier trajectory becomes more difficult to determine, and the estimated capture set can give a good approximation of where to focus in on the surveillance field. Furthermore, this calculations do not take a long time to complete. In both of the examples above, the times to calculate $w(x_i)$ were on the order of 16-17s on a 1.7 GHz Pentium M Processor with 1 GB RAM. However, this number is subject to fluctuate with different

approximation parameters [57].

8.5.2 Agent Tracking with the Reduced Sensor Field

In this section we apply the sensor model from Sec. 8.4 to the surveillance field and compare the probability that a sensor sequence, $\mathcal{S}$, occurs with both the full and reduced sensor networks. We consider four agent trajectories (different from the previous section), two from each agent type shown as sensor sequences in Fig. 8.5 below. Agent Type I is represented



(a) $\mathcal{S}$, Case 1 (b) $\mathcal{S}$, Case 2

(c) $\mathcal{S}$, Case 3 (d) $\mathcal{S}$, Case 4

Figure 8.5: Four trajectories are studied for the probability of a particular path occurring in the surveillance field.

in cases 1 and 2, and agent Type II is represented in cases 3 and 4. To turn the trajectories into a sensor sequence, $x \mapsto \mathcal{S}$, the trajectory was overlaid onto the sensor field. Then the order of the sensors was recorded as the trajectory entered into the sensor's proximity. In general, the steeper the downward trend of the trajectory, the easier it is to detect. Each agent type is given an "easy" trajectory (Figs. 8.5a and 8.5c) and a "harder" trajectory to detect (Figs. 8.5b and 8.5d). In both cases 1 and 3, the initial condition is $x_0 = (0, 10)$ and

$d = 1, \forall x_1$. The initial condition for cases 2 and 4 is $x_0 = (0, 0)$. In case 2,

$$d = \begin{cases} 0, & x_1 < 25 \\ 1, & otherwise \end{cases}$$

and for case IV,

$$d = \begin{cases} 0, & x_1 < 15 \\ 1, & otherwise \end{cases}$$

to generate the initial straight path before the agent diverts toward the alternate exit. Once the sensor sequence for the diverting agents were determined, the joint probability is calculated for an agent traversing sensor sequence, $\mathcal{S}$ for each of the four cases in the presence of a crowd in the hallway. To simulate the crowd, 500 trajectories were created using the crowd dynamics, by setting $d = 0$ and giving the agents a random initial state $x_0 \in X_0$. One hundred of the five hundred trajectories are shown in Fig. 8.6, where we see that the the normal distribution of the initial states places most of the crowd agents in the center of the hallway. Furthermore, the random motion in the x_2 direction is seen clearly. Once the trajectories

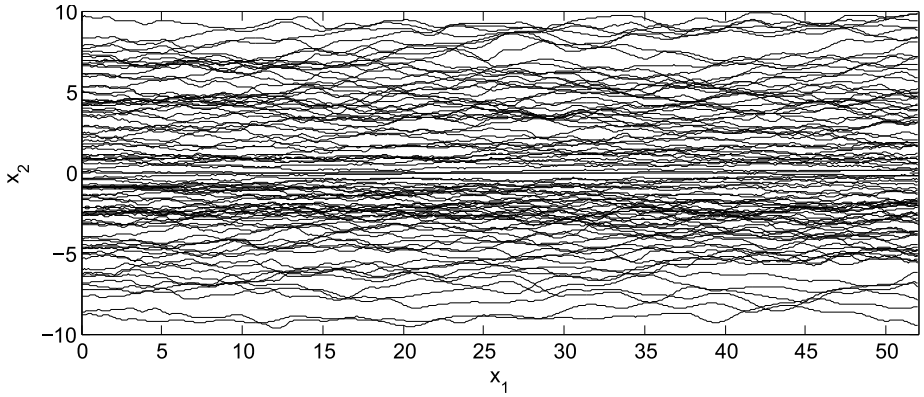


Figure 8.6: This figure shows a sample of 100 trajectories from the total of 500 used to form the crowd hit count.

were recorded, a sensor hit count shown in Fig. 8.7 was created by first assigning each of

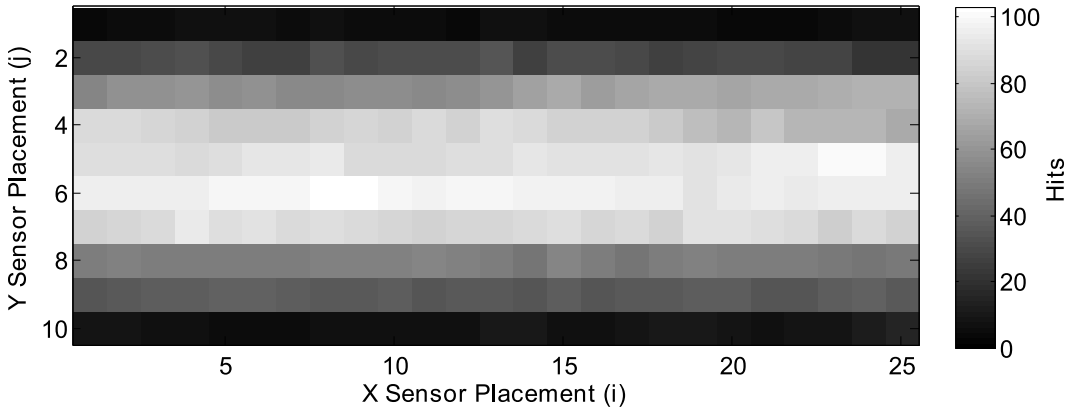
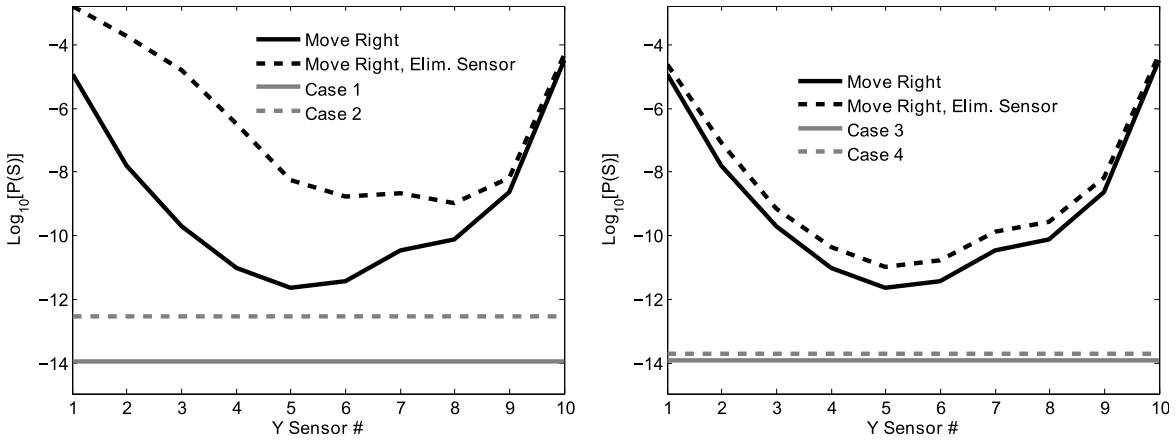


Figure 8.7: This shows the hit count for the random trajectories of 500 agents using the crowd dynamics, $d = 0$.

the trajectories a random entrance time. Then, the hits were recorded as the trajectories entered the proximity region of the sensor. Because the initial condition of the trajectories is a normal distribution, most of the hits appear in the center of the plot. These hit counts are now used to assess the probability of specific sensor sequences occurring.

Eqn. 8.27 was used to find the joint probability of a sensor sequence occurring with the hit counts given in Fig. 8.7. The results of these calculations are shown in Fig. 8.8. Here, the probability of each of the case trajectories from Fig. 8.5 associated with an agent type are plotted against a case of the probability of an agent moving right following a crowd dynamic from each of the possible initial sensors. The black solid line represents the probability of an agent moving right and exiting from E_1 when the full sensor network is utilized. The black dashed line represents the same trajectory, but calculated using the partial sensor network. The gray lines represent probabilities of sensor sequences for their respective cases. We note that the specific case probabilities only occur at the initial sensor of the trajectory, but have been represented by a line across all initial x_2 sensors for ease of comparison.

In both figures 8.8a and 8.8b, the probabilities for the move right case are higher than cases 1-4. The convexity of the probability distribution of the move right scenarios is due to the



(a) Agent Type I (b) Agent Type II

Figure 8.8: Probability of sensor sequences occurring in the surveillance field.

normal distribution of the trajectories. Since more agents tend to be located in the center of the surveillance field, the random walk motion in the vertical direction makes it unlikely for an agent to travel a completely straight line. Anytime the move right scenario is higher than the agent diverging case is beneficial, because the trajectory of an agent heading for E_2 is much smaller than the agents heading for E_1. Therefore, the sensor network would identify this type of behavior as very anomalous. When comparing the "easy" and "hard" cases of each type, the dashed line representing the "hard" cases are always higher than the solid line because these trajectories appear more likely. This is due to the period of time that the agent travels straight before diverting. However, the advantage of the eliminated sensors is seen when comparing the agent types. When comparing the two figures, agent Type I in Fig. 8.8a shows a much larger gap between the dashed and solid lines for the move right scenarios than the Type II agent in Fig. 8.8b. The reason for this increased gap in the Type I agent movement is because more sensors can be eliminated when x_1 is small due to the agent's inability to reach E_2. On the contrary, agent Type II dynamics offer little warning before tending toward the alternate exit. Therefore, there are very few clues early on that can help to determine the ultimate path of the agent.

From these results, scenarios that will benefit most from the sensor reduction techniques are those that can eliminate sensors early along a trajectory. When these types of cases arrive, the dynamics of the agent are such that their intentions are governed more by the environment than cases where the sensors cannot be removed sooner. Therefore, the environment acts as a causal inferencing tool with which we can make assumptions of the agent's dynamics to estimate the outcome of an agent's trajectory. The result is a sensor network with a reduced number of sensors.

8.6 Conclusions

In this work, we exploit properties of the value function produced by solving the Hamilton-Jacobi-Bellman equation for infinite horizon optimal control problems. With an estimate of the dynamics of the agents and environment of the surveillance field, it is possible to determine where sensors should be placed to determine if an agent will reach a specified target. This information allows for sensor reduction of a sensor network, leading to more efficient detection capabilities. This theory is applied to a hallway detection scenario where it is desired to detect if an agent intends to leave the hallway through an alternate exit. With this scenario we have shown how the capture set calculation approximates the true capture set for different agent dynamics. Probability calculations are given and we show how the reduced sensor network makes the probability of the alternate trajectory more distinct. Future work on this topic will involve more complex environments and anomaly detection with a realistic data set.

Chapter 9

Conclusions

The work in this dissertation provides control laws to agents involved in pursuit-evasion scenarios. Such applications include unmanned underwater vehicles engaged in surveillance and tracking maneuvers. Because perfect state information and vehicle control without disturbance is not realizable in an actual implementation, it is beneficial to quickly update the vehicles' control law as knowledge of these effects is gained. This dissertation begins by analyzing two different approaches to this problem: finite horizon one-step static game theory and approximate dynamic programming with adaptive fuzzy inference systems. Ideas from these two solution methods are then combined into a third graph partitioning scheme where the Hamilton-Jacobi-Isaacs (HJI) equation is solved using the Hamilton-Jacobi-Bellman (HJB) equation. It is shown how the control law can be updated on-line without assuming any prior knowledge of an opposing agent. Lastly, the partitioning method is applied to collision avoidance and optimal surveillance scenarios.

The one-step Nash equilibrium scheme is desirable for its ability to handle probabilistic pursuit-evasion scenarios. However, the limited horizon of the controller produces undesired trajectories in the long term. For example, it is shown how a limit cycle can appear in the dynamics of the agent when multiple objectives are considered over a finite horizon.

Even approaches using approximate dynamic programming schemes are subject to the same shortcomings. Once again, if a finite horizon reward is used to update a value approximator (e.g. fuzzy inference system), the dynamics are still subject to limit cycles.

To solve this problem, the state space is partitioned into regions where a finite horizon objective is applied. In doing so, the problem shifts to ensuring that the sequence of partitions traversed by the trajectory terminates at the desired target set. Using a graph-theoretic approach, three criteria are established to determine whether the maneuver is possible given a particular partition-control structure. Updates are then made to the partition-control structure using the HJB equation to identify the region of the state space where the agent is capable of reaching when playing against a worst-case opponent. A temporal study of the evolution of the partition-control structure is given for the Homicidal Chauffeur game, but the results of this analysis will vary depending on the type of game studied.

Lastly, the partitioning scheme for a control law is applied in two different scenarios. In the first application, a collision avoidance scenario is considered. A tiered approach is used where the differential game control law provides high-level guidance that generates a path that a low-level vehicle control must track. This design allows for simpler dynamics to be used for the differential game controller. Using an actual robotic system, tests show that the controller can be implemented on-line using noisy measurements. The second application is a sensor reduction scenario where it is desired to survey a hallway with a minimal number of sensors. Multiple agents are tracked to determine which exit they choose in a crowded hallway. The concept of reachability is used to determine areas of the state space where it is not possible for an agent to enter, eliminating the need to survey that particular area. Therefore, a causal relationship can be established via sensors in other areas of the partition, leading to sensor reductions.

9.1 Significance of Research and Results

The main thrust of this research is that by careful partitioning of the state space and appropriate assignment of controls to the partitions, the solution process of the Hamilton-Jacobi-Isaacs equation can be greatly simplified. In this dissertation, a graph-theoretic criterion is presented in Chapter 5 that relates the sequence of partition travel to a guarantee of reaching the desired target. Therefore, adjustments to the control law need only occur within or on the partition boundaries. By solving the Hamilton-Jacobi-Bellman equation on the states within the partitions, it is possible to update the control law based on new state measurements or knowledge of the opposing agent. As a result, the time to calculate a control is greatly reduced, allowing for control processing to occur on-line during the game. Because the graph-theoretic criterion is satisfied by the update scheme, then the trajectory is guaranteed to have infinite horizon reachability to the target set. Furthermore, worst-case play is assumed on the part of the opposing agent, so learning does not need to occur over multiple iterations of the *game* as seen in many approximate dynamic programming methods.

The other aspect of this research is implementing the control law on an actual robotic system. Processing the control law on a FitPC2 mounted on the robot vehicle, shows that it is possible to implement the scheme on a mobile platform. In particular, this dissertation shows how the differential game aspect of the control law can be calculated using simplified vehicle dynamics, but still apply to a vehicle with more complicated dynamics. This is important for systems where the curse of dimensionality due to an increased number of states could render the control law impossible to calculate on-line.

The work in this dissertation reduces the computational requirement needed to calculate the HJI equation for pursuit-evasion scenarios. This research essentially provides a shortcut to reaching the solution. If the method fails, then the calculation would proceed as it would

using traditional methods. As discussed in Chapter 5, the best performance is seen when the agent has the dynamic capabilities to win. Therefore, implementing this controller can only help an unmanned vehicle reach its goal more quickly.

9.2 Suggestions for Future Work

There are numerous possibilities for improvement in order to develop a control for guiding an agent in an unknown environment against an unknown competitor. Although the control law in this dissertation eliminates many unneeded calculations, it is still subject to the curse of dimensionality with more complicated dynamics, grid resolution, etc. Therefore, the topics of future research will focus on ways to reduce the control calculation time as the complexity of the system increases due to multi-agent systems, complex singular surfaces, and measurement error.

Multi-agent Systems

When controlling multi-agent systems there are two differing approaches that can be used: centralized and decentralized control. Centralized control essentially treats each team as one agent, but with many states. This would fit into the framework of this dissertation readily, but the shear increase of states of even a team of two could prevent the algorithm from being applied on-line. Decentralized control, on the other hand, places the control on each agent and relies on collaborative behaviors that spread the global control calculation amongst the team. However, this does not fit within the paradigm of the worst-case control law generated in this dissertation. Therefore, a hybrid design should be investigated similar to that seen seen with Q-learning [81]. Using a leader-based team, one possible solution would be for a designated leader to command subordinate vehicles to follow members of the

opposite team in certain regions of the domain using the sensor problem of Chapter 8. Then, the subordinate agents would pursue their targets using a similar implementation as seen in Chapter 7.

Complex Singular Surfaces

One of the main problems with developing differential game schemes is the emergence of singular surfaces in the value function that make finding a solution difficult. These singular surfaces result because abrupt changes in the control strategy, e.g. a pursuer of finite turn radius located in front of the evader will have an equally beneficial chance of turning left or right. As the dynamics of the agents become more complicated, these surfaces become more difficult to identify. The update algorithm discussed in Chapters 5 and 6 modify the placement of the partitions to the position and shape of the singular surfaces. However, for the update algorithm to terminate, much emphasis is placed on expertise for the initial partition-control law assigned to the agent. Otherwise, the algorithm may never terminate and the control law may not be generated. This is not unreasonable for many scenarios where the magnitude of the opponents' dynamic parameters is unknown, but a general model is known. The ability to adjust different singular surface models would be beneficial in scenarios that involve different vehicle platforms (e.g. truck vs. motorcycle). Therefore, one way to improve upon the control law is to investigate agent types by determining what singular surfaces to associate with certain dynamic capabilities. For example, a vehicle with Ackermann steering will have a surface associated with its turning radius, and a stealth game will have a surfaces associated with acceleration. Such a system can be encoded into a fuzzy inference scheme where adaptive tools already available in the Q-learning literature could be used to adjust the parameters.

Measurement Based Maneuvers

In all of the simulations conducted in this dissertation, either perfect knowledge or measurements of state information is given to the agent. Although not completely unrealistic, it is often the case that the state information cannot be directly detected, and must be estimated through other means such as sequences of distance and bearing information. Depending on the type of measurements available, Target Motion Analysis (TMA) will play a major role in the success of the control law guiding an agent to reach its target. For example, measurements that are bearings-only, in the case of passive sonar, will require that the control law have specific maneuvers [94] to make the problem observable for estimating state information. Such maneuvers could appear as time-varying boundary constraints that would produce a time-varying value function that will not be as easily decomposed as seen in Chapter 7.

There are still many more possibilities for further research and development on this topic. However, this dissertation presents a control law formulation that significantly improves upon the calculation time for differential game theoretic approaches to pursuit-evasion scenarios. As technology improves, the number of applications for this control-law will only increase in complexity and speed, broadening the realm of autonomous control, if only a small margin.

References

[1] M. Aboelaze and F. Aloul. Current and future trends in sensor networks: a survey. In *Wireless and Optical Communications Networks, 2005. WOCN 2005. Second IFIP International Conference on*, pages 551 – 555, march 2005.

[2] T. Alpcan and T. Basar. A game-theoretic framework for congestion control in general topology networks. In *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*, volume 2, pages 1218 – 1224 vol.2, dec. 2002.

[3] Béla Andrásfai. *Graph Theory: flows, matrices*. Akadémiai Kiadó, 1991.

[4] P. J. Antsaklis, K. M. Passino, and S. J. Wang. Towards intelligent autonomous control systems: Architecture and fundamental issues. *Journal of Intelligent & Robotic Systems*, 1:315–342, 1989. 10.1007/BF00126465.

[5] Abraham Bachrach, Ruijie He, and Nicholas Roy. Autonomous flight in unknown indoor environments. *International Journal of Micro Air Vehicles*, 1(4):217–228, 2009.

[6] Tamer Başar and Geert Jan Olsder. *Dynamic Noncooperative Game Theory*. Number 23 in Classics in Applied Mathematics. SIAM, 2 edition, 1999.

[7] J. Baillieul and P.J. Antsaklis. Control and communication challenges in networked real-time systems. *Proceedings of the IEEE*, 95(1):9 –28, jan. 2007.

[8] A. Balluchi, A. Bicchi, A. Balestrino, and G. Casalino. Path tracking control for dubin's cars. In *IEEE International Conference on Robotics and Automation*, April 1996.

[9] A. Balluchi, A. Bicchi, and P. Souéres. Path-following with a bounded-curvature vehicle: a hybrid control approach. *International Journal of Control*, 78(15):1228–1247, October 2005.

[10] S.R. Barrett. Optimizing sensor placement for intruder detection with genetic algorithms. In *Intelligence and Security Informatics, 2007 IEEE*, pages 185 –188, May 2007.

[11] R. Behringer. Road recognition from multifocal vision. In *Intelligent Vehicles '94 Symposium, Proceedings of the*, pages 302 – 307, October 1994.

[12] Richard Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[13] Carolyn A. Berry, Benjamin F. Hobbs, William A. Meroney, Richard P. O'Neill, and William R. Stewart Jr. Understanding how market power can arise in network competition: a game theoretic approach. *Utilities Policy*, 8(3):139 – 158, 1999.

[14] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*, volume 2. Athena Scientific, 2 edition, 2001.

[15] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*, volume 1. Athena Scientific, 3 edition, 2007.

[16] R. Bishop. A survey of intelligent vehicle applications worldwide. In *Intelligent Vehicles Symposium, 2000. IV 2000. Proceedings of the IEEE*, pages 25 –30, 2000.

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

[18] Sotiris Brakatsoulas, Dieter Pfoser, Randall Salas, and Carola Wenk. On map-matching vehicle tracking data. In *Proc. 31st VLDB Conference*, 2005.

[19] R. R. Brooks, Jing-En Pang, and C. Griffin. Game and information theory analysis of electronic countermeasures in pursuit-evasion games. *IEEE Trans. Syst., Man, Cybern. A*, 38(6):1281–1293, November 2008.

[20] Hui Cao, Emre Ertin, and Anish Arora. Minimax equilibrium of networked differential games. *ACM Trans. Auton. Adapt. Syst.*, 3:14:1–14:21, December 2008.

[21] Hui Cao, Emre Ertin, Vinodkrishnan Kulathumani, Mukundan Sridharan, and Anish Arora. Differential games in large-scale sensor-actuator networks. In *Proceedings of the 5th international conference on Information processing in sensor networks*, IPSN '06, pages 77–84, New York, NY, USA, 2006. ACM.

[22] Michael R. Caputo. *Foundations of Dynamic Economic Analysis*. Cambridge University Press, Cambridge, UK, 2005.

[23] P. Cardaliaguet, M. Quincampoix, and P. Saint-Pierre. Some algorithms for differential games with two players and one target. *RAIRO Mathematical Modeling and Numerical Anaylsis*, 28(4):441–461, 1993.

[24] P. Cardaliaguet, M. Quincampoix, and P. Saint-Pierre. Optimal times for constrained nonlinear control problems without local controllability. *Applied Mathematics & Optimization*, 36:21–42, 1997.

[25] K. Chakrabarty, S.S. Iyengar, Hairong Qi, and Eungchun Cho. Grid coverage for surveillance and target location in distributed sensor networks. *Computers, IEEE Transactions on*, 51(12):1448 – 1453, December 2002.

[26] D.E. Chang, S.C. Shadden, J.E. Marsden, and R. Olfati-Saber. Collision avoidance for multiple agent systems. In *Proceedings. 42nd IEEE Conference on Decision and Control*, volume 1, pages 539 – 543, December 2003.

[27] Hyeong Soo Chang and Steven I. Marcus. Two-person zero-sum markov games: Receding horizon approach. *IEEE Trans. Autom. Control*, 48(11):1951–1961, November 2003.

[28] Han-Lim Choi, Min-Jea Tahk, and Hyochoong Bang. Neural network guidance based on pursuit-evasion games with enhanced performance. *Control Engineering Practice*, 14(7):735 – 742, 2006.

[29] Chern Chung. *Reachable Sets Analysis in the Cooperative Control of Pursuer Vehicles*. PhD thesis, University of New South Wales, July 2007.

[30] Timothy Chung, Geoffrey Hollinger, and Volkan Isler. Search and pursuit-evasion in mobile robotics. *Autonomous Robots*, pages 1–18, 2011.

[31] Thomas Clouqueur, Veradej Phipatanasuphorn, Parameswaran Ramanathan, and Kewal K. Saluja. Sensor deployment strategy for target detection. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, WSNA '02, pages 42–48, New York, NY, USA, 2002. ACM.

[32] Jon Cohen. Drone spy plane helps fight california fires. *Science*, 318(5851):727, November 2007.

[33] Rémi Coulom. High-accuracy value-function approximation with neural networks applied to the acrobot. In *ESANN'2004 proceedings - European Symposium on Artificial Neural Networks*, pages 7–12, April 2004.

[34] Emiliano Cristiani and Maurizio Falcone. A fast marching method for pursuit-evasion games. *Communications to SIMAI Congress*, 1, 2006.

[35] Xiaohui Dai, Chi-Kwong Li, and A.B. Rad. An approach to tune fuzzy controllers based on reinforcement learning for autonomous vehicle control. *Intelligent Transportation Systems, IEEE Transactions on*, 6(3):285 – 293, sept. 2005.

[36] Jason C. Derenick, John R. Spletzer, and M. Ani Hsieh. An optimal approach to collaborative target tracking with performance guarantees. *Journal of Intelligent and Robotic Systems*, 56(1-2):47–67, 2009.

[37] Sameh F. Desouky and Howard M. Schwartz. Q(λ)-learning adaptive fuzzy logic controllers for pursuitevasion differential games. *International Journal of Adaptive Control and Signal Processing*, 25(10):910–927, 2011.

[38] Sameh F. Desouky and Howard M. Schwartz. Self-learning fuzzy logic controllers for pursuit-evasion differential games. *Robot. Auton. Syst.*, 59:22–33, January 2011.

[39] S.S. Dhillon and K. Chakrabarty. Sensor placement for effective coverage and surveillance in distributed sensor networks. In *Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE*, volume 3, pages 1609 –1614, March 2003.

[40] Prashant Doshi and Piotr J. Gmytrasiewicz. Monte carlo sampling methods for approximating interactive POMDPs. *Journal of Artificial Intelligence Research*, 34:297–337, 2009.

[41] P.M. Driver and D.A. Humphries. *Protean Behavior: The Biology of Unpredictability*. Oxford University Press, Oxford, 1988.

[42] M. Falcone. Numerical methods for differential games based on partial differential equations. *International Game Theory Review*, 8(2):231–272, 2006.

[43] Mike Farish. Driving blind. *Engineering & Technology*, February 2010.

[44] Allan Feldman. *Welfare Economics and Social Choice Theory*. Kluwer, 1980.

[45] Carl H. FitzGerald. The princess and monster differential game. In *Decision and Control including the Symposium on Adaptive Processes, 1979 18th IEEE Conference on*, volume 18, pages 946 –947, dec. 1979.

[46] U. Franks, H. Loose, and C. Knoppel. Lane recognition on country roads. In *Intelligent Vehicles Symposium, 2007 IEEE*, pages 99 –104, June 2007.

[47] Drew Fudenberg and David K. Levine. *The Theory of Learning in Games*, volume 1 of *MIT Press Books*. MIT Press, December 1998.

[48] Shmuel Gal. Search games with mobile and immobile hider. *SIAM J. Control and Optimization*, 17(1):99–122, 1979.

[49] Brian P. Gerkey, Sebastian Thrun, and Geoff Gordon. Visibility-based Pursuit-evasion with Limited Field of View. *The International Journal of Robotics Research*, 25(4):299–315, 2006.

[50] A. Gern, U. Franke, and P. Levi. Advanced lane recognition-fusing vision and radar. In *Intelligent Vehicles Symposium, 2000. IV 2000. Proceedings of the IEEE*, pages 45 –51, October 2000.

[51] W. M. Getz and M. Pachter. Two-target pursuit-evasion differential games in the plane. *Journal of Optimization Theory and Applications*, 34:383–403, 1981. 10.1007/BF00934679.

[52] S.N. Givigi, H.M. Schwartz, and Xiaosong Lu. An experimental adaptive fuzzy controller for differential games. In *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*, pages 3017 –3023, oct. 2009.

[53] Sidney N. Givigi Jr., Howard M. Schwartz, and Xiaosong Lu. A reinforcement learning adaptive fuzzy controller for differential games. *Journal of Intelligent and Robotic Systems*, 2009.

[54] Piotr Gmytrasiewicz and Prashant Doshi. A framework for sequential planning in multi-agent settings. *Journal of Artificial Intelligence Research*, 24:49–79, 2005.

[55] B. Goode, A. Kurdila, and M. Roan. Pursuit-evasion with acoustic sensing using one step nash equilibria. In *American Control Conference (ACC)*, pages 1925 –1930, 30 2010-july 2 2010.

[56] Brian Goode, Andrew Kurdila, and Mike Roan. Adaptive fuzzy control of switched objective functions in pursuit-evasion scenarios. In *IEEE Conference on Decision and Control (CDC)*, pages 5762 –5767, 2010.

[57] Brian Goode, Andrew Kurdila, and Mike Roan. A graph theoretical approach toward a switched feedback controller for pursuit evasion scenarios. In *Proc. American Control Conference*, pages 4804–4809, 2011.

[58] Brian Goode, Andrew Kurdila, and Mike Roan. A partitioning scheme for a switched feedback control law in two agent pursuit-evasion scenarios. *ASME Journal of Dynamic Systems, Measurement, and Control*, 2011. to appear.

[59] Brian Goode, Andrew Kurdila, and Mike Roan. Performance of a switching controller for pursuit-evasion scenarios with noisy measurments. In *Proc. IASTED Conference on Control and Applications*, pages 217–224, 2011.

[60] Luis Guilamo, Benjamin Tovar, and Steven M. Lavalle. Pursuit-evasion in an unknown environment using gap navigation graphs. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, April 2004.

[61] Joao Hespanha, Maria Prandini, and Shankar Sastry. Probabilistic pursuit-evasion games: A one-step nash approach. In *Proc. of 39th IEEE Conf. on Decision and Control*, pages 2272–2277, September 2000.

[62] Michael Hoel. Distribution and growth as a differential game between workers and capitalists. *International Economic Review*, 19(2), June 1978.

[63] P.A. Ioannou and C.C. Chien. Autonomous intelligent cruise control. *Vehicular Technology, IEEE Transactions on*, 42(4):657 –672, nov 1993.

[64] Rufus Isaacs. *Differential Games*. John Wiley and Sons, Inc., New York, NY, 1965.

[65] Steffen Jorgensen and Georges Zaccour. *Differential Games in Marketing*. Kluwer Academic Publishers, Norwell, MA, 2004.

[66] Shunsuke Kamijo, Yasuyuki Matsushita, Katsushi Ikeuchi, and Masao Sakauchi. Traffic monitoring and accident detection at intersections. *IEEE Trans. Intell. Transp. Syst.*, 1(2), 2000.

[67] Marcin Karpiński, Aline Senart, and Vinny Cahill. Sensor networks for smart roads. In *Proc. Fourth Annual IEEE International Conference on Pervasive Computing and Communications Workshops*, 2008.

[68] P.B. Key and D.R. McAuley. Differential QoS and pricing in networks: where flow control meets game theory. *Software, IEE Proceedings* -, 146(1):39 –43, feb 1999.

[69] Hassan K. Khalil. *Nonlinear Systems*. Prentice Hall, New Jersey, third edition, 2002.

[70] H.J. Kim, R. Vidal, D.H. Shim, O. Shakernia, and S. Sastry. A hierarchical approach to probabilistic pursuit-evasion games with unmanned ground and aerial vehicles. In *Decision and Control, 2001. Proceedings of the 40th IEEE Conference on*, volume 1, pages 634 –639, 2001.

[71] A. Kok, E. Hans, J. Schutten, and W. Zijm. A dynamic programming heuristic for vehicle routing with time-dependent travel times and required breaks. *Flexible Services and Manufacturing Journal*, 22:83–108, 2010.

[72] Y. Koren and J. Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. In *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, volume 2, pages 1398 –1404, April 1991.

[73] N. N. Krasovskiĭ, A. I. Subbotin, and Samuel Kotz. *Game-theoretical control problems*. Springer-Verlag New York, Inc., New York, NY, USA, 1987.

[74] Andreas Krause, Ajit Singh, and Carlos Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *J. Mach. Learn. Res.*, 9:235–284, June 2008.

[75] R. Lachner. Collision avoidance as a differential game: real-time approximation of optimal strategies using higher derivatives of the value function. In *Systems, Man, and Cybernetics, 1997. 'Computational Cybernetics and Simulation'., 1997 IEEE International Conference on*, volume 3, pages 2308 –2313, oct 1997.

[76] R. Lachner, M. H. Breitner, and H. J. Pesch. Real-time collision avoidance: Differential game, numerical solution, and synthesis of strategies. *ANNALS- INTERNATIONAL SOCIETY OF DYNAMIC GAMES*, 5:115–136, 2000.

[77] L.E. Kinsler, A.R. Frey, A.B. Coppens, J.V. Sanders. *Fundamentals of acoustics*. John Wiley and Sons, 2000.

[78] Ernest B. Lee and Lawrence Markus. *Foundations of Optimal Control Theory*. John Wiley and Sons, 1967.

[79] Pai-Shih Lee and Ling-Ling Wang. Collision avoidance by fuzzy logic control for automated guided vehicle navigation. *Journal of Robotic Systems*, 11(8):743–760, 1994.

[80] Jing Li, Ruchuan Wang, Haiping Huang, and Lijuan Sun. Voronoi-based coverage optimization for directional sensor networks. *Wireless Sensor Network*, 1:417–424, 2009.

[81] Renping Liu and Cai ze su. A novel approach based on evolutionary game theoretic model for multi- player pursuit evasion. In *Computer, Mechatronics, Control and Electronic Engineering (CMCE), 2010 International Conference on*, volume 1, pages 107 –110, aug. 2010.

[82] Mark Mansfield, J. Collins, Malachy Eaton, and Thomas Collins. N-learning: A reinforcement learning paradigm for multiagent systems. In Shichao Zhang and Ray Jarvis, editors, *AI 2005: Advances in Artificial Intelligence*, volume 3809 of *Lecture Notes in Computer Science*, pages 684–694. Springer Berlin / Heidelberg, 2005.

[83] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000.

[84] W.M. McEneaney. A curse-of-dimensionality-free numerical method for solution of certain HJB PDEs. *SIAM journal on control and optimization*, 46(4), 2008.

[85] Theo F. Meijman. Mental fatigue and the efficiency of information processing in relation to work times. *International Journal of Industrial Ergonomics*, 20(1):31 – 38, 1997.

[86] A. W. Merz. Implementing air combat guidance laws. *Journal of Dynamic Systems, Measurement, and Control*, 111(4):605–608, 1989.

[87] Antony Merz. The game of two identical cars. *J. Optim. Theory Appl.*, 9(5):324–343, 1972.

[88] Antony W. Merz. *The Homicidal Chauffeur - A Differential Game*. PhD thesis, Stanford University, March 1971.

[89] Hans-Jrg Mettenheim and Michael H. Breitner. Numerical solution of the game of two cars with a neurosimulator and grid computing. In Odile Pourtallier, Vladimir Gaitsgory, and Pierre Bernhard, editors, *Advances in Dynamic Games and Their Applications*, volume 10 of *Annals of the International Society of Dynamic Games*, pages 1–24. Birkhuser Boston, 2009.

[90] Geoffrey F. Miller and Dave Cliff. Protean behavior in dynamic games: Arguments for the co-evolution of pursuit-evasion tactics. In *From animals to animats 3: Proc. 3rd. Int. Conf. on Sim. of Adapt. Behavior*, pages 411–420, 1994.

[91] T. Miloh and M. Pachter. Ship collision-avoidance and pursuit-evasion differential games with speed-loss in a turn. *Computers & Mathematics with Applications*, 18(1-3):77 – 100, 1989.

[92] Todd Moon and Wynn Stirling. *Mathematical Methods and Algorithms for Signal Processing*. Prentice Hall, Upper Saddle River, NJ, 2000.

[93] Rafael Murrieta-Cid, Teja Muppirala, Alejandro Sarmiento, Sourabh Bhattacharya, and Seth Hutchinson. Surveillance strategies for a pursuer with finite sensor range. *Int. J. Rob. Res.*, 26:233–253, March 2007.

[94] S. Nardone, A. Lindgren, and Kai Gong. Fundamental properties and performance of conventional bearings-only target motion analysis. *Automatic Control, IEEE Transactions on*, 29(9):775 – 787, sep 1984.

[95] John Nash. Non-cooperative games. *The Annals of Mathematics*, 54(2):pp. 286–295, 1951.

[96] John Von Neumann and Oskar Morgenstern. *Theory of games and economic behavior*. Princeton University Press, 1944.

[97] T. Ogawa and K. Takagi. Lane recognition using on-vehicle lidar. In *IEEE Intelligent Vehicles Symposium*, pages 540 –545, June 2006.

[98] Meeko Oishi, Claire Tomlin, Vipin Gopal, and Datta Godbole. Addressing multiobjective control: Safety and performance through constrained optimization. In Maria Di Benedetto and Alberto Sangiovanni-Vincentelli, editors, *Hybrid Systems: Computation and Control*, volume 2034 of *Lecture Notes in Computer Science*, pages 459–472. Springer Berlin / Heidelberg, 2001.

[99] Nejat Olgac, Rifat Sipahi, and Ali Fuat Ergenc. The homicidal chauffeur problem with time delayed feedback. *ASME Conference Proceedings*, 2003(37130):1053–1054, 2003.

[100] Joseph O'Rourke. *Art Gallery Theorems and Algorithms*. The International Series of Monographs on Computer Science. Oxford University Press, New York, NY, 1987.

[101] Vladimir E. Ostashev, D. Keith Wilson, Lanbo Liu, David F. Aldridge, Neill P. Symons, and David Marlin. Equations for finite-difference, time-domain simulation of sound propagation in moving inhomogeneous media and numerical implementation. *The Journal of the Acoustical Society of America*, 117(2):503–517, 2005.

[102] M. Pachter and P.R. Chandler. Challenges of autonomous control. *Control Systems, IEEE*, 18(4):92 –97, aug 1998.

[103] M. Pachter and Y. Yavin. A stochastic homicidal chauffeur pursuit-evasion differential game. *Journal of Optimization Theory and Applications*, 34:405–424, 1981.

[104] Meir Pachter and Wayne M. Getz. The geometry of the barrier in the game of two cars. *Optimal Control Applications and Methods*, 1(2):103–118, 1980.

[105] V. S. Patsko and V. L. Turova. Numerical solution to the acoustic homicidal chauffeur game. In *Proceedings of the 19th IFIP TC7 Conference on System Modelling and Optimization*, pages 227–250, Deventer, The Netherlands, The Netherlands, 2000. Kluwer, B.V.

[106] V.S. Patsko and V. L. Turova. Numerical study of the homicidal chauffeur game. In D. Bainov, editor, *Proceedings of the Ninth International Colloquium on Differential Equations*, 1999.

[107] Viktor Vasil'evich Prasolov and Simeon Ivanov. *Problems and theorems in linear algebra*, volume 134. American Mathematical Society, 1994.

[108] R. Tyrrell Rockafellar. *Convex Analysis*. Princeton University Press, Princeton, NJ, 1970.

[109] L. Schenato, Songhwai Oh, S. Sastry, and P. Bose. Swarm coordination for pursuit evasion games using sensor networks. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 2493 – 2498, april 2005.

[110] J A Sethian. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences*, 93(4):1591–1595, 1996.

[111] Steven E. Shladover, Charles A. Desoer, J. Karl Hedrick, Masayoshi Tomizuka, Jean Walrand, Wei-Bin Zhang, Donn H. McMahon, Huei Peng, Shahab Sheikholeslam, and Nick McKeown. Automatic vehicle control developments in the path program. *IEEE Trans. Veh. Technol.*, 40(1), 1991.

[112] A. Sinha and D. Ghose. Control of multiagent systems using linear cyclic pursuit with heterogenous controller gains. *Journal of Dynamic Systems, Measurement, and Control*, 129(5):742–748, 2007.

[113] Adam Smith, Hari Balakrishnan, Michel Goraczko, and Nissanka Bodhi Priyantha. Tracking Moving Devices with the Cricket Location System. In *2nd International Conference on Mobile Systems, Applications and Services (Mobisys 2004)*, Boston, MA, June 2004.

[114] Steven Strogatz. *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry and Engineering*. Westview Press, 1994.

[115] Darbha Swaroop, J. Karl Hedrick, and S. B. Choi. Direct adaptive longitudinal control of vehicle platoons. *IEEE Trans. Veh. Technol.*, 50(1), 2001.

[116] K. Takagi, K. Morikawa, T. Ogawa, and M. Saburi. Road environment recognition using on-vehicle lidar. In *IEEE Intelligent Vehicles Symposium*, pages 120 –125, June 2006.

[117] C.J. Tomlin, J. Lygeros, and S. Shankar Sastry. A game theoretic approach to controller design for hybrid systems. *Proceedings of the IEEE*, 88(7):949 –970, jul 2000.

[118] Walter M. van Buijtenen, Gerard Schram, Robert Babuška, and Henk B. Verbruggen. Adaptive fuzzy control of satellite attitude by reinforcement learning. *IEEE Trans. Fuzzy Syst.*, 6(2):185–194, 1998.

[119] C. Vasudevan and K. Ganesan. Case-based path planning for autonomous underwater vehicles. *Autonomous Robots*, 3:79–89, 1996. 10.1007/BF00141149.

[120] R. Vidal, S. Rashid, C. Sharp, O. Shakernia, Jin Kim, and S. Sastry. Pursuit-evasion games with unmanned ground and aerial vehicles. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 3, pages 2948 – 2955 vol.3, 2001.

[121] R. Vidal, O. Shakernia, H.J. Kim, D.H. Shim, and S. Sastry. Probabilistic pursuit-evasion games: theory, implementation, and experimental evaluation. *IEEE Trans. Robot. Autom.*, 18(5):662 – 669, oct 2002.

[122] Marcos Vieira, Ramesh Govindan, and Gaurav Sukhatme. Scalable and practical pursuit-evasion with networked robots. *Intelligent Service Robotics*, 2:247–263, 2009.

[123] Heinrich von Stackelberg. *Market Structure and Equilibrium*. Springer-Verlag, 1 edition, 2011. trans. D. Bazin, R. Hill, and L. Urch.

[124] Li-Xin Wang. *Adaptive Fuzzy Systems and Control*. PTR Prentice Hall, Englewood Cliffs, NJ, USA, 1994.

[125] Rongben Wang, Youchun Xu, Libin, and Yufan Zhao. A vision-based road edge detection algorithm. In *Intelligent Vehicle Symposium, 2002. IEEE*, volume 1, pages 141 – 147, June 2002.

[126] Greg Welch and Gary Bishop. An introduction to the kalman filter. Technical report, University of North Carolina, 2006.

[127] Paul Werbos. ADP: Goals, opportunities, and principles. In Jennie Si, Andrew G. Barto, Warren B. Powell, and Donald Wunsch II, editors, *Handbook of Learning and Approximate Dynamic Programming*. IEEE Press, 2004.

[128] Thomas A. Wettergren and Russell Costa. Optimal placement of distributed sensors against moving targets. *ACM Trans. Sen. Netw.*, 5:1–25, June 2009.

[129] Byung-Wook Wie. A differential game model of nash equilibrium on a congested traffic network. *Networks*, 23(6):557–565, 1993.

[130] A.M. Wildberger. Autonomous adaptive agents for distributed control of the electric power grid in a competitive electric power industry. In *Knowledge-Based Intelligent Electronic Systems, 1997. KES '97. Proceedings., 1997 First International Conference on*, volume 1, pages 2 –11, may 1997.

[131] Franz Wirl. The dynamics of lobbying - a differential game. *Public Choice*, 80(3-4), September 1994.

[132] Bin Xu, A. Kurdila, and D.J. Stilwell. A hybrid receding horizon control method for path planning in uncertain environments. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 4887 –4892, Oct. 2009.

[133] Chao Yong and Eric J. Barth. Real-time dynamic path planning for dubins' nonholonomic robot. In *Proc. 45th IEEE Conference on Decision & Control*, 2006.

[134] J. Yuh. Design and control of autonomous underwater robots: A survey. *Autonomous Robots*, 8:7–24, 2000. 10.1023/A:1008984701078.

[135] Michael Zavlanos and George Pappas. Distributed hybrid control for multiple-pursuer multiple-evader games. In Alberto Bemporad, Antonio Bicchi, and Giorgio Buttazzo, editors, *Hybrid Systems: Computation and Control*, volume 4416 of *Lecture Notes in Computer Science*, pages 787–789. Springer Berlin / Heidelberg, 2007.

[136] E. Zermelo. Über das navigationproble bei ruhender oder veranderlicher windverteilung. *Z. Angrew. Math. und. Mech.*, 11, 1931.

[137] Sijian Zhang, Gangfeng Yan, and Weihua Sheng. An efficient planning method for mobile robot based 3d digitization. In *Intelligent Control and Automation (WCICA), 2010 8th World Congress on*, pages 1348 –1353, july 2010.

Appendix A

Pictures of the Experimental Setup

This section provides a few additional pictures of the test setup from Chapter 7.

Figure A.1: Start of a scenario. Each agent is placed in opposite corners of the room. Each agent is equipped with a FitPC2 mounted with velcro and a listener Cricket module mounted to the top of the Roomba. The robot and processor were interfaced via the IRobot serial connection.

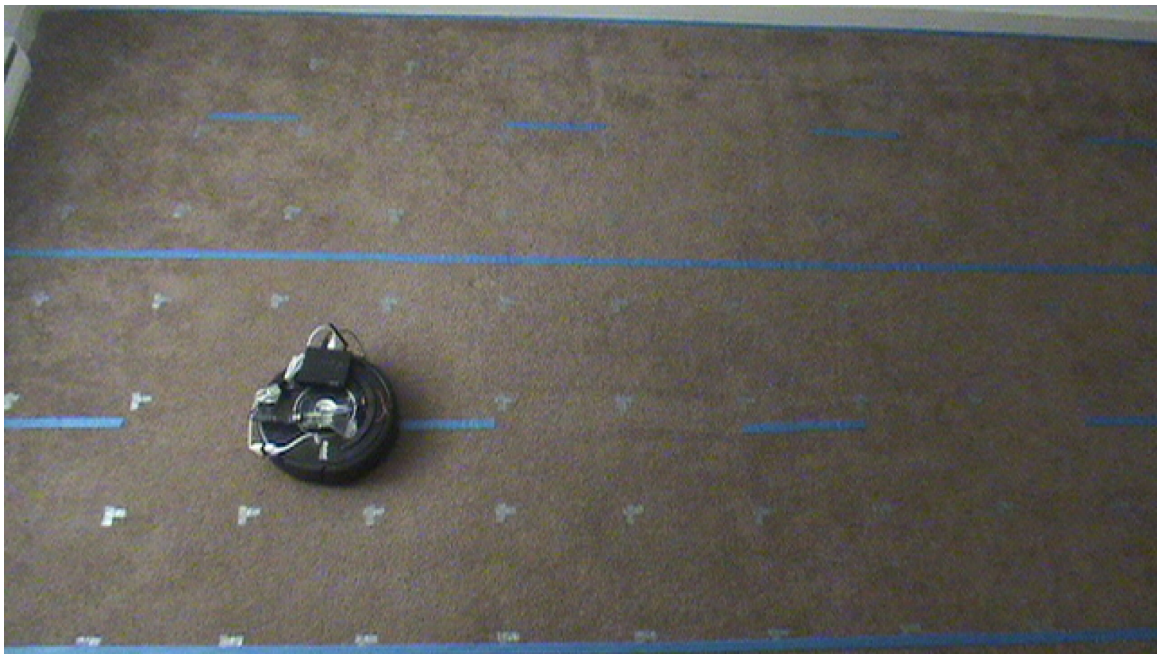


Figure A.2: An overhead view of the evading agent in the middle of its lane. The blue tape represents the counter-flow lanes that make up the road. The gray tape is the grid used to calibrate the Cricket measurement system.

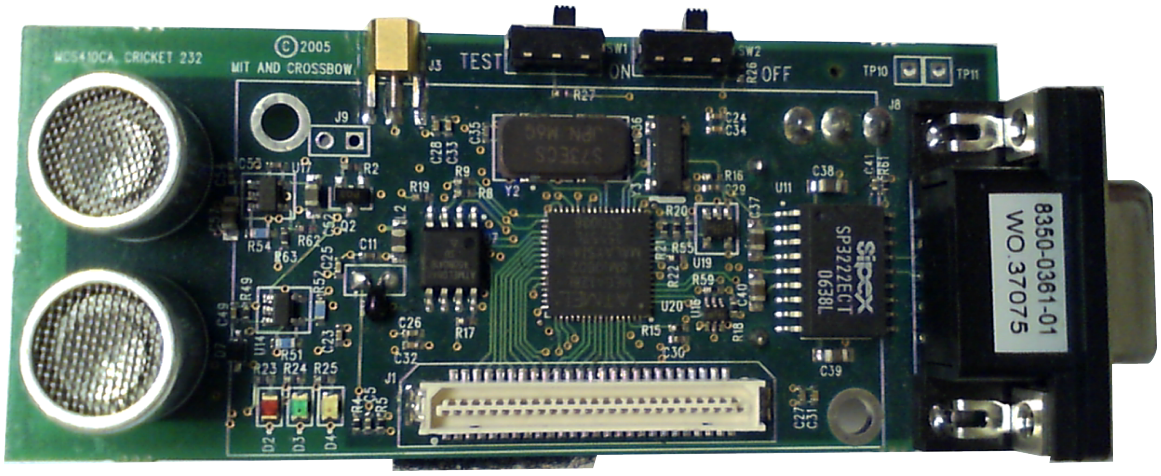


Figure A.3: One of the many beacons/listeners used for position measurements. These devices use both ultrasonic and radio frequency transmissions to determine the distance between each transmitter/receiver pair.

Figure A.4: A portion of the beacon network installed on the ceiling. All beacons were hard-wired to avoid complications with weakened batteries.

Appendix B

Updated Sections 5.3.3 and 5.3.4

Due to copyright restrictions, these sections are updated as follows.

B.1 Properties of the Adjacency Matrix

The controller may be represented by a digraph where the partitions, Σ, are nodes, and edges occur if the trajectory transitions to another partition,

$$\exists(w \in \mathcal{W}, \bar{t}) \; s.t. \; x^t \in \sigma_k, x^{\bar{t}} \in \sigma_{j \neq k}, t < \bar{t} < \infty.$$

Edges represent transitions across partitions, and transitions can be coupled along a trajectory to form a path. A loop edge occurs when the pursuer wants to transition, but fails and remains in the same partition as seen in Example 1. The digraph may be represented as a square adjacency matrix, A

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,q+1} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,q+1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{q+1,1} & a_{q+1,2} & \cdots & a_{q+1,q+1} \end{bmatrix} \tag{B.1}$$

populated according to

$$a_{i,j} = \begin{cases} 0, \text{ if there is no edge from node } i \text{ to } j \\ 1, \text{ if there is an edge from node } i \text{ to } j \end{cases} \tag{B.2}$$

One useful property is that, A^q, contains a count of how many paths of length q exist from node i to j [3].

B.2 Inclusion of a Partition in the Capture Set

Trajectories starting in $\mathcal{C}_F(d(\cdot), \mathcal{M}(x))$ satisfy:

(1) $(x, u, d) \in \mathcal{W}$ with $u = \mathcal{M}(x)$.

(2) Termination in the target set in finite time.

If an edge exists between two nodes, then the trajectory between the partitions is valid. The problem of determining whether or not states in a partition are a member of the capture set is now cast in terms of the adjacency matrix.

Theorem 10. *Assume a switched feedback law, $\mathcal{M}(x)$, is given to an agent and the opposing agent uses a control, $d(\cdot)$. Set $\bar{A} = \sum_{k=1}^{q+1} A^k$. If*

(1) A has all 0 eigenvalues,

(2) the entry, $\bar{a}_{i,1} > 0$ for $i > 1$, and

(3) for all entries, $\bar{a}_{j,1} = 0, j > 1$, then, $\bar{a}_{i,j} = 0$.

then partition, $\sigma_{i-1} \in \mathcal{C}_F(d(\cdot), \mathcal{M}(x))$.

Proof. A matrix is nilpotent if and only if it has all 0 eigenvalues (for proof, see [107]), and there is an integer, $k \in \{1, \ldots, q\}$ such that $A^k = \mathbf{0}$. For higher powers, $A^j = A^k A^{j-k}, j > k$ such that $A^j = \mathbf{0}, \forall j \geq k$. Since A^k determines how many paths of length k exist between two partitions, zero eigenvalues guarantee that all paths from any node to another node are finite. It follows that all trajectories will reside in a single partition after a finite amount of time, because there are no cycles or loops in which the trajectory can be caught.

To show that the final partition can only be the target set, consider a finite path of the digraph that terminates in any partition $\sigma_{j \neq 0}$. Termination in a partition implies positive invariance because it never leaves that set of states upon entry. There are two modes by which this will occur: (1) the trajectory started in σ_j or (2) the trajectory traversed into σ_j from another partition. If (1) is true, then $\bar{a}_{i,1} = 0, i = j + 1$ which contradicts condition 2. Likewise, if (2) is true, then $\bar{a}_{j,1} = 0$ for some $j > 1, i \neq j + 1$ and $\bar{a}_{i,j} \geq 1$, which contradicts condition 3. Therefore, if the conditions 1-3 hold, then the trajectory is guaranteed to terminate in σ_0. □

Appendix C

Copyright Permissions

Enclosed in order are the following copyright permissions:

1. American Automatic Control Council letter of permission - for chapter 3.
2. IEEE statement of permission - chapter 4.
3. ASME copyright (see pg. 2) - chapter 5.
4. IASTED letter of permission - chapter 6.
5. SPIE letter of permission - chapter 8.

Author is copyright owner of all remaining chapters.

Brian Goode

From: "Pradeep Misra" <p.x.misra@gmail.com>
To: <bjgoode@vt.edu>
Sent: Wednesday, October 12, 2011 1:18 PM
Subject: Re: [Contact Information] Reprint paper for dissertation

None - only requirements is that you refer to the ACC paper where you are using the content of those papers.

On Wed, Oct 12, 2011 at 1:16 PM, <bjgoode@vt.edu> wrote:

> Brian J. Goode sent a message using the contact form at http://a2c2.org/contact.
>
> Dear sir,
>
> I am an author of two publications in the ACC conference and I was wondering what steps I need to take to get permission to include them in my dissertation. Thank you!
>
> Brian

Copyright Clearance Center RightsLink®

Home Account Info Help

IEEE Requesting permission to reuse content from an IEEE publication

Title:	Adaptive fuzzy control of switched objective functions in pursuit-evasion scenarios
Conference Proceedings:	Decision and Control (CDC), 2010 49th IEEE Conference on
Author:	Goode, B.; Kurdila, A.; Roan, M.
Publisher:	IEEE
Date:	15-17 Dec. 2010

Copyright © 2010, IEEE

Logged in as:
Brian Goode
LOGOUT

Thesis / Dissertaion Reuse

We are happy to grant this permission. Our only requirement is that the requester provides a full credit notice to the original source (author, paper, publication), followed by the IEEE copyright notice.

BACK CLOSE WINDOW

Copyright © 2011 Copyright Clearance Center, Inc. All Rights Reserved. Privacy statement.
Comments? We would like to hear from you. E-mail us at customercare@copyright.com

COPYRIGHT AGREEMENT (as of March 2010)

ASME Publishing • Three Park Avenue • New York, NY 10016
For questions about Conference paper copyright, please e-mail copyright@asme.org
For questions about Journal paper copyright, please email journalcopyright@asme.org

Before publication of your paper in a conference proceedings or journal, ASME must receive a signed Copyright Agreement. For conference papers, this form should be received by the deadline indicated by the Conference. Other forms may NOT be substituted for this form, nor may any wording on the form be changed.

PAPER NUMBER (for conference/journal papers): DS-10-1338

TITLE: A partitioning Scheme for a Switched Feedback Control Law in Two-Agent Pursuit-Evasion Scenarios

AUTHOR(s): Brian J. Goode , Andrew Kurdila, Michael Roan

CONFERENCE NAME: -------

JOURNAL NAME: Dynamic Systems, Measurement, &Control

ASME requests that authors/copyright owners assign copyright to ASME in order for a conference or journal paper to be published by ASME. Authors exempt from this request are direct employees of the U.S. Government, whereby papers are not subject to copyright protection in the U.S., or non-U.S. government employees, whose governments hold the copyright to the paper. Otherwise, the author/ copyright owner(s) of the Paper should sign this form as instructed below. Please refer to the section below "Who Should Sign" and also to ASME's FAQ page for more information regarding copyright ownership and the copyright process.

WHO SHOULD SIGN

Only the copyright owner(s) of the Paper, or an authorized representative, can sign this form. If one of the following applies, you may not own the copyright to the paper, or you may not be authorized to sign this agreement, and you may need to have the appropriate copyright owner(s) or organization representative sign this Agreement:
(1) You created the Paper within the scope of your employment, and your employer is the copyright owner
(2) You created the Paper under an independent contractor agreement**
(3) You received a grant that funded your Paper.

Please review your company policies regarding copyright, and if you are not authorized to sign this agreement, please forward to the appropriate organization representative. Please review applicable company, institutional, and grant policies and your employment/independent contractor agreement to determine who holds the rights to your Paper. For more information, please refer to the FAQs.

****Note to U.S. Government Contractors:** If you created the Paper under contract with the U.S. Government, e.g., U.S. Government labs, the paper may be subject to copyright, and you or your employer may own the copyright. Please review your company/institutional policies and your contractor agreement. Your Paper may also require a footer acknowledging contract information and also the following statement:

"The United States Government retains, and by accepting the article for publication, the publisher acknowledges that the United States Government retains, a non-exclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this work, or allow others to do so, for United States Government purposes."

It is your responsibility to ensure that the final PDF version of the Paper you submit includes all necessary footers and statements required under your contract.

COPYRIGHT ASSIGNMENT

The following terms of copyright assignment refer to Sections 1, 2, and 3. Sections 4 and 5 may not be subject to copyright.

The undersigned hereby assigns irrevocably to ASME all worldwide rights under copyright in the Paper.

Authors retain all proprietary rights in any idea, process, procedure, or articles of manufacture described in the Paper, including the right to seek patent protection for them. Authors may perform, lecture, teach, conduct related research, display all or part of the Paper, and create derivative works in print or electronic format. Authors may reproduce and distribute the Paper for non-commercial purposes only. Non-commercial applies only to the sale of the paper per se. For all copies of the Paper made by Authors, Authors must acknowledge ASME as original publisher and include the names of all author(s), the publication title, and an appropriate copyright notice that identifies ASME as the copyright holder.

PLEASE READ THE TERMS AND CONDITIONS WHICH ARE FULLY INCORPORATED IN THIS AGREEMENT.

1. PAPERS OWNED BY ONE AUTHOR OR JOINT AUTHORS; DESIGNATED AUTHORS (For jointly authored works, all authors should submit a signed Agreement, or one Designated Author may sign on behalf of the other authors, but **ONLY IF** the designated author has secured written authorization to do so from all other authors. The designated author must be able to produce such written authorization if requested.)

Designated authors, please sign below and list the names of the co-authors for whom you are signing. Please include full contact information for each author. Attach additional sheets if necessary. Author(s)/co-author(s) not covered by the Designated author, please sign in the appropriate section below and provide full contact information.

Author, Co-Author, or Designated Author

Name: Brian J. Goode Signature:  Date: 5/19/2011

Affiliation: Virginia Tech (Company or Institution) Job Title: Research Assistant

Address: 136 Durham Hall (Street Address) Blacksburg (City) VA (State) 24060 (Zip Code) US (Country)

Phone: 757-871-5328 Fax: ---- Email: bjgoode@vt.edu

Author

Name: Andrew Kurdila Date: 5/19/2011

Affiliation: Virginia Tech (Company or Institution) Job Title: Professor

Address: 305 Durham Hall (Street Address) Blacksburg (City) VA (State) 24060 USA (Zip Code) (Country)

Phone: (540) 231-4162 Fax: ----- Email: kurdila@vt.edu

2. PAPERS OWNED BY EMPLOYER OF AUTHOR(s) (Author may sign if so authorized; otherwise, an officer or other authorized agent of the employer should sign below.)

Name: _____ Signature: _____ Date: _____

Affiliation: _____ (Company or Institution) Job Title: _____

Address: _____ (Street Address) (City) (State) (Zip Code) (Country)

Phone: _____ Fax: _____ Email: _____

Author

The following terms of copyright assignment refer to Sections 1, 2, and 3. Sections 4 and 5 may not be subject to copyright.

The undersigned hereby assigns irrevocably to ASME all worldwide rights under copyright in the Paper.

Authors retain all proprietary rights in any idea, process, procedure, or articles of manufacture described in the Paper, including the right to seek patent protection for them. Authors may perform, lecture, teach, conduct related research, display all or part of the Paper, and create derivative works in print or electronic format. Authors may reproduce and distribute the Paper for non-commercial purposes only. Non-commercial applies only to the sale of the paper per se. For all copies of the Paper made by Authors, Authors must acknowledge ASME as original publisher and include the names of all author(s), the publication title, and an appropriate copyright notice that identifies ASME as the copyright holder.

PLEASE READ THE TERMS AND CONDITIONS WHICH ARE FULLY INCORPORATED IN THIS AGREEMENT.

1. PAPERS OWNED BY ONE AUTHOR OR JOINT AUTHORS; DESIGNATED AUTHORS (For jointly authored works, all authors should submit a signed Agreement, or one Designated Author may sign on behalf of the other authors, but **ONLY IF** the designated author has secured written authorization to do so from all other authors. The designated author must be able to produce such written authorization if requested.)

Designated authors, please sign below and list the names of the co-authors for whom you are signing. Please include full contact information for each author. Attach additional sheets if necessary. Author(s)/co-author(s) not covered by the Designated author, please sign in the appropriate section below and provide full contact information.

Author, Co-Author, or Designated Author

Name: Michael Roan Signature: _____ Date: 5/11/11

Affiliation: Virginia Tech (Company or Institution) Job Title: Professor

Address: 114-G Randolph (Street Address) Blacksburg (City) VA (State) 24060 (Zip Code) USA (Country)

Phone: (540)231-5846 Fax: _____ Email: mroan@vt.edu

Author

Name: _____ Date: _____

Affiliation: _____ (Company or Institution) Job Title: _____

Address: _____ (Street Address) (City) (State) (Zip Code) (Country)

Phone: _____ Fax: _____ Email: _____

2. PAPERS OWNED BY EMPLOYER OF AUTHOR(s) (Author may sign if so authorized; otherwise, an officer or other authorized agent of the employer should sign below.)

Name: _____ Signature: _____ Date: _____

Affiliation: _____ (Company or Institution) Job Title: _____

Address: _____ (Street Address) (City) (State) (Zip Code) (Country)

Phone: _____ Fax: _____ Email: _____

Author

Name: _____ Date: _____

Affiliation: _____ Job Title: _____
(Company or Institution)

Address: _____
(Street Address) (City) (State) (Zip Code) (Country)

Phone: _____ Fax: _____ Email: _____

3. **PAPERS CREATED BY U.S. FEDERAL OR STATE GOVERNMENT CONTRACTORS** (Please sign below if you created the Paper under contract with the U.S. Federal or State government (e.g., U.S. government labs), and you are authorized to sign on behalf of your organization; otherwise, an officer or other authorized agent should sign. Please include any required footers on the final PDF version of your Paper.)

Name: _____ **Signature:** _____ **Date:** _____

Affiliation: _____ Job Title: _____
(Company or Institution)

Address: _____
(Street Address) (City) (State) (Zip Code) (Country)

Phone: _____ Fax: _____ Email: _____

Author

Name: _____ Date: _____

Affiliation: _____ Job Title: _____
(Company or Institution)

Address: _____
(Street Address) (City) (State) (Zip Code) (Country)

Phone: _____ Fax: _____ Email: _____

PAPERS NOT SUBJECT TO COPYRIGHT ASSIGNMENT

4. PAPERS CREATED BY U.S. FEDERAL GOVERNMENT EMPLOYEES

Please sign below if you created the Paper within your scope of your employment by the U.S. Federal Government, and you are authorized to sign on behalf of your agency or department; otherwise, an officer or authorized agent should sign. Please include the following footer on the final PDF version of the Paper you submit: "This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States. Approved for public release; distribution is unlimited."

Name: _____ Signature: _____ Date: _____

Affiliation: _____ Job Title: _____
(Company or Institution)

Address: _____
(Street Address) (City) (State) (Zip Code) (Country)

Phone: _____ Fax: _____ Email: _____

Author

Name: _____ Date: _____

Affiliation: _____ Job Title: _____
(Company or Institution)

Address: _____

(Street Address) (City) (State) (Zip Code) (Country)

Phone:_____ Fax:_____ Email: _____

---

5. PAPERS CREATED BY NON-U.S. GOVERNMENT EMPLOYEES

Please fill in and sign below if you created the Paper within the scope of your duties as an officer or employee of a non-U.S. government, and you are authorized to sign on behalf of your government; otherwise, an officer or authorized agent should sign.

This work was prepared while under employment by the Government of_____ as part of the official duties of the author indicated below, and as such copyright is owned by that Government, which reserves its own copyright under national law. The copyright notice on the paper should read:_____

Name: _____ Signature: _____ Date: _____

Affiliation:_____ Job Title: _____
(Company or Institution)

Address: _____
(Street Address) (City) (State) (Zip Code) (Country)

Phone:_____ Fax:_____ Email: _____

Author

Name: _____ Date: _____

Affiliation:_____ Job Title: _____
(Company or Institution)

Address: _____
(Street Address) (City) (State) (Zip Code) (Country)

Phone:_____ Fax:_____ Email: _____

August 31, 2009

ASME COPYRIGHT FORM TERMS AND CONDITIONS

The following terms and conditions are fully incorporated into the Copyright Form. Please read them carefully.

REPRESENTATIONS, OBLIGATIONS, ACKNOWLEDGEMENTS, AND INDEMNIFICATION

You represent and acknowledge that:

(A) This Paper represents: either the first publication of material or the first publication of an original compilation of information from a number of sources as specifically noted by footnotes and/or bibliography.

(B) You have the right to enter into this Copyright Form and to make the assignment of rights to ASME. If the Paper contains excerpts from other copyrighted material (including without limitation any diagrams, photographs, figures or text), you have acquired in writing all necessary rights from third parties to include those materials in the Paper, and have provided appropriate credit for that third-party material in footnotes or in a bibliography.

(C) If you are signing this Form on behalf of any co-authors or other copyright holders, you have obtained express authorizations from all those authors and/or copyright holders to make this assignment of rights to ASME.

(D) To the best of the author's knowledge, all statements contained in the Paper purporting to be facts are true or supported by reasonable scientific research, the Paper does not contain any defamatory or libelous material and does not infringe any third party's copyright, patent, trade secret, or other proprietary rights and does not violate the right of privacy or publicity of any third party or otherwise violate any other applicable law; furthermore that to the best of your ability, you are responsible for ensuring the accuracy of your research and the Paper's content.

(E) If the Paper was produced in the course of an author's employment by, or contractual relationship with, the U.S. Federal or State Government and/or contains classified material, it has been appropriately cleared for public release and such is indicated in the paper.

(F) The Paper is not subject to any prior claim, encumbrance or form and is not under consideration for publication elsewhere.

(G) You have appropriately cited and acknowledged all third parties who have contributed significantly in the Paper's technical aspects.

(H) ASME is not responsible for any misrepresentation, errors or omissions by those signing this copyright form.

(I) All print and electronic copies of the Paper submitted to ASME become ASME's physical property regardless of whether or not ASME publishes the Paper, and that ASME is not obligated to publish your paper (see the Termination Section below if your paper is not published).

(J) ASME is not responsible for any of your expenses incurred in connection with preparing the Paper or attending meetings to present it, nor will ASME pay you any financial compensation if it publishes your Paper.

(K) Subject to and to the maximum extent permitted by law, you agree to indemnify and hold harmless ASME from any damage or expense related to a breach of any of the representations and warranties above.

TERMINATION

If ASME decides not to publish your Paper, this Form, including all of ASME's rights in your Paper, terminates and you are thereafter free to offer the Paper for publication elsewhere.

GENERAL PROVISIONS

This Copyright Form, the Terms & Conditions, and ASME Copyright Guidelines, constitutes the entire agreement between you and ASME, and supersedes all prior or current negotiations, understandings and representations, whether oral or written, between you and ASME concerning the Paper.

This Agreement is governed by, and should be construed in accordance with, the laws of the State of New York, United States of America, applicable to agreements made and performed there, except to the extent that your institution is prohibited by law from entering contracts governed by New York law, in which limited case this Agreement is governed by, and should be construed in accordance with, the laws of the jurisdiction in which your institution is located. Any claim, dispute, action or proceeding relating to this Agreement may be brought only in the applicable state and federal courts in the State and County of New York, and you expressly consent to personal jurisdiction and venue in any of those courts.

Brian J. Goode

From: IASTED Calgary [Calgary@iasted.org]
Sent: Monday, July 25, 2011 3:37 PM
To: 'bjgoode@vt.edu'
Subject: RE: IASTED Paper 729-042

Dear Brian,

Thank you for your email. You can include the relevant parts in your dissertation as long as you reference where the sections were originally published.

If you have any more questions, please do not hesitate to contact me.

Yours sincerely,

Nadia Leamy

-----Original Message-----
From: bjgoode@vt.edu [mailto:bjgoode@vt.edu]
Sent: Saturday, July 23, 2011 6:47 PM
To: IASTED Calgary
Subject: IASTED Paper 729-042

Good evening,

My name is Brian Goode and I am the primary author for paper 729-042.
This was submitted to the Control and Applications Conference, 2011. I would like permission to include parts of this paper in my dissertation. What do I need to do to make this possible? Thank you for your time.

Brian

Brian J. Goode

From: Scott McNeill [scottm@spie.org]
Sent: Monday, October 03, 2011 11:39 AM
To: Brian J. Goode
Subject: RE: Reprint for Dissertation

Dear Brian Goode,

Thank you for seeking permission from SPIE to reprint material from our publications. As an author of the cited work, you retain co-owner rights to the original content therein. Publisher's permission is hereby granted under the following conditions: (1) the material to be used has appeared in our publication without credit or acknowledgment to another source; and (2) you credit the original SPIE publication. Include the authors' names, title of paper, volume title, SPIE volume number, and year of publication in your credit statement.

Sincerely,

Scott McNeill for
Eric Pepper, Director of Publications
SPIE
P.O. Box 10, Bellingham WA 98227-0010 USA
360/676-3290 (Pacific Time) eric@spie.org

From: Brian J. Goode [mailto:bjgoode@vt.edu]
Sent: Friday, September 30, 2011 9:29 PM
To: reprint_permission
Subject: Reprint for Dissertation

Dear sir or madam:

I am defending my dissertation on Oct. 21st and would like permission to use this paper, of which I am an author as a chapter:

Goode, B., Chin, P., Roan, M. "Sensor Reduction Techniques using Bellman Optimal Approximations of Target and Environment Dynamics." SPIE Conference on Defense, Security, and Sensing, 2011, vol. 8050, 805018.

Please advise on what actions I need to take. Thank you very much!

Brian

_____
Brian J. Goode
Ph. D Candidate
Vibrations and Acoustics Lab
136 Durham Hall, Virginia Tech