MODE IDENTIFICATION USING STOCHASTIC HYBRID MODELS WITH APPLICATIONS TO CONFLICT DETECTION AND RESOLUTION

BY

ASAL NASERI KOUZEHGARANI

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Aerospace Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2010

Urbana, Illinois

Doctoral Committee:

Assistant Professor Natasha Neogi, Chair
Professor Petros Voulgaris
Professor P.R. Kumar
Associate Professor Esa Rantanen

Abstract

Most models of aircraft trajectories are non-linear and stochastic in nature; and their internal parameters are often poorly defined. The ability to model, simulate and analyze realistic air traffic management conflict detection scenarios in a scalable, composable, multi-aircraft fashion is an extremely difficult endeavor. Accurate techniques for aircraft mode detection are critical in order to enable the precise projection of aircraft conflicts, and for the enactment of altitude separation resolution strategies.

Conflict detection is an inherently probabilistic endeavor; our ability to detect conflicts in a timely and accurate manner over a fixed time horizon is traded off against the increased human workload created by false alarms —that is, situations that would not develop into an actual conflict, or would resolve naturally in the appropriate time horizon-thereby introducing a measure of probabilistic uncertainty in any decision aid fashioned to assist air traffic controllers. The interaction of the continuous dynamics of the aircraft, used for prediction purposes, with the discrete conflict detection logic gives rise to the hybrid nature of the overall system. The introduction of the probabilistic element, common to decision alerting and aiding devices, places the conflict detection and resolution problem in the domain of probabilistic hybrid phenomena.

A hidden Markov model (HMM) has two stochastic components: a finite-state Markov chain and a finite set of output probability distributions. In other words an unobservable stochastic process (hidden) that can only be observed through another set of stochastic processes that generate the sequence of observations. The problem of self separation in distributed air traffic management reduces to the ability of aircraft to communicate state information to neighboring aircraft, as well as model the evolution of aircraft trajectories between communications, in the presence of probabilistic uncertain dynamics

as well as partially observable and uncertain data. We introduce the Hybrid Hidden Markov Modeling (HHMM) formalism to enable the prediction of the stochastic aircraft states (and thus, potential conflicts), by combining elements of the probabilistic timed input output automaton and the partially observable Markov decision process frameworks, along with the novel addition of a Markovian scheduler to remove the non-deterministic elements arising from the enabling of several actions simultaneously. Comparisons of aircraft in level, climbing/descending and turning flight are performed, and unknown flight track data is evaluated probabilistically against the tuned model in order to assess the effectiveness of the model in detecting the switch between multiple flight modes for a given aircraft. This also allows for the generation of probabilistic distribution over the execution traces of the hybrid hidden Markov model, which then enables the prediction of the states of aircraft based on partially observable and uncertain data.

Based on the composition properties of the HHMM, we study a decentralized air traffic system where aircraft are moving along streams and can perform cruise, accelerate, climb and turn maneuvers.We develop a common decentralized policy for conflict avoidance with spatially distributed agents (aircraft in the sky) and assure its safety properties via correctness proofs.

Acknowledgments

I would like to express my gratitude and appreciation to Prof. Natasha Neogi for serving as my advisor and providing guidance, motivation, and support for this work. I will always admire her brilliance and her insight for interdisciplinary research. Along with being my mentor, she has always been a good friend to me during the good and not so good times. I have been lucky to have you as my advisor and as my friend. I will always treasure your kindness and friendship.

I was fortunate enough to have the help and advice of Prof. Esa Rantanen. He was always ready to offer his wisdom and his experience as a former controller to keep me in track in regards to human factor issues. And he made it possible for me to test the applicability of my theories by generously sharing his vast amount of real flight data. I must thank the other memebers of my committe, Prof. P.R. Kumar and Prof. Petros Voulgaris, who stirred me in the right direction with their infinite knowledge and insight into control theory.

My fellow graduate students, Andres Ortiz and Daniel Uhlig, deserve many thanks. I will never forget your help while I was preparing for my preliminary and final defense exams. You were always there for practice talks and took care of all the other details.

I wish to thank my parents and my family for their support and encouragement of my academic pursuits. And, most importantly, to my husband Amir, you are my everything, thanks for being with me.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Air Traffic Control

The goal of air traffic control ATC) system is to accomplish the safe, efficient flow of traffic from origin to destination. The task of air traffic control includes ground operations from the gate to the taxiway to the runway, takeoff and climb operations to reach a cruising altitude, cross-country flight to the destination, approach and landing operations at the destination, and taxi back to the gate. There are three general classes of controllers. Ground and local controllers (referred to as tower controllers) handle aircraft on the taxiways and runways and also through takeoffs and landings. Radar controllers handle aircraft from their takeoff to their cruising altitude (departure control) and then return them through their approach at the destination (approach control). The busy region surrounding airport facilities is called the terminal radar control area, TRACON. En-route controllers working at the air route traffic control center (ARTCC) manage the flow of traffic along the airways between the TRACON areas. Flow of aircraft across the entire United States is managed by the Air Traffic Control System Command Center in Herndon, Virginia. [1]

In the current ATC system, the pilot determines the flight's objectives and decides how those objectives can best be met. The objectives include the

destination airport, route of flight, proposed altitude, cruising airspeed, time of departure, climb and descent profiles, and speed schedules [2]. However, the controller gets only a limited number of these objectives through the flight plan. Using this limited information, the controller is responsible for separating aircraft.

Detection of conflicts —defined here as the potential loss of prescribed separation [3] —between two aircraft is the primary purpose of air traffic control. This task, multiplied by a large number of aircraft pairs under a single controller's responsibility at any time exacts a high toll on human controllers in terms of mental workload; furthermore, the potential consequences of a miss —a midair collision —are simply unacceptable within the modern air transportation systems. Therefore, ATC procedures have been designed to maximize safety (i.e., to minimize the potential of conflicts) with a necessary tradeoff in reduced system capacity.

Safety is ensured, in large part, by guaranteeing minimum vertical and lateral separation. To ensure total safety, the aircraft would never fly. And to ensure a greater safety level than what we have today, separation thresholds would be greater than what we currently practice. However, this compromise the second goal of ATC, efficiency, which is demanded by consumers and pilots. Every airport has a capacity, i.e., the number of aircraft it can receive per unit of time. And the goal is to meet this capacity. Optimization is limited by changes in the weather, wake vortices following the passage of heavy aircraft and the limited ability to predict the future.

For the current ATC system, for en-route airspace the minimum horizontal separation if 5 nmi, while the minimum vertical separation is 1000 ft. When the controller is separating aircraft, it is required to ensure that the airspace reserved for one aircraft does not overlap the airspace reserved for

another. Controllers use four methods to separate aircraft: vertical, lateral, longitudinal, and visual separation [2]. The basic *vertical separation method* is for the controller to request that the pilot report passing through or leveling off at a particular altitude. Once this altitude has been reported, the controller can assign another altitude to a different aircraft, as long as the two altitudes differ by at least 1000 feet. One exception to this rule is that if both aircraft are climbing, once the higher aircraft has reported leaving an assigned altitude and is climbing to an altitude at least 1000 feet higher, the lower aircraft may be assigned the altitude the first aircraft has vacated.

Exclusive use of vertical separation can result in inefficient use of airspace. Thus, controllers consider alternating methods of separating aircraft, one of which is *lateral separation*. With lateral separation, each aircraft must be established on an airway whose protected airspaces do not overlap. If this cannot be accomplished, one of the other methods of separation should be used.

Holding patterns are used whenever an aircraft does not have sufficient airspace to continue toward its destination. An aircraft is restricted to a small area when it is within a holding pattern. By clearing aircraft to fly either above or below other holding aircraft, vertical separation is applied. To ensure lateral separation (when the reserved airspace for the holding aircraft does not overlap with the reserved airspace belonging to other aircraft) the controller should take into account the speed of the aircraft. Since the inbound leg of a standard holding pattern is one minute, faster aircraft will cover a greater distance in that time.

Longitudinal separation between two aircraft that are flying along the same route can be applied when the aircraft are flying at or near the same airspeed or the leading aircraft is significantly faster.

Visual separation is one of the most flexible methods. In this method, it is required that either of the pilots sees the other aircraft and provide the required separation, or that the controller is able to see both aircraft and provide safe separation.

Air traffic is expected to grow by 5% annually over the next decade. In view of the current ATC limitations, the Federal Aviation Administration (FAA) and the aviation community are working together on two initiatives: *Free Flight*, and ATC modernization. Free flight allows pilots to choose their own routes, altitudes, and speeds and modify them in real time as they see fit. User preference would be restricted only in congested airspace or to prevent unauthorized entry of special use airspace. Free flight is potentially feasible because of enabling technologies, like Global Positioning system (GPS), datalink communications like Automatic Dependence Surveillance-Broadcast (ADS-B), Traffic Alert and Collision Avoidance Systems (TCAS), and powerful on-board computation.

To increase system throughput and the number of aircraft individual controllers can safely monitor under the demand of increased traffic levels, several automated aids that detect conflicts based on complex algorithms have been developed. However, such automated tools are inherently imperfect [4, 5]. NASA's Center-TRACON Automation System (CTAS) [6] and MITRE's URET [7] are decision support tools for ground controllers.

CTAS is a system developed by NASA that provides users with airspace capacity improvement, delay reductions and fuel savings by using computer automation to assist controllers in efficiently descending, sequencing, and spacing arriving aircraft. It provides four functions: traffic management advisor, descent advisor, final approach spacing toll, and expedite departure path.

The User Request Evaluation Tool (URET) was developed at MITRE Corporation's Center for Advanced Aviation System Development (CAASD) to assist controllers with detection and resolution of predicted problems. URET combines real-time flight plan and track data (from the ARTCC Host computer) with site adaptation, aircraft performance characteristics, and winds and temperatures (from the National Weather Service) in order to construct four-dimensional flight profiles. It also adapts each trajectory to the observed behavior of aircraft (speed, climb and descent rate) [7]. URET uses its predicted trajectories to continuously check for conflicts up to 20 minutes into the future and displays an alert to the appropriate sector. It also provides a trial plan which allows a controller to check a desired flight plan amendment for potential conflicts before issuing a clearance. [3]

Combined with the innately probabilistic nature of conflict detection and the asymmetrical valuing of the two erroneous outcomes, misses and false alarms —that is, situations that would not develop into an actual conflict, or would resolve naturally in the appropriate time horizon —automated conflict alerting systems frequently produce large numbers of false alarms that again contribute to increased workload of the human controllers overseeing the system. Such probabilistic uncertainty associated with automated decision aids greatly reduces their usability and acceptability in the workplace; conversely, improvements in the reliability (i.e., reduced false alarm rates) of automated conflict detection systems will help to bring about their full potential in increasing the system capacity without undue drawbacks in controller workload and trust. The interaction of the continuous dynamics of the aircraft, used for prediction purposes, with the discrete conflict detection logic gives rise to the hybrid nature of the overall system. The introduction of the probabilistic element, common to decision alerting and aiding

devices, places the conflict detection and resolution problem in the domain of probabilistic hybrid phenomena.

1.2 Hybrid System

Hybrid systems involve both continuous dynamics as well as discrete phenomena. The continuous dynamics of hybrid systems, generally given by differential equations, may be continuous-time, discrete-time, or mixed. The discrete variable dynamics of hybrid systems are generally governed by a digital automaton, or an input-output transition system with a countable number of states. We can assume a hybrid system to a run with a sequence of steps. The system evolves continuously, within each step, until a transition occurs. Transitions are instantaneous state changes that separate continuous state evolutions.

Due to the inherent uncertainty existing in real world problems, stochastic hybrid and switched modeling formalisms [8] are prevalent in the representation of physical systems. Stochastic hybrid systems arise in numerous applications of systems with multiple nodes, such as flexible manufacturing system, air traffic management, etc. This idea is not new and many models of stochastic hybrid systems have been proposed. The major point of distinction lies in the manner in which the randomness or stochasticity is incorporated into the model.

Generally, the data received for conflict detection purposes is noisy, and limited in its scope (i.e. position, speed and heading). Fortunately, aircraft behavior can be characterized by several standard flight modes, such as steady level flight, climbing flight, turning flight, etc. However, due to the noise associated with the flight track data, and the lack of intent information,

it is often difficult to determine whether a given aircraft is adhering to its declared flight plan. In order to estimate the mode of flight, the observed position, speed and heading data must be analyzed, and a probabilistic distribution over the possible modes must be determined. This task lends itself to the framework of Hidden Markov Modeling (HMM), which is used here to detect flight mode changes.

1.3 Hidden Markov Models

The Hidden Markov Model [9] is comprised of a finite set of *states*, each of which is associated with a (generally multidimensional) probability distribution. Transitions among the states are governed by a set of probabilities called *transition probabilities*. In a particular state an outcome or *observation* can be generated, according to the associated probability distribution. It is only the outcome, not the state, that is visible to an external observer: therefore the states of the system are hidden from the observer, hence the name Hidden Markov Model.

Hybrid Input/Output Automata, HIOAs, are used in description and analysis of hybrid systems. We use the HIOA framework developed in Ref. 10.

1.4 Hybrid Input/Output Automata (HIOA)

HIOA is an automaton framework for describing discrete and continuous behavior. The simplest HIOA consists of sets of internal variables, internal

actions, transitions and trajectories. Valuations of the internal variables define the the states of the automaton, which can change continuously over a period of time (trajectories) and discretely through transitions. The behavior of an HIOA is modeled as an alternating sequence of actions and trajectories, referred to as an execution. An HIOA is a nondeterministic automaton which can communicate both discretely (through shared actions) and continuously (through shared variables) with other HIOAs. The external interface of the HIOA is defined by adding sets of input and output variables and input and output actions. The externally visible behavior corresponding to an execution, called a trace, is obtained by removing all the internal variables and actions from the execution.

In HIOAs, uncertainties are captured as nondeterministic choices. Nondeterminism can describe uncertainty as a set of possible choices. Incorporating probabilities in hybrid systems framework yields a richer language for model construction called Probabilistic Timed I/O Automata (PTIOA), which provides a mathematical framework for modeling and verifying computing systems that interact with uncertain systems. PTIOA supports continuous evolution, nondeterminism, probabilistic transitions, and discrete communications between components.

In order to generate the probability distribution over a set of executions of a hybrid system, the issues of non-determinism must be resolved. If there are several enabled actions for a given state in an execution, one must be chosen via a consistent policy. Possible policies include (i) event chosen independent of execution history, or (*oblivious*), (ii) event dependent on current state, or (*Markovian*) (iii) event chosen dependent on execution history, or (*history-dependent*). Extensive work has been done regarding the first policy [11]. We consider the implementation of a Markovian policy in the context of a

partially observable probabilistic hybrid automata framework, which we call a *Hybrid Hidden Markov Model (HHMM)*. This framework arose through the consideration of the distributed air traffic management problem, where individual airplanes (agents) have full information of their own state and model, but only partial information regarding their neighbors' states. We utilize the HHMM formalism to address the problem of self-separation of aircraft along a parametrized route in the included example.

1.5 Decentralized Conflict Avoidance

In the decentralized air traffic system, each aircraft talks to its closest neighbors, i.e., it has the flight data of its neighboring aircraft. Since there are data dropouts, uncertainties in the data, an unknown time-varying number of maneuvering aircraft and unknown identities, the assumption of full observability does not stand. At each time instant, each aircraft detects potential conflicts with its neighboring aircraft and then makes a decision to resolve conflicts. The combination of probabilistic uncertain dynamics and probabilistic observability yields the probability distribution over the state which is called a belief state.

The mathematical definition of the decentralized conflict detection algorithm for an aircraft, represented by an HIOA, which then receives as inputs the position, velocity and heading of all aircraft within sensor range, allows for the aircraft in question to run HHMMs of each aircraft, and to probabilistically predict what mode of the HIOA each sensed aircraft has engaged. This allows for proactive conflict detection and resolution maneuvers that can be enacted by each aircraft, without involving overt communication between aircraft.

1.6 Thesis Outline

In the next chapter, a literature review of the state of the art techniques in stochastic hybrid modeling, probabilistic conflict detection and intent information are outlined. Chapter four provides the theoretical background for Hidden Markov Modeling, and the types of problems best suited for this formalism. The hybrid hidden Markov model (HHMM) is introduced chapter five along with presenting the HIOA and PTIOA frameworks and their properties. A description of the air traffic data and sector area being used is given in chapter six. The model which uses intent information in the HHMM is described here as well. The models derived and their efficacy as well as the comparison between HMM and other approaches are then discussed in this chapter. In chapter seven, a decentralized conflict avoidance algorithm using the HHMM framework and the HIOA language is developed. Conclusions and future work are outlined in chapter eight.

Chapter 2

Survey of the Field

2.1 Probabilistic Conflict Detection Algorithms

In this section we present a brief review of some of the work done in probabilistic conflict detection. Kuchar and Yang [12] present an excellent survey of over 60 different conflict detection and resolution schemes, where these methods are classified according to a taxonomy that includes dimensions of state (vertical, horizontal or both), prediction methods (nominal, worst-case or probabilistic), conflict detection threshold, conflict resolution methods and maneuvers, and conflict management (pairwise or global).

Carpenter and Kuchar [13] have developed a prototype airborne collision alerting logic for aircraft on approach to closely-spaced parallel runways. The alerting decision is based on the estimated probability of a collision (using a series of Monte Carlo simulations), in contrast to using a standard spatial or temporal alerting criterion (Traffic Alert and Collision Avoidance System (TCAS) or Precision Runway Monitoring (PRM)) to avoid large time delays and also unacceptable false alert rates. The authors have developed a dynamic model of aircraft on approach that includes uncertainties in sensor measurement and in the intentions of the aircraft. In this model having knowledge of the relative position, speed, heading and turn rate of parallel

traffic is important to determine whether a situation is hazardous. Here the threatened aircraft follows a normal approach path while the intruder follows either a normal or a blunder approach path. In case of an alert, the climbing-turn maneuver is initiated. Since this is an airborne probabilistic logic, as the authors mention, the pilot may have some difficulties in understanding why alerts occur. This may result in a lack of trust in the system, hence a decrease in its efficiency.

Paielli and Erzberger [14, 15] present a method that accurately estimates the probability of conflict for aircraft pairs in free flight. They make two important assumptions: First, the prediction errors are approximated as normally distributed. Hence, the two error covariances of the aircraft pair can be combined into a single covariance of their relative position. Paielli and Erzberger [14, 15] assign this combined covariance to one of the aircraft, referred to as the stochastic aircraft, while the other aircraft (reference aircraft) is regarded to have no positional uncertainty. Secondly, it is assumed that the planned velocities and prediction errors of both aircraft are constant throughout the encounter or the period of potential conflict. The probability of conflict is then determined as the intersection of the ellipse corresponding to the combined error covariance and the circular conflict zone (5 nmi radius). By projecting the circular conflict zone along a line parallel to the relative velocity, an extended conflict zone is formed. The total probability of the encounter is the intersection of the combined error ellipse and this extended conflict zone. With a coordinate transformation the combined error ellipse transforms into a unit circle, which simplifies the probability computation and leads to an analytical computation of the total probability of conflict, which in turn can be used to find the optimal time to initiate a resolution maneuver. Paielli [16] tests the conflict detection scheme proposed in Ref. 14

using actual flight data, considering only aircraft pairs in level flight over FL290.

Blin et al. [17] propose different aircraft position error models that they tune by reproducing the results obtained in Ref. 14, 15. Their models are based on three considerations: (1) a position error normally distributed with a constant rate that grows linearly with time, (2) a position error resulting from a velocity error modeled as a Brownian process, and (3) a position error resulting from an acceleration error modeled as a Brownian process. According to the authors, a combination of these three models is a possible solution. This new position error model is generic and can be used for different probabilistic methods. In Ref. 18 the authors have developed an environment to quantitatively evaluate and compare three different conflict detection algorithms (geometric, probabilistic and enhanced probabilistic). They use minimum horizontal separation expansion and altitude shift to create conflicts with real air traffic data. Like Ref. 16, the authors develop their prediction error model with simple trajectory prediction, i.e., straight paths at constant speed. It is shown that the modifications to the error model of Ref. 14 proposed in Ref. 17 result in a decrease of the missed detection rate while maintaining the false alarm rate.

Prandini et al. [19] present a decentralized algorithm for CD&R. In this model, randomized optimization is used for estimating the criticality measure, that is, the maximum instantaneous probability of conflict, and formulating a conflict detection algorithm for two aircraft encounters. The advantage of randomized techniques is that they tend to be computationally efficient.

Hwang et al. [20] use the trajectory prediction error model proposed in Ref. 14 to develop a conflict detection algorithm which is based on hybrid

models of aircraft. For flight mode estimates they propose a modified version of the Interacting Multiple Model (IMM) algorithm [21, 22] called Residual Mean IMM. This work is restricted to two dimensions.

2.2 Stochastic Hybrid Systems (SHS)

Due to the inherent uncertainty existing in real world problems, stochastic hybrid and switched modeling formalisms [8, 23–25] are prevalent in the representation of physical systems. Stochastic hybrid systems arise in numerous applications of systems with multiple nodes, such as flexible manufacturing system, air traffic management, etc. This idea is not new and many models of stochastic hybrid systems have been proposed (see Ref. [26] for an overview). The major point of distinction lies in the manner in which the randomness or stochasticity is incorporated into the model.

One choice in modeling stochastic hybrid systems is to replace the deterministic dynamics by a stochastic differential equation. In this case, starting from a fixed initial state, depending on the solutions of the stochastic equations, different guards can be activated. Hence different discrete transitions occur randomly, although the discrete state is deterministic [27].

Another choice is to replace the deterministic transitions between discrete states by random ones governed by some prescribed probabilistic law. Two types of discrete transitions are studied in different models. The first occur at the boundaries of the state space, where continuous evolution becomes impossible. These transitions are called forced transitions. The second class of transitions, known as spontaneous transitions, can take place in the interior of the state space at random times (e.g. using a generalized Poisson process) [24].

Some stochastic hybrid models allow diffusion to model continuous evolution (Switched Diffusion Processes [28, 29]; Stochastic Hybrid Systems [27]) while others do not (e.g. Piecewise Deterministic Markov Processes [30]). Some models only allow forced transitions [27], others allow random time transitions [28], while still others allow both. Next, some stochastic hybrid models are reviewed.

2.2.1 Piecewise Deterministic Markov Processes (PDMP)

Piecewise Deterministic Markov Processes (PDMP) are a model of stochastic hybrid systems in which randomness occurs only in the discrete transitions. The evolution of the continuous state is according to a deterministic nonlinear differential equation, but transitions occur either when the state hits the state space boundary (guards), or according to a generalized Poisson process. After each transition the hybrid system is reset instantaneously according to a probability distribution which depends on the state before the transition [29].

2.2.2 Switched Diffusion Process (SDP)

A controlled switching diffusion is an example of a hybrid system that arises in numerous applications like fault-tolerant control systems, multiple target tracking and flexible manufacturing systems. The continuous state is governed by a stochastic differential equation (controlled diffusion process), while the discrete state is a controlled Markov chain which its transition matrix depends on the continuous state. Also the continuous state jumps at random times. The characteristic feature of SDP is that the continuous state

evolves without jumps, i.e. it can be assumed to be a continuous function of time [24, 28].

2.2.3 Polynomial Stochastic Hybrid Systems (PSHS)

Stochastic Hybrid Systems (SHS) are a class of nonlinear stochastic continuous time hybrid dynamical systems. They are characterized by a hybrid state defined by continuous and discrete states. The continuous state evolves according to a stochastic differential equation that depends on the hybrid state. The discrete dynamics produces transitions in both continuous and discrete states. Transitions are either forced or spontaneous. After each transition, the hybrid state is reset instantaneously to a new value according to a probability law depending on the pre-jump location. This class allows diffusion processes in the continuous evolution, spontaneous discrete transitions, forced transitions, and probabilistic reset of the hybrid state as a result of discrete transitions [30].

PSHS generally correspond to stochastic SHS with polynomial continuous vector fields, reset maps, and transition intensities. For this model both the continuous and the discrete component of the hybrid state are stochastic processes. The evolution of the continuous state is determined by a stochastic differential equation and the discrete state is governed by a transition or reset map. These discrete transitions are triggered by stochastic events (like transitions in PDMP) [25].

Although SHS can model large classes of systems, their formal analysis presents significant challenges, such as having an analytical solution for the partial differential equations that express the evolution of the probability distribution function for the states. But in the PSHS model, an infinite

vector is created that contains the probabilities of all discrete modes and the multi-variable statistical moment of the continuous state. The dynamics of this vector are governed by an infinite-dimensional linear ordinary differential equation. It is shown in Ref. [25] that these linear ODEs can sometimes be approximated by a finite-dimensional nonlinear ODE with relative precision.

2.3 Intent Information

In predicting the potential of future loss of separation between two aircraft it is imperative to know the respective intentions of the aircraft (e.g., will they continue on their present straight-line trajectories, or will one or both change altitudes and/or headings, etc.). Human controllers may infer aircraft intentions from the filed flight plans but also possess large amounts of intent information from their communications with pilots as well as their own planning processes. The latter two sources of information are clearly extremely valuable in this task due to their greater immediacy and because flight plans are frequently changed via voice communications, based on controllers' active planning of traffic flows. Automated systems have access to flight plan information but not to moment-to-moment information exchanges between controllers and pilots via voice radio, and obviously not to controllers' covert planning processes, and therefore are in certain disadvantage compared to human controllers.

Some CD&R algorithms have used intent information to predict the future position of an aircraft. These algorithms may use different types of intent information. Predefined flight plans and ATC clearances are called implicit intent information. Implicit intent is available via a filed flight plan, and/or communications between the ATC controller and the pilot. Infor-

mation exchange between two or more aircraft or between aircraft and ATC through digital data link provides the explicit form of intent. The Flight Management System (FMS) flight plans are explicit intent. FMS route information consists of the position of each waypoint, the estimated speed at each waypoint and the estimated time of arrival at each waypoint.

In Ref. 31 the initial set of explicit intent information, transferred via data link between aircraft, required to perform conflict detection is described. Kuchar and Yang [32] describe some of the fundamental issues that arise in CD&R problems involving intent by developing a generalized model of conflict detection when intent information is available. Their model is composed of a strategic model (economic-based) and a tactical one (safety-based).

In Ref. 33 a method to compute the probability of conflict in the presence of intent information and trajectory uncertainties is described. This approach relies on Monte Carlo simulations using a series of straight-line trajectories.They use a series of Monte Carlo simulations for various trajectory errors and intent information.

Carreno and Munoz [34] present an intent based conflict detection algorithm as well. They use published data to predict aircraft future locations. Their intent based conflict detection algorithm utilizes intent projection when the aircraft is on a nominal path, otherwise it reverts to state projection when the aircraft diverts from its nominal path by more than a predeterminded margin. This algorithm reduces the number of false alarms as opposed to state based conflict detection algorithms.

Zhao et al. [35] propose several classifications of pilot intent models. They specify four groups of intent parameters: motive intent, objective intent, trajectory intent, and cost intent. Motive intent can be used to predict

aircraft trajectories in a short time, in turn objective intent can reduce the number of false alarms when used in tactical CD&R aids.

Conformance monitoring in air traffic control is required to detect deviations of aircraft from their assigned trajectories. This promotes safety, security, and efficiency. In Ref. 36 fault detection techniques are used for conformance monitoring, where non-conforming behavior of an aircraft is considered a fault. Use of this approach and the insight gained through it is demonstrated by simple implementations of the framework with flight-test data.

Krozel and Andrisani [37] present a method for inferring the intent of a pilot in real time. Their method is based on artificial intelligence and links weather state information, special-use airspace region boundaries and other factors to the inferred intent. This technique can be used for conformance monitoring as well but has not been extended to this application as of yet.

Chapter 3

Hidden Markov Models

A hidden Markov model (HMM) has two stochastic components: a finite-state Markov chain and a finite set of output probability distributions. In other words an unobservable stochastic process (hidden) that can only be observed through another set of stochastic processes that generate the sequence of observations.

The basic theory of HMMs was introduced in the late 1960s by Baum and his colleagues [38–41]. In the classic paper of L.R. Rabiner [9] an introduction to the theory of HMMs and their application in speech recognition has been presented. In this work, an example of an HMM is introduced where there is a collection of N urns, each containing a different proportion of colored balls with M possible colors for each ball. The observation sequence occurs by choosing a new urn based on only the previously chosen urn, and then choosing with replacement a ball from this new urn. The ball choices are observable but the sequence of urn choices are hidden.

HMMs are currently being used in automatic speech recognition [42–44], language modeling [45, 46], biological sequence analysis [47], network intrusion detection systems [48, 49] and various other applications.

3.1 Discrete Markov Processes

Consider a system with a set of N states, $S = \{s_1, s_2, \ldots, s_N\}$. At any given time, the system is in one of the states and moves to another state in the next time step according to a set of probabilities associated with the state. If the process is a first order Markov chain, the transition probability does not depend upon which states the chain was in before the current state, i.e.,

$$a_{ij} = P\left\{q_{t+1} = j \,|q_t = i\right\}, \quad 1 \leq i,\ j \leq N$$

The state transition probabilities have the following properties:

$$a_{ij} \geq 0, \quad 1 \leq i, j \leq N$$

$$\sum_{k=1}^{M} a_{ij} = 1, \quad 1 \leq i \leq N$$

Since the output of the process is the set of states at each instant of time and each state corresponds to a physical, i.e., observable, event, this stochastic process can be called an observable Markov model.

3.2 Hidden Markov Model

The Hidden Markov Model [9, 42, 50–53] is comprised of a finite set of *states*, each of which is associated with a (generally multidimensional) probability distribution. Transitions among the states are governed by a set of probabilities called *transition probabilities*. In a particular state an outcome or *observation* can be generated, according to the associated probability distribution. It is only the outcome, not the state, that is visible to an external

observer: therefore the states of the system are hidden from the observer, hence the name Hidden Markov Model.

In order to define the HMM completely, the following five elements are needed:

1) The number of states of the model, N (although the states are hidden).

2) The number of distinct observation symbols per state, i.e, the discrete alphabet size, M. If the observations are continuous then M is infinite.

3) The state transition matrix, $\Lambda = \{a_{ij}\}$, and the set of state transition probabilities,

$$a_{ij} = P\left\{q_{t+1} = j \,|q_t = i\right\}, \quad 1 \leq i,\ j \leq N$$

where q_t denotes the current state. Transition probabilities should satisfy the normal stochastic constraints,

$$a_{ij} \geq 0, \quad 1 \leq i, j \leq N \ and \ \sum_{k=1}^{M} a_{ij} = 1, \quad 1 \leq i \leq N.$$

4) A probability distribution in each of the states, $B = \{b_j(k)\}$ where

$$b_j(k) = P\left\{o_t = v_k \,|q_t = j\right\},\ 1 \leq j \leq N,\ 1 \leq k \leq M\ ,$$

where v_k denotes the k^{th} observation symbol in the alphabet, and o_t is the current observation. The following stochastic constraints must be satisfied.

$$b_j(k) \geq 0,\ 1 \leq j \leq N,\ 1 \leq k \leq M \ and \ \sum_{k=1}^{M} b_j(k) = 1, \quad 1 \leq j \leq N.$$

5)The initial state distribution, $\Pi = \{\pi_i\}$ where,

$$\pi_i = P\{q_1 = i\},\ 1 \leq i \leq N.$$

Therefore we can use the compact notation $\lambda = (\Lambda, B, \Pi)$ to denote an HMM with discrete probability distributions.

3.2.1 The three basic problems for HMMs

There are three basic problems for HMMs that are useful in real-world applications. Given the model parameters, the evaluation problem involves computing the probability of a particular output sequence and is solved using the forward algorithm. The decoding problem entails determining the hidden sequence most likely to have generated a sequence of observations, and is solved by the Viterbi algorithm. In the learning problem, the model parameters that are most likely to have generated a sequence of observations are determined using the Baum-Welch algorithm.

1) Given a model $\lambda = (\Lambda, B, \Pi)$ and the observation sequence $O = o_1 o_2 \ldots o_T$, how do we efficiently compute the probability of the observation sequence, $P\{O|\lambda\}$?

2) Given a model $\lambda = (\Lambda, B, \Pi)$ and the observation sequence $O = o_1 o_2 \ldots o_T$, what is the most likely state sequence in the model, $Q = q_1 q_2 \ldots q_T$ that produces the observations?

3) Given a model $\lambda = (\Lambda, B, \Pi)$ and the observation sequence $O = o_1 o_2 \ldots o_T$, how should we adjust the model parameters in order to maximize $P\{O|\lambda\}$?

3.2.2 The Decoding Problem and the Viterbi Algorithm

The solution to the decoding problem depends upon the way the "most likely state sequence" is defined. One approach is to find the most likely state q_t at time t and to concatenate all such q_t's. However this method does not necessarily give a physically meaningful state sequence. Therefore we use another method, commonly known as the *Viterbi algorithm*, where the entire state sequence with the maximum likelihood is found. In order to facilitate this computation, we define an auxiliary variable, which gives the highest probability that a partial observation sequence and state sequence up to time t can have, given that the current state is i.

$$\delta_t(i) = \max_{q_1 q_2 \cdots q_{t-1}} P\left\{q_1, q_2, \cdots, q_{t-1}, q_t = i, o_1, o_2, \cdots, o_{t-1} \,|\lambda\right\} \tag{3.1}$$

It is easy to observe that the following recursive relationship holds.

$$\delta_{t+1}(i) = b_j(o_{t+1}) \left[\max_{1 \le i \le N} \delta_t(i) a_{ij}\right], \quad 1 \le i \le N,\ 1 \le t \le T-1, \tag{3.2}$$

where, $\delta_1(j) = \pi_j b_j(o_1),\ 1 \le j \le N$.

So, the procedure to find the most likely state sequence starts from the calculation of $\delta_T(j),\ 1 \le j \le N$, and the state j^*, is found where

$$j^* = arg \max_{1 \le j \le N} \delta_T(j). \tag{3.3}$$

The sequence of states is then back-tracked to find the most likely.

This whole algorithm can be interpreted as a search in a graph whose nodes are formed by the states of the HMM in each of the time instants

$t,\ 1 \leq t \leq T$.

3.2.3 The Learning Problem

Generally, the learning problem involves adjusting the HMM parameters, so that the given set of observations (called the *training set*) is represented by the model in the best possible way for the intended application. Thus it is clear that the "quantity" we wish to optimize during the learning process differs depending on the nature of the application. In other words, there may be several *optimization criteria* for learning, out of which a suitable one is selected depending on the application.

There are two main optimization criteria found in the literature; Maximum Likelihood (ML) and Maximum Mutual Information (MMI). The solution to the learning problem under the Maximum Likelihood method is described below.

In ML we try to maximize the probability of a given sequence of observations O^w, belonging to a given class w, given the HMM λ_w of the class w, with respect to the parameters of the model λ_w. This probability is the total likelihood of the observations and can be expressed mathematically as

$$L_{tot} = P\left\{O^w \,|\lambda_w\right\}. \tag{3.4}$$

Since we consider only one class w at a time, we can drop the subscript and superscript w's.

However, there is no known way to analytically solve for the model $\lambda = (\Lambda, B, \pi)$, which maximizes the quantity L_{tot}. Nonetheless, we can

choose model parameters such that it is locally maximized, using an iterative procedure, like the Baum-Welch method, which is described below.

This method can be derived using simple "occurrence counting" arguments, or using calculus to maximize the auxiliary quantity

$$Q(\lambda, \bar{\lambda}) = \sum_{q} P\left\{q \,|O, \lambda\right\} \log\left[P\left\{O, q, \bar{\lambda}\right\}\right]. \tag{3.5}$$

A special feature of this algorithm is that it is guaranteed to converge.

To describe the Baum-Welch algorithm, also known as the Forward-Backward algorithm, we define two auxiliary variables, in terms of the classical forward and backward variables, $\alpha_t(i)$ and $\beta_t(j)$ respectively.

$$\alpha_t(i) = P\left\{o_1, \cdots, o_t, q_t = i \,|\lambda\right\} \tag{3.6}$$

$$\alpha_{t+1}(i) = \sum_{i=1}^{N} \alpha_t(i) a_{ij} b_j(o_{t+1}) \quad 1 \leq j \leq N \ , \ 1 \leq t \leq T-1 \tag{3.7}$$

$$\beta_t(j) = P\left\{o_{t+1}, \cdots, o_T \,|q_t = j, \lambda\right\} \tag{3.8}$$

$$\beta_t(j) = \sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j) \quad 1 \leq i \leq N \ , \ t = T-1, T-2, \cdots \tag{3.9}$$

The first variable is defined as the probability of being in state i at time t and in state j at time $t+1$. Formally,

$$\xi_t(i,j) = P\left\{q_t = i, q_{t+1} = j \,|O, \lambda\right\}. \tag{3.10}$$

This is the same as,

$$\xi_t(i,j) = \frac{P\left\{q_t = i, q_{t+1} = j \,|\lambda\right\}}{P\left\{O \,|\lambda\right\}}. \tag{3.11}$$

Using forward and backward variables this can be expressed as,

$$\xi_t(i,j) = \frac{\alpha_t(i)a_{ij}\beta_{t+1}(j)b_j(o_{t+1})}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)a_{ij}\beta_{t+1}(j)b_j(o_{t+1})}. \tag{3.12}$$

The second variable is the *a posteriori* probability,

$$\gamma_t(i) = P\left\{q_t = i \,|O, \lambda\right\}, \tag{3.13}$$

that is the probability of being in state i at time t, given the observation sequence and the model. In forward and backward variables this can be expressed by,

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N}\alpha_t(i)\beta_t(i)}. \tag{3.14}$$

One can see that the relationship between $\gamma_t(i)$ and $\xi_t(i,j)$ is given by,

$$\gamma_t(i) = \sum_{j=1}^{N}\xi_t(i,j), \quad 1 \le i \le N,\ 1 \le t \le M. \tag{3.15}$$

Now it is possible to describe the Baum-Welch learning process, where parameters of the HMM are updated in such a way as to maximize the quantity, $P\left\{O\,|\lambda\right\}$. Assuming a starting model $\lambda = (\Lambda, B, \pi)$, we recursively calculate and update the HMM parameters according to Eqs. 3.16-3.18, known as re-estimation formulae.

$$\bar{\pi}_i = \gamma_1(i), \quad 1 \le i \le N \tag{3.16}$$

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1}\xi_t(i,j)}{\sum_{t=1}^{T-1}\gamma_t(i)}, \quad 1 \le i \le N,\ 1 \le j \le N \tag{3.17}$$

$$\bar{b}_j(k) = \frac{\sum_{t=1,o_t=v_k}^{T}\gamma_t(j)}{\sum_{t=1}^{T}\gamma_t(i)}, \quad 1 \le j \le N,\ 1 \le k \le M \tag{3.18}$$

The Baum-Welch algorithm is guaranteed to converge, thereby finding a local maxima.

If we now allow the states of the HMM to be represented by continuous, probabilistic functions, the models become Hybrid Hidden Markov Models (HHMM) which are described in the next chapter.

Chapter 4

Hybrid Hidden Markov Models

In the previous chapter, we studied hidden Markov models. If we now allow the states of the HMM to be represented by continuous, probabilistic functions, the models become Hybrid Hidden Markov Models (HHMM). Thus, an HHMM can be used in conjunction with past flight data in a TRACON in order to model the conformance of aircraft to flight paths, or to detect when individual aircraft change their mode of flight (e.g. when an aircraft changes its flight path from steady level flight, to climbing flight etc.).

4.1 Hybrid Automata

A hybrid automaton is a state machine whose states can change by discrete transitions, which result in an instantaneous change of state, or by continuous trajectories, which express the evolution of the state over time intervals.

Definition 4.1.1 *A hybrid automaton, HA, is composed of the seven-tuple:*

$$HA = (W, X, Q, \Theta, E, H, D, T) \tag{4.1}$$

where,

- *The sets W and X are disjoint from each other and correspond to external and internal variables, respectively. The set of variables is*

defined as $V = W \bigcup X$.

- Q *represents the set of states,* $Q \subset val(X)$.
- Θ *is the non-empty set of start states,* $\Theta \subseteq Q$.
- *The set of actions* Ac *consists of a set* E *of external actions and a set* H *of internal actions which are disjoint from each other.*
- *The set* D*, of discrete transitions,* $D \subseteq Q \times Ac \times Q$*, is often denoted in shorthand by* $x \xrightarrow{a} x'$*. If* $x \xrightarrow{a} x'$*, it is said that* a *is enabled in* x.
- T *is the set of trajectories for* V*. For a trajectory* $\tau \in T$*, the first and last (if* τ *is closed) states are denoted by* $\tau.fstate$ *and* $\tau.lstate$ *respectively. The set* T *satisfies the following closure properties.*
  - P_1 *Prefix Closure: For every* $\tau \in T$ *and every prefix* τ' *of* τ*,* $\tau' \in T$.
  - P_2 *Suffix Closure: For every* $\tau \in T$ *and every suffix* τ' *of* τ*,* $\tau' \in T$.
  - P_3 *Concatenation Closure: If* $\tau_0, \tau_1, \cdots \in T$ *is a sequence of trajectories such that* $\tau_i.lstate = \tau_{i+1}.fstate$*, then their concatenation is also in* T.

These closure properties are needed for parallel composition of hybrid automata. In a composed system, any trajectory of any automaton can be interrupted at any time by a discrete transition of another automaton. Prefix closure ensures that the part of trajectory up to the transition is a trajectory, while the remainder being a trajectory is guaranteed by suffix closure. And finally the concatenation closure is required so that the automaton can follow changes in its continuous dynamics.

Note: Each variable (a component of the system's state) v is associated with a static type and a dynamic type. The static type of v, $type(v)$, is the set of values that v can take. $val(V)$ is the set of all valuations of V and a valuation is a function that associates for each $v \in V$ a value in $type(v)$.

4.1.1 Executions and Traces

Execution fragments and traces are used to describe the behavior of the automata. An execution fragment of a hybrid automaton is an (Ac, V)-sequence (action-trajectory sequence)

$$\alpha = \tau_0 a_1 \tau_1 a_2 \tau_2 \cdots \tag{4.2}$$

where

- Each $\tau_i \in T$
- Each $a_i \in Ac$
- If τ_i is not the last trajectory in α, then the last state in τ_i can map, under some action a_{i+1}, to a first state in some trajectory τ_{i+1}; $\tau_i.lstate \overset{a_{i+1}}{\rightarrow} \tau_{i+1}.fstate$.

An execution fragment records all the details of a particular run of a system, including all the discrete state changes and also the changes to the state and external variables as time advances. The set of execution fragments of HA is denoted by $frags_{HA}$. An execution fragment α is called an execution if the first state of α is a start state of the hybrid automaton HA. The set of executions of HA is defined as $execs_{HA}$.A state of the hybrid automaton is reachable if it is the last state of some execution of the automaton. An

invariant property (or an invariant) is a predicate on the state variables that is true in all reachable states.

Traces capture the externally visible behavior of a hybrid automaton. Traces record external actions and the trajectories that describe the evolution of external variables. The trace of an execution fragment α, $trace(\alpha)$, is the (E, W)-restriction of α, i.e., the restriction of α to the external actions and external variables. A trace fragment of a hybrid automaton HA, from a state x of HA, is the trace of an execution fragment of HA whose first state is x. A trace of HA is the trace of an execution of HA. The set of traces of HA is defined as $traces_{HA}$.

Definition 4.1.2 *Hybrid automata HA_1 and HA_2 are said to be comparable if $W_1 = W_2$ and $E_1 = E_2$, i.e., they have the same external interface. Furthermore, if HA_1 and HA_2 are comparable, then HA_1 implements HA_2 if $traces_{HA_1} \subseteq traces_{HA_2}$.*

4.1.2 Composition

Parallel composition is an operation on hybrid automata, which allows an automaton representing a complex system to be constructed by composing automata representing individual system components. The composition operation in Ref. 10 identifies external actions with the same name in different component automata, and likewise for external variables. When any component automaton performs a discrete step which involves an action a, all the other automata that have a in their action sets do the same. Similarly, when any component automaton performs a trajectory which involves a particular evolution of values for an external variable v, then all the other automata that have v in their variable set do the same.

In Ref. 10, composition is defined as a partial, binary operation on hybrid automata. Since internal actions of an automaton HA_1 are intended to be unobservable by any other automaton HA_2, HA_1 is composed with HA_2 only if the internal actions of HA_1 are disjoint from the actions of HA_2. Likewise, disjointness of the internal variables of HA_1 and the variables of HA_2 is required.

Definition 4.1.3 *Hybrid automata HA_1 and HA_2 are said to be compatible if $H_1 \cap A_2 = H_2 \cap A_1 = \emptyset$ and $X_1 \cap V_2 = X_2 \cap V_1 = \emptyset$.*

Definition 4.1.4 *If HA_1 and HA_2 are compatible, then their composition $HA_1 \| HA_2$ is defined to be the structure $HA = (W, X, Q, \Theta, E, H, D, T)$ where:*

- $W = W_1 \cup W_2$ *and* $X = X_1 \cup X_2$.
- $Q = \{x \in val(X) \,|(x restrictedto X_1) \in Q_1 \wedge (x restrictedto X_2) \in Q_2\}$.
- $\Theta = \{x \in val(X) \,|(x restrictedto X_1) \in \Theta_1 \wedge (x restrictedto X_2) \in \Theta_2\}$.
- $E = E_1 \cup E_2$ *and* $H = H_1 \cup H_2$.
- *For each* $x,\ x' \in Q$ *and each* $a \in A$, $x \xrightarrow{a_A} x'$ *iff for* $i = 1, 2$, *either*
  - $a \in A_i$ *and* $x restrictedto X_i \xrightarrow{a_i} x' restrictedto X_i$ *or*
  - $a \notin A_i$ *and* $x restrictedto X_i = x' restrictedto X_i$.
- $T \subseteq trajs(V)$ *is given by* $\tau \in T \Leftrightarrow \tau restrictedto V_1 \in T_1 \wedge \tau restrictedto V_2 \in T_2$.

.

The next theorem states that the class of hybrid automata is closed under composition.

Theorem 4.1.5 *If HA_1 and HA_2 are hybrid automata, then $HA_1 \| HA_2$ is a hybrid automaton.*

The proof is found in Ref. 10. A projection lemma derived from this theorem states that the executions of a composition of hybrid automata project to give the executions of the component automata.

4.2 Hybrid Input/Output Automata (HIOA)

HIOAs are used in description and analysis of hybrid systems. This model is based on the concept of infinite state machines whose states can change by discrete actions or by continuous trajectories [10]. The HIOA model is a refinement of the hybrid automaton where external actions and variables consist of input and output actions and variables.

Definition 4.2.1 *A hybrid input/output automaton $\mathcal{A} = (HA, U, Y, I, O)$ consists of*

- *A hybrid automaton HA (4.1).*
- *A set U of input variables and a set Y of output variables, $W = U \bigcup Y$. As before, $V = W \bigcup X$. The set $Z = X \bigcup Y$ is called the set of locally controlled variables.*

- *A set I of input actions and a set O of output actions, $E = I \bigcup O$. As before $Ac = E \bigcup H$. The actions in the set $L = H \bigcup O$ are called locally controlled actions.*

In Addition $\mathcal{A}$ satisfies:

- E_1 Input Action Enabling: For every $x \in Q$ and $a \in I$, there exists $x' \in Q$ such that $x \xrightarrow{a} x'$.
- E_2 Input Trajectory Enabling: $\mathcal{A}$ should be able to accept any trajectory for the input variables.

Input trajectory enabling states that an HIOA should be able to accept any trajectory, i.e., any trajectory for the input variables, either by letting time advance for the entire duration of the input trajectory, or by reacting with a locally controlled action after some part of the input trajectory has occurred.

An automaton with distinguished inputs and outputs that does not necessarily satisfy the axioms E_1 and E_2 is called a pre-HIOA.

An execution of a pre-HIOA $\mathcal{A}$ is defined to be an execution of HA, and a trace of $\mathcal{A}$ is defined to be a trace of HA. Similarly the execution fragments and trace fragments of a pre-HIOA are the execution fragments and trace fragments of the corresponding hybrid automaton.

A pair of HIOAs are comparable if they have the same external interface.

Definition 4.2.2 *Two pre-HIOAs $\mathcal{A}_1$ and $\mathcal{A}_2$ are comparable if $I_1 = I_2$, $O_1 = O_2$, $U_1 = U_2$, and $Y_1 = Y_2$. If $\mathcal{A}_1$ and $\mathcal{A}_2$ are comparable then $\mathcal{A}_1$ implements $\mathcal{A}_2$.*

Since internal actions of an HIOA $\mathcal{A}_1$ are intended to be unobservable by any other HIOA $\mathcal{A}_2$, $\mathcal{A}_1$ is only allowed to be composed with $\mathcal{A}_2$ if the set of internal actions and variables of $\mathcal{A}_1$ are disjoint from the sets of actions and variables, respectively, of $\mathcal{A}_2$. Similarly the disjointness of the sets of output actions of $\mathcal{A}_\infty$ and $\mathcal{A}_\in$ and disjointness of the sets of output variables of $\mathcal{A}_1$ and $\mathcal{A}_2$ is required.

Definition 4.2.3 *Pre-HIOAs $\mathcal{A}_1$ and $\mathcal{A}_2$ are compatible if $H_1 \bigcap Ac_2 = H_2 \bigcap Ac_1 = \emptyset$, $X_1 \bigcap V_2 = X_2 \bigcap V_1 = \emptyset$ and $Y_1 \bigcap Y_2 = O_1 \bigcap O_2$. If $\mathcal{A}_1$ and $\mathcal{A}_2$ are compatible pre-HIOAs, then their composition, $\mathcal{A}_1 \| \mathcal{A}_2$, is defined to be the structure $\mathcal{A} = (X, Y, U, Q, \Theta, H, O, I, D, T)$ where*

- $X = X_1 \bigcup X_2$ *and* $Y = Y_1 \bigcup Y_2$
- $Q = \{x \in val(X) \,|x\lceil X_1 \in Q_1 \wedge x\lceil X_2 \in Q_2\}$
- $\Theta = \{x \in Q \,|x\lceil X_1 \in \Theta_1 \wedge x\lceil X_2 \in \Theta_2\}$
- $H = H_1 \bigcup H_2$ *and* $U = (U_1 \bigcup U_2) - Y$
- $O = O_1 \bigcup O_2$ *and* $I = (I_1 \bigcup I_2) - O$
- *For each* $x, x' \in Q$ *and each* $z \in Ac$, $x \xrightarrow{a}_{Ac} x'$ *iff for* $i = 1, 2$, *either (1)* $a \in Ac_i$ *and* $x\lceil X_i \xrightarrow{a}_i x'\lceil X_i$, *or (2)* $a \notin A_i$ *and* $x\lceil X_i$.
- $T \subseteq trajs(V)$ *is given by* $\tau \in T \Leftrightarrow \tau \downarrow V_1 \in T_1 \wedge \tau \downarrow V_2 \in T_2$.

In general, the composition of two HIOAs $\mathcal{A}_1$ and $\mathcal{A}_2$ is not necessarily an HIOA. This happens when the input variables of $\mathcal{A}_1$ are output variables of $\mathcal{A}_2$ and vice-versa, and the trajectories of these variables defined by the automata are inconsistent. Hence, the composed $\mathcal{A}_1 \| \mathcal{A}_2$ does not satisfy the input trajectory enabling axiom. Therefore, we have the following weaker theorem [10].

Theorem 4.2.4 *If $\mathcal{A}_1$ and $\mathcal{A}_2$ are compatible pre-HIOAs then $\mathcal{A}_1 \| \mathcal{A}_2$ is a pre-HIOA.*

In the next section we describe the Probabilistic Timed I/O Automata framework which supports continuous evolution of states through trajectories, nondeterministic or probabilistic discrete state transitions (which can be based on continuous distributions), and discrete communications between components.

4.3 Probabilistic Timed Input/Output Automata (PTIOA)

A Probabilistic Timed Input/Output Automaton $\mathcal{A} = (X, (Q, \mathcal{F}_Q), \bar{x}, A, D, \mathcal{T})$ consists of [11]:

- A set X of internal or state variables.
- A set $Q \subset val(X)$ of states, a measurable space $(Q, \mathcal{F}_Q)$ called the state space and the start state $\bar{x} \in Q$.
- Disjoint sets H, I, and O of internal, input and output actions and the countable set of actions $A = H \cup I \cup O$ ($L = O \cup H$ is the set of local actions and $E = O \cup I$ is the set of external actions).
- A set $D \subseteq Q \times A \times \mathcal{P}(Q, \mathcal{F}_Q)$ of probabilistic transitions.
- A deterministic set T of trajectories for Q which is closed under prefix, suffix, and concatenation.

This closure property is needed for parallel composition of PTIOA, where any trajectory of any automaton can be interrupted at any time by a probabilistic transition of another automaton. Prefix closure ensures that the part of the trajectory up to the transition is a trajectory, while the remainder being a trajectory is guaranteed by suffix closure. Concatenation closure ensures that the automaton can follow changes in its continuous dynamics.

It should be noted that unlike HIOAs, PTIOAs do not have external variables ($W = U \cup Y$) and it is assumed that they communicate only through shared actions.

Execution fragments and traces are used to describe the behavior of the automata. An execution fragment is an alternating sequence of actions and trajectories which shows all the changes to the state and external variables (in the case of the HIOA) and all discrete state changes while time advances. Traces capture the externally visible behavior of the hybrid automaton and for a PTIOA record the external actions and duration of the intervening time intervals. Hence, the trace of α, $trace(\alpha)$, is the $(E, \emptyset)$-restriction of α.

In the PTIOA framework there are probability measures over the set of executions and traces which in the HHM framework translates into the probability distribution in each of the states.

Definition 4.3.1 *[11] Pre-PTIOAs $\mathcal{A}_1$ and $\mathcal{A}_2$ are compatible if $X_1 \bigcap X_2 = H_1 \bigcap A_2 = H_2 \bigcap A_1 = O_1 \bigcap O_2 = \emptyset$. If $\mathcal{A}_1$ and $\mathcal{A}_2$ are compatible pre-PTIOAs, then their composition, $\mathcal{A}_1 \| \mathcal{A}_2$, is defined to be the structure $\mathcal{A} = (X, (Q, \mathcal{F}_Q), \bar{x}, A, D, T)$ where*

- $X = X_1 \bigcup X_2$ *and* $(Q, \mathcal{F}_Q) = (Q_1 \times Q_2, \mathcal{F}_{Q_1} \otimes \mathcal{F}_{Q_1})$, *and* $\bar{x} = (\bar{x}_1, \bar{x}_2)$
- $A = A_1 \bigcup A_2$, $O = O_1 \bigcup O_2$, $I = (I_1 \bigcup I_2) - O$ *and* $H = H_1 \bigcup H_2$

- $D \subseteq Q \times A \times P(Q, \mathcal{F}_Q)$ *is the set of* $((x_1, x_2), a, \mu_1 \otimes \mu_2)$ *such that for* $i \in \{1, 2\}$ *if* $a \in A_i$ *then* $(x_i, a, \mu_i) \in D_i$*, otherwise* $\mu_i = \delta_{x_i}$

- $T \subseteq trajs(V)$ *is given by* $\tau \in T \Leftrightarrow \tau \downarrow X_1 \in T_1 \wedge \tau \downarrow X_2 \in T_2$.

Theorem 4.3.2 *If* $\mathcal{A}_1$ *and* $\mathcal{A}_2$ *are pre-PTIOAs then* $\mathcal{A}_1 \| \mathcal{A}_2$ *is a pre-PTIOA.*

4.4 Partially Observable Markov Decision Processes

A Markov decision process (MDP) is a framework for sequential decision making in a stochastic environment, which can be used to model an agent's synchronous interaction with an environment. In this framework, it is assumed that there is no uncertainty about the agent's current state. MDPs have been applied in economics, operations research, control systems design and artificial intelligence.

A partially observable Markov decision process (POMDP) is an extension of an MDP, where the state of the system is not fully observable. POMDPs have been applied in robotics, networked control systems [54], dialogue management [55] and hazard avoidance alerting systems [56].

Definition 4.4.1 *A POMDP is a tuple* (S, A, T, R, Z, O) *where [57]*

- S *is a finite set of states.*

- A *is a finite set of actions.*

- $T : S \times A \rightarrow \Pi(S)$ *is the state transition function, where* $T(s, a, s^{'}) = P\left\{s_{t+1} = s^{'} \,|s_t = s, a_t = a\right\}\ \forall\ s, s^{'} \in S,\ a \in A$ *and* Π *is the probability distribution.*

- $R : S \times A \rightarrow \Re$ *is the reward function, that gives the expected immediate reward gained by taking each action in each state.*
- Z *is a finite set of observations.*
- $O : S \times A \rightarrow \Pi(Z)$ *is the observation function, where* $O(s, a, z) = P\{z_t = z \,|s_t = s, a_{t-1} = a\} \ \forall z \in Z,\ s \in S,\ a \in A.$

There are two standard approaches to solving POMDP problems [54]. In the first approach the control action at a given time depends explicitly on the complete history of observations which is called the information state and grows without bound as time progresses. On the other hand, the combination of probabilistic uncertain dynamics and probabilistic observability yields the probability distribution over the state which is called a belief state. In the second approach, the belief state is viewed as another state variable with its own state space (belief space).

4.5 Hybrid Hidden Markov Model

In this section, we introduce Hybrid Hidden Markov Models (HHMM) by mapping the PTIOA framework into hidden Markov models. We utilize this technique to preserve the Markov property for state transitions, since an aircraft trajectory can be subdivided into distinct segments, each corresponding to a flight mode, and the switching between flight modes is a finite-state Markov process. From a given state x of a PTIOA, multiple actions may be enabled, and one of these actions is chosen nondeterministically, which in turn uniquely determines the probabilistic transition at x, and the next state according to the probability distribution. To resolve the nondeterminism, a

scheduler is used which can be independent of the history of the execution (oblivious), history dependent or Markovian.

Conceptually, envision a hidden Markov model that allows for an infinite number of continuous states (which evolve according to continuous differential equations) that can be reduced to a discrete number of abstractions which characterize the modes of the HMM. Using the terminology of the POMDP and PTIOA, we formally define the Hybrid Hidden Markov Model as follows.

Definition 4.5.1 *An HHMM is a tuple* (N, M, H, S, B, Π) *where:*

1. N *is the set of states;* $N = Q \subset val(X)$*, where* X *is a set of internal or state variables.*

2. $M = E \cup W$ *is the union of observable actions defined by the set of external actions (PTIOA),* $E = I \cup O$ *(*I *and* O *are the set of input and output actions, respectively), and the observable part of the state (POMDP),* W*.*

3. H *is the set of internal actions, and* $A = H \cup I \cup O$*.*

4. S *is the state transition function on the measurable state space* $(Q, \mathcal{F}_Q)$ *given by*

$$s_{t,t+1} = P\left\{q_{t+1} \,|q_t, a_t\right\},\ q_t \in val(X(t)),\ a_t \in A \text{ and } S = \left\{s_{t,t+1}\right\}. \tag{4.3}$$

5. B *is the distribution on the observable execution trace* $T_{obs} = m_t m_{t-1} \cdots m_{t=0}$ *(where* $m_t = val(M(t))$ *such that*

$$b_t(k) = P\left\{m_t = val(E \cup W(t)) \,|q_t = val(X(t))\right\},\ k \in M\ , \tag{4.4}$$

6. $\Pi = \{\pi_i\}$ *is the initial distribution of the state given the initial set of start states* Θ

$$\pi_i = P\{q_0 = \theta_i,\ \theta_i \in \Theta\}. \tag{4.5}$$

Note that the state transition function employs a Markovian selection process.

If we then consider a simple pictorial description (figure 4.1) of an aircraft which alternates between the modes of steady level flight (represented by the dynamics contained in the circle on the left) and climbing/descending flight (represented by the dynamics contained in the circle on the right), and the non-deterministic transition functions described upon the arrows denoting the random distribution that determines when the system changes modes, given that the guarding conditions upon the arrow is satisfied (since each epsilon denotes a small value), one can see the characteristics of a hybrid hidden Markov model, whose probability distribution on the observable execution trace is unknown.

Thus, given the HHMM λ and a sequence of observations T_{obs}, we can determine the probability that the observations are generated by the model. Furthermore, given the model λ and a sequence of observations T_{obs}, we can determine the most likely state sequence in the model that produces the observations using a continuous adaptation of the Viterbi Algorithm. Finally, given a model λ and a sequence of observations T_{obs}, we can determine how we should adjust the model parameters $\lambda = (\Lambda, B, \Pi)$ in order to maximize $P\{T_{obs}\,|\lambda\}$ using a continuous adaptation of the Baum-Welch Algorithm. An HHMM can be used in conjunction with actual flight data from a TRACON, in order to model the conformance of aircraft to flight paths, or to detect when individual aircraft change their mode of flight (e.g. when an aircraft

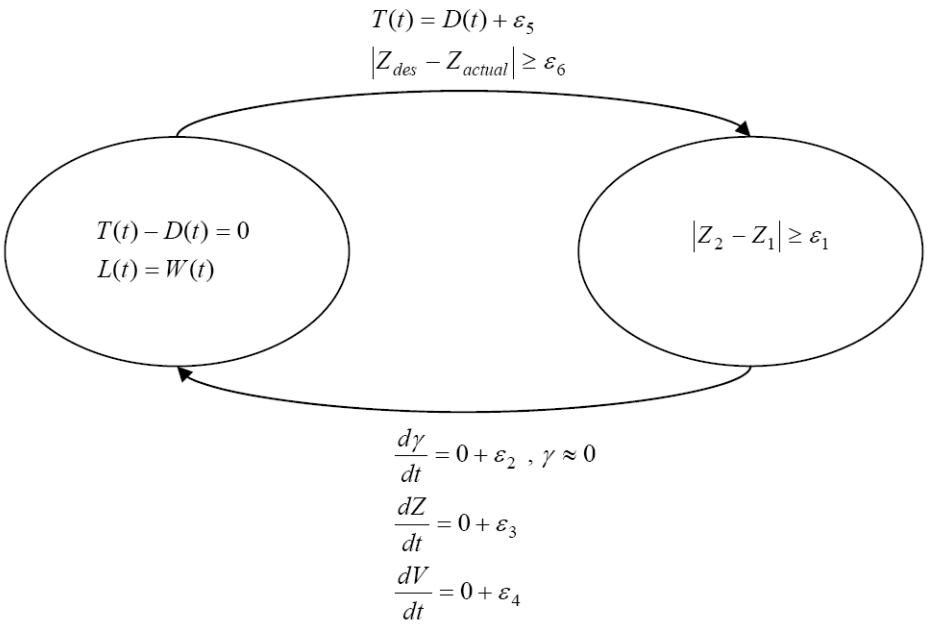


Figure 4.1: Theoretical hybrid hidden Markov model for aircraft in steady level flight or climbing/descending flight

changes its flight path from steady level flight, to climbing flight etc.).

4.6 Advantages of HHMM to Other Formalisms

Using HHMMs allows us to look at the aircraft mode identification problem as a hybrid system with continuous variables (trajectories) instead of a discrete system. This model is most representative of the actual physical evolution of an aircraft in flight, as it behaves in a continuous manner. When the aircraft trajectories are modeled as continuous, we have an infinite number of states; in order to study them we must use a finite abstraction of the states, which is enabled by the notion of levels of abstraction (or equivalence classes) in the HIOA framework. That is why we can use abstract flight modes, such as steady level flight or accelerating turn, to represent the physical dynamics of

the aircraft. Also, to extend the work into decentralized CD&R (that is, the notion that each aircraft has an HHMM onboard and gets information from its neighboring aircraft), we need these models to be composable, a property that is guaranteed by using the HIOA framework. The receptiveness property of the HIOA does not allow the state machine to block time or to contribute to producing Zeno behavior (infinite number of discrete transitions in a finite time).

Modeling the aircraft dynamics as continuous coupled differential equations allows us to detect mode changes in a more timely fashion. That is, we do not have to wait for the next discrete time interval to update our estimate of the mode of the system. As inputs arrive asynchronously to the HIOA, the boundary between two modes of flight requires that the final valuations of all continuous variables in the prior mode become the initial values in the subsequent mode (unless they are overwritten by the triggering input). The inherent inertia of an HHMM is sufficiently less than that of its equivalent HMM (which is essentially a predictive altitude threshold model). That is, the HHMM is better able to model, then distinguish, between changes attributed to noise and/or disturbances, as opposed to actual changes in commanded flight paths. This is because the HHMM formalism allows for the incorporation of continuous, stochastic elements, such as random walks (wind disturbances) and Gaussian noise.

Chapter 5

Mode Detection Results

5.1 Aircraft Flight Data

We have used two different sets of real flight data, NDMS data and FEWS data, in the evaluation of the HHMM framework.

The FAA System Analysis Recording (SAR) stores all flight and radar information in Air Route Traffic Control Centers (ARTCCs). The SAR data are reduced into reports generated by two computer programs, the Data Analysis Reduction Tool (DART) and National Track Analysis Program (NTAP). A National Airspace System (NAS) Host computer can generate these reports. The NAS Data Management System (NDMS) was developed to organize the text-based NTAP reports into hour-long Microsoft Access database files [58]. Traffic samples used in this paper are from the Indianapolis ARTCC (ZID) for both busy and slow times of day. Each file of the database corresponds to a single aircraft with the aircraft identifier as its name. The data in the files are recorded approximately every six seconds and include

- Digital time (ZULU) in hhmmss format
- Position of the aircraft and its altitude
- Ground speed

- Heading angle
- Assigned altitude
- Controlling sector number

The second set of data was obtained from the FAA's evaluation simulation of Future En route WorkStation (FEWS). The simulations took place during the summer of 2005 at the Research and Development Human Factors Laboratory (RDHFL) at the FAA's Technical Center at Atlantic City International Airport, NJ. FEWS data includes more information for each aircraft, and the data are recorded approximately every twelve seconds. The additional data that we used for our HHMM at each time instant consists of: true airspeed (kts), indicated airspeed (kts), flight path angle (deg), total weight of the aircraft (lb), thrust (lb), drag (lb), lift coefficient, desired alt (ft), desired heading angle (deg) and desired indicated speed (kts).

One of the difficulties in predicting aircraft positions is modeling the uncertainties. Wind and errors in tracking, navigation, and control affect the future motion of an aircraft. These uncertainties can be modeled as the sum of a large number of independent random perturbations acting in disjoint time intervals and, thus, it is expected to be Gaussian [14, 19].

The fidelity of the data does not allow for the use of comprehensive stochastic models (for wind and/or radar errors). To the best of our knowledge, this is a common occurence in treatise dealing with actual flight data [16, 59] and is a subject of future work.

The simple linear model used to predict motion along cross-track and

along-track axes for the NDMS data is given below.

$$\begin{aligned} \dot{x}(t) &= v(t)\sin(\pi - \psi(t)) \\ \dot{y}(t) &= v(t)\cos(\pi - \psi(t)), \end{aligned} \tag{5.1}$$

where x is the position of the aircraft on the along-track axis, y is the position of the aircraft on the cross-track axis, $v(t)$ is the ground speed of the aircraft at time t, and $\psi(t)$ is the heading angle of the aircraft at time t.

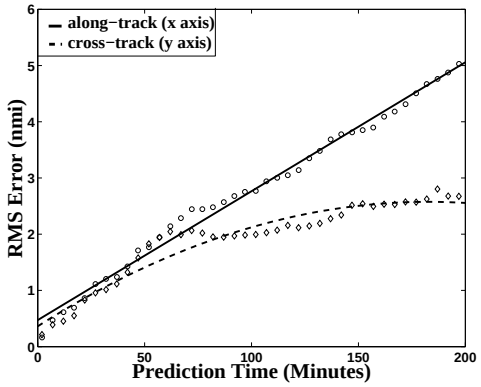


Figure 5.1: Position prediction error statistics

To calculate the uncertainty due to Eq. (5.1), the prediction error statistics are computed by using the flight data of 30 aircraft in steady level flight. A prediction time range of 20 minutes (typical look ahead time for mid-range conflict detection algorithms [19]) is divided into 200 six-second intervals, and for each interval the mean, standard deviation, and root mean square (rms) position prediction errors are computed. The rms errors are shown in figure 5.1. These values can be taken as a bound on uncertainty for the linear prediction model (Eq. (5.1)), using the aircraft data in order to estimate future conflicts, and agree with the literature [16]. The standard deviation errors of the along-track and cross-track axes start at 0.154 nmi and 0.19 nmi, respectively, for zero prediction time.

5.2 The Use of Intent Information

To model the flight path of aircraft by automata (finite state machines) two approaches can be taken. In one model each flight mode of aircraft is one state of the automaton and in the other there are only two states; one corresponding to the horizontal flight modes (the flight modes that do not involve any altitude changes, e.g., steady level flight, level turn, speed up/down and etc.) and the other state representing the vertical flight modes (the flight modes that necessarily include altitude changes, e.g., climb, descend, accelerated climb, and etc.). In this paper the second model is used. Figure 5.2 shows this hybrid input/output automaton (HIOA).

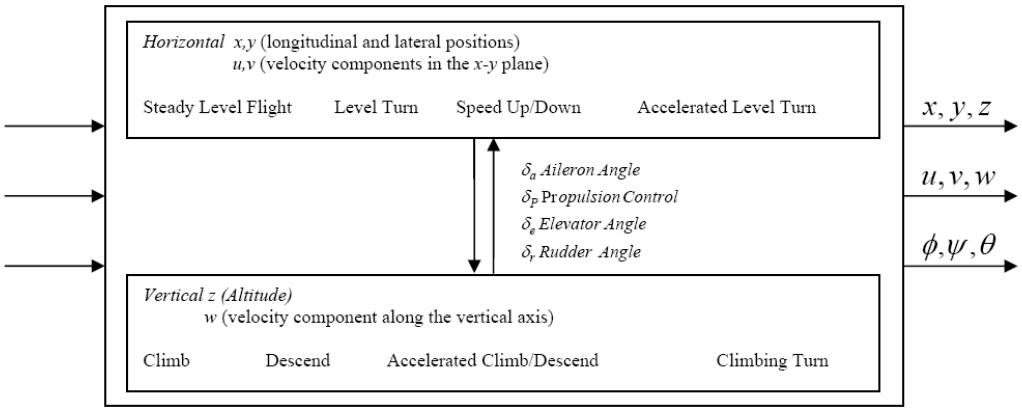


Figure 5.2: Aircraft hybrid input/output automata

Explicit intent information can be used in two different ways in the HHMM described before. First it can be used for better mode identification. The predefined flight plans (position of waypoints, etc.) can be interpreted as flight modes at each time instant. Then the HHMM can use this extra information to identify the actual modes faster and more accurately. Secondly intent can be used to determine whether an aircraft is conforming to its flight plan or not. In this paper intent is used for conformance monitoring.

We denote the radar measurements (track files) by $x,\ y,\ z,\ \theta,\ \phi,\ \psi,\ v$ which are respectively, positions of aircraft along the longitudinal and lateral axes, the altitude, pitch (attitude), roll (bank angle), yaw (heading), and speed. Then by using the HHMM we attempt to find the probability of each sequence of states and choose the path that has the maximum probability. We call this path $\hat{S}$.

Explicit intent is known via published flight plans and communications between the controller and the pilot. The published flight plans (which we call offline intent information) give us the exact sequence of states and the path of aircraft. We call this sequence S_0 and the positions $x_0,\ Y_0,\ Z_0$. We assume that the communications between the controller and the pilot (shared information in controller/pilot interactions [60]) only include controller's commands and we call them online intent information. These commands should allow us to get the deviation between exact offline intent and the HHMM estimated sequence.

Intent is a continuous variable and HHMMs can be used to carry continuous intent. By using intent information along with the Viterbi algorithm in the hybrid hidden Markov model we can estimate likely mode transitions which lead to a better mode identification. We will also be able to determine whether the aircraft are conforming to their predefined flight path or ATC controller's commands. Based on the HIOA shown above, we can determine conformance for vertical and horizontal states separately.

5.3 Mode Detection Results and Discussion

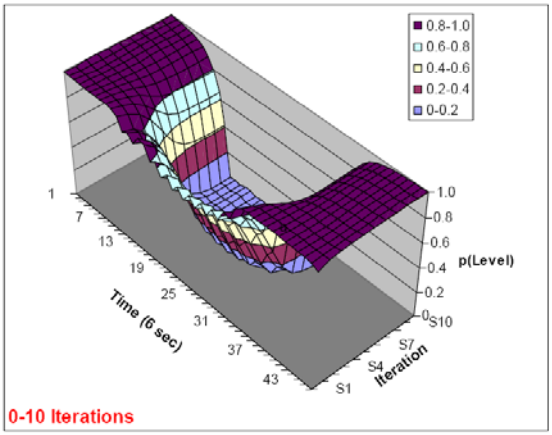
The actual aircraft flight data, described in section 5.1 is used in conjunction with the HHMM described in the previous chapter, to perform conformance

monitoring by detecting mode changes in the flight of individual aircraft. If this model is coupled with the flight plan filed by the aircraft (intent information), then the degree of conformance the aircraft exhibits with respect to its intent can be described probabilistically, by solving the decoding problem. Alternatively, the flight modes of the aircraft can be described in a stochastic hybrid setting, and the flight data taken can then be used to train the HHMM in order to detect transitions between classical modes of flight (i.e., climbing, climbing turn, level flight, speed-up etc.). This produces an HHMM that can then be used to give a probabilistic evaluation of the flight mode of an aircraft after only a few data cycles, in a real-time setting. This latter task is the focus of this paper, and in the sections below, we describe the parameters of the trained HMM and HHMM for the flight modes of steady level flight, climbing/descending flight, turning flight, and climbing/descending turning flight. The accuracy of the detection of the probabilistic switching function between these four modes of flight is then discussed.

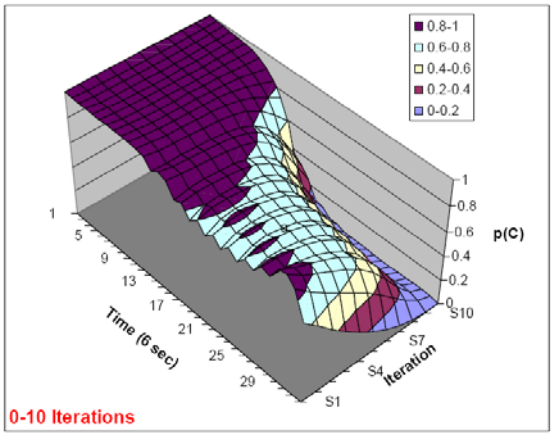
5.3.1 Parameter Adjustment for Mode Detection using NDMS Data

In Ref. 61 the altitude data of 30 aircraft, flying initially above FL 260, over a period of 25 to 65 measurement cycles, were analyzed using the Baum-Welch algorithm in order to refine the parameters of an HMM for transition detection between flight modes. These traffic samples were from the Indianapolis ARTCC (ZID) flying in DAY sector during the busy time of day, 21:00-22:00. Hence, deviation due to aircraft type was not accounted for in the model, as it is assumed that the class of planes capable of flight at these altitudes is relatively uniform in terms of flight characteristics. Three abstract observation

classes, comprised of level flight, climbing flight, and descending flight, were used for the mode detection model, with satisfactory preliminary results. This simplified model and its results are used as a benchmark throughout this section.



(a) Climbing flight (FL 260-290)



(b) Descending flight (FL 290-260)

Figure 5.3: Probability of Level Flight Mode during 10 Training Iterations

In Ref. 62 we divide the aircraft into the following modes: aircraft in steady level flight, aircraft in descending or climbing flight, and aircraft in turning flight (using the same aircraft data as in Ref. 61). To capture the turn mode, the altitude data is insufficient. Hence, the position of the aircraft on the along-track and cross-track axes, in addition to the altitude, is used

to distinguish between an aircraft in steady level flight and an aircraft which is turning in the $x - y$ plane. The heading angle of the aircraft is used as a refining parameter to improve this metric.

Two aircraft, one in climbing flight and another in descending flight, are used as illustrative examples to demonstrate the trained HMM's ability to detect mode changes, and thus determine the conformance of aircraft. Figure 5.3 shows the convergence of this model during 10 training iterations of the Baum-Welch algorithm. In this figure, P(Level) corresponds to the value of the probability of being in level flight at each time step. Since the data used in this paper was recorded at six second intervals, each time step in this figure stands for six seconds, so the tic-mark corresponding to 7 on the time axis refers to a time passage of 42 seconds. Here the probability of being in level flight is plotted for 10 consecutive iterations, where at each iteration the model processes all the data and adjusts the parameters. The climbing flight case used 45 measurements (4 min 30 sec) of data, and the descending flight case used 39 measurements (3 min 48 sec) of data. The aircraft in figure 5.3(a) starts climbing from FL260 to FL290 at the 11th time instant and levels off when it reaches FL290 at time 186 seconds. This is clearly illustrated by the pronounced "U" shape in the graph at the 10^{th} iteration. But the aircraft shown in figure 5.3(b) is at first in steady level flight at FL290 and then starts descending to FL260 at the 11th time instant and levels off at FL260 at the 31st time step.

The corresponding re-estimated transition and emission probabilities for both the above scenarios after 10 iterations are given in Table 5.1. For graphical display purposes, we assign a structure using the label "C" to correspond to the mode of steady, level flight, and the character "H" corresponds to the mode of non-level (climbing, descending or turning) flight. For display pur-

Table 5.1: Final probabilities for HHMM trained on data for climbing and descending flight after 10 iterations

Climbing	$p(\cdots\|C)$	$p(\cdots\|H)$	$p(\cdots\|START)$
$p(1\,\|\cdots)$	0.00	0.44	
$p(2\,\|\cdots)$	1.00	0.56	
$p(3\,\|\cdots)$	0.00	0.00	
$p(C\,\|\cdots)$	0.90	0.05	1.00
$p(H\,\|\cdots)$	0.05	0.95	0.00
$p(STOP\,\|\cdots)$	0.05	0.00	0.00
Descending	$p(\cdots\|C)$	$p(\cdots\|H)$	$p(\cdots\|START)$
$p(1\,\|\cdots)$	0.00	0.53	
$p(2\,\|\cdots)$	0.84	0.47	
$p(3\,\|\cdots)$	0.16	0.00	
$p(C\,\|\cdots)$	0.88	0.05	1.00
$p(H\,\|\cdots)$	0.06	0.95	0.00
$p(STOP\,\|\cdots)$	0.06	0.00	0.00

poses, the three classes of observations for predicted flight over a 6 second measurement interval are denoted by: Class 1, climbing/descending flight, Class 2, steady level flight, and Class 3, turning flight. The observation alphabet used is thus $\{1, 2, 3\}$. Note that in the full HMM the distinction is made between climbing and descending flight (as is done in Ref. 61), and between a climbing turn and a descending turn, leading to an observation alphabet of $\{1, 2, 3, 4, 5, 6\}$. This however is not conductive to concise graphical or lexical display, and thus the above abstraction is used for visual reasons and for notational simplicity. As can be seen from Table 5.1, the probabilities for the mode transitions (i.e. $P(C\,|H\,)$ and $P(H\,|C\,)$) are almost identical for both cases.

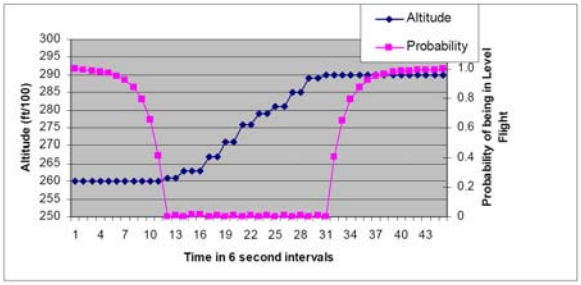
In Table 5.1, $p(2\,|C)$ (or $p(2\,|H)$ is the probability of being in steady level flight if the aircraft was in a level (or non-level) flight mode at the previous time instant. Hence,

$$p(1\,|C) + p(2\,|C) + p(3\,|C) = p(1\,|H) + p(2\,|H) + p(3\,|H) = 1 \qquad (5.2)$$

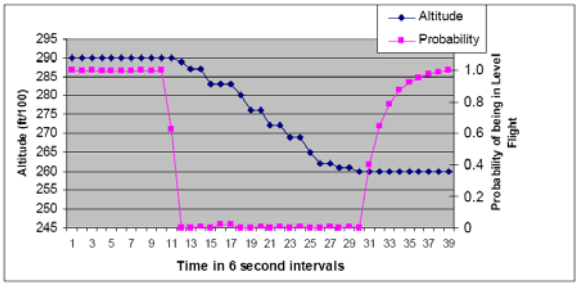
Similarly $P(H\,|C\,)$ is the probability of being in a non-level flight mode if the aircraft was in a level mode at the previous time step.

$P(STOP\,|\cdots)$ and $p(\cdots|START)$ depend on the data set that the HHM is being trained on. In the case of the climbing aircraft the probability of being in a level flight mode at the beginning of the data set, $p(C\,|START)$, is one. For the descending aircraft the probability of reaching the end of the data while being in a non-level flight mode, $p(STOP\,|H)$, is zero. Hence,

$$p(C\,|C) + p(H\,|C) + p(STOP\,|C) = p(C\,|H) + p(H\,|H) + p(STOP\,|H) = 1 \tag{5.3}$$



(a) Climbing flight (FL 260-290)



(b) Descending flight (FL 290-260)

Figure 5.4: Detection Threshold

Figure 5.4 illustrates the fact that the model's ability to detect mode transitions is very accurate, and usually occurs within two data cycles (within approximately 12 seconds). The aircraft used for the climbing flight profile was climbing parallel to its along-track axis, and the y values were almost

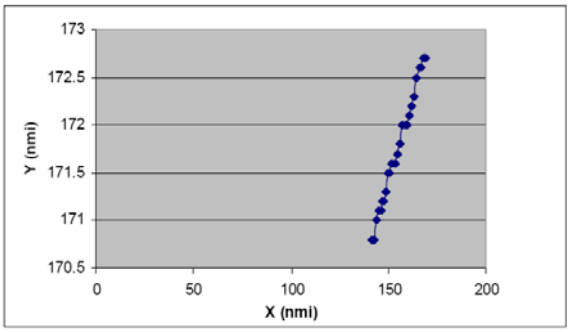


Figure 5.5: Planar flight path of the climbing aircraft

constant (shown in figure 5.5). As can be seen in figure 5.4(a) the probability of being in level flight drops below one before the aircraft starts to climb, which could indicate a turn. But since the aircraft was flying on a straight line with zero degree change in its heading angle, the governing equations for the stochastic hybrid evolution in this flight mode was adjusted with the following condition: if the change in the position of the aircraft along any of its axes is less than 0.2 nmi, then the aircraft is either in steady level flight, or climbing/descending flight, depending on the value of its altitude changes during each time interval. Note, the model is not predicting the climbing mode at this point. Thus, the addition of this pathological case created a refinement in the HMM, allowing it to become more robust in the learning process. This allowed the HMM to make the clear determination of the climbing flight, thereby eliminating the turn mode, within approximately 12 seconds (between measurement intervals 10-12 on figure 5.4(a)). Thus, the probability of $P(3\,|H) = P(3|\,C) = 0$ for the HMM trained on the climbing flight data. The descending aircraft chosen was a clean descent without significant variation in the x-y plane and the model detected the mode change from steady level flight to descending flight within the space of a measurement interval (6 seconds in interval 11).

Measuring information entropy is one way to evaluate a model, and also

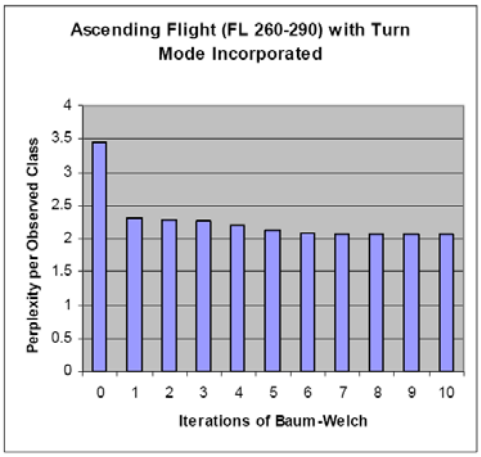
a way to assess the complexity of a modeling task for a given repository of data sets. Entropy is a measure of the level of certainty obtained with the addition of one additional data point, or:

$$H = \sum P(Observations) \times \lg_2(P(Observations)). \tag{5.4}$$

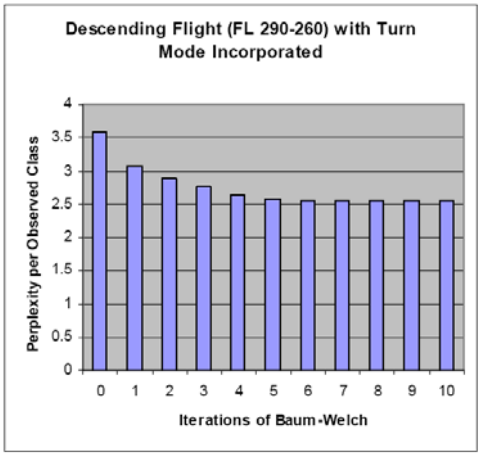
Mathematically, the perplexity of a model is defined as the base two exponent of the entropy. We generally say a model is good to the degree that it assigns high probability (equivalently, low perplexity) to test data. Obviously no modeling technique can guarantee that the model will assign high probability to test data it has never seen, but the Baum-Welch algorithm is guaranteed, at every iteration, to at least improve the probability of the observed training data.

Figure 5.6 illustrates convergence via each iteration of the Baum-Welch algorithm; the uncertainty associated with the transition probabilities is decreasing, and approaches a perplexity value of about two for the climbing flight (figure 5.6(a)) and a value of about 2.5 (figure 5.6(b)) for the descending flight. Thus, the total probability assigned to of all the paths that explain the data is always increasing. The plots in figure 5.6(c) and 5.6(d) show the perplexity of data for the model that does not include turn mode [61].

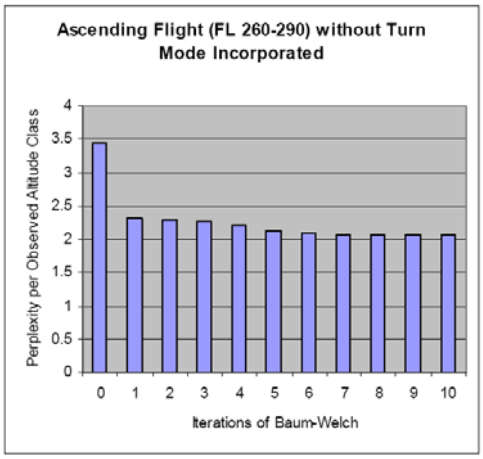
For the climbing flight example, the added condition in the stochastic hybrid evolution equations that uses the additional aircraft track information provided by the heading angle (as well as the x-y positions) results in the perplexity of the HMM with extended modes (level, climbing, descending, level turn, climbing turn, descending turn) being equal to that of the simple model with three modes (level, climbing, descending). Thus, two extra modes were added, but three extra pieces of observational information (per



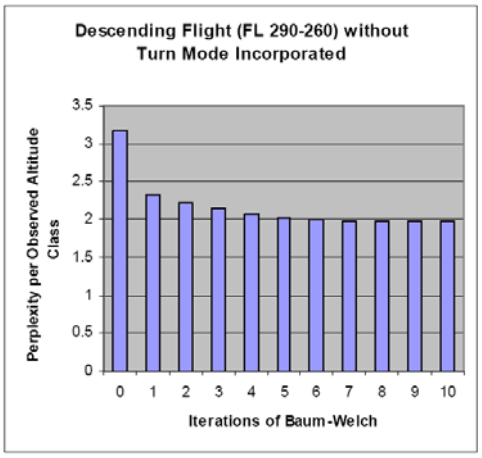
(a) Climbing flight



(b) Descending flight



(c) Climbing flight for the previous model [61]

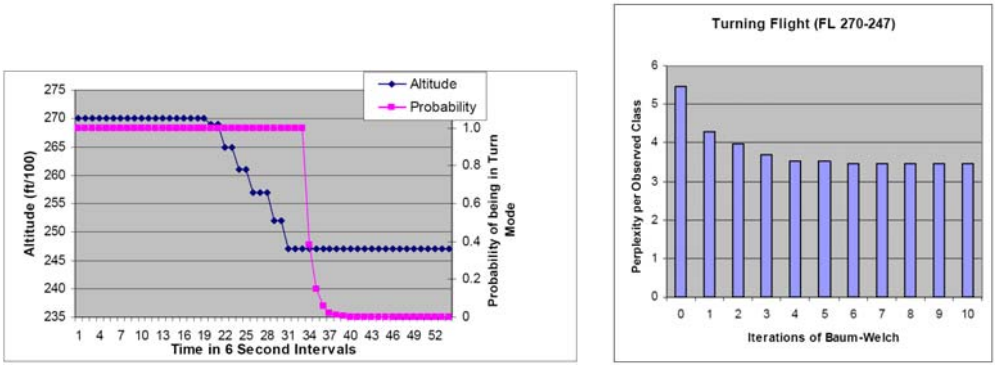


(d) Descending flight for the previous model [61]

Figure 5.6: Perplexity of data over 10 iterations

measurement) were provided in the training set. The heading angle allows for the immediate elimination of the pathological turning case due to large error in the x-y plane, thereby eliminating the need to consider any of the turning modes of flight when determining the mode in the extended model. Hence, the values seen in figure 5.6(a) and figure 5.6(c) are equal, as the extended model reduces to the simple model used in Ref. 61. However, in the descending flight, the initial perplexity and its final value after 10 iterations, are higher when the turn mode is added to the model. This is expected

because the information provided by the heading angle is insufficient to eliminate the uncertainty in determining whether the aircraft is in turning flight in the extended model.



(a) Mode detection threshold (b) Perplexity of data

Figure 5.7: Random Aircraft

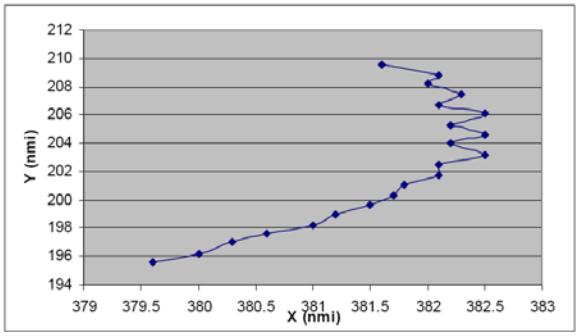


Figure 5.8: Planar flight path of the random aircraft

Now, to evaluate the effectiveness of an HMM at detecting mode changes, it is necessary to score the ability of the model to produce a given series of observations. We can evaluate the performance of the trained HMM against random aircraft trajectories in the specified flight level, many of whom are not well behaved.

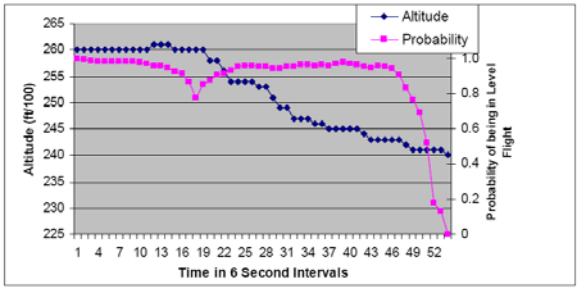
A representative example of a random aircraft trajectory and the associated mode change sequence is given in Fig. 5.9. The perplexity of the data over 10 iterations not only begins at a significantly higher value (Fig. 5.9(b)),

for the first iteration (of 5.5, compared to 3.5 for the well behaved data), but never settles below 3.5, demonstrating that even further refinement may not improve the entropy associated with the model. As can be seen, the perplexity of the data in this extended model is higher than when the turn mode is not included in the simplified model (Fig. 5.9(b)). Fig. 5.10 shows the planar flight path of this aircraft.

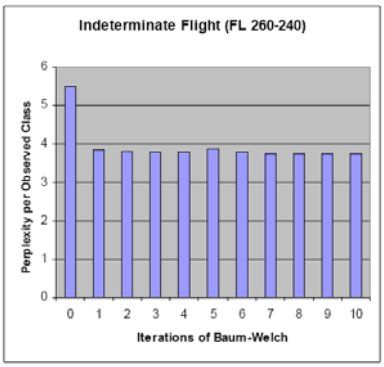
Fig. 5.7 shows another random aircraft. Turn mode detection is obvious in this figure (figure 5.7(a)) and figure 5.8 indeed shows that the path of the aircraft in x-y plane is a turn. Thus, within three measurement intervals, the extended model is able to determine the mode switches between turning flight and descending turning flight, then between descending turning flight and level flight. The perplexity of the data over 10 iterations not only begins at a significantly higher value (figure 5.7(b)), for the first iteration (of 5.5, compared to 3.5 for the well behaved data), but never settles below 3.46, demonstrating that even further refinement may not improve the entropy associated with the model.

5.3.2 Parameter Adjustment for Mode Detection using FEWS Data

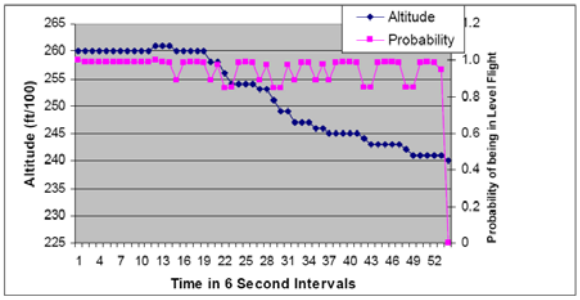
Now we use the HHMM to detect mode changes in a new set of aircraft flight data recorded for FAA'a Future Enroute Workstation Study (FEWS) using the automaton shown in figure 4.1. Table 5.2 shows the final probabilities of the HHMM trained on a descending aircraft data. The three classes of observations for predicted flight over a twelve second interval are denoted by: Class 1, descending flight, Class 2, steady level flight, and class 3, climbing flight. Figure 5.11 shows the mode change sequence and the perplexity of



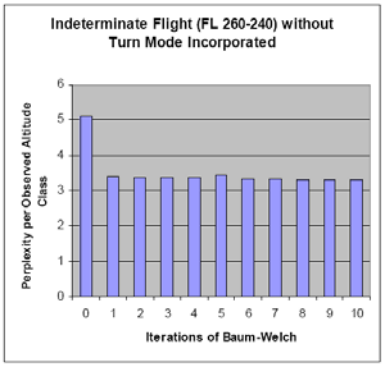
(a) Mode detection threshold



(b) Perplexity of data



(c) Mode detection threshold for the previous model [61]



(d) Perplexity of data for the previous model [61]

Figure 5.9: Indeterminate flight path

data for the descending aircraft.

Figure 5.13 shows a random aircraft. This aircraft is climbing from FL260 and at the 12^{th} time instant, it levels off at FL290 and then starts descending at the 33^{th} time interval and levels off at the 47^{th} time step when it reaches FL220. Figure 5.14 shows the planar path of this aircraft. In figure 5.13 the accuracy of the mode detection of the the HHMM described in this section is compared with the accuracy of the HMM introduced in the previous section. As can be seen the HHMM detects mode changes (e.g. climb to steady level flight) approximately 12 seconds before the HMM. Note that while the initial perplexity of about 4.82 is higher than the 4.72 of the HMM model in figure 5.13(d), the well behaved flight modes evinced by the aircraft in figure

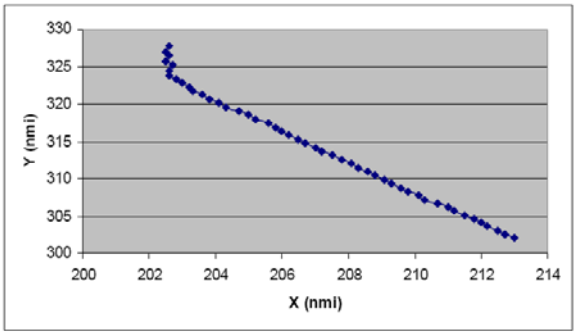


Figure 5.10: Planar flight path of the indeterminate aircraft

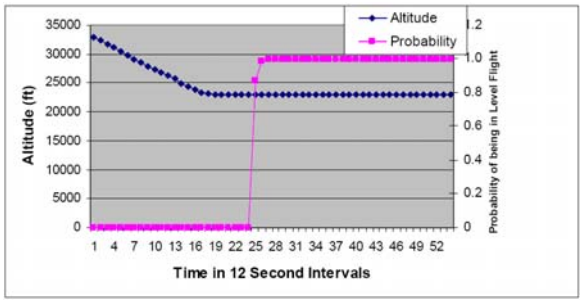
Table 5.2: Final probabilities for HHMM trained on FEWS data for descending flight (FL 320-230) after 10 iterations

Descending	$p(\cdots\|C)$	$p(\cdots\|H)$	$p(\cdots\|START)$
$p(1\,\|\cdots)$	0.00	1.00	
$p(2\,\|\cdots)$	1.00	0.00	
$p(3\,\|\cdots)$	0.00	0.00	
$p(C\,\|\cdots)$	0.97	0.06	0.00
$p(H\,\|\cdots)$	0.00	0.94	1.00
$p(STOP\,\|\cdots)$	0.03	0.00	0.00

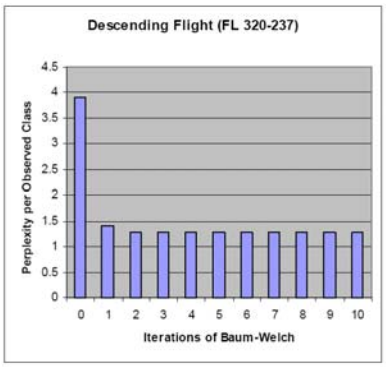
5.13(b) have a final complexity of about 2.67, lower than the 2.85 shown in figure 5.13(d). Thus, the HHMM is more accurate in this case after sufficient iteration.

It should be noted that the classical limitations of hidden Markov models are inherent in the HHMMs produced by this work. The fundamental assumption in all Markov models is that the transition probabilities are independent, which may not necessarily be true of all flight data. The HMM is only as good as the data it was trained upon, and a tendency to iterate too many times results in overfitting the data, thereby decreasing the score of the model against other data sets.

Furthermore, we assume that our model has an inherent inertia, which is characteristic of aircraft flight paths. This acts to smooth the estimation process over iterations. It should be noted, however, that the continuous



(a) Mode detection threshold



(b) Perplexity of data

Figure 5.11: Descending Flight

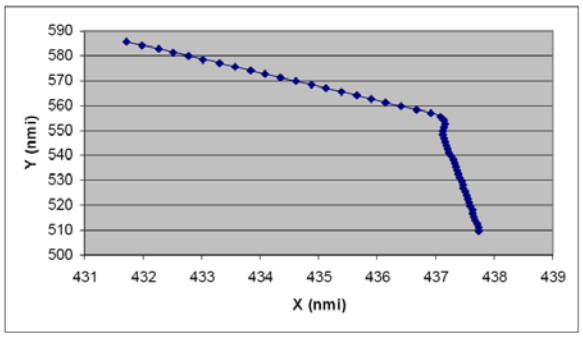
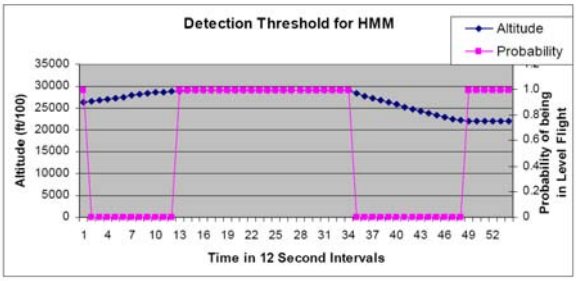


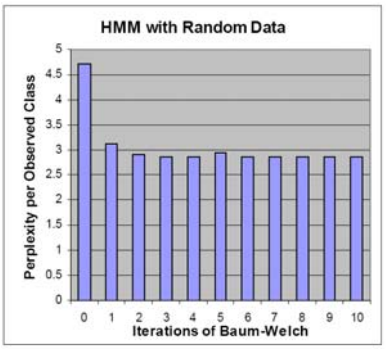
Figure 5.12: Planar flight path of the descending aircraft

model used for the aircraft dynamics had associated with it, an uncertainty equal to approximately 200 ft in the worst case, which is essentially equal to the discrete estimation threshold used to determine the flight observation sequence.

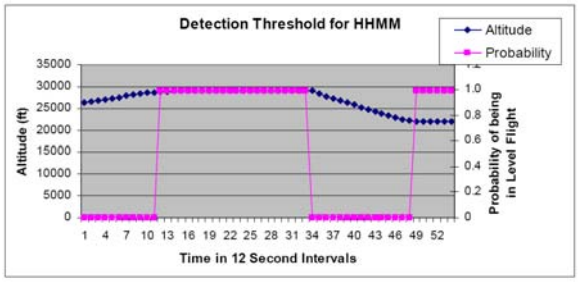
Next, we compare the efficacy of the HHMM with multiple-model Kalman filter algorithms.



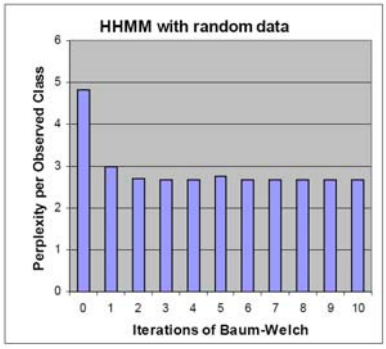
(a) Mode detection threshold with HHMM



(b) Perplexity of data for HHMM



(c) Mode detection threshold with HMM



(d) Perplexity of data for HMM

Figure 5.13: Indeterminate flight path

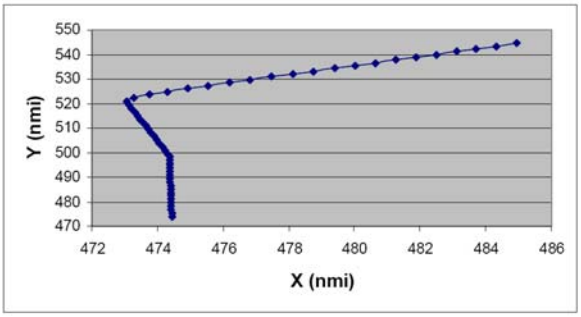


Figure 5.14: Planar flight path of the indeterminate aircraft

5.4 Comparing HMM and State-Dependent-Transition Hybrid Estimation Algorithm

Seah and Hwang [63] propose a stochastic linear system hybrid estimation algorithm to improve tracking of aircraft around airports by using the knowledge of departure or arrival aircraft flight plans and nominal flight profiles.

They use a multiple-model Kalman filter algorithm for aircraft tracking with a new assumption, i.e., they consider mode transition probabilities to be dependent on the continuous state, and they derive a conditional mode transition matrix. They present a discrete-time stochastic linear hybrid system model of the aircraft dynamics:

$$\begin{aligned} x(k+1) &= A_{m(k)}x(k) + B_{m(k)}w_{m(k)}(k) \\ z(k) &= C_{m(k)}x(k) + v_{m(k)}(k) \\ \pi_{ij}(x(k)) &= P\{m(k) = j \,|m(k-1) = i, \quad x(k)\} \end{aligned} \tag{5.5}$$

where the continuous state vector is $x = \left[\xi \ \dot{\xi} \ \ddot{\xi} \ \eta \ \dot{\eta} \ \ddot{\eta} \ h \ \dot{h}\right]^T$ and $m(k) \in \{1, 2, \cdots, r\}$ is the discrete state/mode. Also the process noise and measurement noise are uncorrelated zero-mean Gaussian sequences.

The horizontal flight submodes are: CV (constant velocity and heading), CA (constant acceleration/deceleration) and CT (coordinated turn with constant speed and turning rate). The vertical submodes are: CH (constant altitude) and CD (constant altitude change rate).

First, they model the mode transitions using known/deterministic flight profiles (take-off, landing and movement along SID/STAR). They draw separate finite state automata for horizontal and vertical planes to show flight mode transition models for each flight-mode change point (FCP), where the transition conditions are called guard conditions (C_{ij}). For the stochastic model, they model the flight profiles to be randomly distributed about the nominal profile with a Gaussian distribution uncertainty. They show that the conditional mode transition probabilities π_{ij} can be expressed with multivariate Gaussian density functions instead of multivariate integrals. This algorithm is called State-Dependent-Transition Hybrid Estimation (SDTHE) algorithm.

Figures 5.15 and 5.16 compare the mode detection accuracy of HMM with SDTHE and IMM algorithms under the conditions of Example 2 in Ref. 63. Because of the discrete nature of the SDTHE and IMM algorithms, HMM was used instead of the HHMM. As can be seen, HMM has a better accuracy than IMM and it detects mode changes at the same time instant as SDTHE algorithm.

The hidden Markov models extend to hybrid HMMs which, detect inputs in an asynchronous fashion. On the other hand, Kalman filter techniques (such as IMMs) are restricted in the sense that they have to wait for the next discrete time interval to update their mode estimation. Hence, HHMMs are able to detect changes that occur in an asynchronous fashion, that is, between "update" intervals.

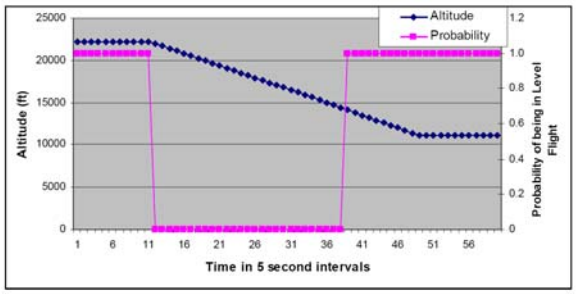


Figure 5.15: Mode detection threshold HMM

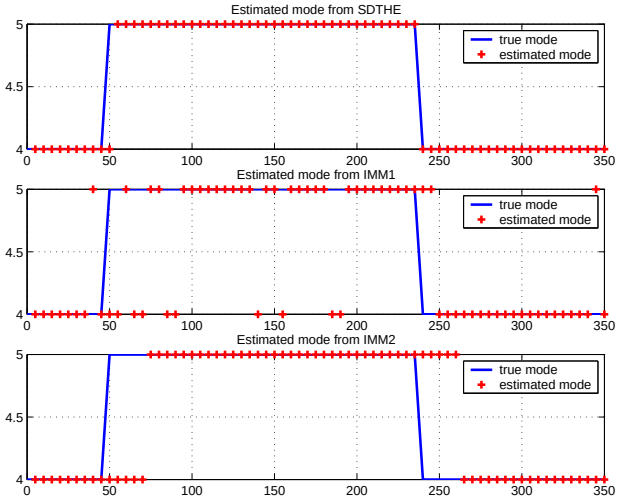


Figure 5.16: Mode detection threshold SDTHE, numbers 4 and 5 correspond to submodes CH and CD respectively.

Chapter 6

Decentralized Conflict Detection and Resolution

Motion coordination/assignment is the general problem of reaching some global spatial pattern of movement in a set of autonomous agents. In [64] a notion of virtual nodes (VN) is developed where a virtual node is an abstract, relatively well-behaved active node that is implemented using less well-behaved real nodes. In this framework, virtual nodes are associated with predetermined, well-distributed locations in the plane, communicating among themselves and with mobile client nodes using local broadcast. The plane is separated into disjoint zones, each belonging to a virtual node, which is emulated by all the mobile nodes present in its zone. The VN abstraction provides a centralized controller with reliable storage. The HIOA framework is used to describe the components of the system. This VN abstraction layer is applied to the problem of coordinating the motion of real mobile nodes in a 2 dimensional space.

A distributed hybrid approach to the assignment problem in distributed motion planning is considered in [65], which simultaneously addresses the discrete assignment of destinations to agents, which is determined dynamically through distributed coordination protocols, as well as the continuous control strategies for driving the individual agents to the destinations.

The problem of safely coordinating the motion of several agents sharing the same environment has received a great deal of attention. In the air

traffic control literature, a control policy for generating provably safe conflict resolution maneuvers for two aircraft is proposed in [66]. The approach allows for uncertainty in the intent of one of the aircraft and calculates the least restrictive control scheme for the other aircraft based on the worst case uncertainty and the minimal unsafe operating region for each aircraft.

Other decentralized algorithms can be found in [67–70].

6.1 Verifying Safety Properties

Proving safety properties over cyber-physical systems involves generating the set of reachable states for the system, and then assuring that a mathematical invariant that embodies the safety property holds over this set. Generating the exact reachable sets of hybrid systems is a non-trivial task, as most fixed point iteration schemes possess no guaranteed termination conditions, rendering the problem intractable. Usually, fast overapproximations of the reachable sets are generated instead, and the invariant property is proved over the conservative set. However, the lack of a proof does not necessarily mean that the exact reachable set does not obey the given property; it then becomes necessary to find a weaker formulation of the safety property, or a closer overapproximation to the exact reachable set.

6.1.1 Automatic Invariance Verification

Definition 6.1.1 *Let $\mathcal{M}$ be a hybrid hidden Markov model with X as the set of internal variables and set of states $N = Q \subset val(X)$. Also let $\mathcal{P}$ be a predicate on X. The set of states satisfying $\mathcal{P}$ is denoted by $\mathcal{P}$ as well. The predicate $\mathcal{P}$ is called an invariant of $\mathcal{M}$ if $\mathcal{R}_{\mathcal{M}} \subseteq \mathcal{P}$, where $\mathcal{R}_{\mathcal{M}}$ is the*

reachable set of $\mathcal{M}$.

An HHMM $\mathcal{M}$ is safe with respect to a particular safety property $\mathcal{S}$, if $\mathcal{S}$ is an invariant for $\mathcal{M}$. If the set of reachable states of $\mathcal{M}$, $\mathcal{R}_{\mathcal{M}}$, is computable, then whether $\mathcal{R}_{\mathcal{M}} \subseteq \mathcal{S}$ can be checked. Computing $\mathcal{R}_{\mathcal{M}}$ is decidable only if $\mathcal{M}$ belongs to a fairly restricted class [71]. These decidable classes are generally called order minimal hybrid automata [72] with linear, polynomial, and exponential state models [69, 73–78].

If generating the exact reachable set is not trivial, another alternative is to compute an overapproximation of $\mathcal{R}_{\mathcal{M}}$, denoted by $\bar{\mathcal{R}}_{\mathcal{M}}$ [79, 80] and then check the invariant property over the conservative set. Algorithms for overapproximation of reachable set for hybrid systems have been developed [81–85].

We now introduce another class of properties of $\mathcal{M}$.

Definition 6.1.2 *Let* $\mathcal{M}$ *be a hybrid hidden Markov model with* X *as the set of internal variables and set of states* $N = Q \subset val(X)$. *A predicate* $\mathcal{P}$ *on* X *is an inductive property of* $\mathcal{M}$ *if any execution that starts from a state satisfying* $\mathcal{P}$ *reaches only states that also satisfy* $\mathcal{P}$. *An inductive property* $\mathcal{P}$ *that is satisfied by all the starting states of* $\mathcal{M}$ *is an invariant.*

In computer science, invariants and levels of abstraction are standard methods for reasoning about discrete systems. In Ref. 86 invariant assertion is used to analyze an acceleration maneuver modeled by hybrid input/output automata. A deductive technique used for verifying invariance has been applied earlier to air traffic control systems in the context of HIOAs and SHIOAs (structured HIOAs) [11, 87] and also to vehicle control systems [86, 88, 89]. In this technique, the desired invariant property $\mathcal{P}$ is deduced from

the specifications of the HHMM $\mathcal{M}$ by first finding an inductive property $\mathcal{P}' \subseteq \mathcal{P}$, and then checking that the transitions and trajectories of $\mathcal{M}$ preserve $\mathcal{P}'$.

6.2 Case Study: Decentralized Policy for Conflict Avoidance of Aircraft in a Stream

6.2.1 Self Separation of Aircraft in a Stream

If we define collision between any two (or more) aircraft PN_i and PN_j as being such that a loss of physical separation occurs (say, as being the two aircraft are no longer laterally separated by 5 nautical miles, or no longer vertically separated by 1000ft), then we can express the act of collision in terms of the reachable sets of the automata (HHMMs) PN_i and PN_j.

For each aircraft PN_i in a bounded sector B in $\Re^3$, all of whom possess unique identifiers $i \in I$, communication occurs via Automatic Dependent Surveillance - Broadcast (ADS-B) technique. That is, all aircraft PN_j within a radius R_p of the agent PN_i who sends broadcast message m at time t, will receive the message m in the bounded time interval $[t, t + \delta_{PN_i}]$, provided that PN_j remains within R_p for the entire interval.

Definition 6.2.1 *Denote the set of states reachable for an HHMM PN in the time interval $[t, t + \delta_{PN}] = \Delta t$ as being $R_{PN} : x_{PN} \longrightarrow 2^{x_{PN}}$; where x_{PN} is the position of PN at time t, and $R_{PN}(x_{PN})$ is the set of all physically attainable points for the aircraft PN in the time interval Δt, given its dynamic*

constraints.

Note that this definition can be extended to encompass a set relation. That is, given a set of possible initial positions, Θ, (and a probabilistic distribution over these positions); we denote the set of reachable states (and its corresponding probabilistic distribution) as being the union of reachable sets derived using each point in the initial set as the starting point in the singleton relation.

Definition 6.2.2 *The set of reachable states for PN in $[t, t + \delta_{PN}]$, given the initial set of start states Θ, is denoted by $R_{PN}(\Theta) = \bigcup_{x_{PN} \in \Theta} R_{PN}(x_{PN})$.*

This leads to the formal definition of a collision as:

Definition 6.2.3 *A collision between aircraft PN_i and PN_j is formally defined as occurring if*

$$C_{PN_{i,j}} : \left\{ \vec{x}_{PN_i} - \vec{x}_{PN_j} \right\} \leq \vec{K} \tag{6.1}$$

where K is the collision threshold. We denote this as $(x_{PN_i}, x_{PN_j}) \in C_{PN_{i,j}}$.

The threshold K is a vector quantity as the threshold for lateral separation differs from that of vertical separation.

For aircraft traveling along a parameterized route lateral separation is the primary concern. If we bound the maximum and minimum acceleration $[\overline{a}, \underline{a}]$ and velocity $[\overline{v}, \underline{v}]$ each aircraft is capable of maintaining, we arrive at the following two conditions necessary to ensure self separation along the parameterized path for the time interval Δt.

Theorem 6.2.4 *If we consider the scenario where three aircraft PN_i, PN_j and PN_k correspond to the leading aircraft, the "own" aircraft and the lagging aircraft, respectively and with the assumption of starting from a safe condition, we must have, for safety and liveness to be guaranteed in the time interval $\Delta t = [t_0, t_0 + \delta t]$:*

$$\exists v_j > \underline{v},\ a(\tau) \in [\overline{a}, \underline{a}] \quad such\ \ that$$

$$K + x_{PN_j}(t_0) + \int_{t_0}^{t_0+\delta t} (v_j(t_0) + a(\tau)tau)d\tau \leq x_{PN_i}(t_0) + \int_{t_0}^{t_0+\delta t} (v_{PN_i}(t_0) + \underline{a}\tau)d\tau \tag{6.2}$$

and

$$x_{PN_j}(t_0) + \int_{t_0}^{t_0+\delta t} (v_{PN_j}(t_0) + a(\tau)d\tau)d\tau \geq K + x_{PN_k}(t_0) + \int_{t_0}^{t_0+\delta t} (v_{PN_k}(t_0) + \bar{a}\tau)d\tau \tag{6.3}$$

This is a conservative bound, which assumes nothing about the behavior of PN_i and PN_k, only that PN_j has received the broadcast message m_i containing x_i and m_k containing x_k.

6.2.2 Problem Definition

Here we consider a decentralized air traffic control system, where aircraft are flying along differentiable curves (i.e., enroute flight paths) that we call streams. The streams are placed in such a way that all the streams on the vertical plane and all the streams on the lateral plane are conflict free.

Hence we can look at one of these streams on which the aircraft are ordered $1 < 2 < \cdots < i-1 < i < i+1 < \cdots N$, where the 1^{st} aircraft is in the lead, and the N^{th} aircraft is trailing. Safety conditions prohibit two aircraft loosing a separation of L ($\vec{K} = (L, D)$), and liveness conditions require each aircraft to maintain a minimum velocity $\underline{v}$. Performance considerations limit maximum acceleration and maximum deceleration, $a \in [\bar{a}, \underline{a}]$, as well as maximum velocity $\bar{v}$. Thus the initial conditions for one stream are:

$$v_i \in [\underline{v}, \bar{v}] \quad \forall i \in [1, \cdots, N] \tag{6.4}$$

$$|x_{i+1} - x_i| > L \quad |x_i - x_{i-1}| > L \quad \forall i \in [2, \cdots, N-1] \tag{6.5}$$

The communication radius for all aircraft is assumed to be R, where $R >> L$.

The composite HHMM for each of these aircraft, H, is composed of four component HHMMs, where each corresponds to a mode of flight and are labeled Cruise, Accelerate, Climb and Turn, as shown in Fig. 6.1. In this model the *Cruise* Automaton acts as a Markovian scheduler/supervisor. It receives the global inputs, i.e., the flight data of the neighboring aircraft, along with the local inputs corresponding to the "own" aircraft. If no potential conflicts are detected, it only evolves the steady level flight trajectories and sends out the flight data of the "own" aircraft. But if a conflict is detected, it determines the resolution maneuver, and then sends out the new flight data to both the next mode automaton and the neighboring aircraft automata. Now we will develop each automaton in HIOA language. But first we will present the language specifications.

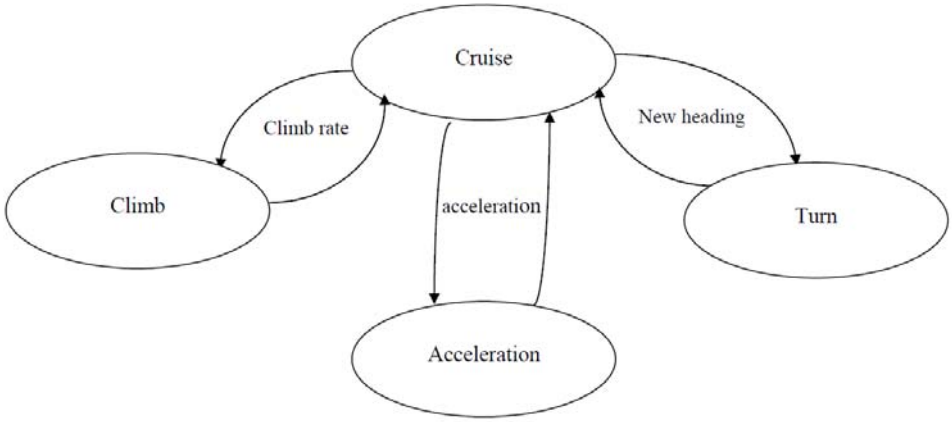


Figure 6.1: Composed HHMM for each aircraft

6.2.3 The HIOA Language

We use the HIOA language described in Ref. 11 to describe the component HHMM.

The first line consists of the keyword **automaton** followed by the automaton's name and a list of formal parameters, which are used to specify sets of objects such as automata, actions, trajectories, etc.

The body of the specification has four sections: signature, variables, transitions, and trajectories. The actions of an automaton are declared by the keyword **signature** followed by a list of actions and their kinds.

The **variables** section declares the variables of the automaton along with their kinds, types, dynamic types and possibly their initial value.

The discrete transitions corresponding to each action of the automaton are defined in the **transitions** section. A precondition can be defined using the keyword **pre** followed by a predicate on the automaton parameters, action parameters and the internal variables. Due to the input enabling condition, preconditions cannot be defined for input actions. The program following the keyword **eff** defines how the state changes when the action occurs. The

statements of the program are assignments, or conditionals or loops.

The **trajectories** section defines a set of state models for the automaton. Each trajectory definition starts with the keyword **trajdef** followed by a name for the state model, an optional invariant condition, a stopping condition using the keyword **stop when** and a set of Differential and Algebraic Inequalities (DAIs).

Figure 6.2 shows the *Cruise* Automaton (HHMM) in HIOA language [11] for H_i. H_i has two sets of global input and output actions corresponding to communication from H_{i-1} and H_{i+1}. *Cruise* receives the global inputs and the local inputs corresponding to H_i. The *receive* action receives the aircraft current flight data, i.e., the position, velocity and heading of each aircraft and evolves the steady level flight trajectories. Then, to detect potential conflicts the safety condition is checked by the internal action, *conflictdetect*. If a conflict is detected, the value of the boolean internal variable *conflict* changes to true and the evolution of the trajectories stops. To resolve the conflict the *resolve* action follows the *conflict* action, where a Markovian scheduler determines the resolving maneuver. The protocol is as follows. The first choice is the acceleration mode, but if acceleration or deceleration cannot be done due to the occurrence of further violations, the climb mode should be considered, however if another aircraft is above or below H_i, then the turn mode is chosen. When the acceleration mode is chosen, a new value for velocity is determined, similarly for the climb mode, a new altitude and for the turn mode a new heading is determined. The *send* output action sends the new values of flight data both to the subsequent automaton as well as to H_{i-1} and H_{i+1} along with the new intended value of altitude, velocity or heading. The automata corresponding to the climb/descend, turn and acceleration maneuvers are shown in Figures 6.3,6.4,6.5 respectively.

Automaton Cruise
Signature
Input
receive(m_i),m_i $\in$(position,velocity,heading)
Ouput
send(m_2, m),m $\in$(position,velocity,heading)
$\bigcup$ (newaltitude,newvelocity,newheading)
Internal
conflictdetect;resolve
Variables
Internal
$(x_c, y_c, z_c) \in \Re^3$;$v_c \in \Re$;$\phi_c \in \Re$
conflict:Bool:=false;d:Bool:=false
d $\in \Re$, d:=0
Transitions
Input receive(position,velocity,heading)
effect
$(x_c, y_c, z_c) \leftarrow$ position
$v_c \leftarrow$ velocity
$\phi_c \leftarrow$ heading
output send(position,velocity,heading)
pre
conflict
output send(newaltitude,newvelocity,newheading)
pre
conflict
Internal conflictdetect
eff
let d=safetycheck(m_1, x_c, y_c, z_c, m_3)
if d=true then
conflict:=true fi
Internal resolve
pre
conflict
eff
let res=resmaneuver(m_1, x_c, y_c, z_c, m_3)
if res=1 then
newvelocity $\leftarrow$ newvelocity
newaltitude,newheading $\leftarrow$ 0
if res=2 then
newaltitude $\leftarrow$ newaltitude
newvelocity,newheading $\leftarrow$ 0
if res=3 then
newheading $\leftarrow$ newheading
newvelocity,newaltitude $\leftarrow$ 0
Trajectories
trajdef motion
evolve
$\dot{x}_c = v_c \ \cos(\phi_c)$
$\dot{y}_c = v_c \ \sin(\phi_c)$
$\dot{z}_c = 0$
stop when
conflict

Figure 6.2: Cruise Automaton

Automaton Climb/Descend

Signature
Input
receive(m),$m \in$(position,velocity,heading,newaltitude)
Ouput
send(m),$m \in$(position,velocity,heading)

Variables
Internal
$(x_{cd}, y_{cd}, z_{cd}) \in \Re^3$;$v_{cd} \in \Re$;$\phi_{cd} \in \Re$;
$(\alpha_{cd}, z_{newcd}) \in \Re^2$

Transitions
Input receive(position,velocity,heading,newaltitude)
effect
$(x_{cd}, y_{cd}, z_{cd}) \leftarrow$ position
$v_{cd} \leftarrow$ velocity
$\phi_{cd} \leftarrow$ heading
$(\alpha_{cd}, z_{newcd}) \leftarrow$ newaltitude

output send(position,velocity,heading)
pre
$z_{cd} = z_{newcd}$

Trajectories
trajdef flightdata
evolve
$\dot{x}_{cd} = v_{cd}\ \cos(\phi_{cd})\ \cos(\alpha_{cd})$
$\dot{y}_{cd} = v_{cd}\ \sin(\phi_{cd})\ \cos(\alpha_{cd})$
$\dot{z}_{cd} = v_{cd}\ \sin(\alpha_{cd})$
stop when
$z_{cd} = z_{newcd}$

Figure 6.3: Climb Automaton

6.2.4 The Conflict Avoidance Policy

In this setup without loss of generality, we assume for each pair of aircraft the leading aircraft is in charge of conflict avoidance, i.e., it first detects the conflict using the mode identification property of HHMM and then performs the resolution manuever. There are four available maneuvers which we describe

Automaton Turn

Signature
Input
receive(m),$m \in$(position,velocity,heading,newheading)
Ouput
send(m),$m \in$(position,velocity,heading)

Variables
Internal
$(x_t, y_t, z_t) \in \Re^3$;$(x_{t0}, y_{t0}, z_{t0}) \in \Re^3$;$v_t \in \Re$;
$\phi_t \in \Re$;$r_{newt} \in \Re$

Transitions
Input receive(position,velocity,heading,newheading)
effect
$(x_t, y_t, z_t) \leftarrow$ position
$(x_{t0}, y_{t0}, z_{t0}) \leftarrow$ position
$v_t \leftarrow$ velocity
$\phi_t \leftarrow$ heading
$r_{newt} \leftarrow$ newheading

output send(position,velocity,heading)
pre
$(x_t^2 + y_t^2) = (x_{t0}^2 + y_{t0}^2)^{(1/2)} + 2r_{newt}$

Trajectories
trajdef flightdata
evolve
$\dot{x}_t = v_t \ \cos(\phi_t)$
$\dot{y}_t = v_t \ \sin(\phi_t)$
$\dot{z}_t = 0$
$\dot{\phi}_t = \dot{\phi}_t$
stop when
$(x_t^2 + y_t^2) = (x_{t0}^2 + y_{t0}^2)^{(1/2)} + 2r_{newt}$

Figure 6.4: Turn Automaton

in detail in the following sections.

Automaton Acceleration

Signature
Input
receive(m),$m \in$(position,velocity,heading,newheading)
Ouput
send(m),$m \in$(position,velocity,heading)

Variables
Internal
$(x_a, y_a, z_a) \in \Re^3$;$v_a \in \Re$;$\phi_a \in \Re$;
$v_{a0} \in \Re$;$(a_{newa}, v_{newa}) \in \Re^2$

Transitions
Input receive(position,velocity,heading,newvelocity)
effect
$(x_a, y_a, z_a) \leftarrow$ position
$v_{a0} \leftarrow$ velocity
$v_a \leftarrow$ velocity
$\phi_a \leftarrow$ heading
$(a_{newa}, v_{newa}) \leftarrow$ newvelocity

output send(position,velocity,heading)
pre
$v_a = v_{newa}$

Trajectories
trajdef flightdata
evolve
$\dot{x}_t = (v_{a0} + a_{newt}t)\ \cos(\phi_t)$
$\dot{y}_t = (v_{a0} + a_{newt}t)\ \sin(\phi_t)$
$\dot{z}_t = 0$
stop when
$v_a = v_{newa}$

Figure 6.5: Acceleration Automaton

The Cruise Maneuver

As long as each pair of aircraft are not in conflict the leading aircraft does not change its speed, heading or pitch.

$$v_i^t = v_i^{t-1} \quad if v_i > v_{i+1} \tag{6.6}$$

The Acceleration Maneuver

If a conflict is detected, the first available resolution maneuver for the leading aircraft is acceleration.

For the acceleration maneuver to be feasible, there are three conditions that should be met:

$$v_i < v_{i+1}$$

$$v_i < v_{i+1},\ (\frac{1}{2}\overline{a}\tau^2 + v_i\tau) - (\overline{v}\tau) \geq L^{'} \tag{6.7}$$

$$v_f < \bar{v} \tag{6.8}$$

where, $L^{'} > L$ is the initial distance between the two aircraft and v_f is the final speed of the leading aircraft at the end of the maneuver.

If these conditions are met, then the transition function for acceleration is

$$v_i^t = v_i^{t-1} + \overline{a}(\tau - \overline{\epsilon}) \geq v_{i+1}^{t+1} \tag{6.9}$$

where $\overline{\epsilon}$ is the maximum acceleration time.

The Climb Maneuver

When the leading aircraft performs the climb maneuver, it increases its speed with maximum acceleration to maximum speed, then starts climbing with the possible maximum pitch to a secondary stream which is D distance above the main stream and then decreases its speed to the minimum speed. This means

that all the aircraft on the secondary stream are flying with minimum speed. Hence, no conflict occurs on the secondary stream except for when the aircraft wants to insert itself into the stream (i.e., when it climbs to the secondary stream and reaches the minimum speed). Thus the turn maneuver is only chosen when the aircraft can insert itself into the stream. The conditions for a safe climb maneuver are as the following

$$x_{i-1} + v_{i-1}T + \frac{1}{2}\overline{a}(T - \overline{\epsilon})^2 \leq x_i + \frac{\overline{a}\mathfrak{t}^3}{6\overline{\epsilon}} + \overline{v}\cos(\overline{\alpha})(T - \mathfrak{t}) \tag{6.10}$$

$$\overline{v}\sin(\overline{\alpha})(T - \mathfrak{t}) \geq D \tag{6.11}$$

where T is the maneuver time, $\mathfrak{t}^2 = \frac{\overline{v} - v_i}{\frac{\overline{a}}{\overline{\epsilon}}}$ and $\frac{da}{dt} \leq \frac{\overline{a}}{\overline{\epsilon}}$.

$$x_{k-1} + \underline{v}[T + \tau] - (x_i + \frac{\overline{a}\mathfrak{t}^3}{6\overline{\epsilon}} + \overline{v}\cos(\overline{\alpha})(T - \mathfrak{t}) + \overline{v}\tau) > L \tag{6.12}$$

where $\tau = \frac{2(\underline{v} - \overline{v})}{\frac{\overline{a}}{\overline{\epsilon}}}$.

The Turn Maneuver

The turn maneuver is always enabled if the distance between each pair of aircraft is enforced to be at least $L^{'} > L$. $L^{'}$ is selected using

$$\int_0^{T_{urn}} \overline{v}\sin(\psi)dt > L^{'} \tag{6.13}$$

where $T_{urn} = f(\dot{\overline{\psi}}, r_{turn})$.

Destination Merge

Since the climb and turn maneuvers result in secondary streams of aircraft, we need a policy for merging the aircraft at the destination (e.g., an airport).

There are two possible policies. The first one, which we call merge at the destination, dictates that the aircraft on the secondary streams should wait for all the aircraft on the main stream to merge. According to the second policy, called the priority merge, if the aircraft in the lead has neighbors on both the above and side streams, it should let two aircraft from different streams merge ahead of it. And if the aircraft in the lead has neighbors from one stream, it should let one aircraft from that stream merge ahead of it.

6.2.5 Proof of Safety

Theorem 6.2.5 *N aircraft are moving along a differentiable curve (an en-route flight path). The aircraft can be strictly ordered $1 < 2 < \cdots < i-1 < i < i+1 < \cdots N$, where the 1^{st} aircraft is in the lead, and the N^{th} aircraft is trailing. If the system starts in a safe state, i.e., the aircraft have enough separation at initial time, then it will always remain in a safe state using the composed HHMM and the Markovian scheduler which uses the decentralized common policy.*

Proof The proof of the theorem follows from the fact that trajectories are continuous functions of time. If at any point a pair of aircraft are in conflict the Markovian scheduler of the leading aircraft chooses a resolution maneuver. If this maneuver is acceleration, then the leading aircraft will change its speed until the two aircraft are not in conflict anymore and resumes steady level flight (cruise). If the resolution maneuver is climb (because the acceleration maneuver does not resolve the conflict), then the aircraft starts climbing to the new altitude and resumes the cruise maneuver. However, if the turn mode is chosen, then the aircraft transition to turn mode and when it reaches the desired curve, it goes back to the original cruise mode. Hence,

any conflict between pairs of aircraft is resolved within two transitions. Since this is a common policy between any two pair of aircraft, safety is always assured. Furthermore, we assume that all aircraft have enough fuel so that during the merge process even for the worst case scenario, which is the case where the first aircraft becomes the last aircraft to merge, they all eventually merge.

6.3 Conflict Detection Extension to Higher Dimensions

The decentralized conflict avoidance algorithm that we introduce here is based on the generalized roundabout policy (GR) of [90], in which the problem of collision-free motion planning for a number of nonholonomic mobile agents evolving on the plane is considered. The agents are able to move on the plane at constant speed, along paths with bounded curvature and the environment in which the agents move is unbounded and free of obstacles. The agents are aware of the position and orientation of nearby agents, within a certain sensing or communication radius, but they do not have access to any other information and they are not required to communicate explicitly their intentions or their objectives. However, all agents make decisions based on a common set of a priori decided rules. The GR policy provides provably safe sensor-based motion planning for an arbitrary number of agents.

The mathematical definition of the decentralized conflict detection algorithm for an aircraft, represented by an HIOA, which then receives as inputs the position, velocity and heading of all aircraft within sensor range, allows for the aircraft in question to run HHMMs of each aircraft, and to proba-

bilistically predict what mode of the HIOA each sensed aircraft has engaged. This allows for proactive conflict detection and resolution maneuvers that can be enacted by each aircraft, without involving overt communication between aircraft.

Aircraft are regarded as possessing a 5 nmi radius of lateral space to define its lateral conflict zone, and a 1000 ft vertical height to define its vertical conflict zone. This means that each i^{th} aircraft can be regarded as being centered in a cylindrical buffer, denoted by C_i^{buffer} which has a $d_s = 5$ nmi lateral radius, and a height of $v_s = 2000$ ft. If the cylindrical buffers of any two (or more) aircraft overlap, they are said to be in conflict. Thus, if we define the i^{th} aircraft holding position $(x_i(t), y_i(t), z_i(t))$ at time t, then the cylindrical buffer zone about the i^{th} aircraft is defined as:

$$C_i^{buffer} = \left\{(x, y, z) | (x - x_i)^2 + (y - y_i)^2 \leq (5 * 6080.20)^2 \wedge |z - z_i| \leq 1000\right\} \tag{6.14}$$

where (x,y,z) are specified in feet, with the origin given by standard hemispherical longitude and latitude.

Let us consider creating a more conservative buffering zone: define the maximum acceleration and maximum velocity that the aircraft can attain in the lateral and vertical planes as $\{a_{Lmax}, a_{Vmax}\}$ and $\{v_{Lmax}, v_{Vmax}\}$ respectively through performance specifications, and the maximum pitching and turning angles that the aircraft can execute as $\{\alpha_{max}, \psi_{max}\}$. Then, we can embed the cylindrical buffer in a more conservative cylinder, defined by the maximum vertical distance, d_{Vmax} and lateral distance d_{Lmax} that the aircraft can travel from its present position in the next 12 second update

interval, centered at the position of the i^{th} aircraft as follows:

$$C_i^{con} = \left\{(x,y,z)|(x-x_i)^2 + (y-y_i)^2 \leq (d_{Lmax})^2 \wedge |z-z_i| \leq d_{Vmax}\right\} \tag{6.15}$$

A sufficient condition to ensure safety is that the interiors of conservative buffers C_i^{con} are disjoint at all times; if such a condition is met, conflicts can be avoided if agents hold their conservative buffers fixed, and move within them (by setting $\psi\prime = -1$, which results in circular motion about a fixed center, for normalized and rate limited $\psi\prime$). As a consequence, each point of contact between conservative buffers defines a constraint on further motion for both agents involved. More precisely, if the conservative buffer of agent i is in contact with the conservative buffers of agents with indices in J_i, where $i \subset \{1,...,n\}$ of the n aircraft, the motion of the agents is constrained as follows:

$$x_i'(x_j - x_i) + y_i'(y_j - y_i) \geq 0, \forall j \in J_i \tag{6.16}$$

In other words, the velocity of the i^{th} conservative buffer is constrained to remain in the convex cone determined by the intersection of a number of closed half-planes. Note that the full set of constraints can be computed assuming that each agent is aware of the configuration of all agents within the sum of conservative buffers. In addition, the amount of information needed to compute the bound is uniformly bounded, independent from the total number of agents in the system: in fact, the maximum number of agents whose conservative buffer is in contact with the conservative buffer of the computing agent is twenty.

As previously mentioned, setting $\psi\prime = -1$ causes an immediate stop of an aircraft's conservative buffer's motion. We will say that when $\psi\prime = -1$,

the aircraft is in the *Hold* state. Given that each aircraft is assigned origin-destination pairs, $(g_{i0}, g_{if}) = [(x_{i0}, y_{i0}, z_{i0}), (x_{if}, y_{if}, z_{if})]$, the *Straight* mode is defined as being, at time t:

$$\psi_i(t) = arctan(\frac{y_{if} - y_i(t)}{x_{if} - x_i(t)}) \wedge \psi\prime = 0 \wedge v_L \neq 0 \wedge v_V = 0 \tag{6.17}$$

that is, the heading angle is the planar line of sight angle (discounting pitch) to the destination from the present position, and the aircraft is moving laterally along that heading angle at a non-zero velocity, while not changing altitude.

Define the set-valued map $\Theta : SE(2) \times 2^{SE(2)} \rightarrow 2^{S^1}$, associating to the configurations of an aircraft and of its neighbors the set of allowable lateral directions in which the conservative buffer of the computing aircraft can translate without violating the constraints of 6.16. For a connected, non-empty set $B \subset S^1$, $B \notin \emptyset$; define $max(B)$ and $min(B)$ as the elements on the boundary of B, respectively in the positive and negative direction with respect to the bisectrix of B. Finally, define the map $\Theta^-(g, \bar{g}) = \Theta(g, \bar{g}) \backslash min(\Theta(g, \bar{g}))$. In other words, the output of Θ^- is an open set, obtained removing the boundary in the clockwise direction of the cone of feasible reserved region translations, where $g_i = (x_i, y_i, \psi_i)$ is the lateral position of the i^{th} aircraft. Whenever Θ is a proper subset of S^1, $max(\Theta)$, $min(\Theta)$, and Θ^- are well defined. If $\Theta = \emptyset$, or $\Theta = S^1$, we set $\Theta^- = \Theta$.

Our concept for decentralized conflict-free coordination is based on maintaining the interiors of conservative buffers' disjoint. Assuming that no constraints are violated, an aircraft will attempt to steer the center of its own conservative buffer towards the position it would assume at the target configuration. In a free environment, this can be accomplished switching between

the *Hold* state and the *Straight* state according to the following logic:

$$\psi\prime(t) = \begin{cases} 0, & \psi = arctan(\frac{y_{if}-y_i}{x_{if}-x_i}) \wedge g_i \neq g_f \quad \text{if conservative conflict} \\ -1 & \text{otherwise} \end{cases} \tag{6.18}$$

where a conservative conflict is defined as at least one other aircraft having their conservative buffer in tangential contact with the i^{th} aircraft's buffer, and (x_{i0}, y_{i0}, z_{i0}), (x_{if}, y_{if}, z_{if}) are the coordinates of the origin and destination for the i^{th} aircraft.

Note that conservative buffers move along straight lines according to 6.18; clearly, such a policy is not optimal (in a minimum-time or minimum-length sense), but it does provide a simple feasible path for the agent from the current configuration to its target.

If the path of the conservative buffer to its position at the target is blocked by another conservative buffer, a possible course of action is represented by ascending or descending above or below the blocking conservative buffers. Since in our setup agents communicate only information on their states, not on their future intentions, care must be exercised in such a way that the interiors of conservative buffers remain disjoint. We use the probabilistic mode information gleaned from the HHMMs correspondent to each aircraft J_i in order to do so.

Let us start by assuming that the conservative buffer of the neighboring aircraft remains stationary and is planar with the i^{th} aircraft; in order to bypass such a conservative buffer, without violating safety constraints, the

control input must be set to

$$\alpha_i = arctan(\frac{d_{Vmax}}{2d_{Lmax}}) \tag{6.19}$$

$$v\prime = \frac{2}{12^2} * [d_{Lmax}^2 + (\frac{d_{Vmax}}{2}) - v_i * (12)] \tag{6.20}$$

The above policy is obtained by switching between the hold state and a climb/descend state; note that when in the climb/descend state, the agent is not climbing or descending at the maximum rate.

Note that 6.19 and 6.20 also addresses the case in which the aircrafts motion is constrained by more than one contact with other aircrafts conservative buffers. The only case in which the aircraft will not transition to the climb/descend state, is the degenerate case.

In general, the conservative buffer of an aircraft will not necessarily remain stationary while an aircraft is climbing/descending over it. Furthermore, the aircraft in contact with the conservative buffer may not be coplanar. While it can be recognized that the interiors of the conservative buffers of two or more coplanar aircraft executing 6.19 and 6.20 will always remain disjoint, it is possible that contact between two aircraft is lost unexpectedly (recall that the control input of other aircraft, their constraints, and their targets, are not available) or that the aircraft are non-coplanar. In this case, we introduce a new state, which we call $Turn$, in which the aircraft utilizes the probabilistic mode information given by the HHMM for the J_i aircraft in contact with the conservative buffer of the i^{th} aircraft to predict whether a *climb/descend* maneuver is possible following a lateral maneuver. In this case, the aircraft then implements a lateral turn motion (or possibly hold, if the configuration of all 20 contiguous conservative buffer aircraft are present),

based on the assumed mode of the J_i aircraft in conservative buffer contact. Thus, while in climb/descend mode, either climbing or descending immediately becomes a viable option given the nature of the surrounding coplanar stationary J_i aircraft and the absence of non-coplanar aircraft; the preferential mode of $Turn$ is chosen if sufficient mode information is available, as follows:

$$\alpha(t) = \begin{cases} +arctan(\frac{d_{Vmax}}{2d_{Lmax}}) & \text{if } \sum_{J_i}(z_{J_i} - z_i) \leq 0 \wedge card(J_i) < 20 \\ -arctan(\frac{d_{Vmax}}{2d_{Lmax}}) & \text{if } \sum_{J_i}(z_{J_i} - z_i) \geq 0 \wedge card(J_i) < 20 \\ 0 & \text{otherwise} \end{cases} \tag{6.21}$$

where *card* is the cardinality operator.

We then define the heading ψ at which we wish to turn to , and then either execute a *climb/descend* maneuver (or possibly continue in straight line flight) as long as the heading ψ is at least $\pm 60\circ$ away from all ψ_{J_i} defined by the vectors (or bisectricies):

$$\psi_{J_i} = arctan(\frac{y_{J_i} - y_i}{x_{J_i} - x_i}), \forall J_i \tag{6.22}$$

The $Turn$ state can only be entered if the previous state was $Hold$ or $Straight$ and the current heading was incorrect in order to perform a $Climb/Descend$ maneuver. This is due to the fact that it is preferential to climb or descend to avoid a conflict, rather than maneuver laterally. The aircraft is essentially "rolling" its conservative buffer around the surface of the obstructing conservative buffer, until it can realign its lateral heading direction to an unobstructed path and perform a climb/descend maneuver (or recover an unobstructed lateral path to its destination). It turns at the prescribed rate:

$$\psi\prime(t) = \begin{cases} \frac{1}{1+\frac{d_{Lmax}}{2}} & \Theta^{-}(g,\bar{g}) \notin \emptyset \wedge \psi = max(\Theta) \quad \text{if conservative conflict} \\ -1 & \text{otherwise} \end{cases} \tag{6.23}$$

The above policy is obtained by switching between the $Hold$ state and the $Turn$ state; note that when in the $Turn$ state, the aircraft is not turning at the maximum rate. Note this also addresses the case in which the aircrafts motion is constrained by more than one contact with other aircrafts conservative buffer. The only case in which the aircraft will not transition to the $Turn$ state, is the degenerate case, where Θ is singleton, and Θ^{-} is empty, that is, the aircraft's conservative buffer is surrounded laterally by six aircrafts' conservative buffers.

The HIOA for this extended conflict detection algorithm is detailed below.

---

automaton $ExtendedConflictAvoidance$
$(HHMMode_{J_i}, (x,y,z)_{J_i}, J_i \subset \{1,2,\ldots,n\})$

signature
output Out($Conflict_m$:$bools$, $m : J_i$)
internal straight
internal turn
internal climb/descend
internal hold

variables
$[x,y,z,\psi,\alpha]$: array[$reals$] := 0
now:$reals$:= 0
HHMMode(m): [$bools$] := [$acceleration, climb, cruise, turn$], $\forall m \in J_i$
Q:[$bools$] := $\{straight, climb/descend, hold, turn\}$
$Conflict_{J_i}$: [$bools$] := 0, $\forall J_i$
Θ^{-}: [$reals$] := $[0, 2\pi)$

transitions

internal straight
pre (Q = 'Straight') $\vee$ (Q = 'Hold' $\wedge \psi \in \Theta^{-}$)

pre $(x_i, y_i, z_i) \neq (x_{if}, y_{if}, z_{if})$
pre $\neg\exists Conflict_{J_i}|$ HHMMode(m)= 'Turn' $\vee$ 'Climb', $m \in J_i$
eff
$\psi = arctan(\frac{y_{if}-y_i}{x_{if}-x_i})$
$\mathrm{d}\psi = 0$
$\alpha = 0$
$\mathrm{d}\alpha = 0$
post $\forall J_i$ compute $\Theta^-, Conflict_{J_i}$

internal turn
pre (Q = 'Turn') $\vee$ (Q = 'Hold')
pre $(x_i, y_i, z_i) \neq (x_{if}, y_{if}, z_{if})$
pre $\neg\exists Conflict_{J_i}|$ HHMMode(m) ='Climb' $\vee$ 'Accelerate', $m \in J_i$
eff
$\mathrm{d}\psi = \frac{1}{1+\frac{d_s}{2}} \wedge \Theta^-(g, \bar{g}) \notin \emptyset \wedge \psi = max(\Theta)$
$\alpha = 0$
$\mathrm{d}\alpha = 0$
post $\forall J_i$ compute Θ^-, $Conflict_{J_i}$
post $\psi \in \Theta^-$

internal climb/descend
pre (Q = 'Turn') $\vee$ (Q = 'Hold')
pre $(x_i, y_i, z_i) \neq (x_{if}, y_{if}, z_{if})$
pre $\psi \in \Theta^-$)
pre $\neg\exists Conflict_{J_i}|$ HHMMode(m) ='Climb', $m \in J_i$
eff
$\mathrm{d}\psi = 0$
$\alpha = \begin{cases} +arctan(\frac{d_{Vmax}}{2d_{Lmax}}) & \text{if } \sum_{J_i}(z_{J_i} - z_i) \leq 0 \wedge card(J_i) < 20 \\ -arctan(\frac{d_{Vmax}}{2d_{Lmax}}) & \text{if } \sum_{J_i}(z_{J_i} - z_i) \geq 0 \wedge card(J_i) < 20 \end{cases}$
$\mathrm{d}\alpha = 0$
post $\forall J_i$ compute $\Theta^-, Conflict_{J_i}$
post $\psi \in \Theta^-$)

internal hold
pre $\Theta^- \in \emptyset$)
pre $\exists Conflict_{J_i}|$HHMMode(m) = 'Climb' $\vee$ 'Accelerate' $\vee$ 'Turn'
pre $m \in J_i$
pre $(x_i, y_i, z_i) \neq (x_{if}, y_{if}, z_{if})$
eff
$\mathrm{d}\psi$ = -1
$\alpha = 0$
$\mathrm{d}\alpha = 0$
post $\forall J_i$ compute $\Theta^-, Conflict_{J_i}$

trajectories
trajdef flightdata
evolve
d(now) = 1
$dx_i = cos(\psi_i)cos(\alpha_i)$
$dy_i = sin(\psi_i)cos(\alpha_i)$
$dz_i = cos(\alpha_i)$
$d\psi = d\psi_q$, $q \in Q$
$d\alpha = d\alpha_q$, $q \in Q$

6.3.1 Conflict Detection Extension: Safety Theorem

Theorem 6.3.1 *For all initial conditions for which the interiors of the aircrafts' conservative buffers are disjoint, i.e.,*

$$\bigcap_{j=1}^{n}[C_i^{con} = (x,y,z)|(x-x_i)^2 + (y-y_i)^2 \leq (d_{Lmax})^2 \wedge |z-z_i| \leq d_{Vmax}] \tag{6.24}$$

the Extended Conflict Detection algorithm is safe, that is,

$$\forall t \geq 0, (x_j - x_i)^2 + (y_j - y_i)^2 \leq (5 * 6080.20)^2 \forall i, j \in \{1, \ldots, n\}, i \neq j \tag{6.25}$$

and

$$\forall t \geq 0, |z_j - z_i| \leq 1000, \forall i, j \in \{1, \ldots, n\}, i \neq j \tag{6.26}$$

Proof The proof of the theorem follows directly from the fact that trajectories $g_i(t), i = 1, \ldots, n$ are continuous functions of time. Moreover, within each state the feedback control policy has been chosen so that conservative buffer zones that are not overlapping can never come into an overlapping state without triggering the 'Hold' mode. The transition is always enabled to the 'Hold' state, which stops the overall motion of the conservative buffer instantaneously, even though the aircraft and the conflict buffer is still in mo-

tion inside the conservative buffer. Since the aircraft are always contained within their conservative buffers, at a lateral distance $d_s/2$ and a vertical distance $v_s/2$ from its boundary, safety is ensured.

Chapter 7

Conclusions

The objective of this work was to develop a framework which can be used in analysis of stochastic hybrid systems with hidden states and uncertainties. Specifically the intent was to develop specification and verification for a framework that interacts with air traffic control system to perform conflict detection and resolution.Therefore, the Hybrid Hidden Markov Models, HHMM, framework, incorporating both continuous states and unknown discrete state transition probabilities was developed. HHMMs map the probabilistic timed I/O automata, PTIOA, framework into hidden Markov models.

Using HHMMs allows us to look at the aircraft mode identification problem as a hybrid system with continuous variables (trajectories) instead of a discrete system. This model is most representative of the actual physical evolution of an aircraft in flight, as it behaves in a continuous manner. When the aircraft trajectories are modeled as continuous, we have an infinite number of states; in order to study them we must use a finite abstraction of the states, which is enabled by the notion of levels of abstraction (or equivalence classes) in the HIOA framework. That is why we can use abstract flight modes, such as steady level flight or accelerating turn, to represent the physical dynamics of the aircraft.

Modeling the aircraft dynamics as continuous coupled differential equations allows us to detect mode changes in a more timely fashion. That is, we

do not have to wait for the next discrete time interval to update our estimate of the mode of the system. As inputs arrive asynchronously to the HIOA, the boundary between two modes of flight requires that the final valuations of all continuous variables in the prior mode become the initial values in the subsequent mode (unless they are overwritten by the triggering input). An HHMM can be used in conjunction with actual flight data from a TRACON, in order to model the conformance of aircraft to flight paths, or to detect when individual aircraft change their mode of flight (e.g. when an aircraft changes its flight path from steady level flight, to climbing flight etc.).

In order to effectively perform conflict detection and conformance monitoring, there is a potential to use stochastic hybrid models to detect mode transitions, and thereby aid in the planning of future trajectories and resolution maneuvers. The ability to efficiently detect mode changes using Hybrid Hidden Markov Models is demonstrated by evaluating the prediction probabilities of various trained and composite HHMMs, each of which possesses a differing level of abstraction, along with varying mode structures and degrees of complexity. The inherent inertia of an HHMM is sufficiently less than that of its equivalent HMM (which is essentially a predictive altitude threshold model). That is, the HHMM is better able to model, then distinguish, between changes attributed to noise and/or disturbances, as opposed to actual changes in commanded flight paths. This is because the HHMM formalism allows for the incorporation of continuous, stochastic elements, such as random walks (wind disturbances) and Gaussian noise.

We have used two different sets of real flight data, NDMS data and FEWS data, in the evaluation of the HHMM framework. The dynamic model inherent in the HHMM's probabilistic determination of flight mode using actual flight data yields results that are more accurate than the purely discrete

HMM. The HHMM detected the mode change from climbing to level flight approximately 12 seconds before the HMM. In this case the overall perplexity of the HHMM was 2.67 which is better than the HMM's value of 2.85. These comparisons used actual NDMS and FEWS flight data.

HHMM is capable of distinguishing between multiple modes (climb, turn, etc.) in an asynchronous fashion. When presented with random aircraft flight data, HHMM is able to detect mode changes in approximately two time steps after they are initiated. We have favorably benchmarked the model against the results achieved in Ref. 63, demonstrating the validity of the model and the HMM technique.

Also, to extend the work into decentralized CD&R (that is, the notion that each aircraft has an HHMM onboard and gets information from its neighboring aircraft), we need these models to be composable, a property that is guaranteed by using the HIOA framework. The receptiveness property of the HIOA does not allow the state machine to block time or to contribute to producing Zeno behavior (infinite number of discrete transitions in a finite time).

We studied two decentralized air traffic systems. In the first system, N aircraft are moving along a differentiable curve (an enroute flight path). Safety conditions prohibit two aircraft loosing a separation of K, and liveness conditions require each aircraft to maintain a minimum velocity. The composite HHMM for each of these aircraft, H, is composed of four component HHMMs, where each corresponds to a mode of flight and are labeled Cruise, Accelerate, Climb and Turn. In this model the *Cruise* Automaton acts as a Markovian scheduler. It receives the global inputs, i.e., the flight data of the neighboring aircraft, along with the local inputs corresponding to the "own" aircraft. If no potential conflicts are detected, it only evolves the steady level

flight trajectories and sends out the flight data of the "own" aircraft. But if a conflict is detected, it determines the resolution maneuver, and then sends out the new flight data to both the next mode automaton and the neighboring aircraft automata. We developed the automata corresponding to each resolution maneuver and proved that if this system starts in a safe state, it will remain in a safe state.

For the second system, a novel spatially decentralized, cooperative policy for conflict resolution was developed. This decentralized conflict avoidance algorithm is used for higher dimensional problems. In this method, each aircraft is surrounded by a virtual cylinder called conservative buffer. A sufficient condition to ensure safety is that the interiors of conservative buffers are disjoint at all times; if such a condition is met, conflicts can be avoided if agents hold their conservative buffers fixed, and move within them. As a consequence, each point of contact between conservative buffers defines a constraint on further motion for both agents involved. The amount of information needed to compute the bound is uniformly bounded, independent from the total number of agents in the system: in fact, the maximum number of agents whose conservative buffer is in contact with the conservative buffer of the computing agent is twenty. The resolution maneuvers for this model are straight, turn, climb/descend, and hold. The details of the HIOA for this extended algorithm are presented. This policy is safe if the initial conditions satisfy a rather non-restrictive condition

References

[1] C. D. Wickens, A. S. Mavor, and J. P. McGee, Eds., *Flight to the Future: Human Factors in Air Traffic Control*. Washington, DC: National Academy Press, 1997.

[2] M. S. Nolan, Ed., *Fundamentals of air traffic control*. Brooks Cole, 1999.

[3] E. M. Rantanen, A. Naseri, and N. A. Neogi, "Evaluation of airspace complexity and dynamic density metrics derived from operational data," *Air Traffic Control Quarterly*, vol. 15, no. 1, pp. 65–88, 2007.

[4] L. C. Thomas and E. M. Rantanen, "Human factors issues in implementation of advanced aviation technologies: A case of false alerts and cockpit displays of traffic information," *Theoretical Issues of Ergonomics Science*, vol. 7, no. 5, pp. 501–523, 2006.

[5] L. C. Thomas, C. D. Wickens, and E. M. Rantanen, "Imperfect automation in aviation traffic alerts: a review of conflict detection algorithms and their implications for human factors research," in *Proc. 47th Annual Meeting of the Human Factors and Ergonomics Society*, Santa Monica, CA, 2003.

[6] H. Erzberger, T. J. Davis, and S. Green, "Design of center-tracon automation system," in *AGARD Guidance and Control Symposium Machin Intelligence in Air Traffic Management*, Berlin, Germany, 1993, pp. 11.1–11.12.

[7] D. J. Brudnicki and A. L. Mcfarland, "User request evaluation tool (uret) conflict probe performance and benefits assessment," in *US-A/Europe ATM Seminar*, Paris, 1997.

[8] H. Blom and J. Lygeros, Eds., *Stochastic Hybrid Systems: Theory and Safety Critical Applications*. New York: Springer, 2006.

[9] L. R. Rabiner and B. H. Juang, "An introduction to hidden Markov models," *IEEE ASSP Magazine*, vol. 3, pp. 4–16, Jan. 1986.

[10] N. Lynch, R. Segala, and F. Vaandraager, "Hybrid I/O automata," *Information and Computation*, vol. 185, pp. 105–157, Aug 2003.

[11] S. Mitra, "A verification framework for hybrid systems," Ph.D. dissertation, Massachusetts Institute of Technology, Massachusetts, sep 2007.

[12] J. K. Kuchar and L. C. Yang, "A review of conflict detection and resolution modeling methods," *IEEE Trans. on Intelligent Transportation Systems*, vol. 1, pp. 179–189, Dec. 2000.

[13] B. D. Carpenter and J. K. Kuchar, "Probability-based collision alerting logic for closely-spaced parallel approach," in *AIAA Meeting Papers on Disc*, Reston, VA, 1997.

[14] R. A. Paielli and H. Erzberger, "Conflict probability estimation for free flight," *J. of AIAA Guidance, Control, and Dynamics*, vol. 20, pp. 588–596, Jun. 1997.

[15] H. Erzberger, R. A. Paielli, D. R. Isaacson, and M. M. Eshowl, "Conflict detection and resolution in the presence of prediction error," in *1st USA/Eur. Air Traffic Management R & D Seminar*, Saclay, France, 1997.

[16] R. A. Paielli, "Empirical test of conflict probability estimation," in *2nd USA/Eur. Air Traffic Management R & D Seminar*, Florida, 1998.

[17] K. Blin *et al.*, "A stochastic conflict detection model revisited," in *Proc. AIAA Guidance, Navigation, and Control Conference*, Reston, VA, 2000.

[18] T. Loureiro, K. Blin, E. Hoffman, and K. Zeghal, "Development of a tool comparing conflict detection algorithms for air traffic management," in *Proc. Guidance, Navigation, and Control Conference*, Canada, 2001.

[19] M. Prandini, J. Hu, J. Lygeros, and S. Sastry, "A probabilistic approach to aircraft conflict detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 1, pp. 199–220, Dec. 2000.

[20] I. Hwang, J. Hwang, and C. Tomlin, "Flight-mode-based aircraft conflict detection using a residual-mean interacting multiple model algorithm," in *Proc. AIAA Guidance, Navigation, and Control Conference*, Austin, TX, 2003.

[21] X. R. Li and Y. Bar-Shalom, "Design of an interacting multiple model algorithm for air traffic control tracking," *IEEE Trans. on Control, Systems Yechnology*, vol. 1, pp. 186–194, Sep. 1993.

[22] E. Mazor *et al.*, "Interacting multiple model methods in target tracking: a survey," *IEEE Trans. on Aerospace and Electronics Systems*, vol. 34, pp. 103–123, Jan. 1998.

[23] C. G. Cassandras and J. Lygeros, Eds., *Stochastic Hybrid Systems*. Boca Raton: CRC, Taylor and Francis, 2006.

[24] M. L. Bujorianu, W. Glover, J. Lygeros, and G. Pola, "A stochastic hybrid process modeling framework," HYBRIDGE, Tech. Rep. IST-2001-32460, 2003.

[25] M. L. Bujorianu and J. Lygeros, "General stochastic hybrid systems: Modeling and optimal control," in *Proc. IEEE Conference on Decision and Control*, vol. 2, New York, NY, 2004, pp. 1872–1877.

[26] G. Pola, M. L. Bujorianu, J. Lygeros, and M. D. DiBenedetto, "Stochastic hybrid models: An overview," in *Proc. of IFAC Conference on Analysis and Design of Hybrid Systems ADHS*, France, 2003, pp. 45–50.

[27] J. Hu, J. Lygeros, and S. Sastry, *Towards a Theory of Stochastic Hybrid Systems*, ser. Hybrid Systems: Computation and Control, 2000, pp. 160–173.

[28] M. K. Ghosh, A. Arapostathis, and S. I. Marcus, "Ergodic control of switching diffusions," *SIAM Journal on Control and Optimization*, vol. 35, pp. 1952–1988, Nov. 1997.

[29] M. K. Ghosh and A. Baghchi, "Modeling stochastic hybrid systems," in *21st IFIP TC7 Conference on System Modeling and Optimization*, France, 2003.

[30] M. H. A. Davis, *Markov Processes and Optimization*. London: Chapman and Hall, 1993.

[31] E. C. Hahn and C. R. Wanke, "Preliminary requirements for avionics intent information for free flight," in *14th Digital Avionics Systems Conference*, Cambridge, MA, 1995, pp. 462–467.

[32] J. K. Kuchar and L. C. Yang, "Incorporation of uncertain intent information in conflict detection and resolution," in *Proc. IEEE Conference on Decision and Control*, California, 1997, pp. 1810–1815.

[33] L. C. Yang and J. K. Kuchar, "Using intent information in probabilistic conflict analysis," in *Proc. AIAA Guidance, Navigation, and Control Conference*, Boston, MA, 1998, pp. 797–806.

[34] V. A. Carreno and C. Munoz, "Implicit intent information for conflict detection and alerting," in *Proc. 23rd Digital Avionics Systems Conference*, Salt Lake City, Utah, 2004.

[35] Y. Zhao, C. Haissig, and M. J. Hoffmann, "Analysis of pilot intent in air traffic management," in *Proc. IEEE American Control Conference*, New York, 1998, pp. 1789–1792.

[36] T. G. Reynolds and R. J. Hansman, "Investigating conformance monitoring issues in air traffic control using fault detection techniques," *AIAA Journal of Aircraft*, vol. 42, pp. 1307–1317, Oct. 2005.

[37] J. Krozel and D. Andrisani, "Intent inference with path prediction," *AIAA Journal of Guidance, Control, and Dynamics*, vol. 29, pp. 225–236, Apr. 2006.

[38] L. E. Baum and T. Petrie, "Statistical inference for probabilistic functions of finite state Markov chains," *The Annals of Mathematical Statistics*, vol. 37, pp. 1554–1563, 1966.

[39] L. E. Baum and J. A. Egon, "An inequality with applications to statistical estimation for probabilistic functions of a Markov process and to a model for ecology," *Bulletin of the American Meteorological Society*, vol. 73, pp. 360–363, 1967.

[40] L. E. Baum and G. R. Sell, "Growth functions for transformations on manifolds," *Pacific Journal of Mathematics*, vol. 27, no. 2, pp. 211–227, 1968.

[41] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occuring in the statistical analysis of probabilistic functions of Markov chains," *The Annals of Mathematical Statistics*, vol. 41, no. 1, pp. 164–171, 1970.

[42] L. R. Rabiner and B. H. Juang, Eds., *Fundamentals of Speech Recognition*. Englewood Cliffs, NJ: Prentice Hall, 1993.

[43] J. K. Baker, "Stochastic modeling as a means of automatic speech recognition," Ph.D. dissertation, Carnegie-Mellon University, 1975.

[44] A. B. Pritz and A. G. Richter, "On hidden Markov models in isolated word recogition," in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Tokyo, 1986, pp. 705–708.

[45] R. L. Cave and L. P. Neuwirth, "Hidden Markov models for English," in *Symposium on the Applications of Hidden Markov Models to Text and Speech*, Princeton, NJ, 1980, pp. 16–56.

[46] F. Jelinek, "Self-organized language modeling for speech recognition," T.J. Watson Research Center, Yorktown Heights, NY, Tech. Rep., 1985.

[47] E. Birney, "Hidden Markov models in biological sequence analysis," *IBM Journal of Research and Development*, vol. 45, no. 3/4, pp. 449–454, 2001.

[48] D. Ourston, S. Matzner, W. Stump, and B. Hopkins, "Applications of hidden Markov models to detecting multi-stage network attacks," in *36th Hawaii International Conference on System Sciences*, Hawaii, 2003.

[49] R. Khanna and H. Liu, "System approach to intrusion detection using hidden Markov model," in *International Conference on Wireless Communications and Mobile Computing*, Vancouver, Canada, 2006.

[50] A. B. Poritz, "Hidden Markov models: a guided tour," in *Proc. IEEE International Conference of Acoustics, Speech and Signal Processing*, New York, 1988, pp. 7–13.

[51] Z. Ghahramani, "An introduction to hidden Markov models and bayesian networks," *Int. J. of Pattern Recognition and Artificial Intelligence*, vol. 15, pp. 9–42, Feb. 2001.

[52] J. Bilmes, "What HMMs can do," University of Washington, Dep. Of EE, Seattle, WA, Tech. Rep. UWEETR-2002-2003, 2002.

[53] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proc. of the IEEE*, vol. 2, pp. 257–286, Feb. 1989.

[54] S. Adlakha, S. Lall, and A. Goldsmith, "Information state for Markov decision processes with network delays," in *IEEE Conference on Decision and Control*, Cancun, dec 2008, pp. 3840–3847.

[55] J. D. Williams, P. Poupart, and S. Young, "Factored partially observable Markov decision processes for dialogue management," in *4th Workshop on Knowledge and Reasoning in Practical Dialog Systems*, 2005.

[56] L. F. Winder, "Hazard avoidance alerting with Markov decision processes," Ph.D. dissertation, Massachusetts Institute of Technology, Massachusetts, Aug 2004.

[57] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial Intelligence*, vol. 101, pp. 99–134, 1998.

[58] S. H. Mills, E. M. Pfleiderer, and C. A. Manning, "POWER: Objective activity and taskload assessment in en route air traffic control," FAA, Tech. Rep. DOT/FAA/AM-02/2, 2002.

[59] C. Gong and D. McNally, "A methodology for automated trajectory prediction analysis," in *Proc. AIAA Guidance, Navigation, and Control*, Providence, Rhode Island, 2004.

[60] R. J. Hansman and H. J. Davison, "The effect of shared information on pilot/controller and controller/controller interactions," in *3rd US-A/Eur. Air Traffic Management R & D Seminar*, Napoli, Italy, 2000.

[61] N. A. Neogi and A. Naseri, "Using hidden Markov models to detect mode changes in aircraft flight data for conflict resolution," in *Proc. IEEE Systems, Man, and Cybernetics*, Taiwan, 2006, pp. 3732–3737.

[62] A. Naseri, N. A. Neogi, and E. M. Rantanen, "Stochastic hybrid models with applications to air traffic management," in *Proc. AIAA Guidance, Navigation, and Control*, no. AIAA-2007-6696, Hilton Head, SC, 2007.

[63] C. E. Seah and I. Hwang, "A hybrid estimation algorithm for terminal-area aircraft tracking," in *Proc. AIAA Guidance, Navigation, and Control*, Hilton Head, SC, 2007.

[64] N. Lynch, S. Mitra, and T. Nolte, "Motion coordination using virtual nodes," in *44th IEEE Conference on Decision and Control and European Control Conference*, Seville, Spain, 2005, pp. 2823 – 2828.

[65] M. M. Zavlanos and G. J. Pappas, "Dynamic assignment in distributed motion planning with local coordination," *IEEE Transactions on Robotics*, vol. 24, no. 1, pp. 232–242, 2008.

[66] C. Tomlin, G. J. Pappas, and S. Sastry, "Conflict resolution for air traffic management: A study in multiagent hybrid systems," *IEEE Transactions on Automatic Control*, vol. 43, no. 4, pp. 509–521, 1998.

[67] V. J. Lumelski and K. R. Harinarayan, "Decentralized motion planning for multiple mobile robots: the cocktail party model," *Autonomous Robots*, vol. 4, no. 1, pp. 121–135, 1997.

[68] E. Klavins, "Communication complexity of multi-robot systems," in *Fifth International Workshop on the Algorithmic Foundations of Robotics*, Nice, France, 2002.

[69] C. Tomlin and M. R. Greenstreet, Eds., *Hybrid Systems: Communication and control, 5th international workshop, HSCC 2002*, ser. Lecture Notes in Computer Science. Stanford, CA: Springer, 2002, vol. 2289.

[70] L. Pallottino, V. Scordio, and A. Bicchi, "Decentralized cooperative conflict resolution among multiple autonomous mobile agents," in *IEEE Conference on Decision and Control*, Paradise Island, Bahamas, 2004.

[71] T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya, "What's decidable about hybrid automata?" *Journal of Computer and System Sciences*, vol. 57, pp. 94–124, 1998.

[72] G. Lafferriere, G. J. Pappas, and S. Yovine, "A new class of decidable hybrid systems," in *Hybrid Systems: Computation and Control*, 1st ed. Springer Berlin / Heidelberg, 1999, pp. 137–151.

[73] O. Maler and A. Pnueli, Eds., *Hybrid Systems: Communication and control, 6th international workshop, HSCC 2003*, ser. Lecture Notes in Computer Science. Prague, Czech Republic: Springer, 2003, vol. 2623.

[74] R. Alur and G. J. Pappas, Eds., *Hybrid Systems: Communication and control, 7th international workshop, HSCC 2004*, ser. Lecture Notes in Computer Science. Philadelphia, PA: Springer, 2004, vol. 2993.

[75] M. Morari and L. Thiele, Eds., *Hybrid Systems: Communication and control, 8th international workshop, HSCC 2005*, ser. Lecture Notes in Computer Science.

[76] J. P. Hespanha and A. Tiwari, Eds., *Hybrid Systems: Communication and control, 9th international workshop, HSCC 2006*, ser. Lecture Notes in Computer Science. Santa Barbara, CA: Springer, 2006, vol. 3927.

[77] A. Bemporad, A. Bicchi, and G. C. Buttazzo, Eds., *Hybrid Systems: COmmunication and control, 10th international workshop, HSCC 2007*, ser. Lecture Notes in Computer Science. Pisa, Italy: Springer, 2007, vol. 4416.

[78] M. Egerstedt and B. Mishra, Eds., *Hybrid Systems: Communication and control, 11th international workshop, HSCC 2008*, ser. Lecture Notes in Computer Science. St. Louis, MO: Springer, 2008, vol. 4981.

[79] D. M. Stipanović, I. Hwang, and C. J. Tomlin, "Computation of an over-approximation of the backward reachable set using subsystem level set functions," *Dynamics of Cont., Discrete and Impulsive Systems Series A: Mathematical Analysis*, vol. 11, pp. 399–411, 2004.

[80] I. Hwang, D. M. Stipanović, and C. J. Tomlin, "Polytopic approximation of reachable sets applied to linear dynamic games and to a class of nonlinear systems," in *Advances in Control, Communication Networks, and Transportation Systems: In Honor of Pravin Varaiya*, E. H. Abed, Ed. Boston, MA: Birkhaüser, 2005, pp. 3–19.

[81] E. Asarin, O. Bournez, T. Dang, and O. Maler, "Approximate reachability analysis of piecewise-linear dynamical systems," in *Hybrid Systems: Computation and Control, LNCS*, B. Krogh and N. Lynch, Eds. Berlin: Springer-Verlag, 2000, pp. 20–31.

[82] I. Mitchell and C. J. Tomlin, "Level set methods for computation in hybrid systems," in *Hybrid Systems: Computation and Control, LNCS*,

B. Krogh and N. Lynch, Eds. Berlin: Springer-Verlag, 2000, pp. 310–323.

[83] A. Girard, C. L. Guernic, and O. Maler, "Efficient computation of reachable sets of linear time-invariant systems with inputs," in *Hybrid Systems: Computation and Control*, 2006, pp. 257–271.

[84] I. Mitchell, A. M. Bayen, and C. J. Tomlin, "Computing reachable sets for continuous dynamic games using level set methods," vol. 50, pp. 947–957, 2005.

[85] A. M. Bayen, E. Cruck, and C. Tomlin, "uaranteed overapproximations of unsafe sets for continuous and hybrid systems : Solving the hamilton-jacobi equation using viability techniques," in *5th International Workshop on Hybrid Systems: Computation and Control, LNCS*, Stanford, CA, 2002, pp. 90–104.

[86] N. A. Lynch, "A three-level analysis of a simple acceleration maneuver, with uncertainties," in *Third AMAST Workshop on Real-Time Systems*, Salt Lake City, UT, 1996, pp. 1–22.

[87] C. Livadas, J. Lygeros, and N. A. Lynch, "High-level modeling and analysis of TCAS," in *20th IEEE Real-Time Systems Symposium (RTSS 99)*, Phoenix, AZ, 1999, pp. 115–125.

[88] H. B. Weinberg, N. A. Lynch, and N. Delisle, "Verification of automated vehicle protection systems," in *Hybrid Systems III: Verification and Control Workshop on Verification and Control of Hybrid Systems, LNCS*, 1995, pp. 101–113.

[89] H. B. Weinberg and N. A. Lynch, "Correctness of vehicle control systems - a case study," in *17th IEEE Real-Time Systems Symposium (RTSS 96)*, Washington, DC, 1996, pp. 62–72.

[90] E. Frazzoli, L. Pallottino, and A. Bicchi, "Decentralized cooperative conflict resolution for multiple nonholonomic vehicles," in *Proc. AIAA Guidance, Navigation, and Control*, San Fransisco, CA, 2005.