HAZARD AVOIDANCE ALERTING WITH MARKOV DECISION PROCESSES

Lee F. Winder and James K. Kuchar

Report No. ICAT-2004-4
August 2004

MIT International Center for Air Transportation
Department of Aeronautics & Astronautics
Massachusetts Institute of Technology
Cambridge, MA 02139 USA

HAZARD AVOIDANCE ALERTING WITH MARKOV DECISION PROCESSES

BY

LEE F. WINDER AND JAMES K. KUCHAR

ABSTRACT

This thesis describes an approach to designing hazard avoidance alerting systems based on a Markov decision process (MDP) model of the alerting process, and shows its benefits over standard design methods. One benefit of the MDP method is that it accounts for future decision opportunities when choosing whether or not to alert, or in determining resolution guidance. Another benefit is that it provides a means of modeling uncertain state information, such as knowledge about unmeasurable mode variables, so that decisions are more informed.

A mode variable is an index for distinct types of behavior that a system exhibits at different times. For example, in many situations normal system behavior is safe, but rare deviations from the normal increase the likelihood of a harmful incident. Accurate modeling of mode information is needed to minimize alerting system errors such as unnecessary or late alerts.

The benefits of the method are illustrated with two alerting scenarios where a pair of aircraft must avoid collisions when passing one another. The first scenario has a fully observable state and the second includes an uncertain mode describing whether an intruder aircraft levels off safely above the evader or is in a hazardous blunder mode.

In MDP theory, outcome preferences are described in terms of utilities of different state trajectories. In keeping with this, alerting system requirements are stated in the form of a reward function. This is then used with probabilistic dynamic and sensor models to compute an alerting logic (policy) that maximizes expected utility. Performance comparisons are made between the MDP-based logics and alternate logics generated with current methods. It is found that in terms of traditional performance measures (incident rate and unnecessary alert rate), the MDP-based logic can meet or exceed that of alternate logics.

This document is based on the thesis of Lee F. Winder submitted to the Department of Aeronautics and Astronautics at the Massachusetts Institute of Technology in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Aeronautics and Astronautics.

Acknowledgements

This work was supported by the National Aeronautics and Space Administration Ames Research Center under grant NAG2-1425 and by the FAA/NASA Joint University Program.

Contents

Nomenclature

Functions

$\mathbf{b_s}$	Belief state
E	Expected value
O	Sensor function
P	Probability
R	Reward function
T	State transition function
U	State expected utility
U_τ	Trajectory utility

Scalar Parameters and Variables

a	Alerting system action
a_0	Deferral (nominal) action
a_p	Previous action
AS	Alert status index
AS_p	Previous alert status
BS	Belief state index
CS	Combination state index
D	Nominal vertical level-off separation
c	Number of policy or value iterations
γ	Reward discounting factor
k	Discrete time index
m	Mode variable, also number of utility recursions
m_i	Index of the i^{th} mode
n	Number of states in situation state space, S
q	Number of actions in alert action space, A
t	Time
t_c	Current time
Δt	Time increment
U_{CR}	Correct rejection utility
U_{SA}	Successful alert utility
U_{UA}	Unnecessary alert utility
v_{climb}	Constant evasion climb rate
$v_{closing}$	Constant horizontal closing rate
$v_{descent}$	Constant nominal intruder descent rate
v_w	Climb rate disturbance input
x	Horizontal separation
x_f	Final horizontal separation
x_p	Previous horizontal separation
Δx	Horizontal position increment
y	Vertical separation

y_f	Final vertical separation
y_p	Previous vertical separation

Sets

A	Alert space
H	Hazard space
O	Observation space
S	Situation space

Trajectories

τ	A trajectory in S
τ_a	A trajectory of alerting actions

Vectors

$\mathbf{a}$	Vector alert action signal
$\mathbf{a}_k$	k^{th} alert action
$\mathbf{d}_h$	Disturbance input to humans
$\mathbf{d}_p$	Disturbance input to plant
$\mathbf{n}_s$	Sensor noise input
$\mathbf{o}$	Alerting system observations of situation
$\mathbf{s}$	Situation state
$\mathbf{u}$	Human input to plant
$\mathbf{x}_a$	Alerting system state
$\mathbf{x}_h$	Human state
$\mathbf{x}_p$	Plant state
$\mathbf{y}$	Human observations of plant

Table of Figures

1. Introduction

1.1 Hazard Avoidance Alerting

Automatic alerting systems are often included in situations where humans interact with machines, such as in vehicles and industrial processes, and unnoticed failures can have serious consequences. An alerting system monitors a situation and, when necessary, generates alerts and other signals to prevent an undesirable incident. It can be as simple as an alarm triggered by an out-of-range sensor reading, but in recent years more complex alerting systems have appeared that gather comprehensive information, use sophisticated decision algorithms, and provide guidance to aid the human operator after the initial alert. Such systems are possible thanks to increasing availability of the needed sensor, communications and computer technology. Examples of successful alerting systems are the mid-air collision (Harman, 1989; O'Hara, 1998; Nordwall, 2002) and terrain (Phillips, 2001; Feith, 2002) avoidance systems installed in many aircraft cockpits. New alerting systems are frequently created or proposed, especially in aviation where they help counteract the negative safety effects of increasing use of airports and airspace, and reduced separation requirements (Scott, 2001; Jones, 2002; Cassell et al, 2001; Carpenter & Kuchar, 1997; Samanant et al, 2000; Zhao & Rock, 2002).

An alerting system takes a stream of state measurements as input, and by some mathematical criteria or logic decides whether or not to alert. If an alert has already happened, the logic decides what additional cues to give the operator. Designing the logic traditionally means choosing a candidate logic, evaluating its performance with simulated scenarios, making changes to fix problems, and repeating the process. The final logic may be significantly different or more complicated than the original due to cumulative improvements.

The initial candidate logic can be chosen in different ways. One common approach is to think of the threshold as a boundary between normal and abnormal system behavior, so the alert indicates a loss of conformance. An example is the "rumble strips" along the edges of many highways that alert a driver whose vehicle may be drifting off of the road. A more complex approach to alerting is to predict the trajectory of state variables and issue an alert if the future proximity of the hazard calls for it. Typically these predictive alerting logics use a rough trajectory model (e.g. not taking prediction uncertainties into account) in conjunction with a state space hazard model and limits on prediction time to avoid alerting too early. Though reasoning explicitly about the potential hazard, such a logic still requires performance analysis and tuning, since the model only roughly describes the conditions justifying an alert (Yang & Kuchar, 2002). Appendix B has a more complete discussion of these different philosophies of alerting.

Recent research has produced design methods where alerting decisions are made based on metrics of predicted decision performance at that time, rather than on rough criteria (Yang & Kuchar, 2002; Kuchar & Yang, 2000). The aim is to reduce the need for trial and error adjustment of a logic by more clearly stating the alerting requirements, and making more informed alerting decisions in the knowledge of these. In systems with random dynamics, where only uncertain state predictions are possible, the required decision metrics tend to be probabilistic quantities (e.g. incident frequencies or probabilities of future incidents).

Motivated by two new design considerations to be introduced next, this thesis continues in the vein of probabilistic prediction-based alerting, with the goals of improving alerting system performance and providing insight into alerting problems.

1.2 Modes and Future Decisions

In many cases a specific event can be identified (such as equipment failure or an operator error) as triggering a change in the dynamics of an observed human-controlled system, so that different models describe its behavior better at different times. In

particular, the event may mark a change from safe to unsafe system behavior. An example is a vehicle on a highway becoming dangerous after its driver falls asleep.

Distinct dynamic behaviors a system can exhibit are termed *modes*. Figure 1.1 illustrates the mode idea for an aircraft in a situation where it could either crash into a mountain top (1) or climb safely over it (2), depending on the vertical path mode being

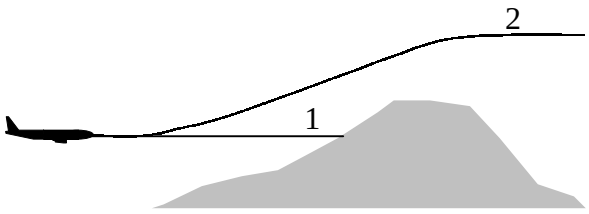


Figure 1.1: Uncertain Situation Modes

used by the aircraft's autoflight system. Uncertainty about which mode the aircraft is in makes it unclear whether an alert is needed: alerting with mode 2 might cause an irritating, unnecessary alert if the human is already aware of the mountain, but failing to alert with mode 1 could allow a crash. Due to the general difficulty of avoiding all such errors in the presence of uncertainty (Kuchar, 1996), the alerting system may be forced to weigh the costs of different errors and make a probability trade-off between the two. Or the alerting system might take steps to reduce the mode uncertainty to allow more exact predictions and better decisions. One objective of this thesis is to discuss the modeling of mode uncertainty, and show the benefits of a Bayesian probabilistic approach to mode uncertainty modeling.

A related issue is how knowledge about the system dynamics is used to achieve the goals of alerting. This includes taking into account for the current alerting decision the fact that there will be new information available in the future, and future choices to make. Figure 1.2 illustrates this issue, again using the aircraft and terrain example. Initially the alerting system is uncertain about the operating mode of the aircraft and it must decide whether or not to alert, triggering a climb avoidance maneuver, or defer the alert for possible use later. If it issues an alert immediately the alert would be safe but

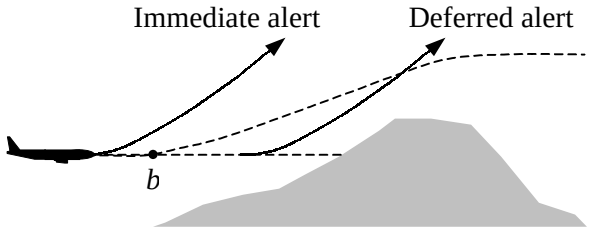


Figure 1.2: Anticipating New Information and Decision Opportunities

maybe unnecessary. Because future observations will reduce uncertainty about the mode, deferring the alert would clarify the need for an alert once the path branching point *b* is passed. The path taken would make the actual mode apparent. If the collision path were taken, then alerting would not risk an unnecessary alert. If deferring would also leave time for a safe avoidance maneuver, then it makes sense to defer the alert. Typically, however, there is no clear branching point like point *b* in actual situations, making the decision to defer alerting more challenging. There is an interesting balance between acting early on uncertain information, ensuring safety but maybe unnecessarily, versus delaying action to better know the mode being used and loosing flexibility for avoiding a threat. Investigating the benefits of such reasoning in an alerting system, but in a more general probabilistic framework, is the second thesis objective.

1.3 Thesis Overview

This thesis presents a new methodology for the design of hazard avoidance alerting, motivated by certain needs not yet addressed by existing design and analysis methods. One need is to incorporate reasoning about future decision opportunities into an alerting and guidance logic. Another is to include reasoning about uncertain, unobservable dynamic modes in the logic. In both cases there is also a need to determine any benefits of such information. Finally, there must be an approach to using the desired information in a way that is efficient enough that alerting decisions can be made in real time.

Chapter 2 gives an overview of probabilistic prediction-based alerting and provides more detailed motivation for the two new considerations. This includes

formally defining model components and showing how different metrics of performance are used for defining and analyzing alerting thresholds.

Chapter 3 contains a review of related research, pointing out similarities to other work and significant differences that make this work necessary.

Chapter 4 covers the basics of Markov decision process (MDP) theory in the context of alerting, using model components defined in chapter 2. After formulating the problem, some basic methods of deriving efficient "policy" solutions are described. After this the use of partially observable Markov decision processes (POMDPs) to model uncertain modes is discussed, along with a basic POMDP solution method. Finally, some considerations about humans in the system are discussed.

Chapter 5 uses an aircraft collision avoidance alerting process to illustrate MDP alerting concepts. A policy is derived that defines the alerting threshold and also generates guidance to maximize safety after the alert. The modeling of alerting preferences in terms of a reward function is shown. Finally, the need for average as well as threshold performance metrics (utilities) is discussed.

Chapter 6 uses a more complex, 2-mode collision avoidance alerting process to look at modeling uncertain modes and updating of uncertain knowledge based on observations. The benefit of such modeling is shown using specific scenarios and metrics of average performance. Finally, the use of belief state domain simplifications is discussed.

In Chapter 7 the main conclusions and contributions are summarized.

2. Probabilistic Alerting Overview and Thesis Motivation

This chapter describes in detail two new concepts for guiding the design of probabilistic alerting systems. One is belief modeling of uncertain mode variables. This is a method for describing uncertainty about which of multiple dynamic modes an observed system is in. Another is anticipating future information gathering and decision opportunities when choosing an alerting system action. Section 2.1 describes the alerting process and modeling assumptions in general. Then section 2.2 covers the different measures of alerting system performance. In section 2.3 the use of probabilistic trajectory prediction to define the alerting logic and achieve specific performance requirements is discussed. This leads into an argument in section 2.4 for the importance of mode belief modeling and for considering future decisions when making the current alerting decision.

2.1 Probabilistic Alerting Process Model[1]

2.1.1 Components and Influences

Figure 2.1 depicts the alerting process using the notation of multivariable systems theory. Blocks represent physical elements, and arrows indicate the influences they have on one another. An element that changes with time is characterized by a set of variables called the *state vector* or *state* of the element. For example, the state of an n-variable element at time t is:

[1] The process and hazard model and the terminology described in this chapter are similar to those used by Kuchar (Kuchar, 1995; Kuchar, 1996).

$$\mathbf{x}(t) = [\, x_1(t) \quad x_2(t) \quad \ldots \quad x_n(t) \,]^T \tag{2.1}$$

Scalars, including time and the state variables, are italicized, while vectors are in boldface. This is the convention throughout this thesis. Another notational practice is to sometimes truncate the labels of parameters with understood dependencies: e.g. $\mathbf{x}(t)$ represented by $\mathbf{x}$.

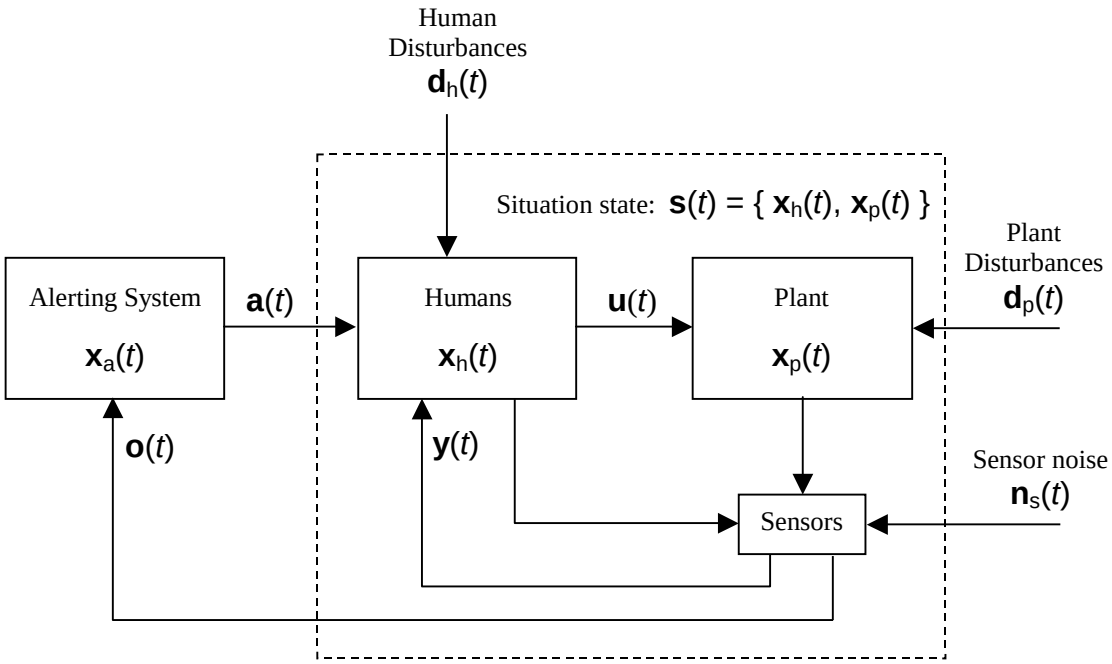


Figure 2.1: Alerting Process

The alerting system, represented by the leftmost element, is a dynamic system with a state, $\mathbf{x}_a$.[2] It makes observations $\mathbf{o}$ and sends inputs $\mathbf{a}$ to some *situation* composed of humans and a *plant* they control. The humans and plant are both dynamic systems with state vectors $\mathbf{x}_h$ and $\mathbf{x}_p$ respectively. The situation is itself a dynamic system with an overall state vector $\mathbf{s}$, made up of the human and plant states. For a physical example, the situation might be highway traffic, including multiple human drivers, their vehicles, and the road they travel on. In that case, the humans are in one block, and the vehicles and road form the plant.

[2] For example, the alerting system state could be an estimate of the current situation state.

The humans make observations **y** of the plant and perform actions **u** to control it. The observations **y** and **o** are the output of a sensors element. This element is a function of **s** that describes which part of the plant state is actually observable to the humans and the alerting system. Sensors may also provide information about the humans themselves, as shown. The humans receive an alerting system signal **a**, which can include both discrete alert values (alerts in the conventional sense) and any continuous resolution guidance the alerting system might provide.

Imperfections in the situation model result in observed state dynamics different from what is expected from the model. Such errors are represented in the diagram by disturbance and noise inputs. The human and plant models are subject to disturbances, $\mathbf{d}_h$ and $\mathbf{d}_p$ respectively, each a vector the same size as the corresponding state. The sensor outputs **y** and **o** are corrupted by random noise that can include bias and high frequency noise components. This is shown as the input vector $\mathbf{n}_s$, which is the combined size of the two sensor outputs.

Note that even though humans are ultimately controlling the plant, the model presented here focuses on the effects of the alerting system on the overall situation (human and plant). In a sense, from the alerting system's point of view, the human and plant can be collectively considered as the controlled element. The design issues then revolve around how to design the alerting system so that the human and plant system responds in an acceptable manner.

2.1.2 Incidents and Hazards

The state **s** changes with time according to the internal situation dynamics and the effects of any alerting system input to the situation. The purpose of the alerting system is to protect against an *incident* by influencing the path **s** takes in its space, *S*. The space *S* is the set of all possible states and is termed the *situation space*. An incident is some negative event that can happen within the situation, such as an accident, rule violation or mechanical failure.

As the state changes with time, there is some probability of an incident happening over a given interval traveled. The probability is different for different paths that **s** can take, as figure 2.2 shows for a 2-dimensional state space. From the current state $\mathbf{s}(t_c)$

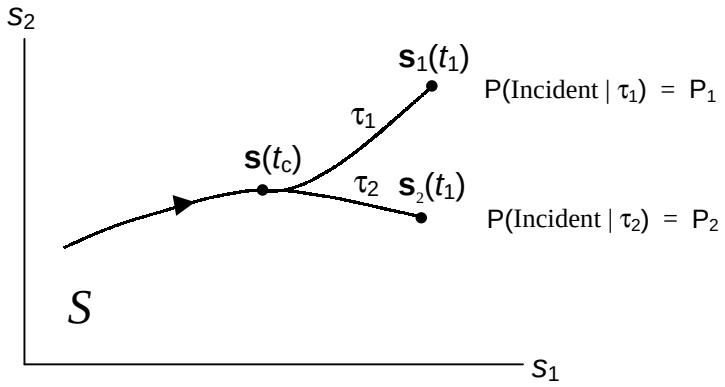


Figure 2.2: Incident Dependence on the Situation Trajectory

two future paths (τ_1 and τ_2) are possible, beginning at the current state and continuing through time t_1. Each path has a different incident probability. A complete description of the situation must include a model specifying these probabilities. The model should give incident probabilities for any possible future path segment beginning at any possible **s**. A common approach is to specify a hazardous region or *hazard space* within *S*, and assume an incident happens with probability 1 if the state trajectory enters hazard space, and 0 otherwise. Figure 2.3 illustrates this kind of incident model in the 2-variable state

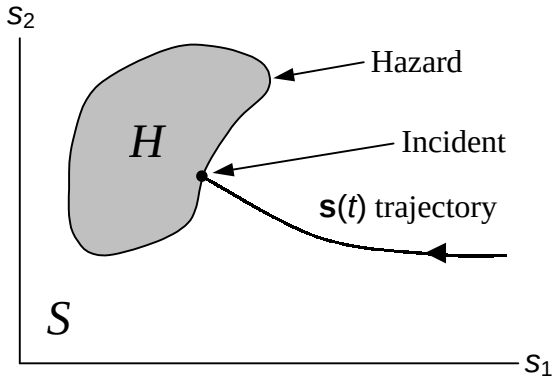


Figure 2.3: Example Incident Model

space, with *H* labeling the hazard space. These incident modeling issues were identified and discussed in more detail by Kuchar (Kuchar, 1995).

2.1.3 Situation Dynamics

The situation is assumed to have uncertain dynamics. That is, knowledge of $\mathbf{s}$ at a given time does not allow an exact prediction of future states. It is further assumed that the uncertainty is amenable to probabilistic modeling. In terms of the variables already defined, this means that, given $\mathbf{s}$ at the current time t_c, the state after some time interval Δt for a known control input τ_a (a trajectory of $\mathbf{a}(t)$) over that interval is known within some probability distribution over the states in *S*. The dynamics can be expressed as a distribution $\mathsf{T}(\cdot,\cdot)$ conditioned on the initial state and control trajectory:

$$\mathbf{s}(t_c+\Delta t) = \mathsf{T}(\mathbf{s}(t_c), \tau_a) \tag{2.2}$$

This future-state distribution is called the *transition function* (Russell & Norvig). Note that the distribution of the future state has no dependence on states prior to $\mathbf{s}(t_c)$. In other words, the assumption is that no information affecting the prediction is lost by forgetting past states. By satisfying this condition, the state $\mathbf{s}$ is said to exhibit the *Markov property*, in the probabilistic sense of the term (Russell & Norvig).

The value of $\mathbf{a}(t)$, chosen by the alerting system, could be a real-valued vector in general, but in practice is sometimes limited to a small set of possibilities:

$$\mathbf{a}(t) \in \{ a_0, a_1, a_2, \ldots a_n \} \tag{2.3}$$

This may be preferable or necessary due to limitations of the alerting display, or in a human's ability to track and follow a command signal (Wickens, 1992). For example, in an impending car accident scenario where the time to predicted collision is short, a hard braking command might be more realistic than complicated instructions to steer around an obstacle. Since a hazard alerting system is meant for rare interventions only, one possible action is the *nominal* or *deferral* action, which is really the lack of an alert. The deferral action is represented by a_0.

Figure 2.4 illustrates probabilistic prediction uncertainty for a situation with a known current state and some implied control signal beginning at t_c. While $\mathbf{s}(t_c)$ is known exactly, $\mathbf{s}(t_c+\Delta t)$ has a distribution of possible states, with darker shading indicating a greater likelihood of the state being at that point. An effect of the trajectory uncertainty is that whether or not future events happen—such as crossing through hazard space—is also uncertain. For the process shown, there is some probability of the state being inside *H* at each point in time, and a probability of **s** passing through *H* over all time. These could be calculated numerically, such as with Monte Carlo simulation, if not analytically.

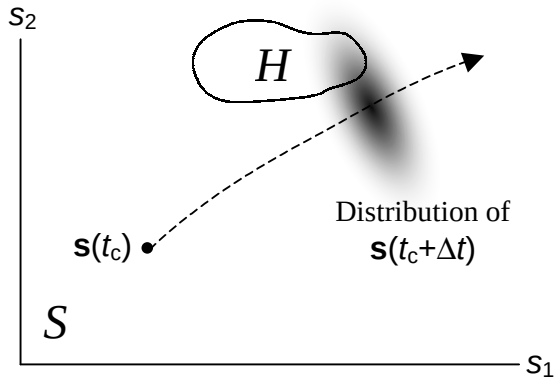


Figure 2.4: Probabilistic Prediction Uncertainty

The probabilistic dynamics and Markov state assumptions are consistent with existing methods of design and analysis of alerting systems (Carpenter & Kuchar, 1997; Yang & Kuchar, 1997, 2002; Kuchar, 1996). While such modeling applies to alerting logic analysis, more relevant to this thesis are design methods where the alerting decision logic is defined in terms of decision metrics computed from a probabilistic prediction model. The reasoning behind such an approach is that it can lead more directly to a logic with the desired performance than trial and error design methods (Yang & Kuchar, 2002).

2.1.4 Situation Observability

To determine how decision options will affect future events, the alerting system uses available knowledge about the current situation state. It may not know the exact state, but have limited information that some states are more likely than others. The less

uncertainty there is, the more accurate predictions can be and the more effective its decisions are likely to be. When state uncertainty is large, the alerting system can reduce it by incorporating information from new observations **o** of the situation into its existing knowledge. An *observation* is a direct measurement of the current situation (e.g. reading a thermometer or other sensor of a physical quantity) (Russell & Norvig).

An ideal observation provides sufficient information to determine the exact and entire current value of **s**. When this is possible, the state is said to be *fully observable* (Russell & Norvig). There is no requirement that the variables in **s** be directly measured. It may be more convenient or realistic to measure a different vector **o** that can be transformed into the value of **s**. For example, in figure 2.5 the relative position of the two aircraft might be defined as one pair of variables $\{ x_m, y_m \}$ (2.5a), but easier to determine by measuring a different, equivalent, pair $\{ r, \theta \}$ (2.5b) that can be transformed into the other variables.

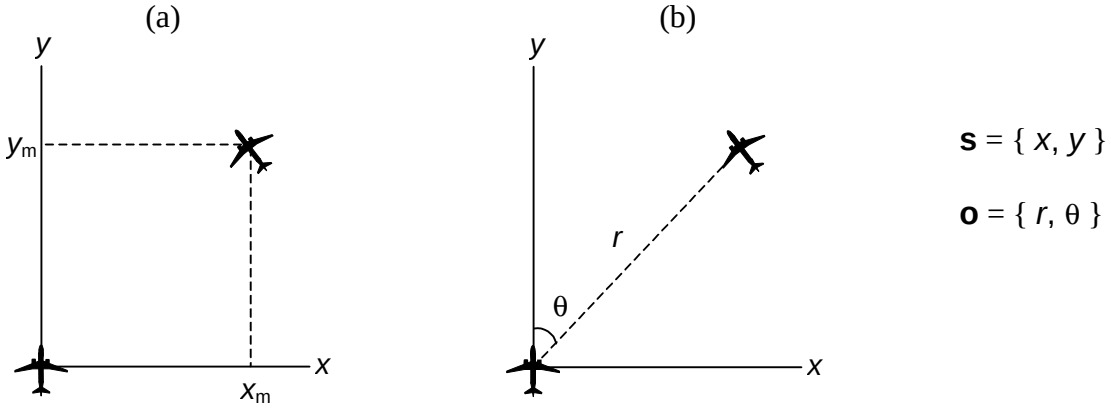


Figure 2.5: State-Observation Relationship

As mentioned in section 2.1.1, an ideal observation of the state is not always possible. One reason is that imperfect sensors can add random measurement errors. The uncertainty due to such errors is often modeled in terms of probability distributions over the space of **o** or *observation space, O*. This kind of observation uncertainty is pictured in figure 2.6a, assuming no state knowledge beforehand (i.e. a uniform state distribution on the *x*-*y* plane).

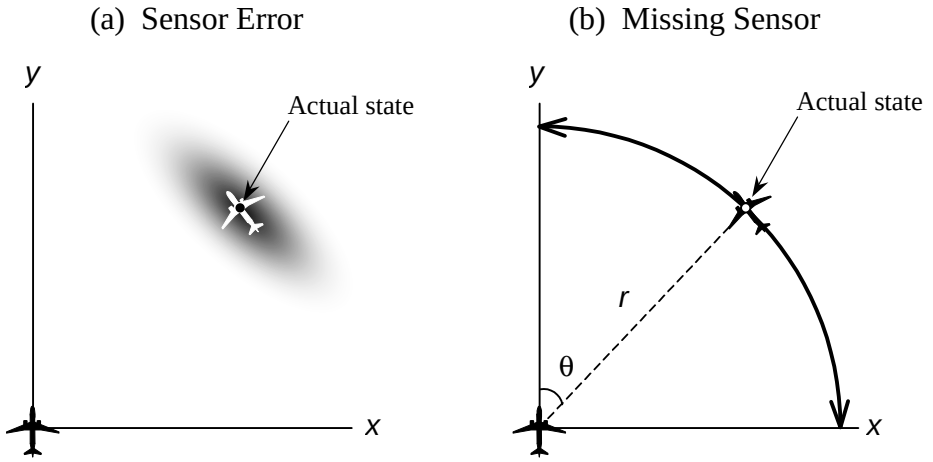


Figure 2.6: Sources of Observation Uncertainty

Even without measurement error, full observability can fail to hold if there are too few variables in **o** to determine the current **s**. For example, in figure 2.6b the observation is reduced to **o** = { *r* }, with *r* exactly measurable and θ unmeasurable, resulting in a locus of possible states all equally likely. Effectively, the measurement error of θ is uniformly distributed if there was no prior knowledge about the variable. If either missing-sensor or sensor-error uncertainty limit the observation, the state is termed *partially observable* (Russell & Norvig).

Assuming **o** consists of measured variables with probabilistic error uncertainty, the relationship between the current state and observation of the state is denoted by the *sensor function* (Russell & Norvig):

$$\mathbf{o}(t_c) = O(\mathbf{s}(t_c)) \tag{2.4}$$

The function $O(\cdot)$ is a probability distribution over the space of possible measurements *O*, conditional on the current state, $\mathbf{s}(t_c)$.

2.2 Alerting System Performance

There are three main aspects of alerting system performance: the system's ability to avoid *unsuccessful alerts*, *unnecessary alerts* and *improper alerts*.

An unsuccessful alert is defined as any alert that is followed by an incident (Yang & Kuchar, 1997). An example would be receiving a collision warning so late that there is not enough time to actually avoid a collision (a *late alert* (*LA*)).

An unnecessary alert (*UA*) occurs when no incident would have occurred had there been no alert (Yang & Kuchar, 1997). This category is relevant because, in calling for an avoidance response, an unnecessary alert interferes with the human operator's achievement of goals, such as maximizing productivity or minimizing time-to-completion of a task. In addition, an unnecessary alert can actually lead to an incident (an *induced incident* (*II*) (Carpenter & Kuchar, 1997)) that would not have happened otherwise. For example, in maneuvering to avoid an incorrectly predicted collision with one car, a driver could crash into another. Note that a successful alert can be either necessary or unnecessary: these alerts are not mutually exclusive events.

An improper alert (*IA*) is an alerting system action perceived as erroneous by a human operator. This could happen for different reasons. For example, the alert might be known or believed to be unnecessary or premature (a *nuisance alert* (*NA*)). Another possibility is that the alert is issued later than preferred (or not at all). Improper alerts, particularly nuisance alerts, are a problem because experiencing these over time can cause an operator to mistrust the alerting system and become less likely to conform to alerts and guidance. This issue was studied in the context of aircraft collision avoidance by Pritchett (Pritchett & Hansman, 1997). In principle, an improper alert can also be an unnecessary alert, an unsuccessful alert, both, or neither. Figure 2.7 illustrates the possible interdependencies of the three error events with a Venn diagram. Drawn this way, an ideal alerting system would never operate inside the ellipses: every alert would be proper, necessary and successful. But as will be discussed, a more realistic goal is not to achieve an ideal alerting system, but one with an acceptable trade-off between different error types.

Whereas unsuccessful and unnecessary alerts are clearly and objectively defined, improper alerts are more ambiguous because of the difficulty of understanding or predicting operator preferences. By some means, however, they must be addressed, and

preferably at an early stage in design. Otherwise, they will have to be dealt with through trial and error with test subjects or user feedback.

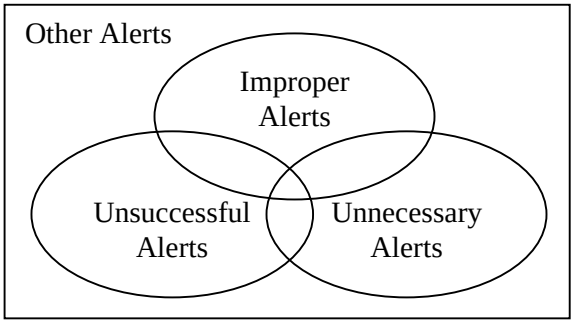


Figure 2.7: Alerting Performance Aspects

Alerting system performance can be quantified using probabilities or frequencies of the three error events. The frequencies of unnecessary alerts and incidents (unsuccessful alerts[3]) in particular are common metrics of overall performance. The performance of the alerting system in a given trajectory scenario can be described in terms of the conditional probabilities of the different events at the time of the alert:

$$P(\text{ Unsuccessful Alert} \mid \mathbf{s} \text{ is at Threshold }) \tag{2.5}$$

$$P(\text{ Unnecessary Alert} \mid \mathbf{s} \text{ is at Threshold }) \tag{2.6}$$

$$P(\text{ Improper Alert} \mid \mathbf{s} \text{ is at Threshold }) \tag{2.7}$$

For any of the error events it is best to have a low probability or frequency, but due to uncertainty in the situation dynamics, it is usually impossible to optimize all metrics at the same time. For instance, adjusting the logic to minimize the probability of an unnecessary alert tends to increase the probability of an unsuccessful alert, necessitating a trade-off between the two. One way to visualize this trade-off is with a *System Operating Characteristic* (*SOC*) plot as in figure 2.8 (Kuchar 1995, 1996).

[3] If an incident happens before there is an alert, the alert can be thought of as happening at the time of the incident { 0, 0 } (Winder & Kuchar, 1999).

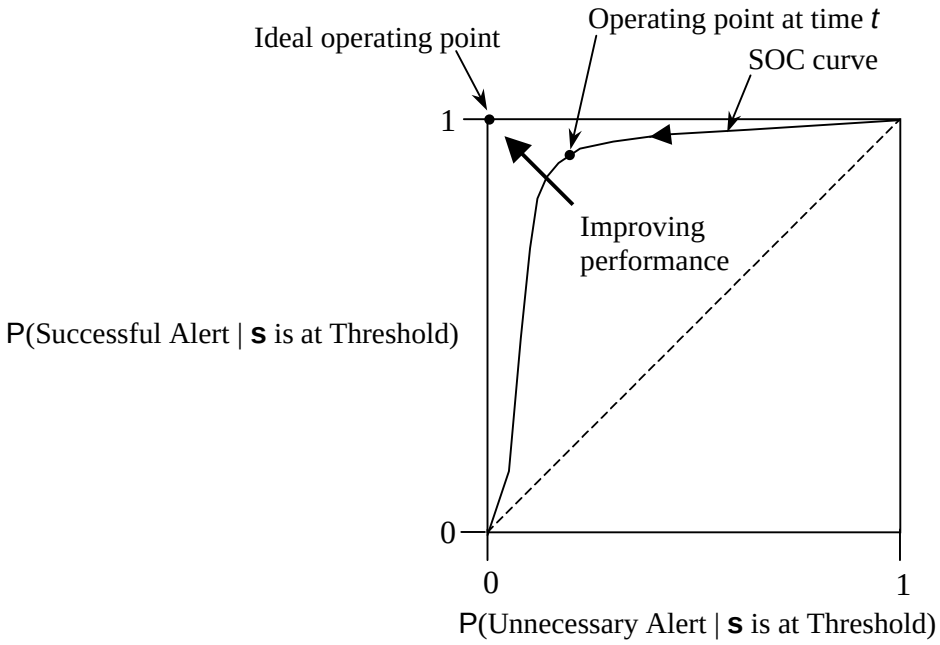


Figure 2.8: System Operating Characteristic Plot

A point on the SOC plane is a plot of the successful alert probability (1 minus the unsuccessful alert probability) against the unnecessary alert probability at a given time. The operating point at the moment an alert is triggered (when **s** is at the threshold) is a measure of alerting performance for that particular scenario. The curve shown corresponds to a hypothetical situation trajectory that ends in an incident along the nominal trajectory if no alert ever occurs. The point traces a path from the upper right corner toward the left as the incident becomes increasingly likely. If evasive action is taken early on, the probability of avoiding the incident is high, as is the probability that the alert is unnecessary. As the alert is deferred, the probability that it would be safe or unnecessary decreases. Ultimately, if the alert is deferred for too long, the incident becomes unavoidable due to the limitations of escape options, and the operating point goes to (0, 0). The ideal place for an alert to occur is at the upper left corner, where the alert is known to be necessary and there is no chance of an incident when the alert occurs. Realistic systems tend not to occupy this point, but a designer may try to achieve alerting points as near to it as possible (Yang & Kuchar, 1997). The nearness that can be achieved depends on the predictability of the situation and the effectiveness of escape options.

For SOC points below the diagonal line an alert would be more likely to cause a collision than prevent one. At the diagonal the alerting system is equally likely to cause a collision as prevent one. So at minimum, alerts should occur over the diagonal.

An SOC plot can also be used as an analysis tool to help visualize the overall performance of an alerting system (Winder & Kuchar, 1999). Whereas in the previous discussion the SOC point was an incident probability at the threshold, the coordinates can also be defined as the global successful and unnecessary alert averages resulting from a set of alerting process trajectories. This use will be important in later chapters.

2.3 Probabilistic Prediction-Based Alerting

Figure 2.9 redraws the alerting process with the alerting system's inner process broken down into two phases: belief state updating and the action logic.

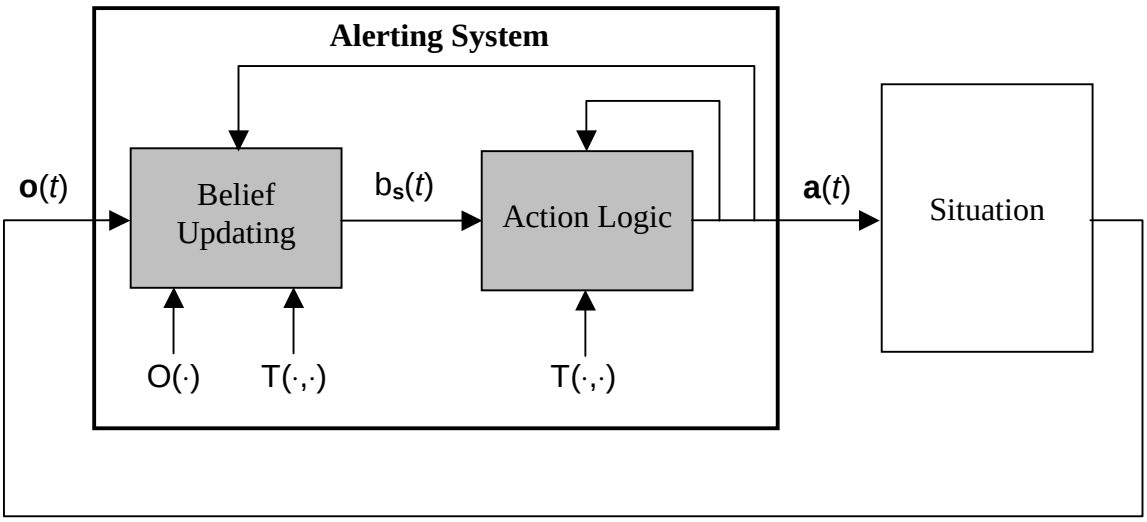


Figure 2.9: Alerting System Structure

2.3.1 Belief State Updating

The Belief Updating block is a process of assimilating the stream of observations of the situation into a probability distribution, $b_s(t)$, describing the overall uncertainty of the current state. Even if each observation has great uncertainty due to the reasons

described earlier, the combined evidence of many observations over time, plus knowledge about the nature of observation errors, of the situation dynamics, and of alerting actions taken, will tend to reduce current-state uncertainty. Because probabilistic state uncertainty can also be thought of as the knowledge or reasonable beliefs about the state, the distribution b_s is called the *belief state* (Russell & Norvig).

Probability theory describes the process of incorporating evidence into the belief state ("*filtering*" (Russell & Norvig)) using the sensor and transition functions that were defined in section 2.1. This process is described in appendix A. Thus, these two functions are included in figure 2.9 as inputs to the belief updating process, along with the observations and past alerting actions. Depending on the situation, belief filtering can be a computationally intensive and time-consuming process. But for special cases of the sensor and transition functions, efficient numerical algorithms exist that make hardware implementation of the process more practical. For example, Kalman filtering is a common technique that produces a valid state distribution when sensor errors are Gaussian noise and the state dynamics can be expressed as linear equations. Such algorithms are common in alerting system design.

2.3.2 Performance Metric-Based Alerting

The Action Logic in figure 2.9 uses the current belief state and the prediction model to choose an alerting action. As shown, it can also take the previous action as an input. Most of the time the alerting system operates in the background, monitoring the state for conditions that justify an alert. During that time the action logic should suggest a deferral action, a_0 (section 2.1.3), because intervention is not yet warranted. Once an alert happens, the logic continues to monitor the state, now using gathered information to best guide the human operators in avoiding an incident.

In prediction-based alerting, the logic chooses its actions based on the relative quality of predicted outcomes for each option. The option that would have the best result (according to some value scheme) is chosen over the others. Predicting outcomes requires the current-state distribution and the state transition model. The incident model, introduced in section 2.1.2, is also required at this stage. As they are determined by

probabilistic process model components, outcome predictions are themselves probabilistic quantities. For example, for a particular sequence of actions it is possible to calculate the probabilities of future events such as hazard encounters, or the probability that an alert issued now would be an unnecessary alert.

As discussed in section 2.2, such probabilities are basic measures of the performance of the alerting threshold. When these quantities are computed in real time within the alerting system, and known at the time an alert happens, the performance of the threshold for that scenario is directly controllable with the decision. For example, in Figure 2.10 an airborne alerting system monitors the nominal probability of crashing into

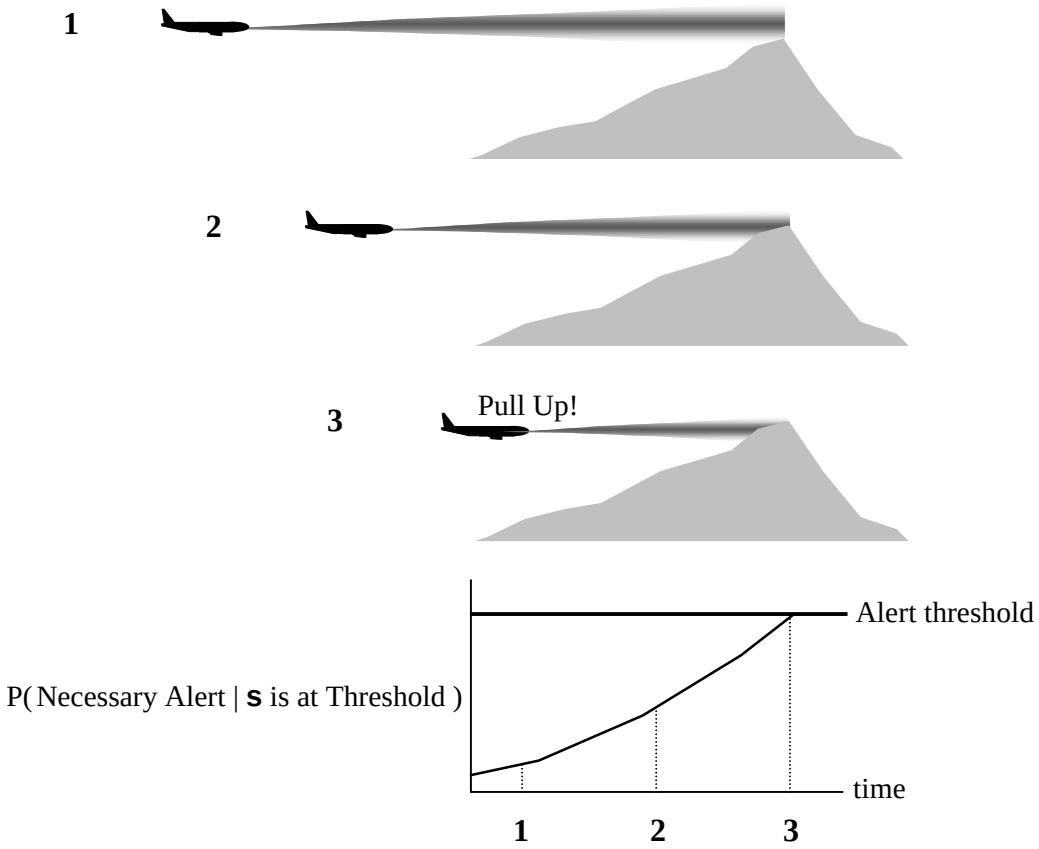


Figure 2.10: Performance Metric Threshold

a mountain, and defers alerting until reaching a threshold probability that the alert is needed to avert a collision. Generally there are requirements on safety of the escape maneuver (successful alert probability) too, so that there may be a defined region in the SOC space where alerts are acceptable if they occur, and outside of which they are not.

The idea of probabilistic prediction-based hazard avoidance alerting is to define the alerting logic directly in terms of predictive metrics such as these, and the path they take through the metric space, as figure 2.11 illustrates (Yang & Kuchar, 1997, 2002).

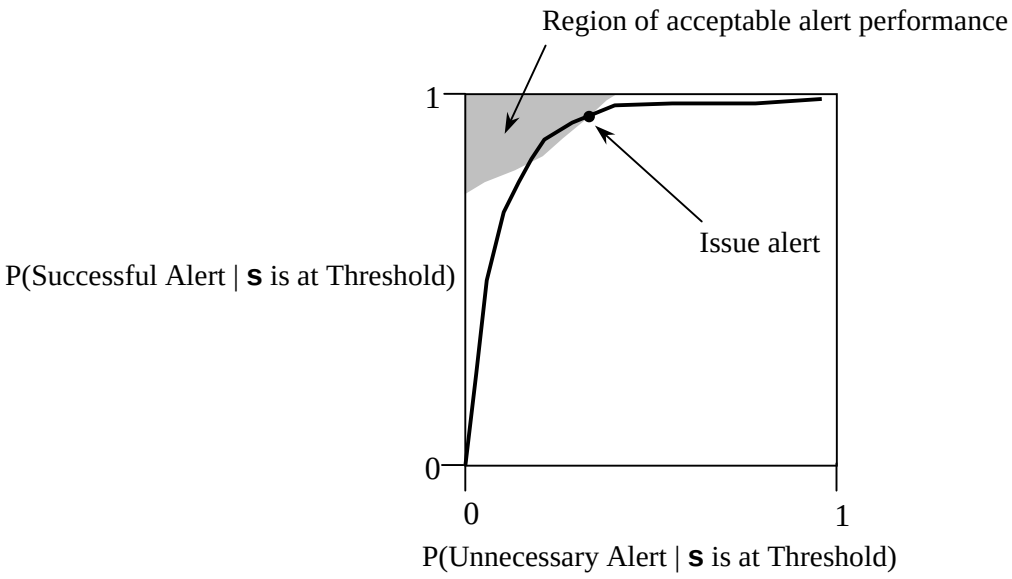


Figure 2.11: SOC Alert Threshold

2.4 New Design Considerations

This thesis looks at two new design considerations for prediction-based alerting, and their effects on performance. In this section the considerations are identified, and their anticipated importance is explained in relation to current design practices.

2.4.1 Mode Beliefs

The situation state, **s**, was described in section 2.1 as a set of variables that satisfies the Markov property, so that probabilistic predictions can be made based on the current state. These are most often real-valued variables representing physical quantities like distance or speed. Another kind of variable, termed a *mode*, is of particular interest in this thesis. A mode is a variable having a discrete domain, and whose value tends to persist over long intervals of time. The mode serves to index a set of distinct behaviors that the situation can exhibit at different times. These can be expressed as a set of transition functions (expression 2.8), one for each possible mode, m_i. For example,

figure 2.12 redraws the terrain collision scenario from chapter 1 in probabilistic form. Though future states are uncertain in either mode, in one the aircraft will nominally tend to crash into the mountain, and in the other it will tend to climb over it, as separate probabilistic transition functions can describe. In general, a mode's value at a given time can be a deterministic function of inputs or other state variables (e.g. an autopilot tracking mode at a known setting), or a random process (e.g. pilot mental lapse, weather conditions). The mode can be observable or unobservable by the alerting system, but only the latter case is of interest in this thesis.

$$\mathbf{s}(t_c+\Delta t) = \begin{cases} T_1(\mathbf{s}(t_c), \tau_a), & m_i = m_1 \\ T_2(\mathbf{s}(t_c), \tau_a), & m_2 \\ T_n(\mathbf{s}(t_c), \tau_a), & \ldots m_n \end{cases} \tag{2.8}$$

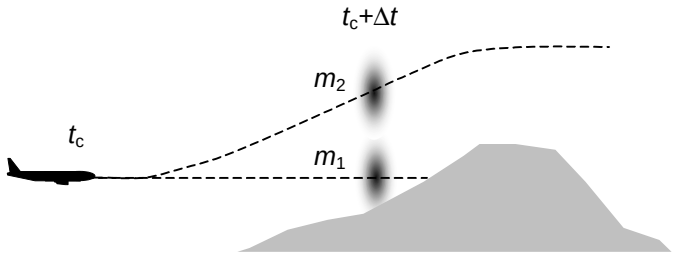


Figure 2.12: Transition Function Indexing with a Mode Variable

When there are mode variables in the situation state, the filtering process described in section 2.3 still applies, resulting in a belief state description of mode uncertainty. In this case the belief state is a discrete probability distribution. Even if no direct measurement of a mode variable is possible, the estimation process will infer information about the mode through its known influence on the transition function. This is similar to using a filter to estimate speed from repeated position measurements when speed is not directly measurable: the equation relating the two allows observations of one to improve knowledge about the other.

The action logic part of the alerting system uses the total belief state at the current time, $b_s(t)$, to make predictions for each possible alerting action. Note that there are two contributors of uncertainty in predicted states: current-state uncertainty (the belief state) and inherent prediction uncertainty due to the transition function. The uncertainty of the predicted state increases with an increase in either component. Figure 2.13 shows the effect of increasing belief state uncertainty on the distribution of predicted states. In a case where the uncertainty contribution of one source is small, neglecting that uncertainty (i.e. replacing the distribution with a single assumed value) may have little effect on the overall prediction uncertainty, and be a reasonable approximation to make. This simplification is commonly done with the output of filtering processes, where the intention is usually to find a best estimate of the state value rather than a full uncertainty description. For example, the mean value or maximum-likelihood value of the belief state might be taken as the estimate.

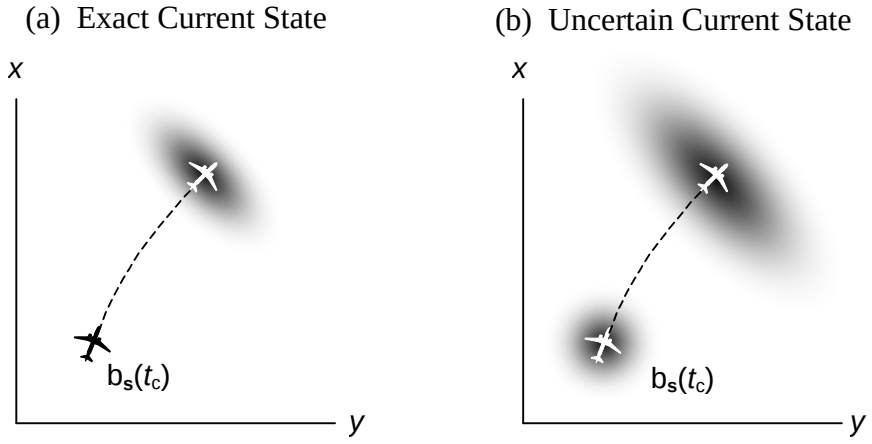


Figure 2.13: Effect of Current-State Uncertainty on Predictions

As figure 2.14 illustrates, with a 1-dimensional state, that a distribution with a sufficiently small variance (2.14a) can ensure that errors due to a state approximation nearly always remain within specified bounds of acceptability. If the filtering process leaves large enough uncertainties in the state, as with the broad distribution in (2.14b), state estimation error becomes unacceptable.

As with continuous variables, a maximum-likelihood approximation can be used to estimate a mode from its distribution. This means assuming that the present mode is the one with the highest probability. This assumed mode can then be used to predict future outcomes, which might allow easier decision metric computation or even avoidance of computations if certain modes are known in advance to be safe or unsafe. This suggests a specialized version of the alerting process where the alerting decision is based on a hypothesis test on the current mode belief state rather than on explicit prediction of the future effects of the current state.

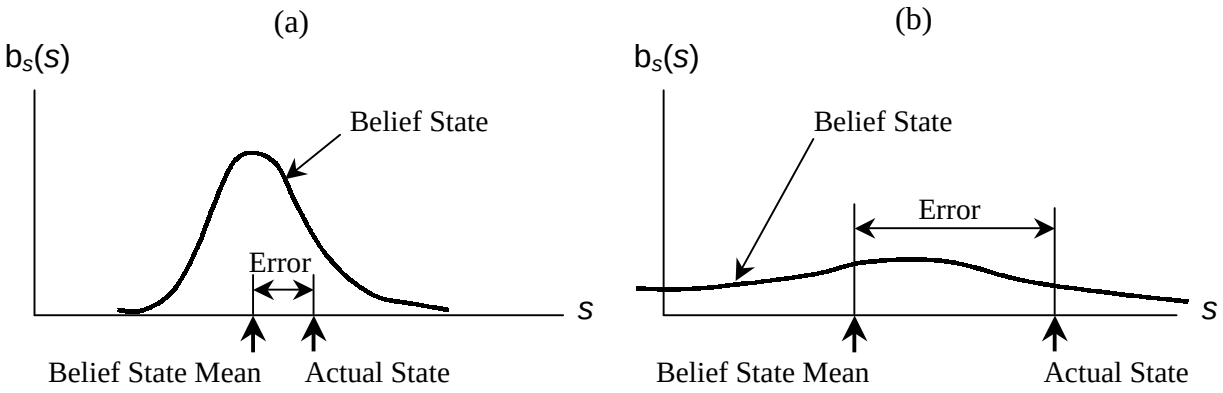


Figure 2.14: State Approximations

Because belief updating can be complicated and slow in some cases, another inclination might be to avoid the Bayes updating process by leaving the belief state or part of the belief state (over some but not all uncertain variables) fixed at a single distribution. This could be a viable option in some cases, if the simplifications are justifiable using known transition and sensor function properties.

2.4.2 Anticipating Knowledge Gain and Decision Opportunities

Up to now the only assumptions about the action logic of the alerting system are that it makes use of a current belief state for the situation, an incident model and probabilistic state predictions to generate metrics for comparing possible actions. Based on existing alerting methods there are different ways to proceed. These will be described and compared next, along with a new method of planning with possible benefits over the existing options.

The simplest way to use state predictions in planning is to base action choices on the predicted outcomes of one or more maneuvers. A *maneuver policy* will refer to a predetermined control law that maps any belief state, $\mathbf{b_s}$, into an alert action, **a**. A *maneuver* is the state trajectory resulting from repeatedly using a single maneuver policy to generate the alerting action. Figure 2.15 illustrates the maneuver idea for the extreme case (for simplicity) of no prediction or current-state uncertainty. The maneuvers shown include the *nominal maneuver* (N), where the maneuver policy is to take the nominal action regardless of $\mathbf{b_s}$, and possible *escape maneuvers* (E_i). Each escape maneuver policy, when used repeatedly, causes the system to seek some state goal. For example, one maneuver policy for an aircraft could involve achieving a certain heading or climb rate, so repeated use of the maneuver would cause convergence to that goal.

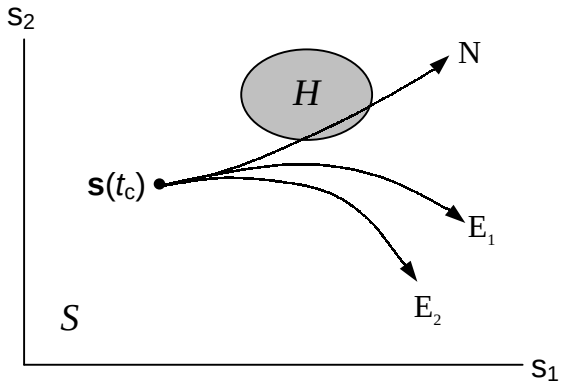


Figure 2.15: Planning with Maneuvers

For each maneuver there is some probability of an incident occurring. The action decision is made by optimizing or satisficing a cost function defined in terms of the probabilities of future events. For example, prior to any alert the rule might be to issue the alert if the probability of an incident along the nominal trajectory exceeds some threshold (as in figure 2.10). Or, it could be based on a maximum allowable probability of an incident for the available escape maneuvers, where reaching a minimum safety level triggers the alert, as done in (Carpenter & Kuchar, 1997). It could also be a rule using both nominal and escape maneuver predictions.

After an alert happens, the simplest approach is to use the safest maneuver, and cease any further planning. Then there are no further choices to make, and the resulting probability of an incident will be as predicted at the time of the alert for that maneuver. For the most predictable evasion performance, the maneuver may take the form of an understood procedure the human operators rehearse, so that no display or interpretation of complex guidance is needed after the alert (e.g. evacuation procedures for fire alarms in a building). In aviation alerting this was assumed during development of the AILS collision avoidance logic for close parallel approaches (Winder & Kuchar, 1999; Samanant et al, 2000), where the tactical nature of the alerts made quick and precise execution of escape maneuvers important.

A more flexible use of maneuver policies is allowing the alerting system to switch between maneuvers (replan) even after an alert occurs. This gives the alerting system greater freedom in acting to attain its objectives, but the method of choosing the next action becomes less clear. Probably the most obvious approach is to choose the maneuver resulting from a fixed maneuver policy assumption as before, but to repeat the decision at later times, each time using the maneuver policy for the maneuver that is predicted to be safest. This could result in a sequence of maneuver segments such as shown in figure 2.16. Note that there is an inconsistency in assuming that the maneuver

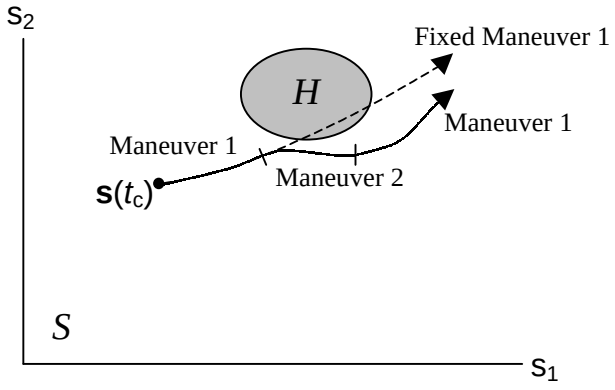


Figure 2.16: Replanning with Maneuvers

policy is fixed when in fact it can be changed later (the fixed-maneuver logic of figure 2.15 has this same inconsistency, since choosing the nominal maneuver does not actually force a nominal trajectory to be followed in the future). However, the escape maneuver

replanning method can still decrease the probability of an incident over that of the fixed-maneuver planning method once an alert has occurred, making it a safer method while not increasing the unnecessary alert rate. The method has been used in alerting systems—for example, by the TCAS logic in the post-alert phase (RTCA, 1983; Harman, 1989).

Figure 2.17 shows the difference between safety as computed using a fixed-maneuver assumption, and the more realistic safety probability the alerting system would predict if it were able to take future decisions into account. The more accurate knowledge would have an impact the alerting decision at t_c, possibly causing the alert to be deferred longer than it would be otherwise. This would reduce the unnecessary alert probability, assuming the SOC trajectory continued moving leftward.

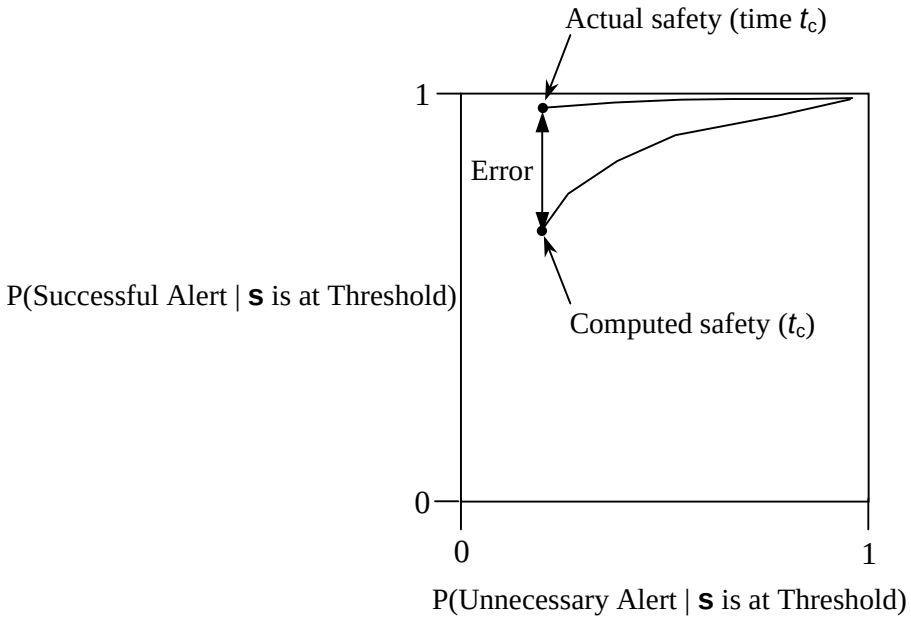


Figure 2.17: Error in Computed Safety

Another important observation to make is that if the alerting system has only uncertain knowledge of the situation state (a belief state), then taking future decisions into account in probability calculations also requires considering future belief states. This is because at future points in time the alerting system will have made additional

observations of the situation, and as a result it will have gained new information that will affect the decision made at that time.

If it can be done, this kind of probability estimate would enable decision making that is more rational and potentially better than others in that, to achieve a specific goal, the decision maker uses more of the information available in the alerting process model and from observations than in simpler methods. Methods currently available for solving this kind of decision problem are described in Markov decision process (MDP) theory, and will be applied later in this thesis. This is a novel approach to designing hazard avoidance alerting systems, because existing probabilistic prediction-based logics tend to use the simpler maneuver-based methods.

An issue that comes up and will be addressed later is that this most general method of probabilistic planning has the potential to be more complex numerically, and too difficult to carry out in real time with trajectory simulation-based methods like Monte Carlo simulation.

2.4.3 Strategy to Avoid Improper Alerts

For the alerting system to be accepted by its users, it should avoid significant numbers of nuisance alerts. Ideally it would avoid all improper alerts, meaning operators would agree with its actions in all scenarios (section 2.2). The design process should include a strategy for minimizing such alerts.

An alerting system's improper alert tendencies are usually detected and eliminated in a process of trial and error, using simulations or other tests along with human judgment. Because improper alerts are inherently subjective, it is unclear how to control them directly through definition of the alerting threshold and avoidance logic. Assumptions are sometimes made, at least implicitly, that improper alerts are closely related to unnecessary alerts, so that minimizing unnecessary alerts also minimizes nuisance alerts. Assuming this relationship is true, it still does not provide a means of avoiding delayed alerts that could also be considered improper.

In this thesis an assumption is made that a more general and complete description is needed of what causes improper alerts, and this description should guide the basic definition of the alerting threshold. Part of the motive for a decision theoretic approach to alerting is an hypothesis that, by approximating a rational decision process and basing it on goals compatible with operator preferences, an alerting logic will tend to avoid improper alerts. This notion will be discussed in more detail in chapter 4.

2.5 Chapter Summary

This chapter described the basic ideas of probabilistic prediction-based alerting as it is typically done, and then argued the need for certain improvements in design: mode uncertainty modeling and accounting for future decision opportunities.

First the alerting process was described in mathematical terms. The alerting system can be though of as a controller of a situation made up of human operators and a plant they interact with. States of dynamic process elements are expressed as state vectors, and the overall situation has a state $\mathbf{s}$. The dynamics of $\mathbf{s}$ are described by a probabilistic transition function $T(\mathbf{s}(t_c), \tau_a)$, which is the conditional distribution of possible next states resulting from a given alert signal τ_a and current state $\mathbf{s}(t_c)$. The alerting system makes observations of the situation through the probabilistic sensor function $O(\mathbf{s}(t_c))$, which is the conditional distribution of possible measurements $\mathbf{o}$ of the current state.

The fundamental aspects of alerting system performance were reviewed. There are three types of error that an alerting system can make: unsuccessful alerts, unnecessary alerts and improper alerts. An unsuccessful alert is one where an alert is followed by an incident, either because the alert was too late or because it actually caused an incident that would not have happened otherwise (an induced incident). An unnecessary alert is one where the alert is not necessary to prevent an incident, and results in an unneeded evasion maneuver or an induced incident. An improper alert is an outcome where a human operator feels an alert is incorrect—either too early or too late—and can harm operator conformance to alerts over time.

Next, the use of probabilistic state filtering in alerting systems was discussed, followed by how probabilistic performance metrics are used to define the alerting threshold.

Finally, new decision considerations were discussed in relation to current methods, with the overall aim of increasing the amount of available information that is applied to the decision. One is the possibility of using belief state filtering to model uncertainty of dynamic modes. Another is the use of information about future decision options, including possible future belief states, to make more informed alerting decisions, both at the threshold and during the evasion guidance phase. Finally, a short discussion of improper alert reduction as a goal of design was included. The need was stressed for use of an improper alert model early in the design to minimize trial and error improper alert reduction later.

3. Related Work

This research concerns methods for designing tactical hazard avoidance alerting systems of the kind increasingly used in aviation safety applications. These systems are characterized by their use as backup safety devices for infrequent hazards, numerous input state variables, complex alert signals such as staged alerts and dynamically generated resolution guidance, and multiple dynamic modes in the monitored situation. This thesis will apply methods from Markov decision Process (MDP) theory to such systems to guide design and improve performance. Figure 3.1 illustrates the relationship

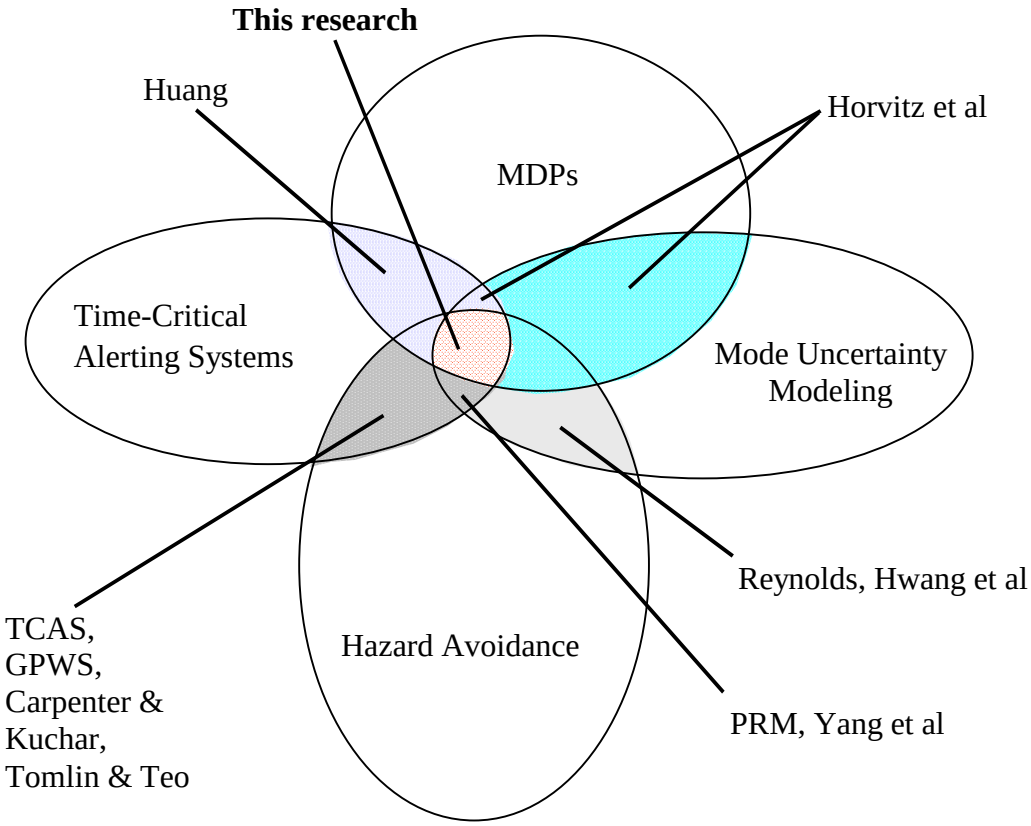


Figure 3.1: Related Research

of this problem to broader existing areas of research. The following sections give an overview of the most closely related work, sharing two or more of the large categories and identified in the diagram with shading.

3.1 Time-Critical Hazard Alerting Systems

Existing alerting system designs can be divided roughly into three categories: conformance-based, simple trajectory-based and complex trajectory-based.

Conformance-based alerting systems operate under the reasoning that abnormal system behavior justifies an alert. An example is the Precision Runway Monitor (PRM) alerting system for preventing collisions during closely spaced parallel runway approaches (Shank, 1994). In this case alerts are triggered when a radar-tracked aircraft crosses into a "no transgression zone" separating two approach paths, as in figure 3.2.

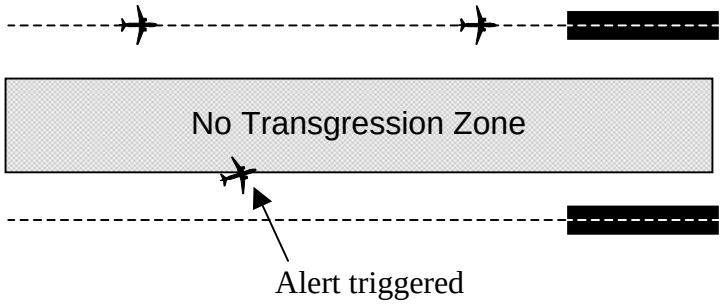


Figure 3.2: PRM Conformance-Based Alerting Threshold

Such an alerting system has no inherent tendency to avoid unnecessary alerts (where no incident would occur without the alert). For example, PRM would trigger an alert even if there were no aircraft in the adjacent approach path to be endangered.

A more sophisticated type of logic is simple trajectory-based alerting. In this method alerts are triggered through a combination of a rough state prediction model and incident proximity criteria. For example, the TCAS aircraft collision avoidance system uses range and time threshold criteria along with an assumption of constant range rate. In such an alerting system there may be several threshold parameters that must be tuned to give acceptable performance in terms of unnecessary alerts and safety. Other examples

of this type of alerting in aviation are the Ground Proximity Warning System (GPWS) terrain avoidance logic and Airborne Information for Lateral Spacing (AILS) logic for parallel approach collision avoidance.

Complex trajectory-based alerting uses a more realistic trajectory model and threshold criteria that are stated directly in terms of alerting performance requirements. An example is the parallel approach logic by Teo & Tomlin (2003) that triggers alerts according to worst-case safety requirements for escape maneuvers. More common are alerting logics that use probabilistic state prediction models and decision metrics. At MIT logics have been developed (Carpenter & Kuchar; Yang & Kuchar, 2002) using probabilistic safety and unnecessary alert performance metrics (P(Successful Alert) and P(Unnecessary Alert)) to define the alerting threshold.

Of these three alerting approaches, the Markov decision process alerting method described in this thesis is most similar to complex trajectory-based alerting with probabilistic modeling and decision metrics. This is because it too uses explicit probabilistic prediction modeling and probabilistic decision metrics to define the logic. It is distinct, however, in that existing approaches have not made use of probabilistic mode uncertainty models or information about possible future decisions. Alerting logics typically involve continuous or observable state variables, and decision metrics are computed with an assumption that a fixed control sequence will be followed.

3.2 Hazard Avoidance with Mode Modeling

An overlapping area of research concerns hazard avoidance using uncertain mode modeling, which includes more strategic forms of detection and avoidance.

At MIT recent attention has been given to conformance monitoring as an approach to avoiding incidents in an airspace environment (Reynolds & Hansman). In this work a fault detection approach was used to judge whether aircraft were conforming or not to an expected path. As with conformance-based tactical alerting systems such as PRM, the driving philosophy in this case is that detection of abnormal system behavior justifies intervention. This point of view is consistent with the current air traffic control

system, where aircraft in controlled airspace are monitored continuously for deviations as they fly along pre-approved routes.

The Markov decision process alerting method differs from this in two main ways. One is that fault detection results in a conclusion that the system is in a specific mode, whereas with the MDP method mode probabilities are maintained and used to weight the outcomes predicted for the different possible modes. Another difference is that in the MDP case alerting decisions depend on predicted final outcomes, whereas interventions are justified by the mode in a fault detection method.

There are other mode-based hazard avoidance methods more similar to MDP alerting in that they use state prediction and mode probability updating. An MIT group (Yang et al, 2004) has applied mode uncertainty modeling and estimation to automotive collision avoidance. Probabilities are computed in real time for a number of possible dynamic models (modes) using a multiple Kalman filter approach. A mode estimate is determined from this and used with a hazard prediction model to choose the alerting action. Another method recently suggested (Hwang et al, 2003) for aircraft conflict detection combines mode estimation with a probabilistic trajectory prediction model to estimate collision probabilities. The MDP approach proposed here differs from both of these methods in that the mode probability distribution rather than a mode estimate taken from the distribution is used in making predictions. Also, in the MDP approach predictions will include information about future decisions yet to be made, whereas both of these assume a predetermined control sequence will be followed, as described in section 2.4.2.

3.3 Alerting with Markov Decision Processes

Markov decision processes have been applied in various decision aids such as navigation advisors that continuously help users optimize their movements. They have also been used for alerting applications.

One relevant case is an alarm system concept designed at Stanford to assist ICU physicians in making optimal use of limited attention resources (Huang, 1999).

Successive alerts would direct physicians to patients most urgently needing attention. This application involved a fully observable Markov patient model, whereas the proposed alerting application allows a partially observable state (uncertain modes).

Another related use of Markov decision processes is work by Horvitz with others on designing MDP-based software agents to help humans accomplish tasks, including during time-critical decision situations such as medical emergencies (Horvitz & Rutledge, 1991; Horvitz & Seiver, 1997). One application combined Markov decision processes with an uncertain user attention model to optimize the timing of email alerts (Horvitz et al, 1999). This latter research is very similar to the proposed use of MDPs, but differs in that the present work concerns time-critical hazard avoidance alerting and specific issues for this kind of alerting, including escape guidance, uncertain dynamic modes. In addition this thesis also discusses MDP-based alerting in the context of current hazard alerting methods, including specific peformance measures such as P(SA), P(UA) and improper alert rates, in order to point out differences and benefits.

4. Alerting with Markov Decision Processes

4.1 Introduction

Richard Bellman introduced Markov decision processes in 1957 as a variant of his more general "dynamic programming" theory of optimal control, itself based on work by Hamilton and Jacobi in the 1800s (Sutton & Barto, 1998; Bellman 1957a,b). Dynamic programming is a method of controlling dynamic systems to optimize some measure of the state trajectory. A typical problem is controlling a vehicle's speed to minimize fuel use on the way to a destination. A Markov decision process is such a problem where time is discrete and the process dynamics exhibit probabilistic randomness. Since its invention, the theory has been extended to allow probabilistic current-state uncertainty due to limited observability. Such a process is termed a partially observable Markov decision process (*POMDP*).

Markov decision processes have been applied in economics, operations research, control systems design, and artificial intelligence (AI) among other areas. In AI research, MDP theory has gained favor as a model of rational decision making in well-defined circumstances where an intelligent agent's outcome preferences can be expressed as a trajectory utility function (Russel & Norvig). This point of view is supported by the success of MDP-based systems in complex reasoning tasks such as playing master-level backgammon against humans (Tesauro, 1994). As demonstrated in that work, MDPs can sometimes be combined with reinforcement learning techniques to avoid the need to directly define utility and reward functions. Russell and Norvig (2003) give a good introduction to MDPs from an AI point of view, and are the main resource for MDP theory and terminology in this chapter. Other popular references include Bertsekas (1987, 1995) and Puterman (1994).

The purpose of this chapter is to give an overview of MDP theory in an alerting systems context. Most of the needed components—the Markov state, probabilistic dynamics and sensor functions—were introduced in chapter 2.

4.2 Problem Formulation

MDP theory requires the alerting process to be modeled in discrete time. Time is represented as a series of integers, k, where each k corresponds to a point in time, t_k. Hazard alerting processes are often discrete anyway, due to the nature of the hardware implementation, which can involve digital computers and sensors with limited update rates.

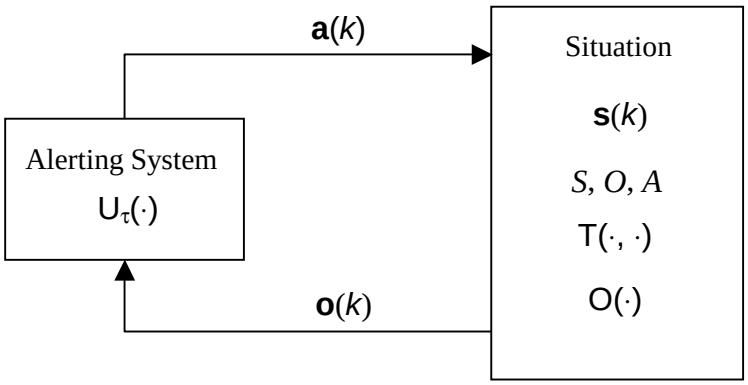


Figure 4.1: Markov Alerting Process

4.2.1 MDP Alerting Model

Figure 4.1 shows the Markov alerting process model and necessary components. A basic process includes:

$\mathbf{s}$, S	Markov situation state and state space	
$\mathbf{a}$, A	Alerting actions and action space	
$\mathbf{s}_{k+1} = \mathsf{T}(\mathbf{s}_k, \mathbf{a})$	Probabilistic transition function describing the distribution of next states within S due to a given action from at the current state	(4.1)

$U_\tau(\mathbf{s}_k, \mathbf{s}_{k+1}, \ldots \mathbf{s}_n)$	Trajectory utility function defining the total utility of a state trajectory τ, where n can be infinitely large	(4.2)

The notation "$\mathbf{s}_k$" is short for $\mathbf{s}(k)$. In a standard MDP as defined above, the state is assumed to be fully observable. A more general version of the problem, called a partially observable Markov decision process or POMDP, allows for an uncertain observation or sensor function:

$\mathbf{o}$, O	Set of possible observations of the state	
$\mathbf{o}_k = O(\mathbf{s}_k)$	Probabilistic observation sensor function describing the distribution of observations within O for a given state.	(4.3)

In a sense any alerting problem is partially observable, since the sensors needed to measure the state are imperfect, but in some cases a regular MDP model may be an appropriate approximation. Systems with unobservable mode variables in the state will likely need a POMDP model.

4.2.2 Trajectory Utility Function

In MDP theory the long-term priorities of the decision maker, in this case the alerting system, are described by a *trajectory utility function*, $U_\tau(\mathbf{s}_k, \mathbf{s}_{k+1}, \ldots \mathbf{s}_n)$. This function is a mapping of each possible system trajectory into scalar value, where the larger the value, the more desirable the trajectory is. For any two outcomes (trajectories), the one with the larger utility is preferred. In sequential decision making, each successive action is chosen to maximize the overall utility of the trajectory, including the future trajectory.

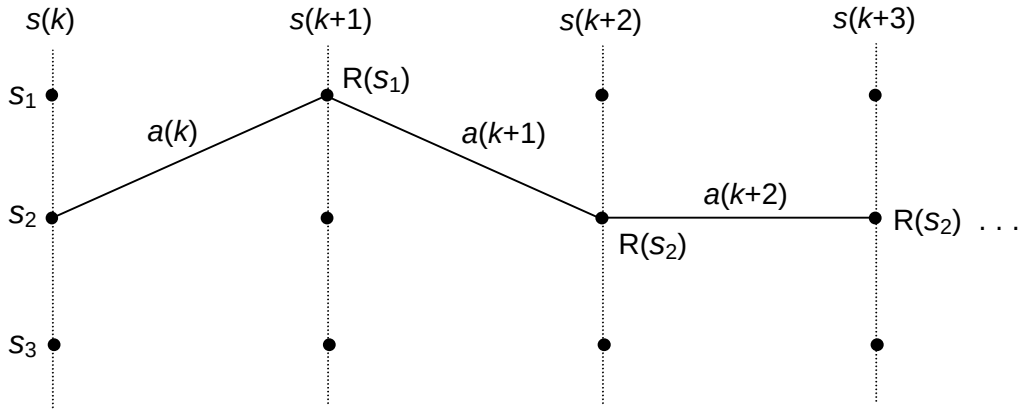
This definition of the utility function is a very general one, and to simplify the problem an assumption can be made that the utility function is *stationary*: that it (and the preferences it represents) stays constant with time. A consequence of the assumption is that the utility function takes the form of a sum of rewards:

$$U_\tau(\mathbf{s}_k, \mathbf{s}_{k+1}, \ldots \mathbf{s}_n) = R(\mathbf{s}_k) + \gamma R(\mathbf{s}_{k+1}) + \gamma^2 R(\mathbf{s}_{k+2}) + \ldots = \sum_{t=k}^{n} \gamma^{t-k} R(\mathbf{s}_t) \qquad (4.4)$$

Here the R factors are rewards gained at each new state, and γ is a discounting factor between 0 and 1 that reduces the reward of future states progressively with projection time. The discounting factor is included because in the case of $n = \infty$ (an *infinite horizon*) a sum of non-discounted trajectory utilities could be infinite, in which case it would be impossible to rank trajectories by utility as needed. In some cases a discounted reward function may also better represent the priorities of the alerting system.

For the case of $\gamma = 1$ the rewards are additive with no discounting. This is allowable if the process has a finite time horizon. A finite horizon can be a reasonable assumption where the potential for incidents exists over identifiable time intervals, such as the time between when an alerting system begins tracking an intruder vehicle and when it has safely passed.

The R function, or *reward function*, can be though of as the most fundamental expression of decision preferences, with U_τ being just a function of R. R is a constant function that assigns an immediate reward to every possible state. To use the Markov framework it will be necessary to describe the requirements of the alerting system in this form, and this may partially dictate the choice of state variables. Figure 4.2 illustrates the reward accumulation process for a 3-state discrete-time system, assuming no reward discounting. Each time an action is taken, causing a transition to the next state, a reward is gained.



$$U_\tau = R(s_1) + R(s_2) + R(s_2) + \ldots$$

Figure 4.2: Reward Accumulation

4.3 Expected Utility-Based Decisions

4.3.1 Maximum Expected Utility Principle

For a system with deterministic dynamics and a known state—where the exact trajectory and utility can be predicted for a sequence of actions—the preferred action at any time is the one that would lead to the highest overall utility. Specifically, for the trajectory utility function defined in section 4.2, the best action is the first action of the sequence that maximizes the utility function. These ideal conditions often fail to hold in realistic system models: current-state uncertainty, imperfect observability, and dynamic uncertainty can make an exact utility prediction impossible. Instead, the utility of a given action sequence becomes a random variable with a distribution of possible values, making it impossible to say which next action maximizes utility.

In MDP theory, and decision theory in general, the m*aximum expected utility principle* is used for decisions under uncertainty. It states that the preferred action is the one producing the greatest expected (mean) outcome utility, rather than the best exact utility:

$$\mathbf{a}_k: \quad \max_{\mathbf{a}_k} E[\, U(\mathbf{s}_{k+1}) \,] \tag{4.5}$$

Where $\mathbf{a}_k$ is the preferred k^{th} action and $U(\mathbf{s}_{k+1})$ is the maximum expected utility possible at the next state, $\mathbf{s}_{k+1}$.

4.3.2 Expected Utility of States

With utility defined by expression 4.4, the action taken at each step should be the one that maximizes the expected utility of the entire state trajectory. By the *principle of optimality*, this also means that the action should maximize the expected utility of the remaining future trajectory. Each state, then, can be thought of as having an associated maximum expected utility[4], $U(\mathbf{s})$. That utility is given by the *Bellman equation*

[4] The maximum expected utility of a state is sometimes referred to as the "utility" of the state for convenience.

$$U(\mathbf{s}_k) = R(\mathbf{s}_k) + \gamma \max_{\mathbf{a}_k} E[\, U(\mathbf{s}_{k+1}) \,] \tag{4.6}$$

This says that the utility of the current state $\mathbf{s}_k$ is the reward that was gained at $\mathbf{s}_k$ plus the discounted maximum expected utility of the following state, $\mathbf{s}_{k+1}$. Since (4.6) also determines $U(\mathbf{s}_{k+1})$ from its following states, $U(\mathbf{s}_k)$ could be calculated by recursively applying (4.6) into the future, if not for the computational difficulties of doing so.

Computing the expected utility of an action in such a way is a difficult task. Finding the expected utility of a given action requires knowing the maximum expected utility of each possible next state (at worst the entire state space) over all possible actions, so in the worst case the number of utility calculations (and amount of computing time) increases in proportion to $n^m q^{(m-1)}$, where n is the number of possible states, q is the number of available actions, and m is the number of action stages considered.[5] The increase in computations with the number of action stages is illustrated in figure 4.3 for the 3-state system of figure 4.2, where m reaches 3. The number of possible state paths

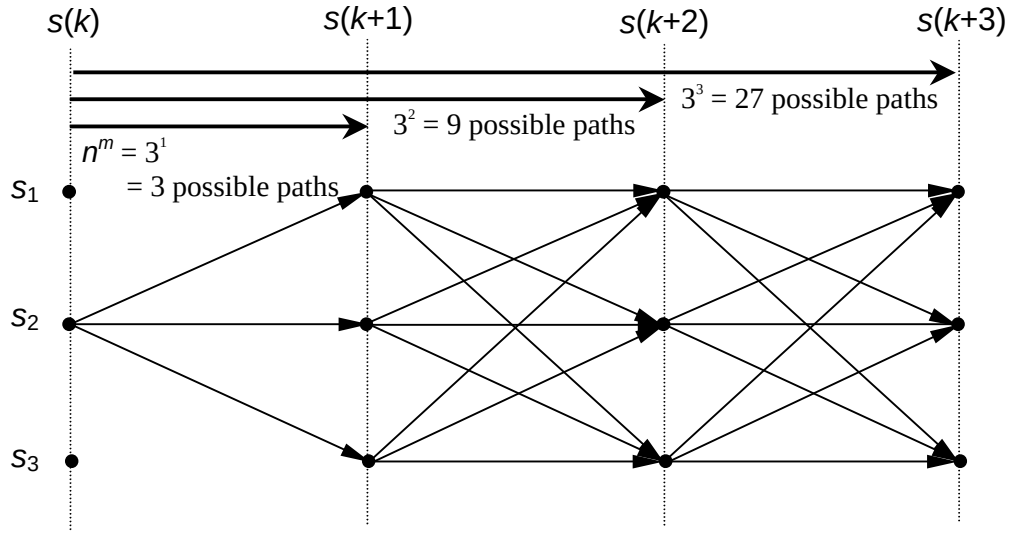


Figure 4.3: Computing Expected Action Utility

increases according to n^m. A single state path in the figure can correspond to more than one sequence of actions, which further expands the number of possible distinct cumulative reward scenarios. At this rate, going beyond a few time steps into the future can quickly become too time and memory consuming to be feasible.

[5] Using a limited number of steps into the future to compute expected utility presumes there is knowledge or an acceptable approximation of the utility of the future-most states.

The mentioned number of possible utility scenarios is for the worst case, while in reality simplifying assumptions are often reasonable. For example, prior constraints on the form of action sequences can reduce the number of possibilities. In addition, state transitions from a given initial state are typically limited to a small part of the state space by nature of the dynamics.

Another way to limit computational complexity is to use approximation methods, such as cutting off expected utility computations at a manageable number of steps and using a heuristic approximation of maximum expected utility at that stage.

The complexity of computing utilities through recursive simulation is significant because it can rule out numerical methods, such as Monte Carlo simulation, that are possible with other alerting philosophies (Yang & Kuchar, 2002).

4.4 Policy Solutions

In general, any function that maps each state in *S* into a unique action is called a *policy* and referenced by the π symbol:

$$\mathbf{a} = \pi(\mathbf{s}) \tag{4.7}$$

The decision rule described in the previous section, based on maximizing expected utility, does this and also ensures a utility-optimal action. An optimal policy is distinguished with an asterisk

$$\mathbf{a}^* = \pi^*(\mathbf{s}) \tag{4.8}$$

The optimal decision rule from section 4.3 would likely be too slow or inaccurate if recursively simulated as described, and a better form would be desirable—for example, this could be a table where pre-computed values can be quickly looked up at each state, or a parameterized function with parameters computed offline.

One efficient policy generation method for the optimal policy is *value iteration*. It follows from the fact that the maximum expected utility at each state, $U(\mathbf{s})$, is related

to that of its neighboring states by the Bellman equation (4.6). If there are n states in S, then there are n Bellman equations $U(\cdot)$ must simultaneously satisfy. Due to the nonlinearity of the Bellman equation there is no guaranteed analytical solution, but iterative numerical methods often work. Starting with an arbitrary function $U(\cdot)$, the Bellman equation is applied at every state in S to generate an updated utility function. This is repeated until convergence of the utility function is achieved. Now, since all of the next-state utilities are all known for any initial state, the policy can be found by applying the maximum expected utility rule (4.5) with a 1-step look-ahead at each state.

Another efficient iterative method is *policy iteration*. Starting with some arbitrary policy π, a utility function $U(\cdot)$ is generated by applying Bellman's equation at each state in a manner similar to value iteration. Then an updated policy is generated using the maximum expected utility rule with $U(\cdot)$. These two steps are repeated until the utility function stops changing.

Both of these common policy generation methods are more efficient than the recursive method described previously, with required memory proportional to n and computation time increasing with n^2qc, where n is the size of S, q is the number of actions, and c is the number of iterations. This is relatively manageable compared to the exponentially increasing resource requirements for forward simulation utility estimation for a single state.

A serious problem remains: the number of states in S can increase exponentially with the number of state variables in the vector $\mathbf{s}$. At worst, the total number of states is the product of the domain sizes of all variables. Because of this, available computing power puts a limit on the number state variables that can be used in the logic. It may be possible to solve a large problem if some part of the domain can be discarded as unlikely to be occupied of physically impossible.

In very large or continuous state spaces it may be necessary to use function approximation methods to model $U(\cdot)$ to reduce the number of parameters that must be computed and stored. For example, neural nets have been used for this purpose

(Tesauro), along with training procedures for determining the necessary value function parameters.

4.5 Partially Observable MDP Alerting

4.5.1 POMDP Solution Issues

Solution methods described in previous sections have assumed full observability so that **s** is known exactly. When the state is only partially observable, so that a belief state must be used, then any recursive prediction approach to calculating expected utility of actions would also involve recursive prediction of future belief states. Or in other words, the decision process requires an awareness of future belief states that could result from different actions. For example, in an alerting problem involving uncertain modes, the alerting system may decide to defer an alert in the knowledge that the mode uncertainty will be reduced by coming events and observations, allowing greater expected safety or a smaller unnecessary alert probability than an immediate alert.

The maximum expected utility for the partially observable case is no longer described by the Bellman equation (4.6), so previously given policy solution methods do not clearly apply. The standard approach to solving POMDP problems is to view the belief state as another state variable with its own state space. The belief state space (*belief space*) is the set of probability distributions possible over the original state space *S*. Viewed this way, the POMDP can be thought of as an MDP where the belief state space replaces *S* as the state domain. Then standard MDP solutions apply.

A remaining challenge is that the belief state space must be indexed so that solution algorithms can step systematically through the space. Another is that even if an index scheme is found, the number of belief states that must be considered may be too large for available methods to be practical, depending on the complexity of possible belief distribution functions. Such POMDP solution issues are an active area of research.

The belief space can be thought of as a parameterized function over the domain *S*, where varying the parameters allows the function to represent the entire range of belief

states. Then, individual belief states correspond to particular parameter choices or coordinates. This makes enumeration of the belief states straightforward. The fewer parameters there are, and the more limited their ranges, the fewer belief states will likely be needed to adequately cover a continuous belief space. This hints of the possibility of approximating a complicated belief space with a reduced-order model, having fewer parameters and needing fewer parameter combinations to span the belief space. Such *belief compression* methods may prove important to design of future alerting systems involving uncertain modes or many state variables.

4.5.2 Belief State Filtering

As described in chapter 2, the combination of probabilistic uncertain dynamics and probabilistic observability leads to uncertainty in the current state in the form of a probability distribution (belief state). The belief state takes into account all available information, including prior knowledge, the transition and sensor models, observations made and past inputs to the system. This combining of information is achieved at each time step using a recursive algorithm:

$$b_{\mathbf{s}(k)} = \alpha \; O(\mathbf{s}(k)) \sum_{\text{All } \mathbf{s}(k-1)} T(\mathbf{s}(k-1), \mathbf{a}(k-1)) \; b_{\mathbf{s}(k-1)} \tag{4.9}$$

The formula states that the updated belief state, $b_{\mathbf{s}(k)}$, is found by summing the transition functions for all possible current $\mathbf{s}(k-1)$, weighting at each state by the current state likelihood $b_{\mathbf{s}(k-1)}$, and then multiplying the resulting function of $\mathbf{s}(k)$ times the observation function, $O(\mathbf{o}(k) \mid \mathbf{s}(k))$. The constant α ensures that the resulting distribution function sums to 1 over its domain. Since the values of $\mathbf{a}(k-1)$ and $\mathbf{o}(k)$ are assumed known, and $\mathbf{s}(k-1)$ is eliminated by the summation, the final expression is a function of $\mathbf{s}(k)$ only. Note that the previous belief state can be either the result of a previous iteration of the formula or a *prior belief state*, $b_{\mathbf{s}(0)}$, that initializes the process. Thus, this formula does incorporate all of the information available in prior knowledge, $b_{\mathbf{s}(0)}$, the process model, $T(\cdot,\cdot)$ and $O(\cdot)$, and the history of past actions and observations (the *evidence*), $\{ \mathbf{o}(0), \mathbf{a}(0), \mathbf{o}(1), \mathbf{a}(1), \ldots, \mathbf{o}(k), \mathbf{a}(k) \}$.

The Markov assumption that was made in defining $T(\cdot,\cdot)$ is important here because it allows the recursive formula shown, as opposed to one that explicitly contains the entire evidence history. Given that the amount of evidence increases with time, such an expression would become increasingly unwieldy and impractical to use with time. For a more detailed discussion of the belief filtering formula and the significance of the Markov assumption, see appendix A.

4.6 Human Modeling Issues

As mentioned in section 4.2.2, the reward function $R(\cdot)$ over the state space *S* is meant as an approximate representation of the goals or priorities of the alerting system. Based on this, the MDP theory generates a policy that is optimal in the utility maximization sense, and presumably rational. The human operators that receive alerts might also be assumed rational, at least ideally, and to act to maximize their own utility function. Such an operator would try to make best use of any signals generated by the alerting system and to do so requires a notion of what the alerting signal means. This is only to point out that whatever $R(\cdot)$ is used for the alerting system involves at least implicit assumptions about operator behavior and preferences. If the reward function conflicts in some way with an operator's understanding of the alerts, or sense of what is justified, the result is improper alerts (section 2.2). This means that when using the MDP design method, which maps alerting preferences directly into the logic, the potential for improper alerts should be considered at the reward function definition stage.

A related issue is the fact that operators learn from observations of the alerting system's behavior, which can cause their understanding of and reactions to alerts to change over time. For this reason it may still be necessary to perform the sort of global trajectory analysis that is commonly used in current methods. For example, Monte Carlo analysis of the alerting logic can provide a more complete picture of the human responses implied by the reward function and operator model that were used, including rates at which evasion commands are modified or reversed, and the overall rates of alerts and incidents. This will help in judging whether the situation model seems consistent with realistic operator behavior. A high alert or incident rate in simulations where an operator

is assumed to conform to alerts could indicate an inherently unacceptable situation where no alerting system would help. In the end, the alerting system design process may take an iterative form where reward and utility functions are arrived at through adjustment of initial estimates.

4.7 Chapter Summary

The alerting problem is restated in terms of the basic elements of a Markov decision process. This includes the state space $\{\mathbf{s}, S\}$, the observation space $\{\mathbf{o}, O\}$, the action space $\{\mathbf{a}, A\}$, the transition function $T(\mathbf{s}, \tau_{\mathbf{a}})$, and the observation function $O(\mathbf{s})$ (for partially observable problems, such as with uncertain modes).

The method also requires defining alerting system performance requirements (outcome preferences) in terms of a trajectory utility function $U_\tau(\tau)$, which associates a utility value with every possible trajectory τ. The utility function is so defined that for any two outcome trajectories, the relative utilities describe the degree of alerting system preference between the two. The trajectory utility function corresponds to a reward function $R(\mathbf{s})$ that gives the reward (utility contribution) of the situation reaching a particular state. The utility of a trajectory is the cumulative reward from the sequence of states passed through.

Decisions are made by the maximum expected utility principle. Ideally, the decision that would result in the maximum utility is preferred, but state uncertainty (due to the stochastic transition and sensor functions) makes exact utility impossible to compute. Expected utility is considered the next best decision metric.

Expected utility can be difficult or impossible to estimate quickly through forward simulation of trajectories. Because alerting decisions must occur in real time, a more efficient means of estimating expected utilities is needed. This is achieved through policy generation, in which a function relating each state, or belief state, to an expected utility and optimal action is pre-computed using Bellman's equation. The policy is stored in a table or other form that allows real-time retrieval of function values.

Alerting processes involving uncertain mode variables have more complicated policy solutions, because the policy is then a function of a belief state rather than a regular state (making it a POMDP problem). One common solution method involves representing the belief state as a parameterized function, so that the range of possible belief states corresponds to a range of parameter values. Then the problem can be approached in the same way as a regular MDP.

The reward function of an alerting system represents the alerting system's preferences, which in turn should be related to and in agreement with the preferences of operators who receive alerts. Otherwise, the alerting system will produce improper alerts that harm long-term performance.

5. Applying the MDP Method to an Aircraft Passing Scenario

5.1 Introduction

In this chapter the benefit of MDP methods is shown for a simplified aircraft collision avoidance problem. Figure 5.1 shows the scenario of interest. Two aircraft approach one another at similar altitudes so that there is some nominal probability of a collision. A collision will be defined as the crossing of one aircraft through a protected region about the other, as shown. One aircraft, called the *evader*, has the option of using a climb evasion maneuver if needed to avoid a collision with the other aircraft, or *intruder*. Each aircraft moves randomly in altitude over time, and this translates into a randomly changing vertical separation between the two. This randomness makes

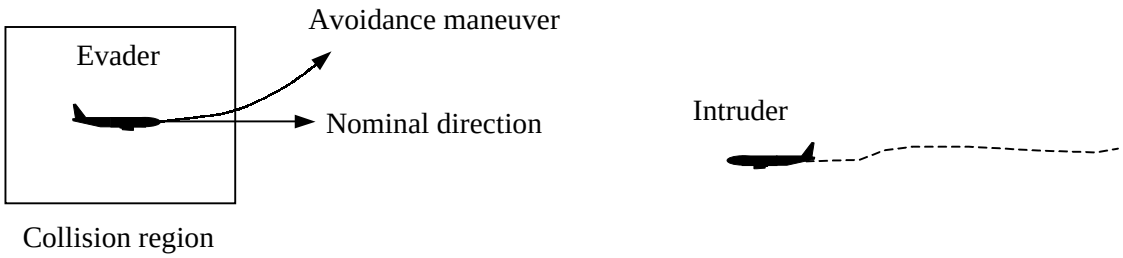


Figure 5.1: Aircraft Encounter Scenario

exact forecasting of a collision impossible, so it is not clear if or when an evasion maneuver is needed. This is a basic example compared to many realistic alerting situations (e.g., where additional avoidance options or 3-dimensional position might be involved), but the simplifications cause no loss of generality in the methods described, and could be relaxed.

Figure 5.2 shows a model of the 2-aircraft system. The aircraft are constrained to a plane, with the evader fixed at the origin of the relative position axes, x and y. The relative horizontal speed is assumed constant, while the relative vertical speed can vary in

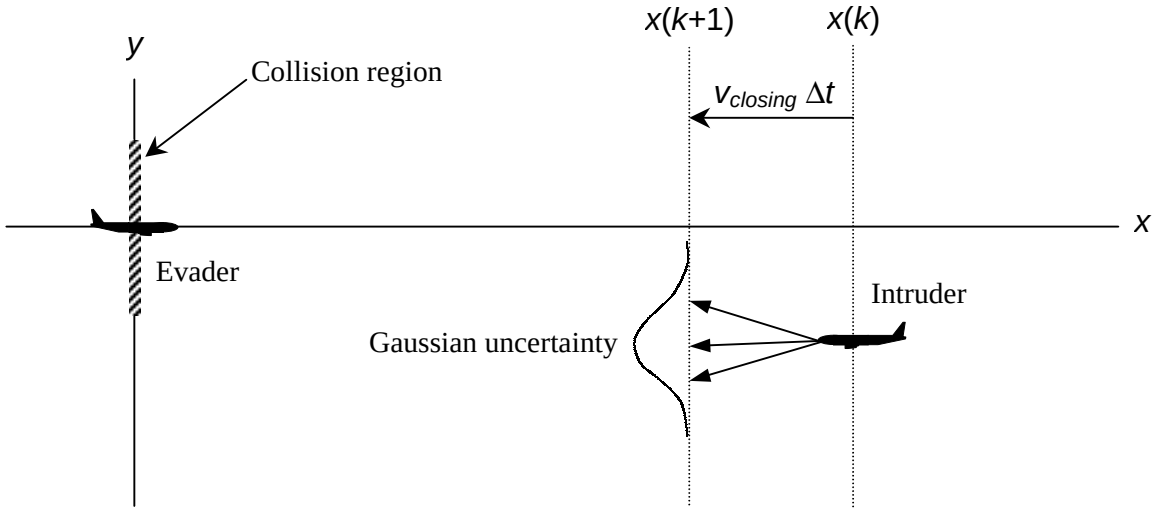


Figure 5.2: Markov Encounter Model – Nominal Dynamics

either direction over time. Prior to any alert, the relative motion is described by a process of the form

$$x(k+1) = x(k) + v_{closing}\,\Delta t \tag{5.1}$$

$$y(k+1) = y(k) + v_w(k)\,\Delta t$$

with the following definitions

$x(k+1)$	Horizontal relative position at the next time step
$y(k+1)$	Vertical relative position at the next time step
$x(k)$	Current horizontal relative position
$y(k)$	Current vertical relative position
Δt	Constant time increment
$v_{closing}$	Constant horizontal closing rate

$v_w(k)$ Gaussian white sequence climb rate disturbance input

The first equation describes the constant horizontal closing rate as a discrete-time process. The second says that the future vertical position of the intruder is a discrete-time Markov random process depending only on relative altitude. Vertical position is predictable only within a normal distribution whose mean is the current position. The variance of predicted vertical position increases linearly with the number of steps into the future (Brown & Hwang).

While the previous equations describe the nominal dynamics that apply before any evasion maneuver occurs, the following one describes the vertical position during a climb escape maneuver (figure 5.3):

$$y(k+1) = y(k) + (v_w(k) + v_{climb}) \Delta t \qquad (5.2)$$

The difference is the addition of a constant bias, v_{climb}, the average climb rate. The horizontal motion is identical for the nominal and evasion cases.

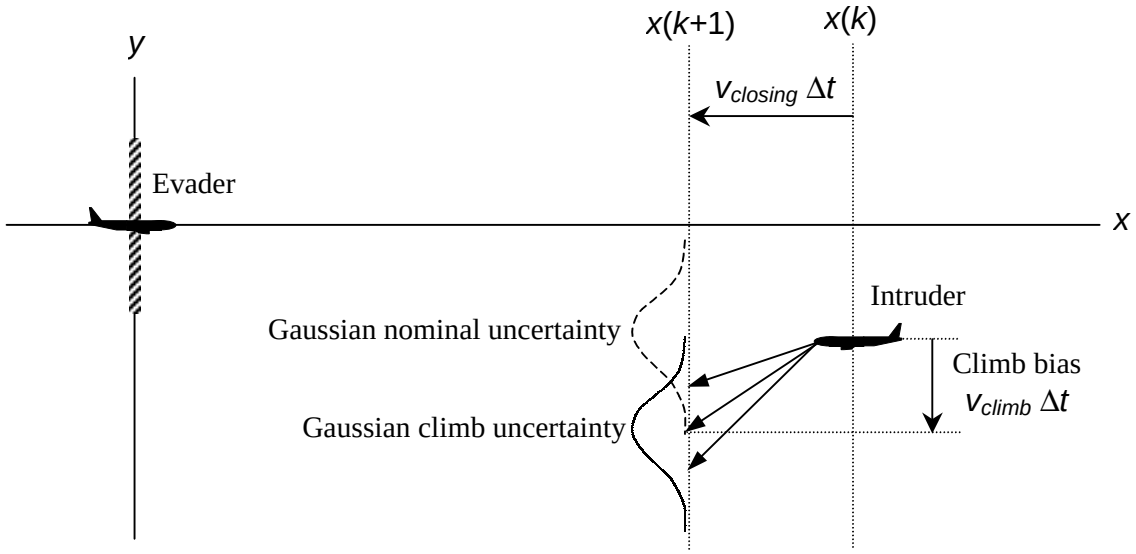


Figure 5.3: Markov Encounter Model – Climb Dynamics

A climb by the evader causes a relative descent by the intruder, as figure 5.3 shows.

Since the evader is fixed at the origin, a collision happens if the intruder crosses through a region about the origin. Because vertical motion is slow relative to horizontal, the collision region is simplified for convenience from an area around the evader to passing within 100 vertical feet at the y-axis, as shown in the figures with a hatched bar.

To simplify the discussion, this model has only two physical state variables: the relative vertical and horizontal positions. A more realistic description of aircraft dynamics could use additional state variables, such as the absolute speeds and positions of each aircraft, but the same design principles would apply. The next chapter will consider one possible improvement, namely the inclusion of a dynamic mode variable to distinguish between normal and failure situations.

Parameters were chosen to simulate an aircraft encounter with a horizontal closing speed of $v_{closing}$ = 440 knots (743 ft/sec). An input noise standard deviation of v_w = 1,858 ft/min (31 ft/sec) was assumed with a 0.32 sec time increment Δt. This is equivalent to a vertical drift standard deviation of 100 ft after a 24,000 ft (approximately 4 nautical mile) horizontal interval. The simplified model produces random trajectories that tend to remain near the initial relative altitude, with possible moderate climbing or descending. A mean evasion climb rate v_{climb} of 1,500 ft/min (25 ft/sec) is assumed.

Figure 5.4 shows an example trajectory resulting from these dynamic assumptions over a 32 second interval. For the first (right) half the nominal dynamics are used, and for the second half the climb dynamics are used. The collision region is also included as a shaded bar about the evader.

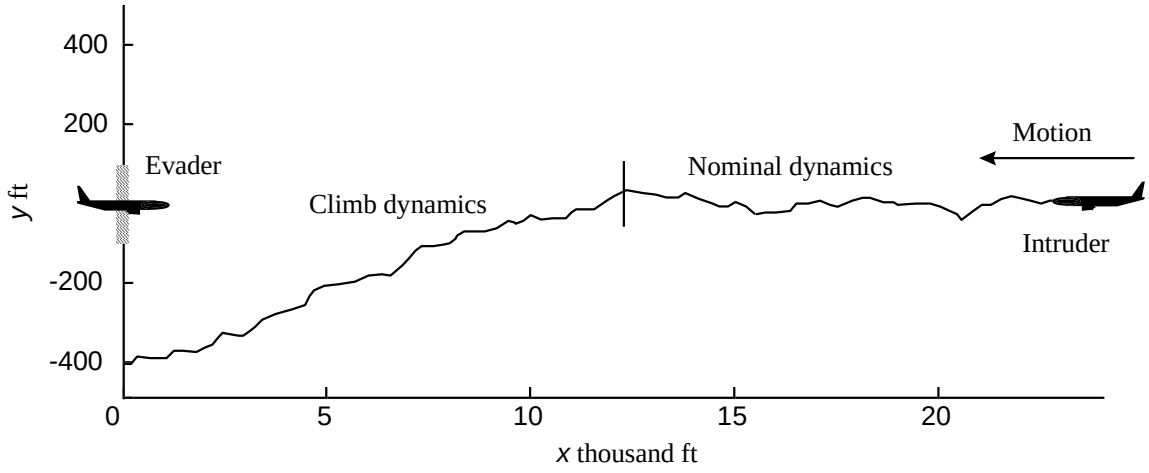


Figure 5.4: Example Trajectory: Nominal and Climb Dynamics

5.2 Basic Reward Function for Alerting

The MDP framework requires that all of the goals or preferences of the alerting system be represented in terms of the relative utilities of events that can happen. Then the decision process becomes a matter of acting to maximize expected utility at each step.

In SOC (system operating characteristic) analysis the quality of alerting decisions is measured in terms of whether alerts happen, whether they are needed, and whether the resulting trajectories are safe. There are desirable and undesirable final outcomes, and as discussed in section 2.2, it is typically impossible to guarantee that only desirable ones happen. A trade-off must be made between them. SOC plots are one tool for analyzing the trade-off, stating it in terms of the probabilities or rates of positive and negative outcomes.

Making a trade-off between different outcomes requires knowing the relative desirability of the outcomes. In a utility approach such preferences are expressed by

assigning numeric values to the outcomes. Outcomes with larger numbers have greater desirability, and the larger the difference between two possibilities, the greater the preference of one over the other. In figure 5.5 possible alerting outcomes are divided into a set of mutually exclusive events. These are all of the possible combinations of the three above-mentioned categories from SOC theory, six in all (the two shaded event blocks are excluded because they are impossible.) Recall that an alert is considered successful if no incident happens, and necessary if an incident would have happened without the alert.

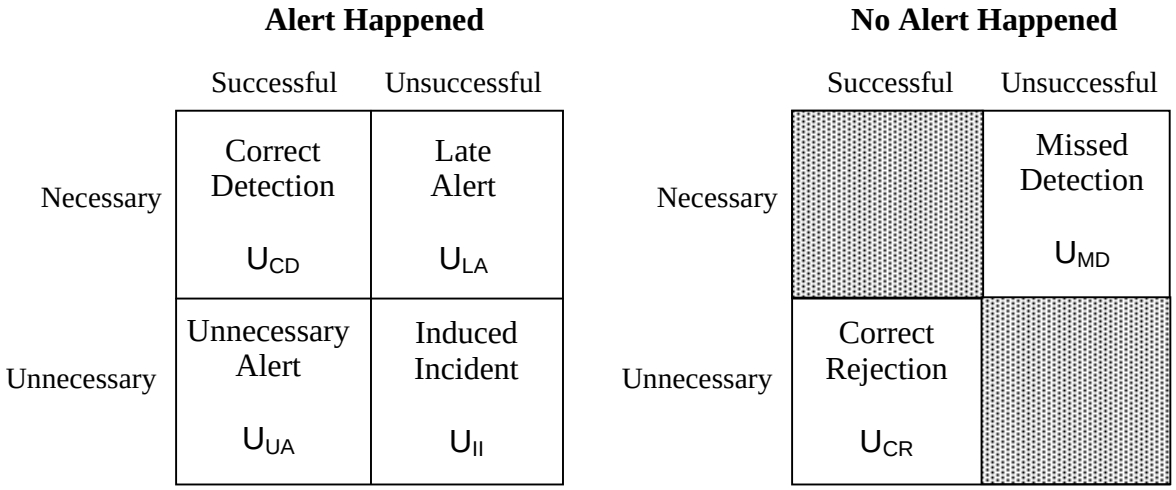


Figure 5.5: Utilities of Alerting Outcomes

In a Markov decision process approach to the alerting problem this information about outcome preferences must be expressed as a trajectory utility function, which will be assumed to take the form

$$U_\tau = \sum_{t=k}^{n} R_t \tag{5.3}$$

where the total utility for the future trajectory is a sum of the rewards gained at each state. This is the additive rewards utility function (eq. 4.4) described in section 4.2, where the discounting factor is set to 1. This is reasonable here assuming rewards are well bounded and that the alerting process ends at some point in time so that the trajectory utility is finite. A convenient end point for the aircraft passing situation is when the intruder passes the y axis: after this point the intruder no longer poses a danger, so the alerting system can stop monitoring it.

A utility ranking of the outcomes could be defined as follows

$$U_{incident} \equiv U_{LA} = U_{II} = U_{MD} \tag{5.4}$$

$$U_{SA} \equiv U_{CD} = U_{UA}$$

$$U_{incident} < U_{SA} < U_{CR}$$

This says that all collision outcomes have equal utility, all successful alert outcomes have equal utility, and correct rejections are preferred over all other outcomes. In some ways this ranking may seem oversimplified; a correct detection seems intuitively better than an unnecessary alert, and an induced collision might be considered worse than a missed detection or late alert. On the other hand, once an alert occurs it is impossible to know whether it was necessary or not in order to make such utility distinctions, because the alert precludes observing the nominal trajectory. So whether there is any practical difference between the different kinds of incident is debatable. Examples in this thesis will assume the (5.4) outcome utility scheme, but it should be noted that the generality of utility does allow for other schemes if needed.

In its favor, this utility scheme conforms to with existing SOC philosophy in that there are two main error types—incidents (unsuccessful alerts) and unnecessary alerts—that should both be avoided if possible to maximize utility. Otherwise some optimal trade-off between the two is needed. This agreement with SOC philosophy allows a direct performance comparison between MDP-based and SOC-based logics in the next chapter.

Specific utility values for the three outcome categories are chosen to best reflect the degree of preference between them. For example, the utility of a successful alert is probably much nearer to that of a correct rejection outcome than a collision. Assuming $U_{incident}$ is fixed, the nearer U_{SA} becomes to U_{CR}, the less the alerting system would expect to gain from deferring alerts, and the earlier alerts would tend to occur. This is true regardless of the utility of a collision, as long as it is less than the other two. Thus, relative utility can be of more significance than absolute utility when assigning utilities.

Recalling that the trajectory utility is a sum of the rewards for each state passed through, one possible reward function is shown in figure 5.6. If the intruder arrives at a point within the collision zone about the evader, then an incident happens and the incident utility, $U_{incident}$, applies. Setting the end-state reward value equal to the total trajectory utility constrains the previous terms of the reward function in that their values must sum to zero. For collision cases, the most obvious solution is to let all previous state rewards be zero. However, there still two kinds of non-collision outcome to consider.

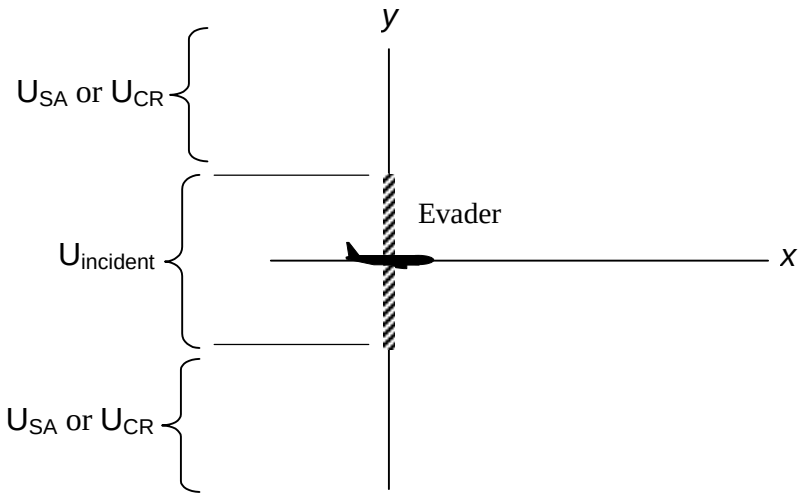


Figure 5.6: End-State Reward Definitions

While there is no ambiguity about the appropriate collision outcome utility—it applies regardless of the previous trajectory taken—the non-collision utility depends on whether or not an alert occurred before the end. An alert happens the first time the climb action is selected. If there was no alert and the trajectory ended in the safe range then U_{CR} is the appropriate outcome utility. If an alert did occur, then the U_{SA} utility applies instead. This conditionality on the alert status suggests that lumping the trajectory utility into a single end-state reward as in collision would require a third state variable in addition to x and y. This will be designated AS (alert status), and is a discrete two-state variable indicating a state either *Before* or *After* an alert.

Figure 5.7 illustrates the 3-variable state space and end rewards. It can be thought of as two position planes, one for each alert status, with different end-position rewards.

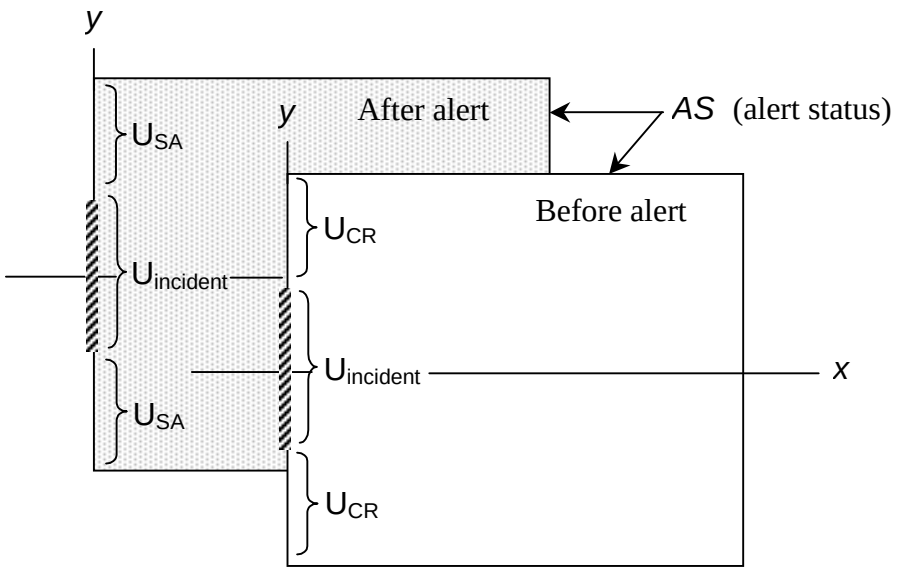


Figure 5.7: State Space with Alert Status Variable

The system trajectory originates in the front plane at the right edge, and moves leftward according to the nominal dynamics as long as no alert occurs. If no alert ever occurs, the trajectory will encounter either the incident or the correct rejection end-state utility. If an alert does occur, the state transitions from the *Before* to the *After* alert status plane, and remains there for the remainder of the scenario. In that case, non-collision end positions have the successful alert rather than the correct rejection reward. For any trajectory, this arrangement assigns the correct outcome utility as defined by the (5.4) expressions. Thus, no additional rewards or penalties are needed from trajectory states before the end state, and this results in a reward function of the form

$$U_\tau = R(x_f, y_f, AS_f) \tag{5.5}$$

5.3 Alerting Policy and Threshold

As mentioned in chapter 4, the most desirable solution to the alerting problem is a policy function that determines the next action from the current state with a minimum of

computation. This section describes the policy generation process for the example alerting problem, based on the reward function defined in the previous section.

This problem is convenient in having a well-defined process end condition—reaching the y position axis. It also has dynamics that guarantee an orderly traversal of the state space, with no chance of occupying a single state twice (the horizontal position always moves forward, and in the same direction.) This allows generating a policy in a direct and fast way that avoids the iterative methods often necessary with MDP problems. An inconvenience of this system is that it has a variable with a continuous domain (y) whereas the MDP method assumes discrete states. This is dealt with by using a discrete approximation of y.

The approach is to generate a function that gives an expected utility for each possible action at each state. Then, by the maximum expected utility principle, the preferred action is the one with the highest expected utility. In this case the utility function has a 4-variable domain, including three state variables in figure 5.7 and the action.

First note that for any end state, the expected utility is the exact utility given by the reward function defined previously (5.5). Next, note that the state one step prior to that one is in the set of states where $x_p = x_f - \Delta x$, where x_f is the location of the end state and Δx is the increment between horizontal positions. Knowing the probabilistic system dynamics T and the end-state expected utilities $U(x_f, y_f, AS_f)$, the expected utility of the prior state, $\mathbf{s}_p = \{ x_p, y_p, AS_p \}$, for a particular action a is given by

$$U_a(x_p, y_p, AS_p) = E(U(x_f, y_f, AS_f) \mid x_p, y_p, AS_p, a) \tag{5.6}$$

$$= \sum_{\text{All } \mathbf{s}_f} U(x_f, y_f, AS_f)\, T(x_f, y_f, AS_f \mid x_p, y_p, AS_p, a)$$

$$= \sum_{\text{All } \mathbf{s}_f} U(\mathbf{s}_f)\, T(\mathbf{s}_f \mid \mathbf{s}_p, a)$$

Figure 5.8 illustrates the variables and functions involved. From the definition of the expected value, $U_a(x_p, y_p, AS_p)$ is the summation of the end-state utility function

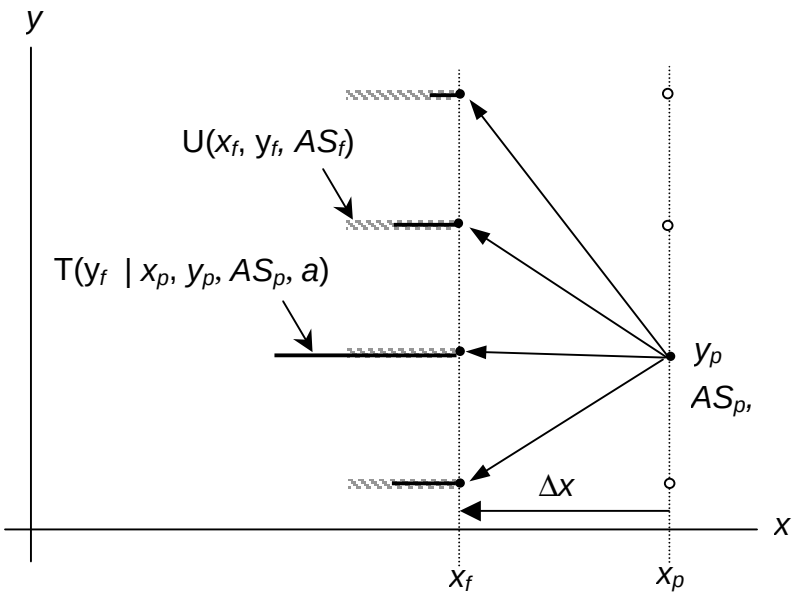


Figure 5.8: Utility and Probability Distribution of Last Step

$U(x_f, y_f, AS_f)$, weighted by the final-state distribution (transition function) $T(x_f, y_f, AS_f \mid x_p, y_p, AS_p, a)$, over all possible end states, $\mathbf{s}_f$. In the figure, the black line at each x_f state represents the probability of transitioning to that state, and the hatched bars represent the expected utility of the state. The two quantities are multiplied at each state and then summed to give the expected utility at $\mathbf{s}_p$. In this case x_f and AS_f are determined exactly by the previous state and a, so the transition function is really a distribution over y_f. The above formula (5.6) gives the utility for a single action, so the calculation must be repeated for each action. The overall utility of the prior state is defined as the maximum value for the possible actions

$$U(x_p, y_p, AS_p) = \max_a [\, U_a(x_p, y_p, AS_p) \,] \qquad (5.7)$$

and the maximizing action is the policy output for that state. Notice that what is referred to as the "utility" of a state is really the maximum expected utility for that state, but the simpler term is sometimes used for convenience.

Using the above formulas (5.7), the utility can be determined for every state in the x_p subset. The resulting utility function then becomes the basis for finding utilities for the next layer back, $x = x_p - \Delta x$, by a similar procedure as with (5.6 and 6.7). This is repeated for every step backward until a complete utility function and action policy is obtained for the entire state space.

Figure 5.9 shows the pre-alert utility function for each action, resulting from applying the described method to the example system with the defined reward function (5.5) and outcome utilities of $U_{CR} = 1.1$, $U_{SA} = 1$ and $U_{incident} = 0$. Each plot is of the utility of taking the indicated action at all positions in the plane. The utility functions are

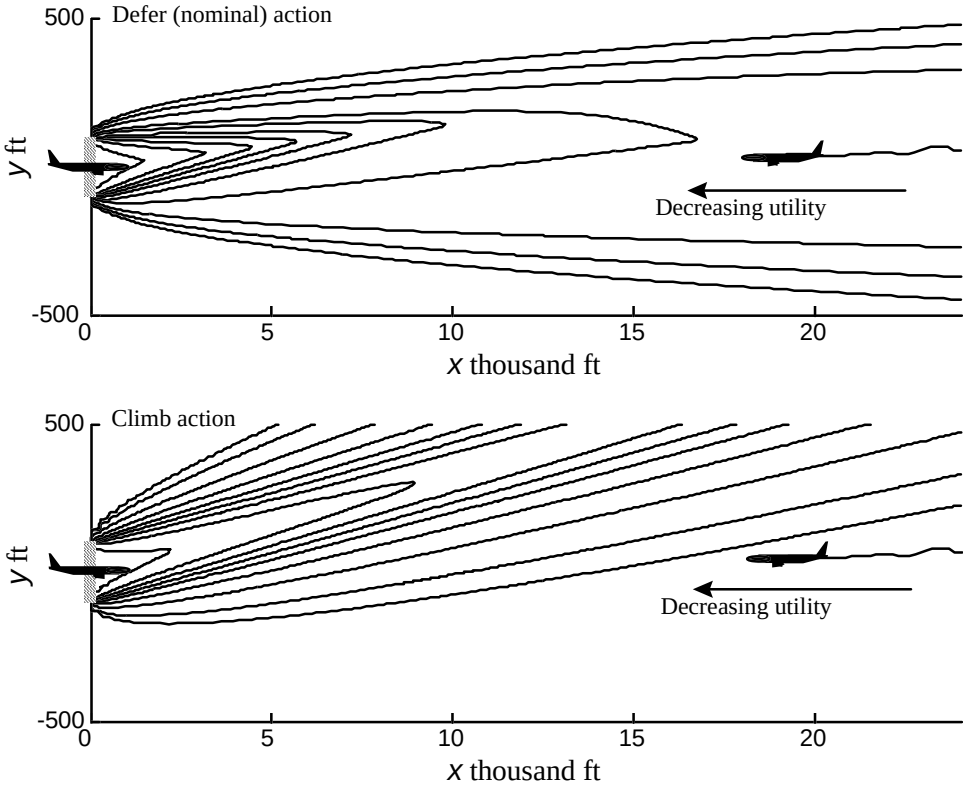


Figure 5.9: Constant-Utility Contours Before Alert

represented by constant-utility contours. In each plot, the utility function minimum is $U_{incident}$ and occurs at the x-y origin, and increases outward from there. In the deferral or nominal action case (top) the maximum utility is U_{CR} and occurs in the broad, level area

outside the outermost contour. In the other case the maximum is U_{SA} and is also in the outermost area.

Figure 5.10 shows the complete utility function, covering both before and after-alert states. The before-alert plots to the left are the ones from figure 5.9. After the alert, the nominal and climb actions are still both available, but unlike before the alert there is no possibility of gaining a U_{CR} reward at the end. Maximizing utility now means maximizing safety, since safe alert outcomes are still worth more than collisions. The after-alert utility functions for the nominal and climb actions are very similar. The reason is that regardless of which action is taken at a given state, both actions are known to be available at the next step, and since the position increment is small, not much safety is lost due to lost time in a single step.

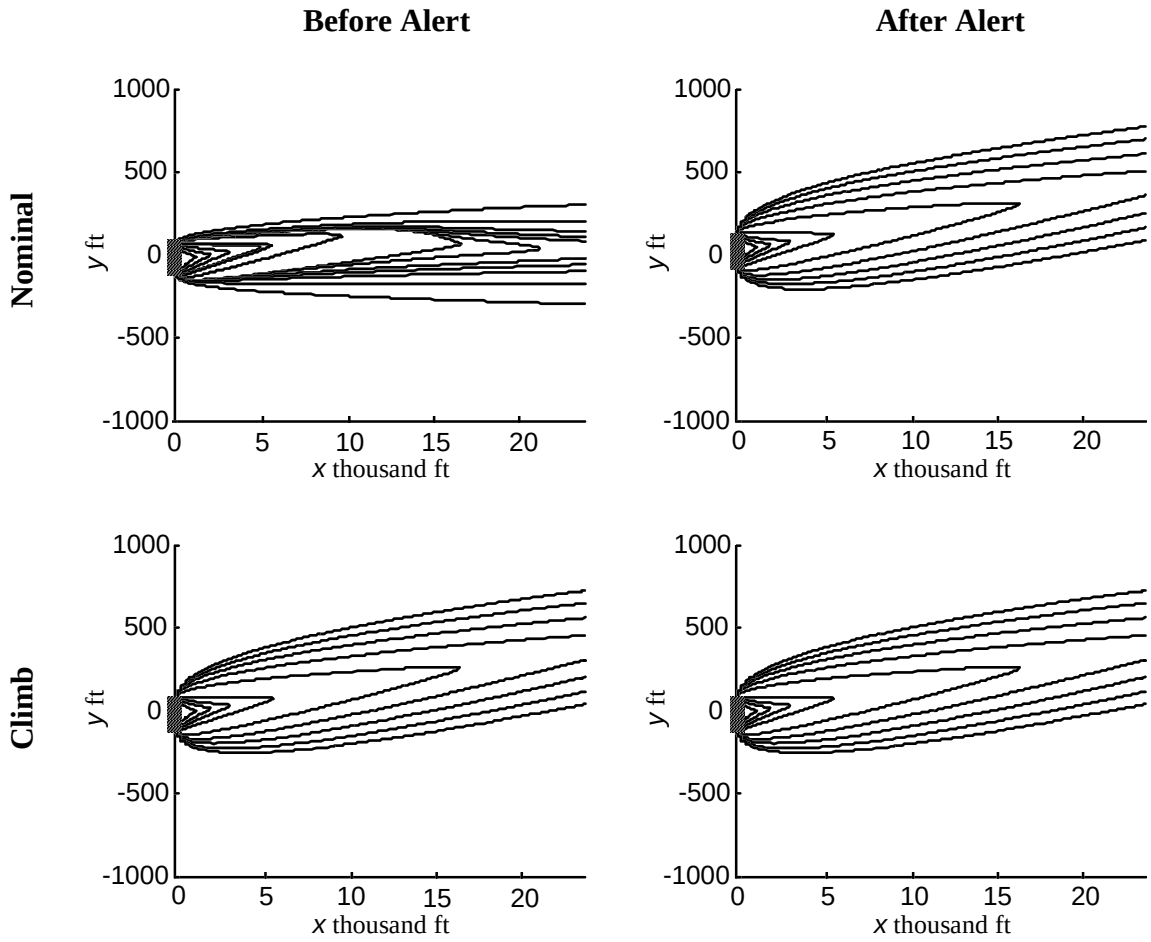


Figure 5.10: Constant-Utility Contours Before and After Alert

The utility functions for the climb action before and after the alert are identical, because the distribution of the next state is the same for both.

If the state begins at **s** = { x = 24,000 ft, y = 0 ft, AS = *Before* } and moves leftward on a collision course, and assuming alerts are suppressed (so no AS transition occurs), a distinct utility trace results for each possible action. These are as shown superimposed in the figure 5.11 plot of utility vs. horizontal distance. Far from the collision there is a slight utility benefit for the nominal action over the climb, hardly visible on the plot. This relative benefit becomes progressively smaller with distance until the nominal and climb actions have the same utility where arrow indicates, at 11,000 ft separation. Beyond this point the nominal action has a lower utility, and both utilities keep decreasing as long as the collision path is followed, due to the decreasing probability of avoiding the collision.

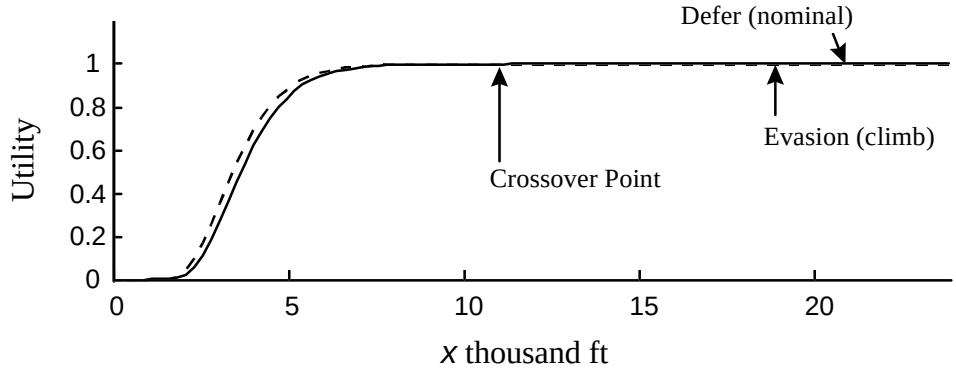


Figure 5.11: Utility vs. Distance for Head-On Collision Trajectory

Following the maximum expected utility principle, an evasion maneuver would have occurred at the crossover point. The location of the crossover depends on the relative values of the basic outcome utilities: U_{CR}, U_{SA} and $U_{incident}$. If U_{CR} becomes greater relative to U_{SA} while holding $U_{SA} - U_{incident}$ constant, then the crossover point will be delayed longer, because it is more worth risking a collision to avoid alerting.

The reason the expected utility benefit of the deferral action is so small initially is that with the defined system dynamics and initial position, the probability of avoiding a collision through normal random drift is small. If the initial vertical position were further

from zero, or the variance of the random process were increased, the utility of the nominal action would be nearer to its maximum of 1.1.

Since the alert threshold is where the two before-alert action options have equal utility, the position-plane threshold is the zero-value contour of their difference, defined by

$$U_{nominal}(x, y, \textit{Before}) - U_{climb}(x, y, \textit{Before}) = 0. \quad (5.8)$$

The two functions $U_{nominal}$ and U_{climb} are the ones from figure 5.9. Figure 5.12 shows the threshold contour, along with the collision trajectory that produced figure 5.11. This is the threshold policy.

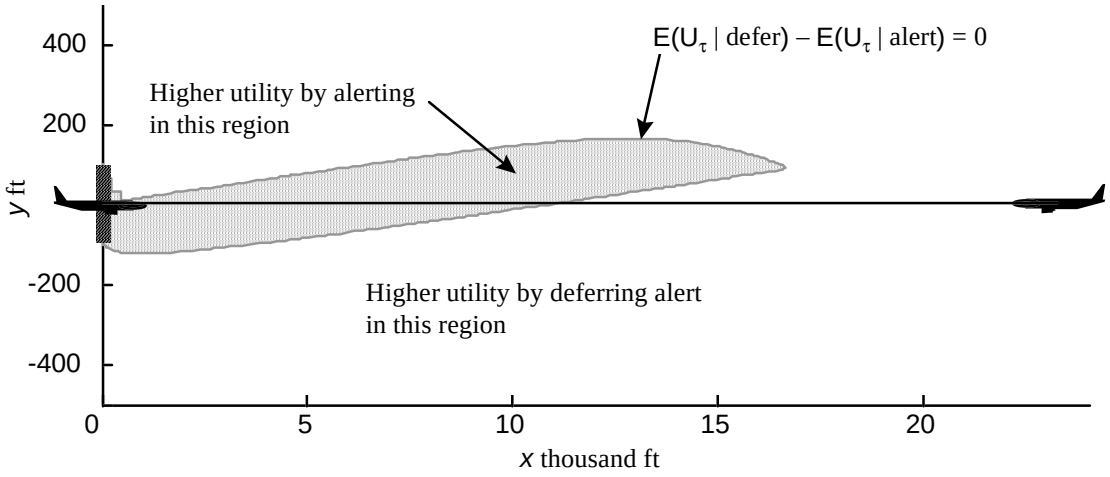


Figure 5.12: Alerting Threshold (Equal-Action-Utility Contour)

The intruder begins at the right edge of the plane and moves at constant speed. According to the assumed dynamics, the vertical position should change randomly, but an idealized trajectory is used to more clearly show the utility trend in figure 5.11. As long as the intruder is outside the threshold contour, the deferral action is preferred. At the threshold, the climb action should be chosen, causing the alert status *AS* to switch to the *After* value.

5.4 After-Alert Guidance

During the deferral phase of an encounter, the assumption is that humans in the situation are in control and the alerting system is only passively monitoring. The first use of the climb action marks the initial alert displayed to the operators. At this point the situation enters the *AS* = *After* state as mentioned. In this state the alerting system still considers both the nominal and climb options, by assumption, but they have different meaning in that the alerting system is now actively guiding the situation. The nominal action is then not a deferral of action, but a command to take a non-climb action. But there is no general requirement that post-alert guidance commands correspond one-to-one to pre-alert options as in this example.

In the same way as the *Before* alert mode, guidance (*After*) mode action switching occurs along a zero-utility-difference contour in the position plane. Now this contour is formed using the two right-side utility functions from figure 5.10. The contour is shown in figure 5.13, along with the original alerting threshold, which no longer applies. Also drawn is the idealized head-on collision scenario, but where alerts are no longer suppressed.

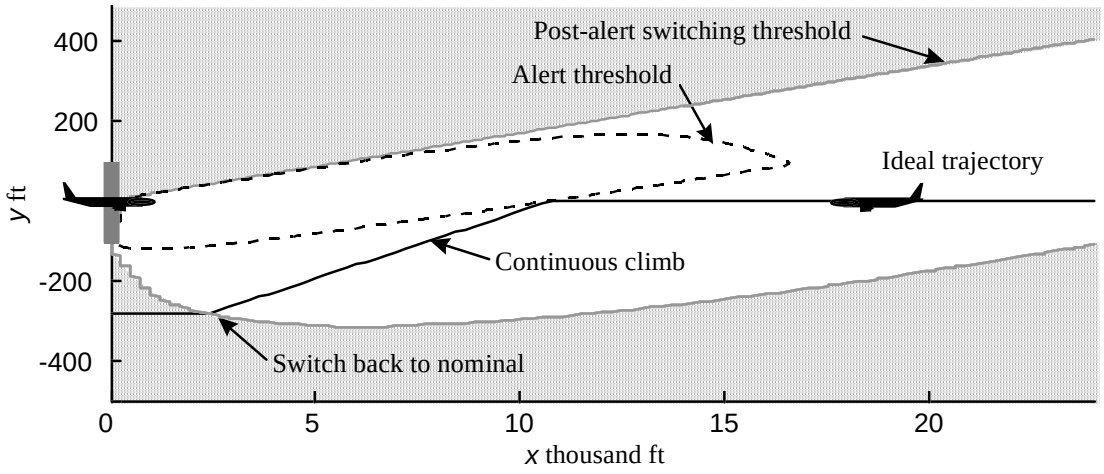


Figure 5.13: Guidance Switching Threshold

When the trajectory reaches the alerting threshold at x = 11,000 ft, a climb action is chosen, the alert status switches to *After*, and the threshold contour is replaced with the

guidance switching contour. Just after the alert, the position is in a region where the climb action has the highest utility, and a continuous climb occurs until the new contour is reached. At the lower contour edge and beyond, the climb and nominal utilities are equal, at least within the precision of the computed utility functions. The nominal action was chosen as the default for such a condition, so the final trajectory segment past $x = 2500$ ft is a sequence of nominal actions. In contrast, at the upper contour edge (the straight line extending from the origin) the switch is critical, because above this the nominal action becomes relatively safe (the utility is higher) compared to the climb.

Figure 5.14 shows an example of command switching at the upper contour. In this case a crossing of the alert threshold occurs, but rather than climbing as directed the vehicle continues on a level path. The suggested action continues to be a climb until the contour crossing at the 9,000 ft point. Figure 5.15 shows the action utility traces

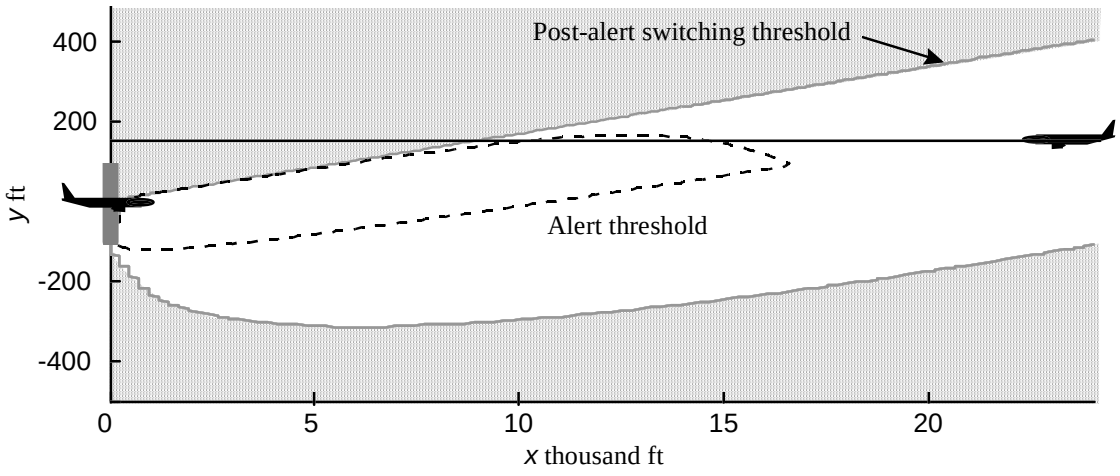


Figure 5.14: Alert Guidance Reversal Case

corresponding to the trajectories in figure 5.13 and 5.14. The vertical dotted lines mark the action switch points. In the normal evasion example, the utility at the alert point is nearly 1 (figure 5.15a), which reflects that there is little chance of a collision (in which case $U_\tau = U_{incident} = 0$) to reduce expected utility below the U_{SA} level ($U_\tau = U_{SA} = 1$). All through the evasion maneuver, expected safety and utility of both actions remains near 1. For the figure 5.14 trajectory, where the climb fails to occur when it should, the initial utility of the nominal action (5.15b) is larger than in (5.15a) due to the 150 ft

vertical offset of the initial position. Nearer to the evader the deferral utility drops due to the decreasing safety of the climb evasion option. After the alert, the utility of both actions declines until the 9000 ft point, where the nominal action is once again safer and a command switch occurs. After this, the nominal action continues to be the safer of the two until the end.

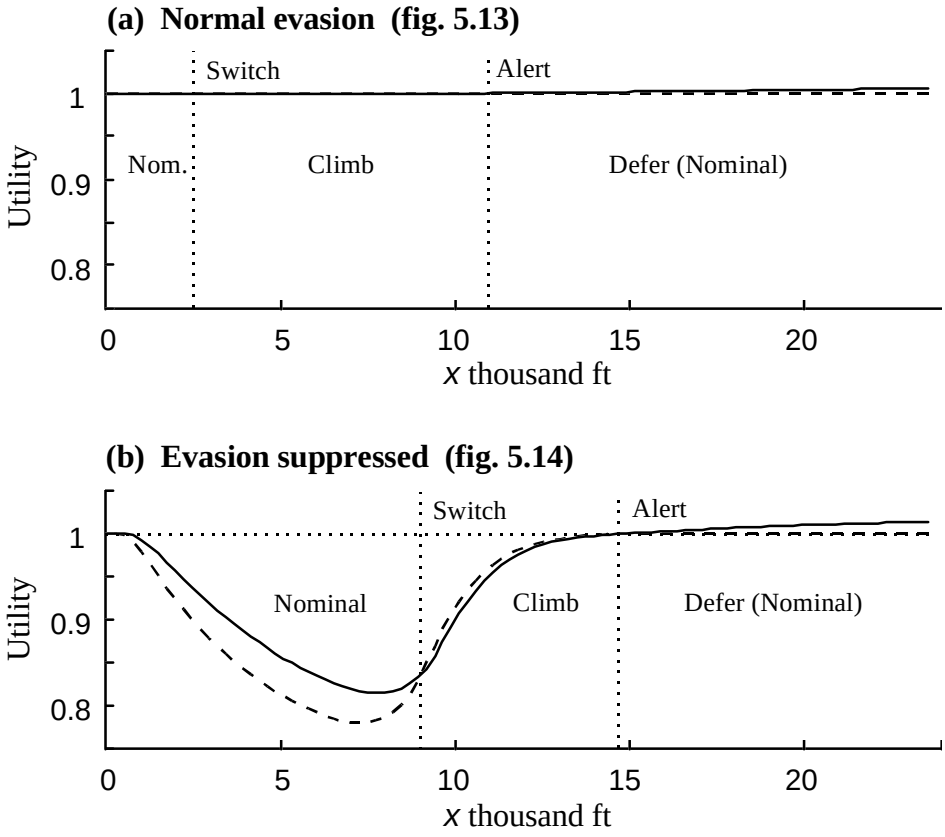


Figure 5.15: Utilities for Guidance Switching Trajectories

Discussing this example has involved statements conflating equating with utility during the after-alert phase. While valid here, it is a special case. In general, safety and utility have a more complicated relationship. For example, in *Before* alert states, a nominal action utility of 1 could result from averaging more than one combination of prior incident ($U_\tau = 0$) and correct rejection ($U_\tau = 1.1$) and successful alert ($U_\tau = 1$) probabilities. After the alert the only two utility outcomes are 0 and 1, so collision probability and utility are equivalent.

5.5 Cumulative Rewards: Excessive Guidance Switching

The reward function defined in section 5.2 works as hoped for the idealized trajectories considered up to now, using only an end-state reward. A more thorough analysis with realistic sample trajectories reveals a problem that figure 5.16 illustrates. The original policy results in a guidance switching contour that is fixed for the duration of the encounter. This means that under some conditions, it may be possible for to the state to cross the threshold in one direction and quickly return, causing rapid switching. This is the case in figure 5.16, where randomness in the vertical position path causes the position to waver at the threshold, resulting in 5 switches over several seconds.

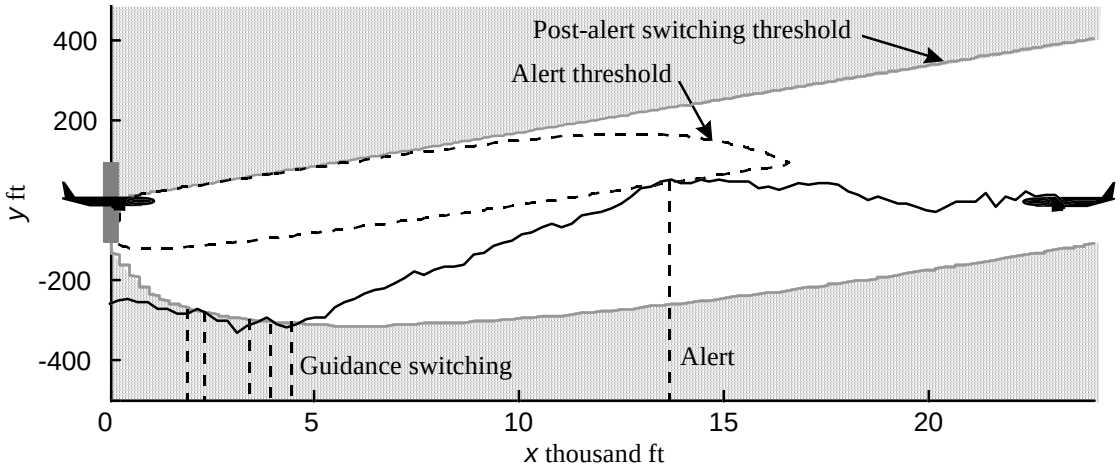


Figure 5.16: Excessive Guidance Switching

Human operators can be expected to follow guidance, but only within limits. Due to unavoidable response delays, rapid command switching may be impossible to follow accurately. In addition such commands can seem irrational or confusing, and cause an unpredictable response.

One way to deal with this is to build operator limitations like response time into the transition function, adding state variables if needed. An example is adding a timer state variable for keeping track of response delays. Another way is to define the reward function with operator preferences in mind, so that alerting system preferences are in agreement with operator preferences. For example, the operator's reluctance to switch

actions during an evasion maneuver can be modeled as a penalty (negative reward) for any step where a switch happens. In this example a combination of both was used to against the excessive switching problem.

The main idea was to apply a penalty for any change of action happening after the initial alert. At each decision point this requires comparing the action options to the previous action. To recall the previous action, a new variable, $a_{prev} \in \{ Nominal, Climb \}$, must be added to the state. The compete state is then $\mathbf{s} = \{ x, y, AS, a_{prev} \}$. Because the policy must assign an action for every possible combination of values of the state variables, this is potentially a function over $N_x \times N_y \times N_{AS} \times N_{a_{prev}}$ states, where each factor is the number of values of the corresponding variable. In this case the set of states can be reduced by noting that any combination where $AS = Before$ and $a_{prev} = Climb$ at the same time is impossible. This reduces the domain size by one fourth.

The new reward function for the current state and candidate action option is

$$R(\mathbf{s}, a) = \begin{cases} R_s(\mathbf{s}, a) + R_f(\mathbf{s}_f) & \text{if } \mathbf{s} = \mathbf{s}_f \\ R_s(\mathbf{s}, a) & \text{otherwise} \end{cases} \tag{5.9}$$

where $R_f(\mathbf{s}_f)$ is the original end-state reward function (5.5), which is independent of a and a_{prev}, and $R_s(\mathbf{s}, a)$ is an additional switching "reward" term defined as

$$R_s(\mathbf{s}, a) = \begin{cases} -\rho & \text{if } a \neq a_{prev} \text{ and } AS = After \\ 0 & \text{otherwise} \end{cases}$$

where ρ is the magnitude of the penalty for an action change. The action utility functions are generated in a similar manner as before, solving backward from the end states, except at each state a penalty is subtracted from the expected next-state utility if a_{prev} differs from the assumed next action.

The resulting alerting policy is shown in figure 5.17, for an assumed penalty of 0.01 per switch, and otherwise the same parameter values as before. The alert threshold is not visibly changed by the new penalty. There is significant change, however, to the after-alert switching thresholds. Whereas the old policy allowed back-and-forth switching of guidance actions along a single contour, the new policy has two contours, one for each action, that are physically separated except along a shared boundary at the top of the alert contour. The separation of the two contours makes rapid back-and-forth switching less likely to happen in the course of evasions similar to the one drawn. In particular, the example scenario in figure 5.16 would most likely have ended with a continuous climb evasion if this policy were used.

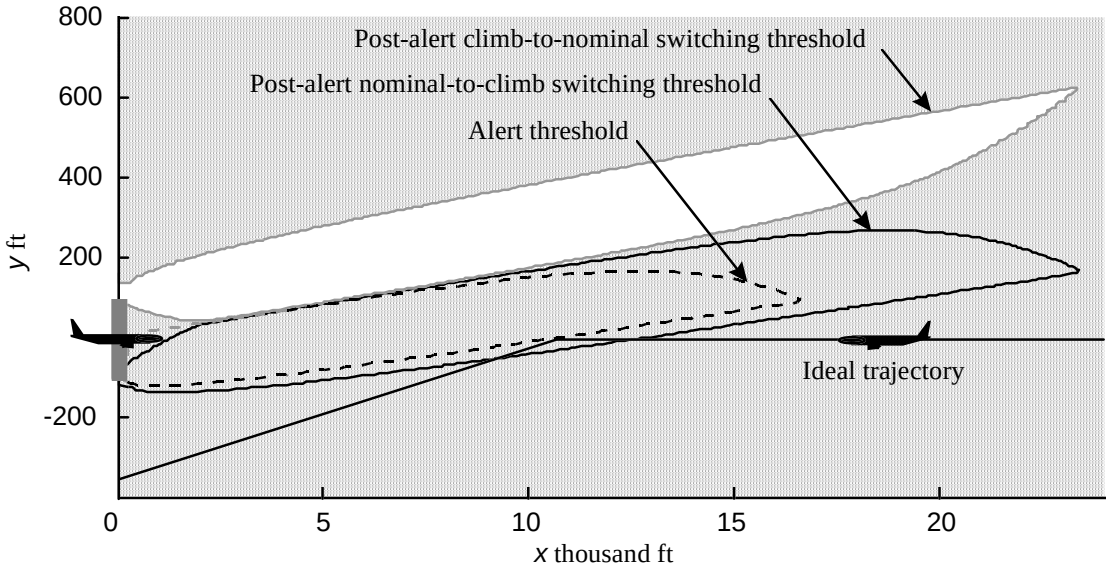


Figure 5.17: Action Switching Contours with Switching Penalty

5.6 SOC Performance Analysis

The mentioned shared switching boundary is reachable by trajectories such as trajectory (a) in figure 5.18. This feature remains despite the penalty on switching because, near the collision zone, the penalty is overwhelmed by a safety-related rapid loss of expected utility as the incident approaches. In addition to this problem, another issue is the possibility of a trajectory like (b), which follows the upper edge of the alert threshold, and ends in a collision without ever triggering an alert. Preventing scenarios like these could require further changes to the reward function or system dynamics, or adding more control options such as a descend action.

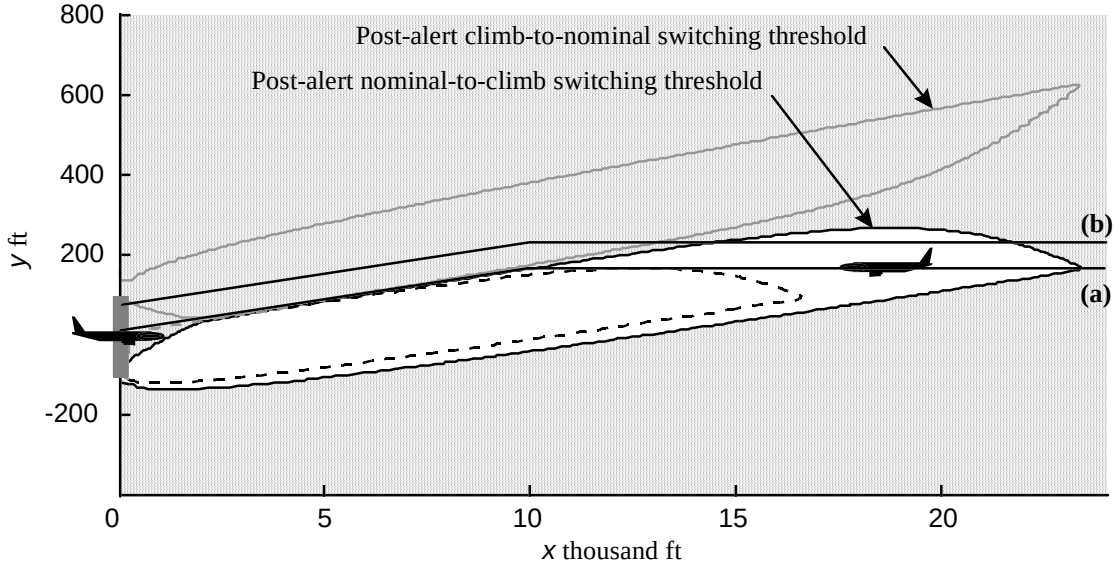


Figure 5.18: Problem Trajectories

Whether changes are needed also depends on the likelihood of the failure. If the prior probability of a failure scenario is low enough, the policy may be considered acceptable as it is. To aid making design choices like this one, or to assess the overall benefit of an alerting system, global performance metrics such as prior incident probability (as mentioned) and unnecessary alert rates remain useful alongside utility decision metrics.

The SOC plot in figure 5.19, for example, helps understand the performance of the example policy, and the effect of adding a switching penalty. Random trajectories were generated beginning at a y position uniformly distributed between –1000 and 1000 ft, and with the alerting logic generating alerts and guidance. This was continued until 10,000 alerts occurred (many trajectories did not result in alerts). Two versions of the logic were simulated in this way: the original logic and the one with penalties on switching. Only trajectories in which an alert (or missed detection, which is treated as an alert) occurred were used in generating the SOC plot, which describes the conditional probabilities of successful and unnecessary alerts, given that an alert happened. As shown, the resulting safety is 0.991 for both, accurate within about 0.002 with 95% certainty. Their difference is within 0.0026 with 95% certainty. The unnecessary alert rates are 0.41 for the original and 0.40 for the version with penalties (within about 0.01 with 95% uncertainty) and their difference is 0.01 within 0.014. While the original logic produces an average of 1.4 action switches after the initial alert, the version with a switching penalty produces 0.04, or an approximate reduction by a factor of 35. This is unsurprising, as the original logic guarantees at least one switch during any successful alert scenario, as figure 5.13 makes clear, and any penalty would likely prevent that switch.

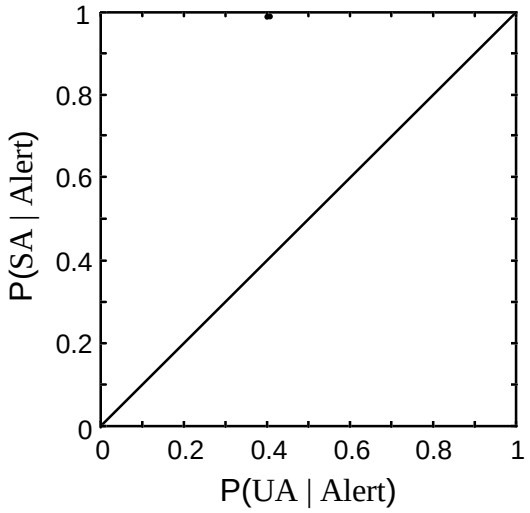


Figure 5.19: Global Performance: With and Without Switching Penalty

5.7 Chapter Summary

This chapter used a simple aircraft collision avoidance process to illustrate the use of Markov decision process modeling to generate an alerting logic from requirements. This involved

- Identifying the Markov state of the process.
- Describing alerting system outcome preferences in the form of a reward function, including defining new state variables if necessary.
- Generating an efficient policy based on the state and reward function.

Also included was a discussion of the need to tailor the alerting threshold and guidance to operator preferences and ability, and demonstration of one method of reducing undesirable guidance switching through state and reward function modifications. Finally, the importance of using global performance analysis (such as Monte Carlo trajectory simulation) along with reward function design was discussed.

6. Aircraft Encounter with Uncertain Modes

6.1 Introduction

The 2-aircraft system from chapter 5 was convenient in that all of the necessary state variables, including position, alert status, and knowledge of the previous alerting system actions, were exactly known at all times. This made the alerting problem amenable to basic MDP methods, including a straightforward policy solution. Many systems of interest lack this property, making it necessary to estimate variables that are unmeasurable or poorly measurable. A special case of this is where a system can operate in multiple modes, changing mode randomly and infrequently. A particular Markov state and transition model might describe its behavior well most of the time, but badly on rare occasions, such as when a failure occurs in the system. A discrete, but unmeasurable mode variable might then be defined as an index between the regular dynamics and an alternate model that better describes the other types of behavior.

In much of airspace, normal operations keep aircraft well separated through standard procedures or air traffic control oversight. Collisions, when they occur, tend to happen after a breakdown of these mechanisms. This is an instance where an unobservable mode variable could improve a Markov model over what is possible with a fully observable encounter model like the one from chapter 5.

6.2 Modified Aircraft Encounter System

Figure 6.1 sketches a multiple mode encounter similar to one considered by Kuchar (Kuchar, 1995). The evader nominally flies level and at a constant speed. The intruder aircraft descends from above and is supposed to level off at an altitude a safe distance D above the evader, as the relative trajectory shows. There is a chance,

however, that it will instead continue descending at the initial rate, passing near or colliding with the evader. The probabilities of these two events are 0.75 and 0.25 respectively.

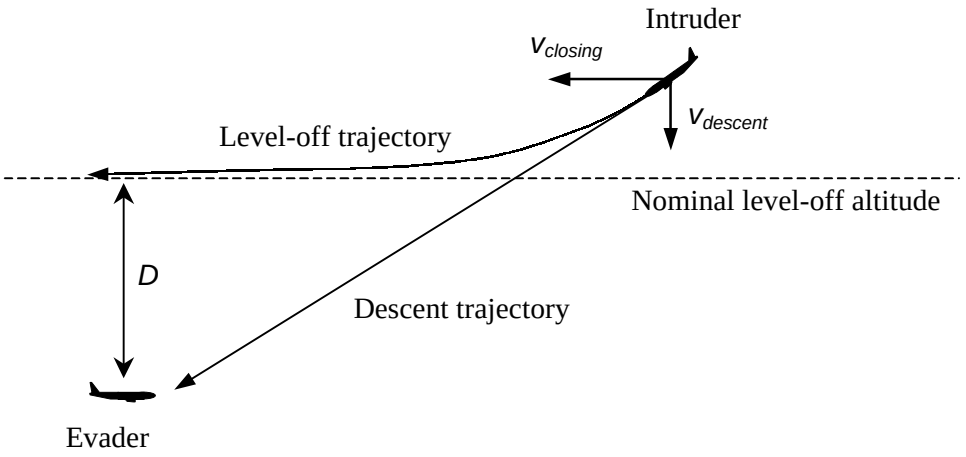


Figure 6.1: Aircraft Encounter with Level-off Mode

As with the chapter 5 system the horizontal speeds of the aircraft are assumed constant and equal, with a total closing rate $v_{closing}$ of 440 knots. Horizontal position and time are discrete with increments of Δx and Δt. The Δx increment is 400 ft, for a Δt of 0.54 sec. The initial descent rate $v_{descent}$ of the intruder is 2500 ft/min and the level-off separation D is 1000 ft. Prior to any alert and before leveling off, the nominal descent is described by the function

$$y(k+1) = y(k) - v_{descent}\,\Delta t \tag{6.1}$$

For the level-off mode and before any alert, the deceleration and level flying phase is described by the function

$$y(k+1) = D + \Delta y_{level}(k)\, e^{-\beta \Delta x} \tag{6.2}$$

where $\Delta y_{level}(k)$ is

$$\Delta y_{level}(k) = y(k) - D \tag{6.3}$$

which is the deviation of the intruder from the level-off separation. The coefficient β is 0.0003 1/ft, which results in a maximum vertical deceleration of 0.29g from the initial 2500 ft/min descent rate or an average of about 0.08g over 17 s until a 1 ft/sec vertical rate is reached, assuming there is a smooth transition from the descend to the level-off function. For the descend mode, the intruder instead maintains the vertical speed $v_{descent}$ all the way to (nominally) a direct collision with the evader, so function (6.1) applies through the entire encounter.

After an alert the evader receives a climb command, and is assumed to accelerate instantaneously to v_{climb}, a 1500 ft/min climb rate. This is to approximate a constant 0.25g pull-up, which is a relatively aggressive maneuver spanning a shorter time compared to the level-off maneuver of the intruder.

Prior to any alert, the evader is assumed to fly level at an altitude a distance D from the intruder's level-off altitude. Once the evader begins a climb, its relative position is no longer a valid reference point for predicting the level-off trajectory of the intruder. To resolve this issue it will be assumed in advance that alerts will not happen before the separation D has been reached. Under this assumption the intruder will either have leveled off already, or be in descend mode where no reference altitude is required, when an alert happens. The system dynamics following an alert are then approximately

$$y(k+1) = y(k) - v_{climb}\,\Delta t \tag{6.4}$$

if the intruder is in level-off mode, and

$$y(k+1) = y(k) - (v_{descent} + v_{climb})\,\Delta t \tag{6.5}$$

if the intruder is in descend mode. If an alert does occur before D is crossed, it will be assumed that the intruder has nearly finished leveling off if in level-off mode, and equation (6.4) will be applied. It should be kept in mind that errors may be large in this range.

Equations (6.1) through (6.5) describe the nominal dynamics of the system. Small disturbances in the vertical separation due to sensor error and other factors (wind, piloting error etc.) are modeled using a Gaussian white sequence input, v_w, as in chapter 5

$$y(k+1) = y(k) - (v_{descent} + v_w)\,\Delta t \tag{6.6}$$

$$y(k+1) = D + \Delta y_{level}(k)\, e^{-\beta \Delta x} + v_w\,\Delta t \tag{6.7}$$

$$y(k+1) = y(k) - (v_{climb} + v_w)\,\Delta t \tag{6.8}$$

$$y(k+1) = y(k) - (v_{descent} + v_{climb} + v_w)\,\Delta t \tag{6.9}$$

This input describes the total deviation from the nominal path due to all noise and disturbances. Since y is relative position, v_w includes the uncertainties for both aircraft. With the level-off dynamics (6.7), the disturbance will cause y to drift randomly, but the exponential term will tend to counteract this and keep it near the level-off altitude in the long run. When any of the other (constant vertical rate) functions are in effect, v_w will induce random walk behavior, where the variance of vertical position uncertainty grows linearly with distance into the future (Brown & Hwang). A v_w value of 1560 ft/min (26 ft/s) is assumed for all dynamic equations. This leads to a 30 ft steady-state standard deviation for vertical position in the level-off mode, and in descend mode or during a climb a 108 ft standard deviation after 24,000 horizontal feet.

Figure 6.2 shows the nominal level-off and descend-mode (or "blunder") trajectories for the figure 6.1 scenario, along with randomly generated sample trajectories for the specified parameter values and without alerts. Again, the evader is fixed at the origin so the relative position of the intruder is what varies with time. Vertical separation is defined as the intruder altitude minus the evader altitude, so that the scenario begins with a positive vertical separation. Horizontal separation is defined similarly.

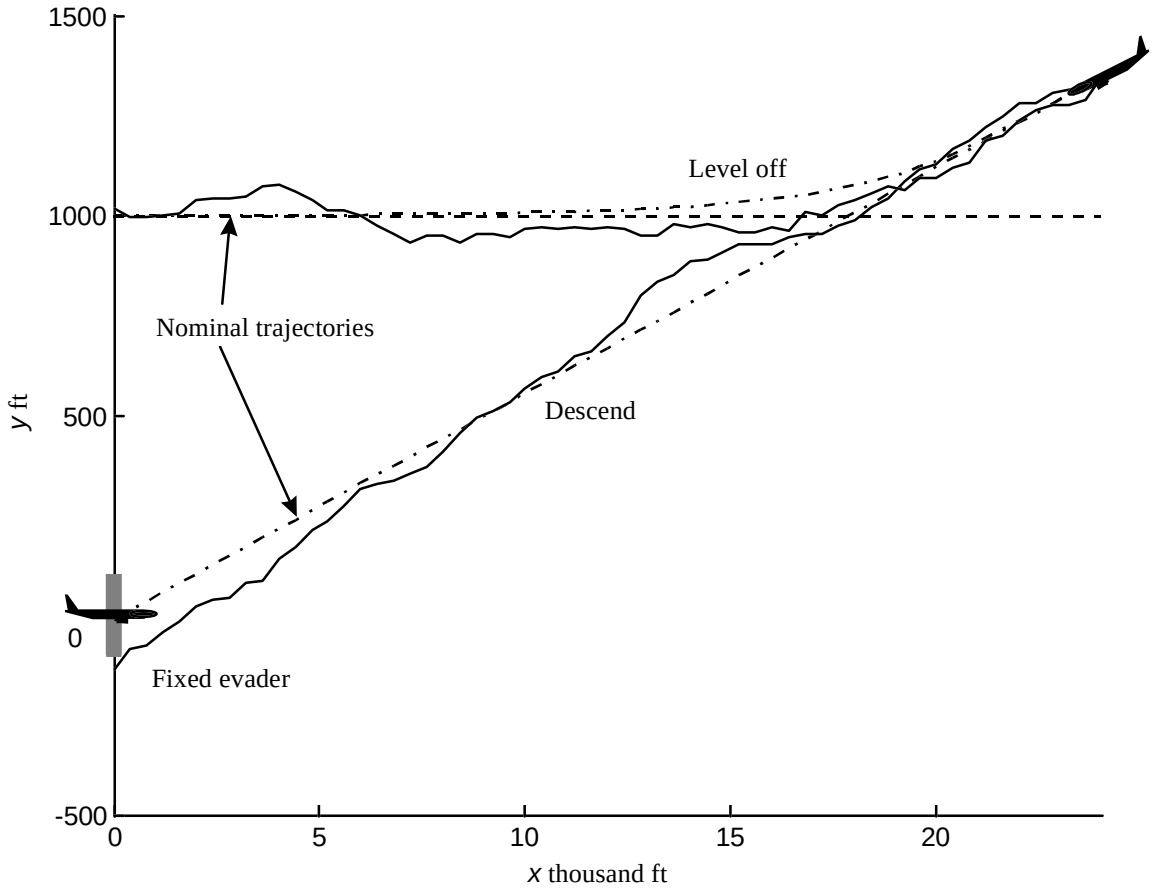


Figure 6.2: Typical Relative Trajectories

6.3 Policy with Belief State

Unlike the variables x, y, AS, and a_{prev} from the previous chapter, the encounter mode, which will be referred to as m, cannot be exactly determined at each time step. Knowledge of m is available in the form of a distribution over its two values. The combination of m and its distribution is a belief state, as defined in section 2.3.1. The domain of the belief state is the set of all possible distributions, of which there are an infinite number in this case. Using a discrete approximation, figure 6.3 illustrates the domain of the m belief state as an orderly progression through the distribution space.

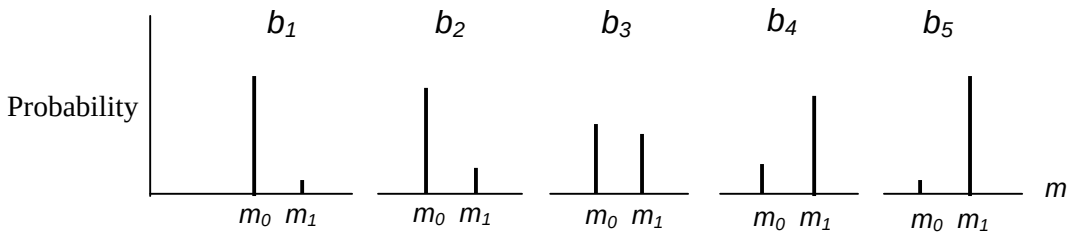


Figure 6.3: Discrete Mode Belief State Domain

In the diagram, m_0 is the level-off mode and m_1 is the descend mode. Represented this way, there is little practical difference between the belief state of m and regular states. Both kinds represent the current, complete knowledge or beliefs about variables in the Markov state, so both are needed to make the most accurate prediction of the future state and events. Both kinds of state are observable in a sense, either by direct measurement or, for the belief state, by Bayes updating based on the previous belief state and input to the system.

The usual approach to MDP problems involving belief states (POMDPs) is to enumerate the belief state domain, if necessary using approximations such as the discrete set in figure 6.3 in place of the full domain, and to include an index for the belief state as a dimension of the state space (Russel & Norvig). Adding the belief state index to the variables previously defined results in the total state

$$\mathbf{s} = \{ x, y, AS, a_{prev}, BS \} \tag{6.10}$$

where BS is the belief state index

$$BS \in \{ b_1, b_2, b_3, b_4, b_5 \} \tag{6.11}$$

The state **s** has a domain at most the size of the product of the sizes of all component variables

$$N_{\mathbf{s}} = N_x \times N_y \times N_{AS} \times N_{a_{prev}} \times N_{BS} \tag{6.12}$$

which increases linearly with the number of possible belief states, N_{BS}.

For ease of finding a policy, the fewer states there are, the better. As in the last example, some states in the above space are clearly not reachable by definition of the corresponding Markov dynamics and possible initial states: any state where $AS = Before$ and $a_{prev} \neq a_0$ (defer) can be eliminated by replacing the two variables with a single combination state variable, CS, whose domain is only the possible combinations of those variables.

$$\mathbf{s} = \{ x, y, CS, BS \} \tag{6.13}$$

Other than the addition of the belief state index, BS, this is similar to the state of the previous example system. The constrained way the state space is traversed, with each x value occurring only once and in fixed order, is also similar. An identical reward function can be used, and for simplicity an action switching cost of zero (see eq. 5.9) will be assumed. The policy can be computed using the same procedure as before, starting at the end-state layer and working backward through state space to generate utility functions for the nominal and climb actions.

Figure 6.4 shows the alerting threshold in the x-y plane. Because there is an additional dimension in the state space, several threshold contours are drawn, representing the different discrete belief states. In other words, a contour crossing will only result in an alert if the belief state at that time is in the belief state range for the contour. In actual use the alerting system will not be forced to choose one of the five discrete belief states in order to use the policy. Instead, a belief state can be maintained in the full, continuous domain and used with interpolation to estimate the action utilities from the discrete policy. The same method is also employed with the vertical position y, since the system dynamics are not discrete in that dimension.

To show the boundaries more clearly, figure 6.5 draws the alert space for each belief state separately.

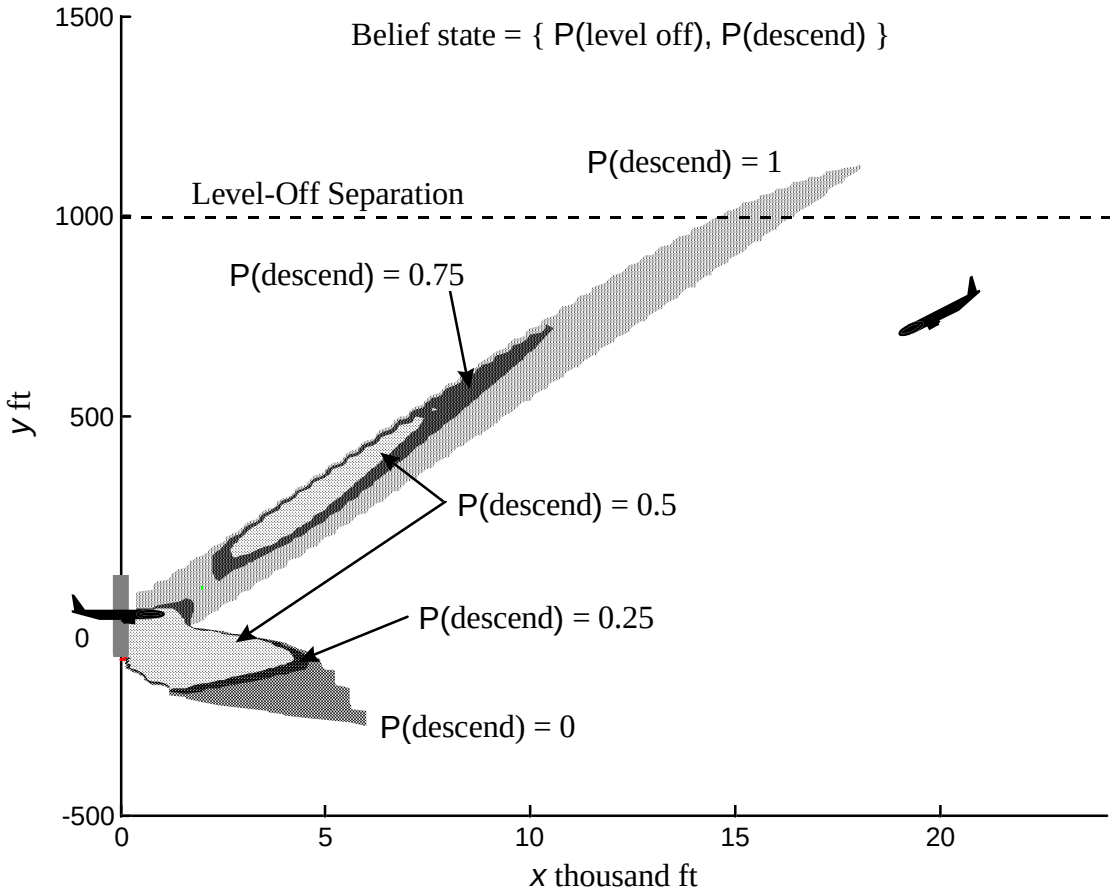


Figure 6.4: Threshold Contours for a Range of Belief States

With the chosen parameters, an alert could apparently occur even before the level-off separation is passed if the descend mode is completely certain (mode belief state b_m = [0 1]). On the other extreme, where there is zero probability of the descend mode (b_m = [1 0]) the intruder would actually have to be at the same altitude or below the evader for an alert to be desirable. In the co-altitude configuration the evader has a clear climb path and can easily escape at the last moment with little chance of an induced collision, regardless of the mode. When the intruder approaches from above in descend mode, the possibility of an induced collision reduces the utility of the climb action relative to deferring. In between the two belief state extremes (e.g. b_m = [0.5 0.5]) there is greater uncertainty about the path of the intruder, so the alerting system tends to be defer alerts longer, knowing that an unnecessary alert and command reversal or an induced collision would likely result.

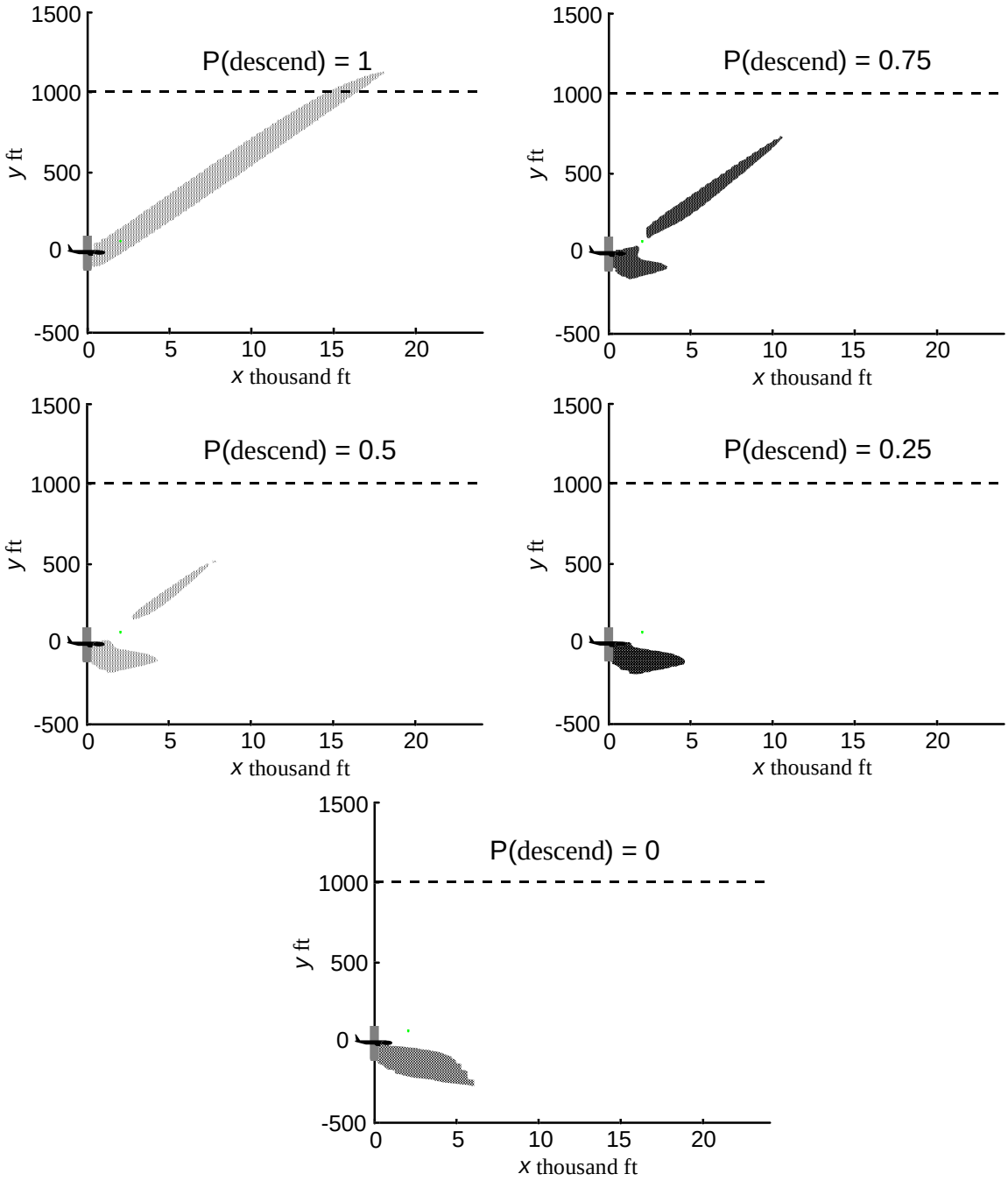


Figure 6.5: Threshold Contours Separated

Because the alerting threshold depends on the current mode belief state, continuous belief updating is necessary as described in chapter 4 and appendix A. This means using measurements of observable state variables (position), the most recent belief

state, and the assumed dynamics for each mode to determine the most reasonable current mode distribution.

6.4 Level-off Scenario

Figure 6.6 shows an idealized (with no disturbance input) level-off scenario, where the intruder begins at a 2500 ft/min descent speed and decelerates smoothly to stop

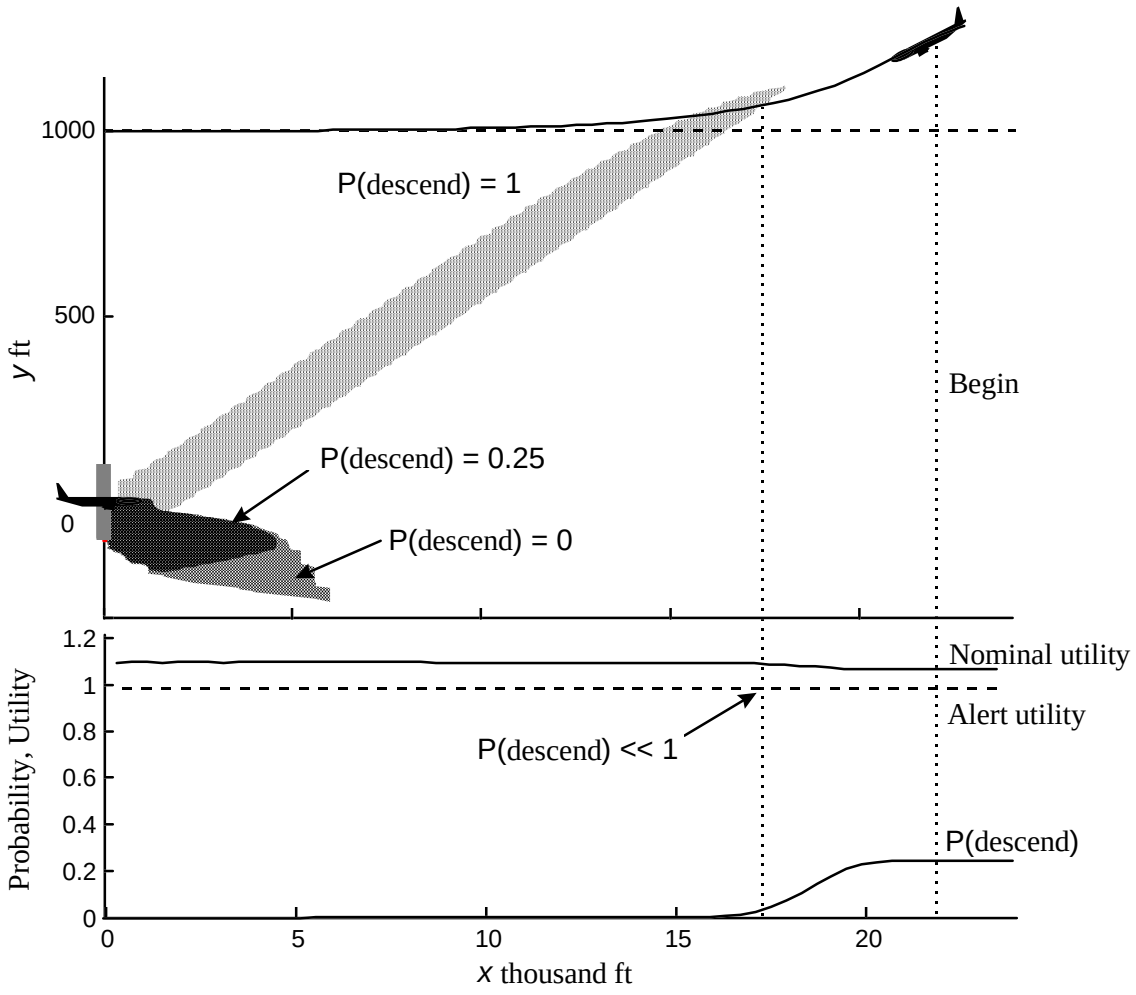


Figure 6.6: Level-Off Scenario

at a target altitude 1000 ft above the evader. In the process the outer alerting contour is crossed, but no alert occurs. The reason for this is made clear in the plot underneath, which shows a trace of the different action utilities and the mode belief state (the descend-mode probability) over the course of the scenario. Initially there is a 0.25

probability that the intruder will fail to level off, posing a significant danger to the evader. As the intruder decelerates, which is more likely in the level-off mode than in the descend mode, the probability that the intruder is in the descend mode decreases to 0.03 by the 17,500 ft horizontal separation point. For this belief state to trigger an alert, the intruder would have to be at or below the altitude of the evader, since the entire alert space range for the belief state range [0.75 0.25] to [1 0] is below this altitude. The contour crossed, corresponding to a descend-mode probability of 1, is not within this range, so no alert occurs.

The alerting threshold was defined as where the alert (climb) action utility is equal to the deferral (nominal) action utility. Outside of the threshold the deferral has a higher utility. In the figure 6.6 utility trace the deferral utility is higher than the alert utility at all times, so no alert is ever needed. As the descend-mode probability approaches zero, the deferral action utility increases toward its maximum possible value of 1.1, because in the level-off mode there is virtually no chance of a lower-utility trajectory (a collision or alert).

6.5 Descend Scenario

Figure 6.7 shows an idealized descend-mode scenario with the same initial conditions as the level-off scenario. Without any alert, the trajectory will end with a collision. Following the alerting policy, an alert occurs at about 760 ft vertical separation. The resulting climb action is continued until the end of the scenario.

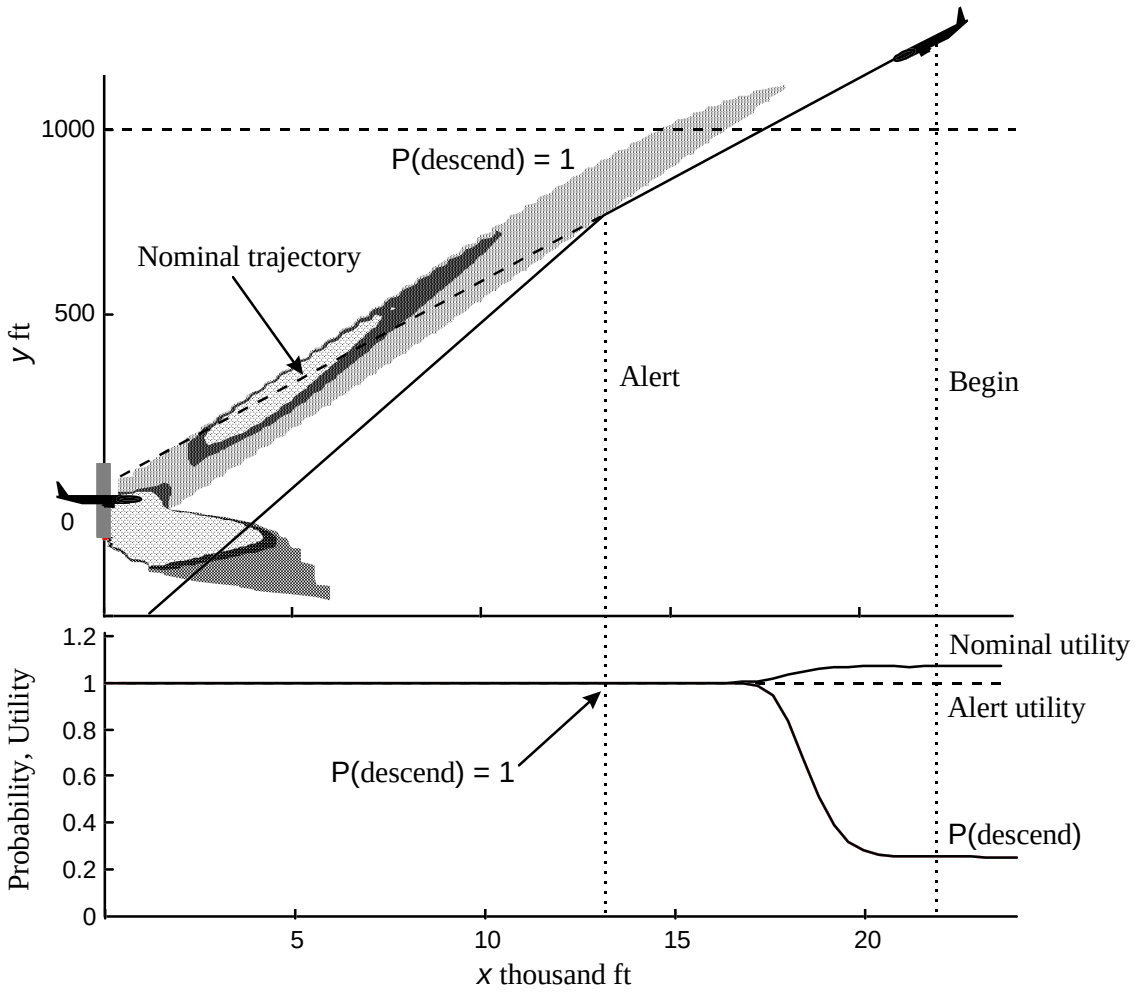


Figure 6.7: Descend-Mode Scenario

The initial descend-mode probability is again 0.25, but in this case the probability increases as evidence is gathered, and by the 15,000 ft horizontal position it has already reached 1 in the plot (visually, though in reality it never quite reaches 1). The [0 1]

contour is encountered about 2,000 ft later. Since the belief state is just under 1 when the [0 1] contour is crossed, the alert is triggered just inside that contour.

Before the alert there is still some chance of a safe, non-alert outcome for which the utility would be high ($U_{CR} = 1.1$), and this keeps the deferral-action expected utility greater than 1 for the first part of the trajectory. After the alert (marked with a vertical line) such an outcome is no longer possible and the greatest outcome utility possible is 1. An expected utility of 1 for an after-alert action means that there is no possibility of a collision occurring. Otherwise the low utility of a collision outcome ($U_{incident} = 0$) would reduce the expected action utility. In this scenario both the nominal and climb actions have nearly the same utility, near 1, after the alert and for the remainder of the encounter. The climb maneuver is aggressive enough that, if at any time during the evasion a nominal action is taken, the climb can be resumed at the next step with only a small loss of safety. Thus, the nominal action has only slightly less utility and safety than the climb.

6.6 Action Switching After the Alert

The same policy that determines the alerting threshold is responsible for choosing alerting system actions after the alert occurs. In the simple descend-mode example in figure 6.7, the optimal alerting input after the alert is to maintain the climb command until the end, because doing otherwise would reduce evasion safety.

Whereas previous scenarios were idealized examples of particular modes, figure 6.8 is a scenario where the intruder behaves in a way that is improbable for either mode.

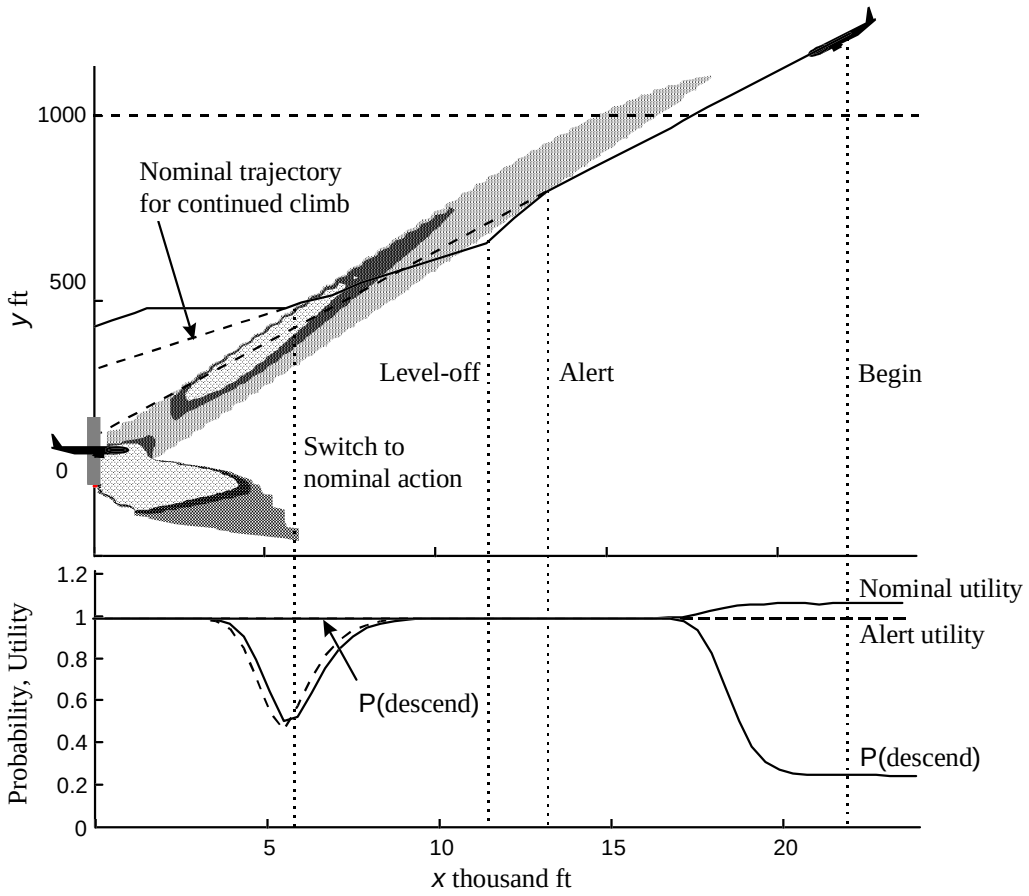


Figure 6.8: Abnormal Level-Off Scenario

It begins with a constant-speed descent at 2500 ft/min, and levels off about 300 ft lower than usual, 11,500 horizontal ft from the evader.

Because the descend-mode probability has converged nearly to 1 by the time the level-off occurs, and because the intruder's behavior is unlikely in either mode, no significant further change occurs in the belief state for the remainder of the scenario. Eventually, the logic sees that the climb maneuver is likely to cause a collision, assuming it is in the descend mode, and reverts back to the safer nominal action. Thus, the alerting system is able to maximize safety after an alert by dynamically choosing guidance, a clear benefit in individual scenarios such as the above. The final switch at 2,000 ft horizontal separation is the result the computed safety difference between the actions dropping to zero at that point, with climb being the default action.

Figure 6.9 shows the overall performance benefit of using reversible evasion maneuvers with this particular system and set of initial conditions, based on the assumed probabilistic dynamics. Average performance data was generated using Monte Carlo trajectory simulation, with the system always beginning at the same position and mode belief state. In one set post-alert command switching was allowed when needed, and in

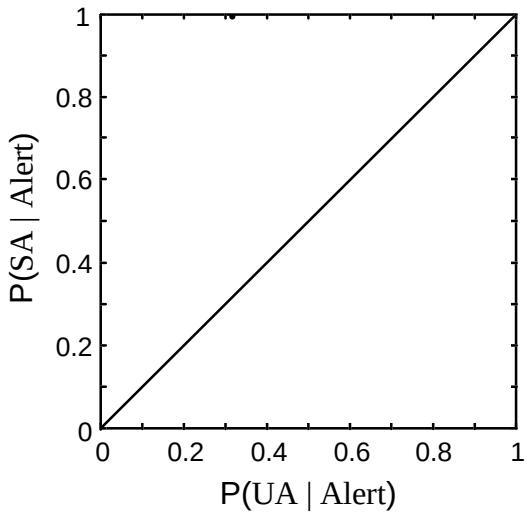


Figure 6.9: Average Performance With and Without Reversible Maneuvers

another the same threshold policy was used, but with the evader constrained to a fixed-climb escape maneuver. In each case trajectories were generated until 10,000 alerts occurred. The resulting average collision rate, P(SA | Alert), is 0.9993 for both, or identical within the precision of this simulation. The true unnecessary alert rate is known

to be the same for both since the same policy determines the threshold. While providing no apparent safety benefit, the logic with switching caused on average 0.73 switches after the alert.

6.7 Value of Anticipating Decision Opportunities

The MDP-based alerting method is distinct from other probabilistic alerting methods in that its decision metrics take into account the possibility of choosing between different actions at future times, so alerts are more informed and performance should be better. The purpose of this section is to demonstrate the performance differences with specific alerting scenarios.

An alternate policy was generated where utilities were defined in terms of the probabilistic (SOC) quantities P(UA) and P(SA), the probabilities of an unnecessary alert and of a successful alert for an alert generated at that moment. Assuming no alert has yet occurred, these were defined as the probability of no incident for each alert option, where the options are either a sustained deferral or a sustained climb respectively. In other words, the assumption is that each option is a complete action sequence covering all future time. This assumption is borrowed from recent research probabilistic alerting systems research (Carpenter & Kuchar; Yang & Kuchar, 1997). P(UA) is undefined after the alert, but an analogous metric for the post-alert case could be called P(SN) (*Safe Nominal* trajectory), defined as the probability of no incident if only nominal actions follow. The two relevant metrics after the alert would be P(SA) and P(SN). The reward function was defined similarly to the regular policy: safe non-alert end states give utility 1.1, safe alert outcomes give 1, and collisions give 0.

The resulting alerting threshold is shown in figure 6.10 along with a level-off mode trajectory. Because the assumption is a that a nominal action will be irreversible if chosen, that action appears to be a hazardous and low-utility option at the intruder's initial position, which is on a nominal collision course with the evader for the descend mode. Even with a descend-mode probability of only 0.25, the overall probability of a collision lowers the expected utility to nearly 0.8. The climb option is relatively safe, and

even though it ensures a maximum outcome utility no greater than 1, it has a higher expected utility at the outset.

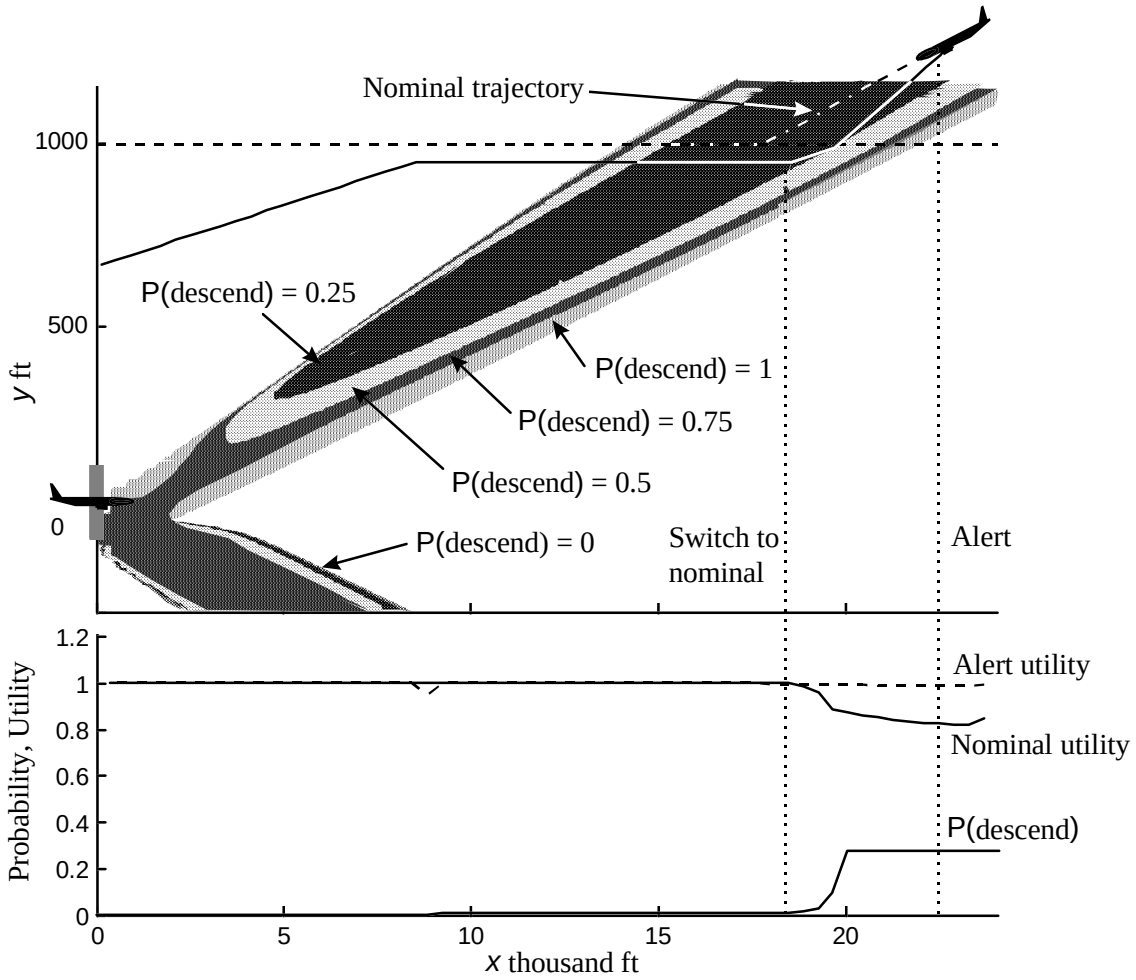


Figure 6.10: SOC-Metric-Based Alert Policy and Level-Off Scenario

The result is that an alert happens immediately. The mode belief updating process continues after the alert, and when the intruder levels off at 19,500 ft separation, P(descend mode) converges to near zero within 2,000 ft with the evasion in progress. At this point the climb trajectory is seen to be less safe than the nominal trajectory, so the nominal action is chosen. So, the logic is able to safely guide the aircraft after the alert, but has committed an unnecessary alert that would have been avoided by the regular logic in this scenario (figure 6.6).

In the descend-mode case (figure 6.11) the alert is triggered at the same time. No switching occurs after the alert, because once mode uncertainty is eliminated the climb path is clearly the safest option. The policy has successfully avoided a collision. The regular policy also avoids the collision (figure 6.7), but defers the alert until after the mode becomes more certain. The SOC-based policy's maneuver avoids the collision by a larger margin, so in one sense gave the more desirable response, but at the cost of a higher unnecessary alert probability.

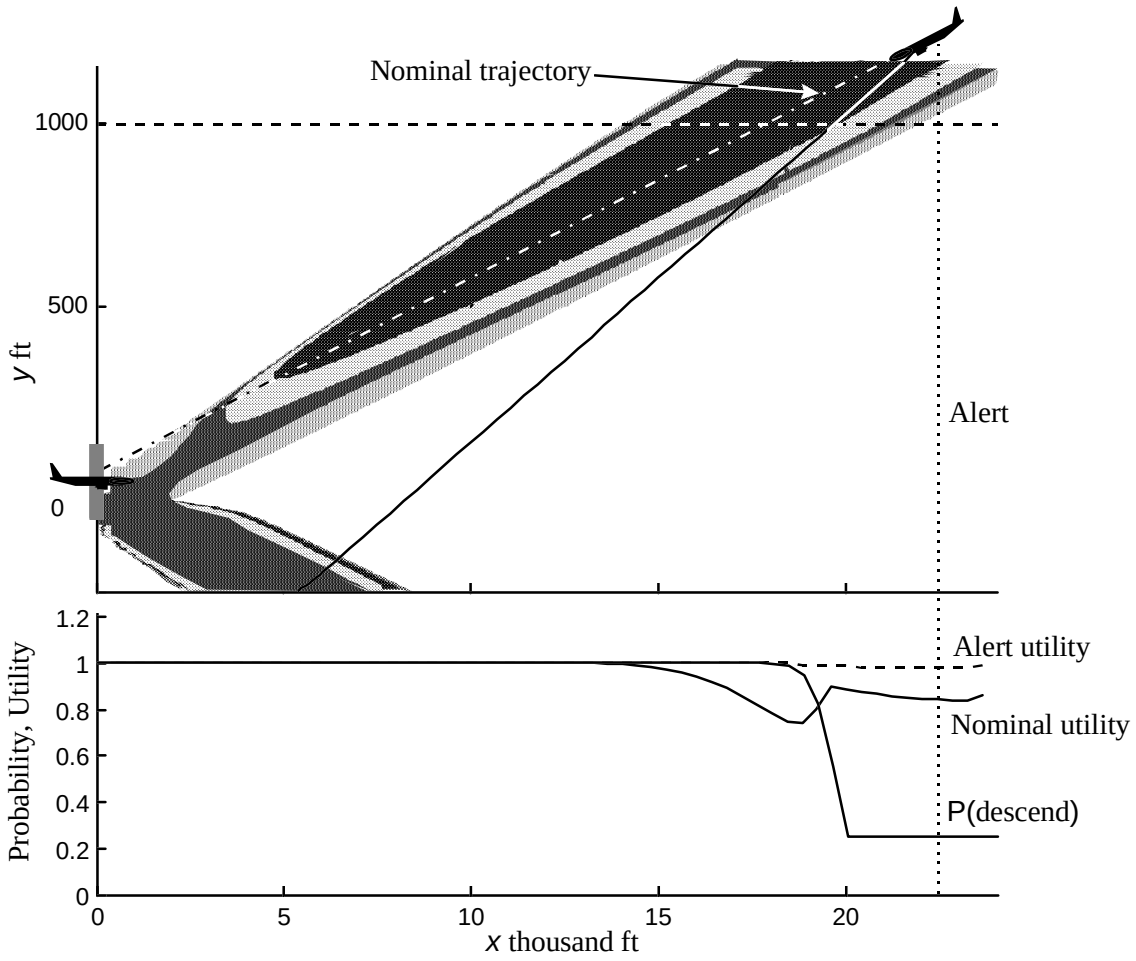


Figure 6.11: SOC-Metric-Based Alert Policy with Descend-Mode Scenario

Because individual scenarios can favor either policy, it is informative to compare average performance over many scenarios representative of the intended operating environment. Figure 6.12 compares the average performance of the two policies, where trajectories were simulated with each to generate 10,000 alert cases, based on the

encounter dynamic model defined in section 6.1, and all beginning at the same initial position and belief state. The metrics used are the average successful alert rate and the average unnecessary alert rate.

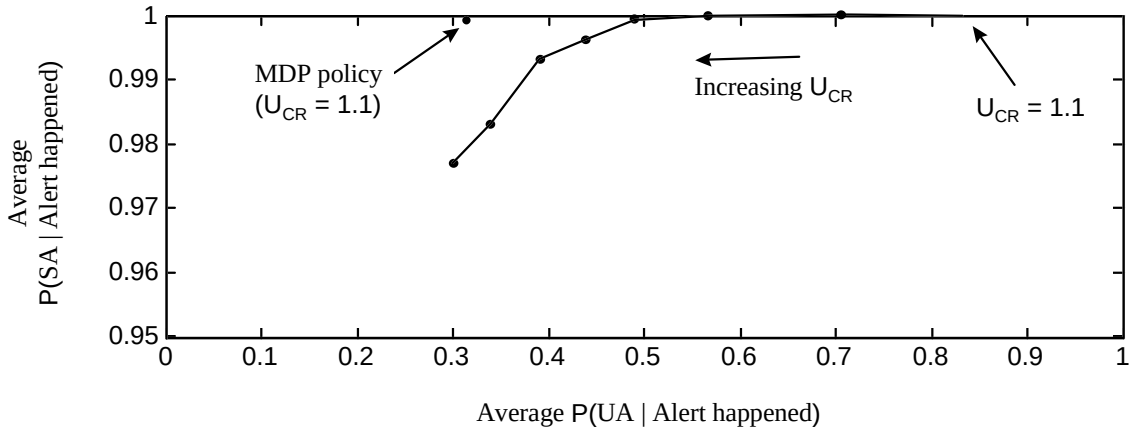


Figure 6.12: Average SOC Performance Comparison for Standard Policy vs. SOC-Based Threshold

As the example scenarios suggest, for a correct rejection utility of $U_{CR} = 1.1$, the SOC-based policy has a safety advantage. The SOC policy avoided any collisions, compared to 7 collisions in 10,000 alerts for the regular policy. At the same time, the SOC policy suffers from a relatively high rate of unnecessary alerts. In a simulation of 10,000 level-off trajectories, the regular policy caused no alerts, while the SOC-based policy caused an alert every time.

Figure 6.13 shows the SOC logic as a region in the SOC space. The alert space is the shaded area bounded by a diagonal line through the origin. The slope of the line is equal to the parameter U_{CR}. At each point in time, the policy generates SOC coordinates, and if these lie outside of the shaded region, the alert is deferred (or the nominal action is taken, if an alert has already happened). From this representation the location of the operating point in figure 6.12 is seen to depend on the SOC threshold slope, which at 1.1 is only slightly steeper than the diagonal.

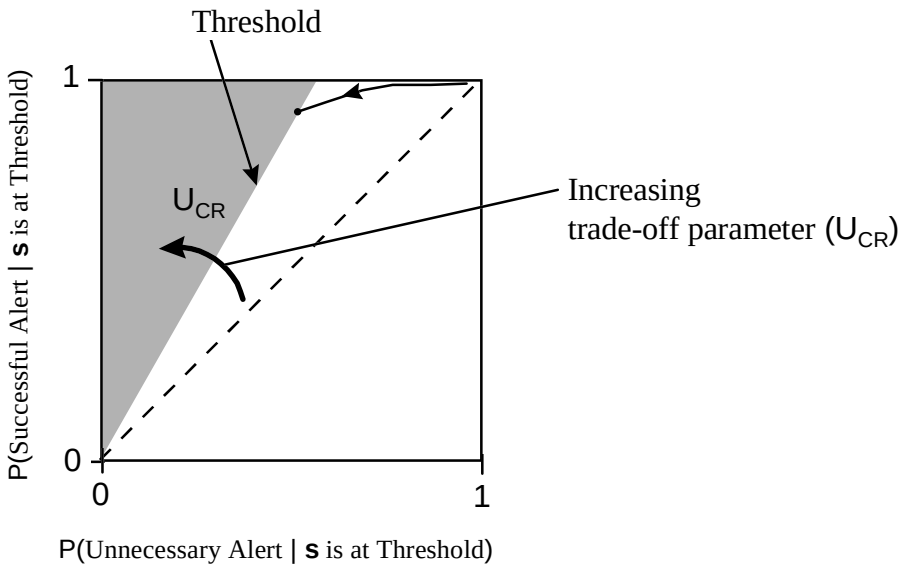


Figure 6.13: SOC-Space Threshold

If U_{CR} is increased, the slope increases, which should result in deferred alerts and a reduced unnecessary alert rate, given that scenario trajectories originate near the top, right corner of the SOC plot. U_{CR} was varied over a range of 1.1 to 3 and the resulting range of average performance is shown as a curve in figure 6.12. Along with the expected reduction in the unnecessary alert rate, there is a gradual decrease in average safety. When the operating point reaches a point directly below the regular policy's SOC position, the overall unnecessary alert rates are the same, and the SOC policy has a P(SA | Alert) of 0.98 compared to the regular policy's 0.999, or 20 times the collision probability, given an alert. From a 10,000 level-off trajectory simulation, the level-off mode unnecessary alert rate at that point is roughly 4 in 1,000 for the SOC policy, compared to none for the regular policy. The total alert rate (level-off and descend cases) is similar for both at 23 per 100. The significance of these differences depends on the particular alerting application, but there is a clear difference between the two policy types, with the MDP-based policy allowing a higher overall safety and fewer total and level-off alerts with the assumed dynamics.

As discussed in section 2.3, there is a more general philosophy that the alerting region in SOC space should take whatever shape necessary to optimize performance. For example, figure 6.14 shows two additional possibilities, each aimed at directly controlling an aspect of performance. The minimum safety threshold ensures that alerting safety is at or above a specified limit, T. The maximum unnecessary alert probability threshold defers alerts until the unnecessary alert probability, a measure of the need for the alert, is acceptably small.

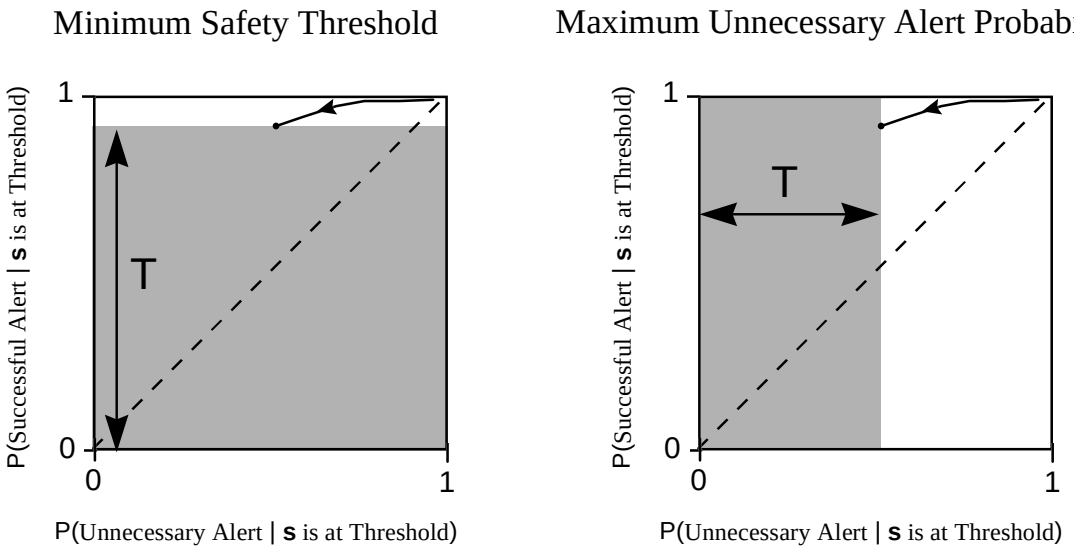


Figure 6.14: Other SOC-Space Thresholds

Figure 6.15 shows corresponding SOC curves for the figure 6.14 logics, generated by varying the tradeoff parameter T over a range. Again, these logics are unable to reach the same trade-off of safety for unnecessary alerts that the MDP-based logic does.

The MDP logic compares favorably with the SOC-based logic because of its more complete use of information available in the belief state and in recognizing future decision opportunities. It is also a result of the reward function chosen for this example, which was constructed with SOC design goals in mind. In general, it should be noted, these may not be the real goals of every alerting system. For example, in systems with normal and failure modes (e.g. Parallel landing approaches (Winder & Kuchar, 1999)) an alert might be considered proper if it occurs during a failure, even if it is an unnecessary

alert. The utility basis of the MDP method is general enough to accommodate such differences.

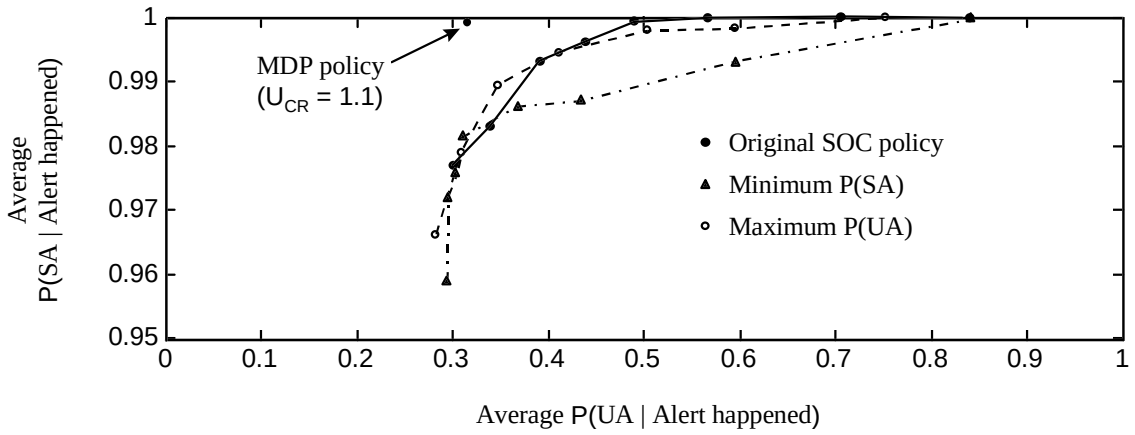


Figure 6.15: Performance of Other SOC-Based Thresholds

6.8 Belief State Modeling Simplification

The MDP-based alerting method assumes that uncertain state variables like modes require maintaining and using a belief state. As described in chapter 2, keeping the belief state current involves a continual filtering process done in real time. The belief state is then used as an input, along with other state variables, to a pre-computed policy that identifies the best action to take. Like other continuous state variables it is often necessary to simplify its domain for describing the policy, such as in the example in this chapter where a continuous range of mode distributions was replaced with a small, discrete set. The smaller the domain, the easier it is to compute a policy using the belief state as a variable. However, oversimplifying the belief state domain could unacceptably impact performance. This section will discuss the relationship of the degree of belief state simplification to the performance of the alerting system.

The simplest way to reduce the complexity of belief state variables is to assume the system is in one or the other mode, and do no belief updating. For example, one option is to assume level-off dynamics always hold. In figure 6.16, this limits the alert space to some region within the belief state rage between P(descend) = 0 and 0.25. While never alerting during level-off scenarios, during descend mode scenarios this logic

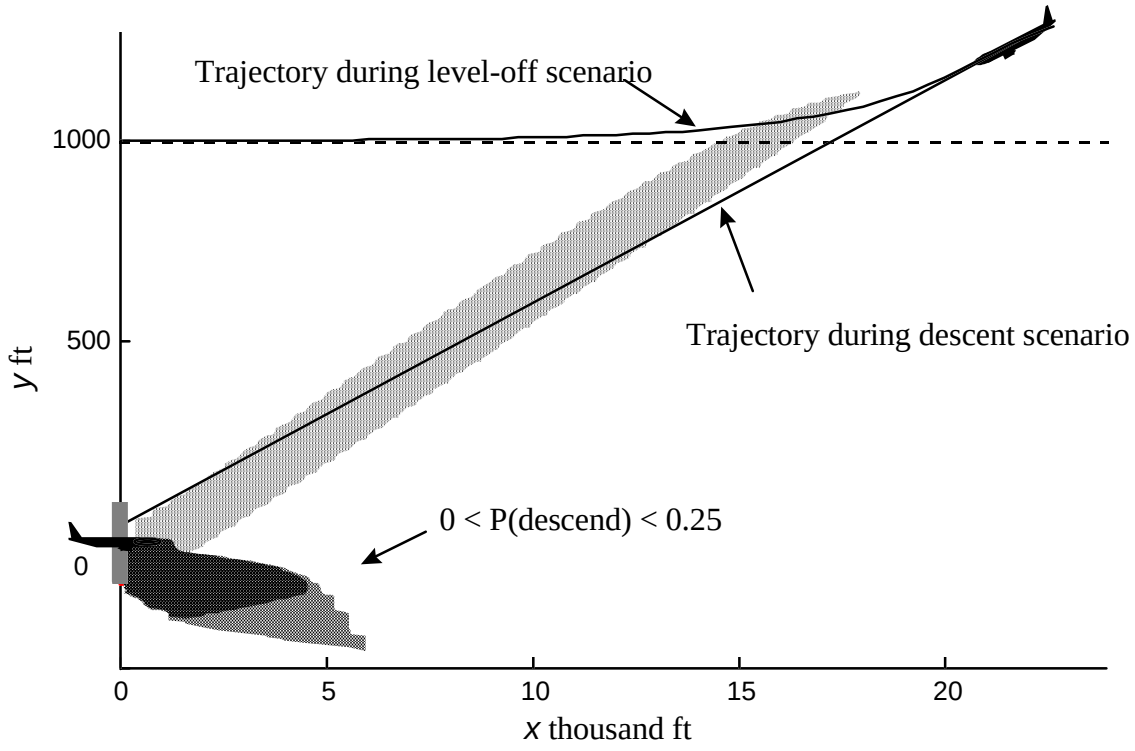


Figure 6.16: Belief State Simplification – Assume No Descend Mode

can allow collisions without providing any alert (missed detections), or issue alerts too late (late alerts).

The other extreme case is to assume that the descend mode always holds (figure 6.17). Now the reverse problem happens: safety during descend-mode scenario is ensured, while unnecessary alerts happen during many level-off cases, since the threshold contour extends beyond the nominal level-off separation.

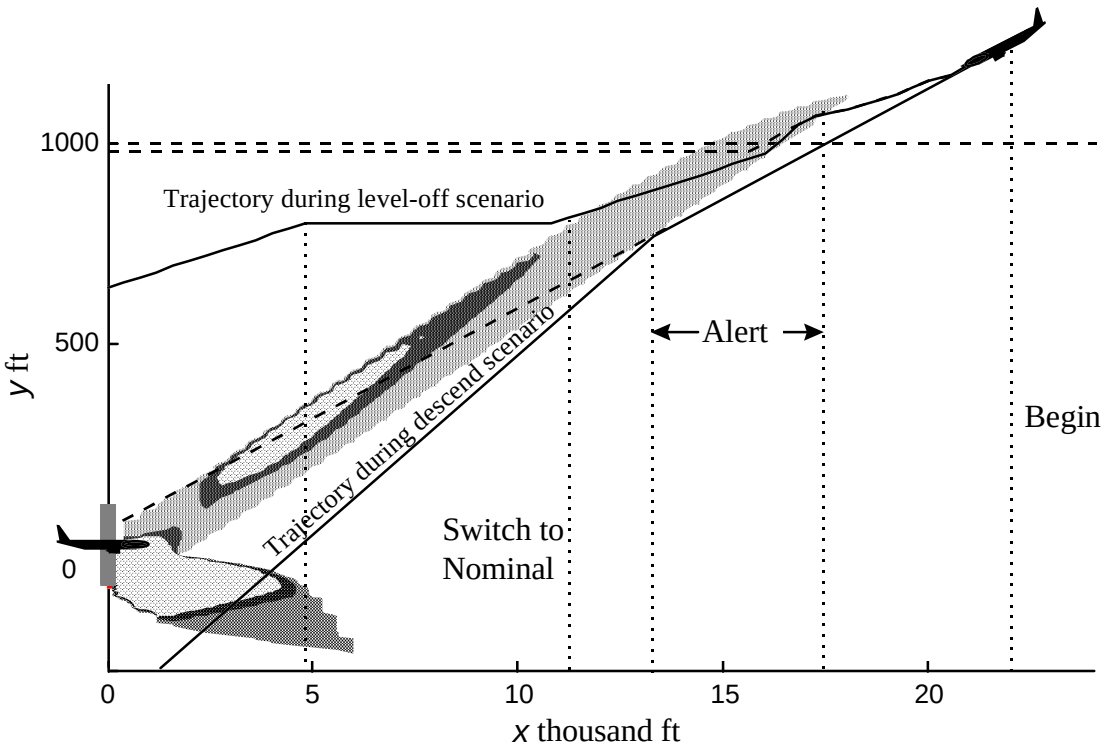


Figure 6.17: Belief State Simplification – Assume No Level-off Mode

Eliminating one mode or the other as discussed has a significant negative effect on the performance of the logic, either increasing unnecessary alerts or reducing safety. A third possible belief state simplification is a 2-value domain including both of the extreme belief cases already considered separately. Recall that the discrete belief state model used in the original policy was a 5-value progression of belief states separated by probability increments of 0.25. Utility function matrices were created over this set, one for each action. Then, to generate action utilities for a given belief state, linear interpolation was done between discrete elements of the utility matrices. The suggestion is to reduce the discrete domain to the smallest set that accounts for both of the extreme belief states, and then assume there is a linear relationship between the belief state and the action utility.

A policy was found for this belief state model and examined in a similar way to previous examples. Because at each extreme belief state there is no possibility of a belief updating cycle changing the belief state, the simplified policy can be found by removing

the middle three belief cases from the existing policy. This is because once the belief state reaches one extreme or other, no future evidence gathered can change the belief state, making the policy at each extreme belief state independent of that at any other belief state.

Figure 6.18 compares the utility traces resulting from each policy version and the idealized descend-mode scenario. Visually they are nearly the same. They both result in

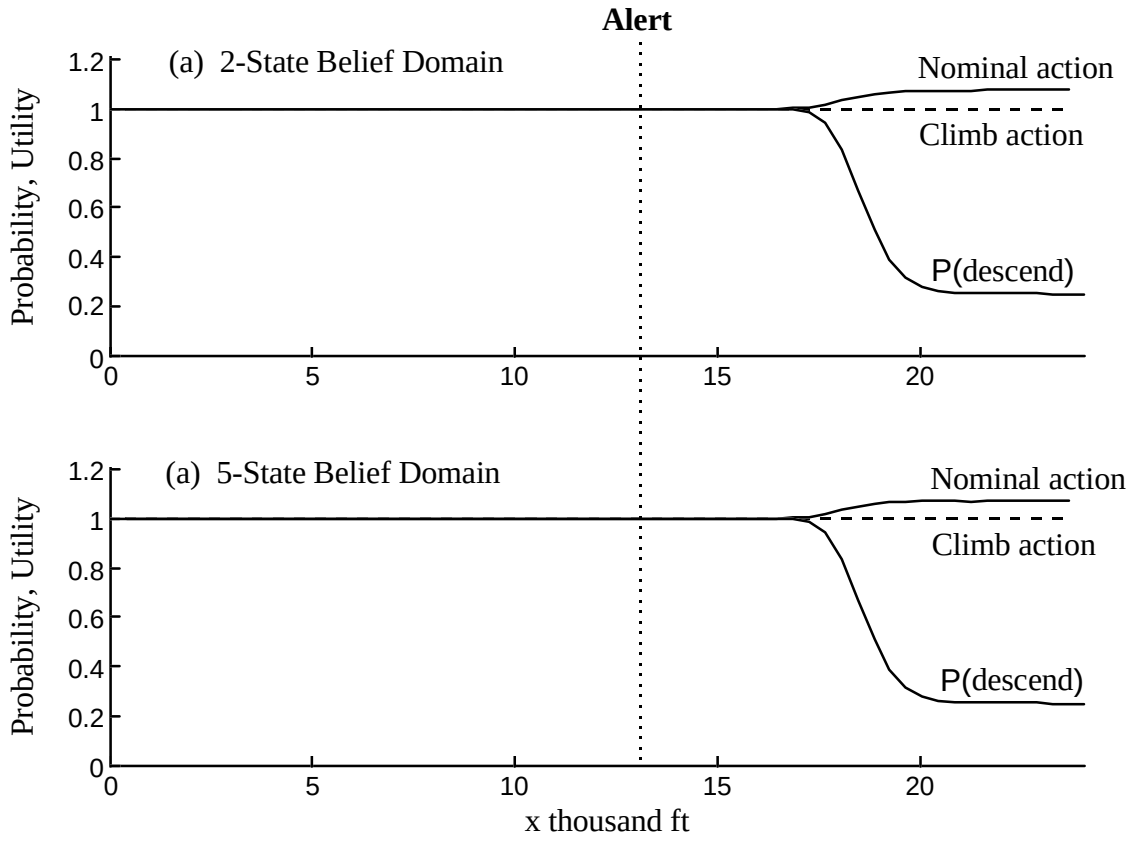


Figure 6.18: Utility Trace Comparison: 2 vs. 5-Belief-State Simplification

deferring the alert until the mode uncertainty is nearly eliminated, and they both result in an alert at the same moment. The policies also react similarly to a level-off scenario, each avoiding an unnecessary alert.

Compared using the Monte Carlo scenario simulation described in section 6.6 with 10,000 alert scenarios, they both had P(SA | Alert) values of 0.9993, and the

sampling error of their difference is 0.001 or less with 95% certainty. The unnecessary alert fractions are 0.314 for the regular logic and 0.322 for the 2-belief state version, different by 0.008. The standard deviation of the sample difference is 0.007, so there is not a clear difference in the unnecessary alert probability between the two logics. Based on 30,000 trajectory level-off simulations, the logics had level-off-mode unnecessary alert rates of 25 (the original) and 19 (2-belief state) per 100,000 level-offs. For the given sample size (30,000), there is no clear difference in level-off alert performance between the two logics.

A possible explanation for the similarity is, besides any coincidental linearity in the actual utility functions, that the structure of the system and utility definitions cause alerts to be put off in every case until mode uncertainty is nearly gone. This means that at the alerting threshold, the belief state is always P(descend-mode) ≈ 1, which is a belief state where action utilities were computed precisely for both policies. The utilities at the threshold are then expected to be similar, even with an approximation of the utility function. To avoid unnecessary alerts in the interval up to this point, it is only necessary that both policies have nominal action utility greater than the climb action utility, so some error due to utility function approximation is tolerable.

6.9 Chapter Summary

In this chapter the use of uncertain modes in an alerting system was investigated, using the example of a 2-aircraft encounter where, depending on an uncertain mode variable, the aircraft may or may not be in danger of colliding. A Markov state was defined for the 2-aircraft system, including a belief state for the mode. This state, a corresponding dynamic model and a reward function describing alerting priorities were used to generate the alerting logic, or "policy." Operation of the policy, both as an alerting threshold and as a post-alert guidance logic, was shown with example scenarios.

The value of taking future decisions into account, as the policy inherently does, was shown using example scenarios and with global performance averages generated by

Monte Carlo trajectory simulation. This involved defining alternative policies where fixed maneuvers were the assumed choices.

The importance of the mode belief state and belief updating process to effective alerting was shown. This was accomplished by looking at the effect of disabling the mode-update process in example scenarios, and comparing global performance metrics from Monte Carlo simulation. In addition it was shown that for acceptable alerting performance a policy may require only limited information about the belief state, depending on the situation. A simplified belief state domain was compared to a more complete belief state model in discussing this.

7. Summary and Contributions

7.1 Summary

In this thesis a framework for designing hazard avoidance alerting systems was presented, based on a Markov decision process model of alerting, and motivated by identified weaknesses with existing methods. Two alerting "logics" were created using the framework and were compared in terms of standard performance measures to logics created with more typical methods, demonstrating a benefit.

The use of MDP methods was motivated by a lack of certain features in existing methods for direct derivation of alerting logics from performance requirements. One of these is the ability to reason about future decision opportunities that might influence the current decision. In particular, such knowledge is important for placement of the alerting threshold, because it is what allows deferral of alerts: knowing whether safe options will be available in the future affects the current decision. Another desired feature is the ability to model and account for uncertain dynamic modes in the observed situation. Modes describe distinct types of behavior a system could exhibit at a given time, and uncertainty in the mode complicates the state predictions needed for decision making. Mode uncertainty also motivates being aware of future decision opportunities, because actions have predictable effects on mode uncertainty. In particular, alerts may be deferred partly in expectation of decreasing mode uncertainty.

The MDP-based methodology requires a Markov state and probabilistic dynamic model of the operator-plant system, a probabilistic observation model, and creation of a reward function that describes the alerting system's (designer's) goals in terms of cumulative rewards that can be gained along future system trajectories. Uncertain mode variables are modeled probabilistically, and the resulting distribution, or belief state, can

be updated at each step to reflect changes in the uncertainty due to new evidence. With these components, MDP theory provides means to derive an efficient alerting policy that allows computations for alerting decisions to be done in real time. The policy determines both the threshold for alerts and the later sequence of cues that guide an operator during resolution of the hazard.

The policy is a function of the current system state that produces the best action from an available set. The state can be a set of variables or a distribution over variables—a belief state—including mode variables. In the belief state case the solution can be less straightforward, but methods exist.

The policy inherently takes into account future decisions through application of Bellman's equation, which itself is an effect of the principle of optimality. Under an assumption of utility-based preferences, this principle says that the utility of an action at a given state depends only on the utility of reaching the next state, assuming the next-state utility is optimal (maximized). Thus, choosing the next action requires no assumption of any particular trajectory being followed later.

The MDP-based methodology was used to derive alerting logics for two kinds of aircraft encounter, one a head-on collision scenario with random altitude variations, and the other an uncertain 2-mode scenario with a safe (level-off) and an unsafe (continued descent) mode. These case studies demonstrated how alerting system goals can be expressed as a reward function, computation of an alerting policy, and use of the policy as an alerting and guidance threshold. The second case study also showed the modeling of an uncertain mode, effects of the mode on policy computation, and the behavior of the resulting logic. In the second case study the MDP-based alerting logic was tested against alternate logics designed according to current practice using standard performance metrics, and the performance benefits of MDP design were made apparent. The importance of using global average performance metrics, including traditional metrics like unnecessary alert and incident rates, alongside reward function requirements was also explained.

A claim is made that the reward function basis of the alerting process must agree with or complement the alerting preferences of the human operators. This is to minimize the rate of improper alerts, defined as alerts that the operators find incorrect, and which include nuisance alerts. However, at this time there is no clear description of this relationship to guide design of the reward function. In the case studies a simple reward function was chosen that makes a trade-off between safety and unnecessary alerts at the threshold. Trading off safety for unnecessary alerts is an established practice in alerting design. The resulting performance compares well with SOC-based alerting, where a threshold is defined in the space of P(SA) and P(UA). In terms of the global SOC performance metrics, the MDP-based logic achieves superior safety to compared SOC-based logics for a given unnecessary alert rate. In addition the MDP-based logic is better able to avoid alerts during level-off mode scenarios while maintaining a given level a safety.

The case study systems were made purposely simple for clarity. This leaves a question of whether MDP methods will also apply to more complex alerting systems requiring more state variables. In principle they do, but because the number of states can increase exponentially with the number of state variables, it is easily possible to run into computing speed and memory limits (Bellman called this problem the "curse of dimensionality.") As a consequence, more complex alerting systems may require policy or utility function approximations that reduce the number of variables and states. The tabular utility function representation and policy derivation methods that were convenient in the case studies may be too inefficient for use general.

7.2 Contributions

A new methodology was described for designing hazard avoidance alerting systems, based on Markov decision process theory. It has a number of advantages:

1. It provides a means of generating an efficient alerting logic directly from requirements, reducing the need for design iterations to meet goals.

a. It was demonstrated with aviation alerting case studies having the core elements of more complex alerting problems.

b. The use of a utility model of performance requirements was described and demonstrated.

2. It makes complete use of available information, including

    a. Information about mode uncertainty.

    b. Information about future states and decision opportunities; in particular, whether deferring an alert leaves sufficient flexibility for future action, and whether future observations will reduce mode uncertainty.

3. It unifies the design of the alerting threshold and guidance, including when flexible escape guidance is needed. This allows knowledge about future guidance to directly affect the threshold. Earlier efforts have focused on optimal definition of the alerting threshold or have assumed fixed escape maneuvers.

References

Bellman, R. E. (1957a). *Dynamic Programming*. Princeton University Press, Princeton, NJ.

Bellman, R. E. (1957b). A Markov decision process. *Journal of Mathematical Mech.*, 6:679-684.

Bertsekas, Dimitri P. (1987). *Dynamic Programming: Deterministic and Stochastic Models*. Prentice-Hall, Upper Saddle River, NJ.

Bertsekas, Dimitri, P. (1995). *Dynamic Programming and Optimal Control*. Athena Scientific, Belmont, Massachusetts. Volumes 1 and 2.

Brown, Robert Grover and Hwang, Patrick Y. C. (1997). *Introduction to Random Signals and Applied Kalman Filtering*. 3rd ed. John Wiley & Sons, Inc.

Carpenter, Brenda D. & Kuchar, James K. (1997). Probability-based collision alerting logic for closely-spaced parallel approach. *AIAA Aerospace Sciences Meeting & Exhibit, 35th*, Reno, NV, Jan. 6-9. AIAA-97-0222.

Cassell, Rick; Evers, Carl; Sleep, Ben; Esche, Jeff. (2001). Initial Test Results of Pathprox — A Runway Incursion Alerting System. *20th Digital Avionics Systems Conference*. Daytona Beach, FL. October 14-20. IEEE.

Feith, Gregory. (2002). Avoiding the Beast Below. *Business & Commercial Aviation*. September 9.

Harman, W. H. (1989). TCAS: A System for Preventing Midair Collisions. *The Lincoln Laboratory Journal*. Vol. 2, No. 3, pp 437-457.

Horvitz, Eric, Andy Jacobs, David Hovel. (July 1999). Attention-Sensitive Alerting. Microsoft Research. *Proceedings of UAI '99 Conference on Uncertainty and Artificial Intelligence*, Stockholm, Sweden, July. Morgan Kaufmann, San Francisco. pp. 305-313.

Horvitz, Eric & Geoffrey Rutledge. (1991). Time-Dependent Utility and Action Under Uncertainty. *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence*. July.

Horvitz, Eric & Seiver, Adam. (1997). Time-Critical Action: Representations and Application. *Proc. 13th Conf. on Uncertainty in Artificial Intelligence*, Morgan Kaufmann, San Francisco, CA, August.

Huang, Cecil (1999). *Intelligent Alarms: Allocating Attention Among Concurrent Processes*. Stanford PhD thesis. March.

Hwang, Inseok; Hwang, Jesse & Tomlin, Claire. (2003). Flight-Mode-Based Aircraft Conflict Detection using a Residual-Mean Interacting Multiple Model Algorithm. *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, Austin. August.

Jones, Denise R. (2002). Runway Incursion Prevention System Simulation Evaluation. *21st Digital Avionics Systems Conference*, Irvine, CA, Octover 27-31. IEEE.

Kuchar, James K. (1995). *A Unified Methodology for the Evaluation of Hazard Alerting Systems*. MIT Aeronautics and Astronautics PhD thesis, Cambridge, MA. ASL-95-1.

Kuchar, James K. (1996). Methodology for Alerting-System Performance Evaluation. *AIAA Journal of Guidance, Control and Dynamics*, Vol. 19, No. 2, pp 438-444. March-April.

Kuchar, James K. & Yang, Lee C. (2000). A Review of Conflict Detection and Resolution Modeling Methods. *IEEE Transactions on Intelligent Transportation Systems*, Vol. 1, No. 4. December.

Nordwall, Bruce D. (2002). TCAS More 'Foolproof' Than Generally Recognized. *Aviation Week & Space Technology*, July 15.

O'Hara, Colleen. (1998). Friendlier Skies. *Federal Computer Week*. FCW Media Group. March 30.

Phillips, Edward H. (2001). Aviation Week: Global Teamwork Called Key to Improving Aviation Safety. *Aviation Week & Space Technology*. July 16.

Pritchett, A. R. & Hansman, R. J. (1997). *Pilot Non-Conformance to Alerting System Commands During Closely Spaced Parallel Approaches*. MIT Aeronautics and Astronautics PhD Thesis, Cambridge, MA. Aeronautical Systems Laboratory Report, ASL-97-2.

Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, NY.

Reynolds, Tom G. & Hansman, R. John. (2003). Investigating Conformance Monitoring Issues in Air Traffic Control Using Fault Detection Approaches. MIT International Center for Air Transportation, Report ICAT-2003-5. November.

RTCA, Radio Technical Committee on Aeronautics. (1983), *Minimum Performance Specifications for TCAS Airborne Equipment*, Document No. RTCA/DO-185, Washington, D.C. September.

Russell, Stuart & Norvig, Peter. (2003). *Artificial Intelligence: A Modern Approach*. 2nd ed. Prentice Hall. Pearson Education, Inc., NJ.

Samanant, Paul; Jackson, Mike; Haissig, Christine M.; Corwin, Bill. (2000). CASPER/AILS: An Integrated DGPS/ADS-B Airborne Alerting System For Closely Spaced Parallel Approaches. *IEEE PLANS, Position Location and Navigation Symposium*, pp 57-65, March 13-16.

Scott, Phil. (2001). Collision Detection: New radar systems may prevent deadly accidents on congested runways. *Scientific American*, February.

Shank, E. M. & Hollister, K. M. (1994). Precision Runway Monitor. *The Lincoln Laboratory Journal*, Vol. 7, No. 2, MIT Lincoln Laboratory, Lexington, MA.

Sutton, Richard S. and Barto, Andrew G. (1998). *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA.

Tesauro, Gerald. (1994). TD-Gammon, A Self-Teaching Backgammon Program, Achieves Master-Level Play. *Neural Computation*, Vol. 6, No. 2, pp 215-219.

Wickens, Christopher D. (1992). *Engineering Psychology and Human Performance*. 2nd ed. Harper Collins Publishers, Inc.

Winder, Lee F. & Kuchar, James K. (1999). Evaluation of Vertical Collision Avoidance Maneuvers for Parallel Approach. *AIAA, Journal of Guidance, Control and Dynamics*. November-December.

Yang, Lee C. & Kuchar, James K. (2002). Performance Metric Alerting: A New Design Approach for Complex Alerting Problems. *IEEE Transactions on Systems, Man and Cybernetics--Part A: Systems and Humans*, Vol. 32, No. 1. January.

Yang, Lee C. & Kuchar, James K. (1997). Prototype conflict alerting system for free flight. *Journal of Guidance, Control and Dynamics*. AIAA. Vol. 20, No. 4, July-August.

Yang Lee C.; Yang, Ji Hyun; Feron, Eric & Griffith, Ernest W. (2004). Multiple Model Estimation for Improving Conflict Detection Algorithm. IEEE.

Zhao, Yiyuan J. & Rock, Dennis. (2002). Fundamental Principles of Alerting Algorithm Design with Applications to Rotorcraft Formation Flight. *AIAA Guidance, Navigation and Control conference*. Montery, CA, August 5-8. AIAA-2002-4754.

Definitions

Action	A decision option available to the alerting system
Action logic	Component of the alerting system that chooses the alerting action based on the current belief state and previous action
Action space	Set of possible alerting system actions
AILS	NASA Airborne Information for Lateral Spacing collision avoidance logic for parallel approach collision avoidance
Alerting system	Automation that monitors a human-operated system and issues alerts and guidance to avoid unexpected hazards
Belief compression	Refers to methods of approximating a range of belief states with a simpler function having fewer parameters
Belief space	The set of all possible belief states
Belief state	A probability distribution over the situation space that describes alerting system uncertainty about the state
Bellman equation	Fundamental equation of MDP theory that describes the relationship of the maximum expected utility at each state to that of neighboring states, reachable by a single action
CD	Correct detection
Correct detection	A state trajectory in which an alert occurs, preventing an incident that would have happened otherwise
Correct rejection	A state trajectory in which no alert occurs and no incident occurs
CR	Correct rejection
Deferral action	Also nominal action. The alerting system action prior to any alert occurring, when it is passively monitoring

Escape maneuver	A maneuver due to repeatedly using a given maneuver policy after an alert has occurred
Evader	Aircraft that is assumed to receive and respond to alerts to avoid the intruder
Evidence	Information gathered by the alerting system in the form of observations of the situation and recalled past actions
Expected utility function	Function mapping each state into an expected utility
Filtering	The process of updating the a belief state, given a stream of observations
Finite horizon	A class of Markov decision processes in which there is assumed to be a finite time remaining before the process ends
Fully observable	The exact situation state is available through a single observation
GPWS	Ground Proximity Warning System. Alerting system for preventing "controlled flight into terrain" accidents
Hazard space	The set of all situation states where it is possible for an incident to occur
IA	Improper alert
II	Induced incident
Improper alert	An alerting system action that an operator believes is incorrect
Incident	An undesirable event that an alerting system is designed to prevent from happening
Induced incident	An incident that occurs after an alert happens, that would not have occurred without the alert
Infinite horizon	A class of Markov decision processes in which there is assumed to be no end point
Intruder	Aircraft that may endanger the evader and is assumed not controllable by the alerting system

Knowledge	The total information available to the alerting system about the situation through observations and prior knowledge. The belief state
LA	Late alert
Late alert	An alert that is necessary but happens too late to prevent an incident
Maneuver	Sequential use of a single maneuver policy over some interval
Maneuver policy	A function that maps a state or belief state into a specific alerting system action
Markov property	The property of a state whereby knowing the state allows a correct state prediction (exact or probabilistic prediction)
Maximum expected utility function	Function mapping each state into the largest expected utility possible for a set of possible actions
Maximum expected utility principle	Decision criterion stating that the preferred decision is the one that gives the maximum expected outcome utility
MD	Missed detection
MDP	Markov decision process
Missed detection	A case where an incident happens without any prior alert
Mode	A discrete state variable that represents distinct dynamic behaviors a system can exhibit
NA	Nuisance alert
Nominal action	A deferral action by the alerting system
Nominal maneuver	Assuming no alert has occurred, the maneuver due to repeating the deferral action indefinitely
Nuisance alert	An improper alert where an operator considers the alert unjustified
Observation	A situation state measurement at a given time
Observation space	The set of possible observations for a particular state space and sensor function

Operator	A human operating within a larger situation, who can receive alerts from an alerting system
Partially observable	Refers to a situation where an observation does not provide the exact and entire state
Plant	A system controlled by operators within a situation
Policy	A function that maps any state (or belief state) into a particular action
Policy iteration	A method of finding a policy for an MDP that involves alternately computing the utility function implied by a candidate policy, and calculating a new candidate policy from the utility function
POMDP	Partially observable Markov decision process
Precision Runway Monitor	Alerting system for preventing collisions during closely spaced parallel approaches
Principle of optimality	In a sequential decision process, maximizing the utility of the system trajectory (past and future) implies maximizing the utility of the future trajectory.
Prior belief state	A belief state that is assumed given at the start of a Markov decision process
PRM	Precision runway monitor
Reward function	A function defined over state space that specifies the reward gained by the alerting system when the state is occupied
SA	Successful alert
SN	Safe nominal trajectory
Sensor function	A function that maps the situation state into an observation
Situation	A system composed of human operators and a plant that they control
Situation space	The set of possible situation states
Situation state	The Markov state describing a situation
SOC	System operating characteristic

State	A state vector
State vector	A set of variables that together describe the condition of a system at a given time
Stationary	Refers to a function that remains unchanged with passing time
Successful alert	An alert that is not followed by an incident
System Operating Characteristic	A pair of performance metrics, either the probability of a safe alert and the probability of an unnecessary alert, given that an alert happens, or the frequency of a safe alert and of an unnecessary alert given that an alert happens.
TCAS	Traffic Alert and Collision Avoidance System. An alerting system for preventing mid-air collisions between aircraft
Trajectory utility function	A function mapping a state trajectory into a utility value
Transition function	The function describing the future state or state distribution for a given initial state and control (action) input
UA	Unnecessary alert
Unnecessary alert	An alert that is not followed by an incident, but where no incident would have occurred without the alert
Unsuccessful alert	An alert followed by an incident
Utility	A scalar value describing the degree of desirability or goodness of something
Utility function	See trajectory utility function. Also, short for a maximum expected utility function
Value iteration	A method of solving Markov decision problems, where the correct maximum expected utility function is arrived at through iterative application of the Bellman equation to a candidate utility function over the entire state domain, each time using the most recent utility function

Appendix A: Overview of Belief State Filtering

Consider a discrete-time alerting process described by the following components:

1. A situation state, $\mathbf{s}$
2. An alert signal, $\mathbf{a}$
3. An observation signal $\mathbf{o}$ determined by a sensor function, $O(\mathbf{o}(k) \mid \mathbf{s}(k))$
4. A state transition function, $T(\mathbf{s}(k+1) \mid \mathbf{s}(k), \mathbf{a}(k))$
5. Prior knowledge about $\mathbf{s}$ in the form of a probability distribution, $b_{\mathbf{s}(0)}(\mathbf{s}(0))$

The functions $O(\cdot)$ and $T(\cdot)$ are in the form of probability distributions over the spaces of $\mathbf{o}$ and $\mathbf{s}$, respectively.

At each point in time, an observation ($\mathbf{o}(k)$) is made and an action ($\mathbf{a}(k)$) is taken by the alerting system. After k time steps, beginning with $k = 0$, the entire history of these values is:

$$\{ \mathbf{o}(0), \mathbf{a}(0), \mathbf{o}(1), \mathbf{a}(1), \ldots , \mathbf{o}(k-1), \mathbf{a}(k-1), \mathbf{o}(k) \}$$

or more compactly,

$$\{ \mathbf{e}(0), \mathbf{e}(1), \ldots , \mathbf{e}(k-1), \mathbf{o}(k) \}$$

where $\mathbf{e}(i)$ is the "evidence" from time step i:

$$\mathbf{e}(i) = \{ \mathbf{o}(i), \mathbf{a}(i) \}.$$

This history is the complete set of observable evidence concerning the situation. By definition, the belief state at k is the probability distribution of $\mathbf{s}(k)$, conditioned on all of this evidence:

$$b_{\mathbf{s}(k)}(\mathbf{s}(k)) = p(\mathbf{s}(k) \mid \mathbf{e}(0), \mathbf{e}(1), \ldots , \mathbf{e}(k-1), \mathbf{o}(k))$$

Applying the definition of conditional probability, this becomes:

$$= \alpha \; p(\, \mathbf{o}(k) \mid \mathbf{s}(k), \mathbf{e}(0), \mathbf{e}(1), \ldots, \mathbf{e}(k\text{-}1)\,) \;\; p(\, \mathbf{s}(k) \mid \mathbf{e}(0), \mathbf{e}(1), \ldots, \mathbf{e}(k\text{-}1)\,)$$

where α is a normalization constant that makes the distribution sum to 1. The rightmost factor is the conditional probability distribution of the current state on all past observable evidence. The middle factor is the conditional distribution of the current observation on the current state and all previous observations and actions. Since the observation depends on the current state and nothing else (by definition of the sensor function), the expression simplifies to:

$$= \alpha \; p(\, \mathbf{o}(k) \mid \mathbf{s}(k)\,) \;\; p(\, \mathbf{s}(k) \mid \mathbf{e}(0), \mathbf{e}(k1), \ldots, \mathbf{e}(k\text{-}1)\,)$$

To evaluate the last factor, it can be expanded to show the implicit conditioning of $\mathbf{s}(k)$ on the previous state, $\mathbf{s}(k\text{-}1)$:

$$p(\, \mathbf{s}(k) \mid \mathbf{e}(0), \mathbf{e}(1), \ldots, \mathbf{e}(k\text{-}1)\,) =$$

$$\sum p(\, \mathbf{s}(k) \mid \mathbf{s}(k\text{-}1), \mathbf{e}(0), \mathbf{e}(1), \ldots, \mathbf{e}(k\text{-}1)\,) \;\; p(\, \mathbf{s}(k\text{-}1) \mid \mathbf{e}(0), \mathbf{e}(1), \ldots, \mathbf{e}(k\text{-}1)\,)$$

The summation is of $\mathbf{s}(k\text{-}1)$ over the space *S*. Due to the Markov property assumed for $\mathbf{s}$, this simplifies to:

$$= \sum p(\, \mathbf{s}(k) \mid \mathbf{s}(k\text{-}1), \mathbf{a}(k\text{-}1)\,) \;\; p(\, \mathbf{s}(k\text{-}1) \mid \mathbf{e}(0), \mathbf{e}(1), \ldots, \mathbf{e}(k\text{-}1)\,)$$

So the belief state is:

$$b_{\mathbf{s}(k)}(\, \mathbf{s}(k)\,) =$$

$$\alpha \; p(\, \mathbf{o}(k) \mid \mathbf{s}(k)\,) \; \sum p(\, \mathbf{s}(k) \mid \mathbf{s}(k\text{-}1), \mathbf{a}(k\text{-}1)\,) \;\; p(\, \mathbf{s}(k\text{-}1) \mid \mathbf{e}(0), \mathbf{e}(1), \ldots, \mathbf{e}(k\text{-}1)\,)$$

Note that $p(\, \mathbf{o}(k) \mid \mathbf{s}(k)\,)$ is just the sensor function, and $p(\, \mathbf{s}(k) \mid \mathbf{s}(k\text{-}1), \mathbf{a}(k\text{-}1)\,)$ is the transition function. The distribution $p(\, \mathbf{s}(k\text{-}1) \mid \mathbf{e}(0), \mathbf{e}(1), \ldots, \mathbf{e}(k\text{-}1)\,)$ is the belief state for the previous time step, $b_{\mathbf{s}(k\text{-}1)}(\, \mathbf{s}(k\text{-}1)\,)$, assuming that it reduces to $p(\, \mathbf{s}(k\text{-}1) \mid \mathbf{e}(0),$

e(1), … , **o**(k-1)) due to causality (the action **a**(k-1) is chosen after the belief state for **s**(k-1) is determined, so it should not influence that belief state).

To summarize in more explicit notation, the belief state is:

$$b_{\mathbf{s}(k)}(\mathbf{s}(k)) = \alpha\ O(\mathbf{o}(k) \mid \mathbf{s}(k)) \sum T(\mathbf{s}(k) \mid \mathbf{s}(k-1), \mathbf{a}(k-1))\ b_{\mathbf{s}(k-1)}(\mathbf{s}(k-1))$$

Thus, for the assumed observation and transition functions, there is a recursive calculation for the belief state. Once the belief state is calculated for one time step, that belief state along with the next action and observation will allow a calculation of the same effort for the next belief state. At initiation of the process, when there is no previously computed belief state, a prior belief state, $b_{\mathbf{s}(0)}(\mathbf{s}(0))$, must be assumed.

Appendix B:
Common Philosophies of Alerting Logic Design

Three common philosophies of alerting logic design have been identified to classify existing or proposed hazard alerting systems. Following is a more detailed and general description of each philosophy. Parallel approach collision prevention serves as an example.

B.1 Trajectory Conformance Monitoring

This type of logic uses non-conformance of a system to established procedures as a basis for alerting. For example, figure B.1 shows a system state with respect to a normal operating region in state space. If the state exits outside the normal operating region an alert is issued. The system exists to prevent occurrences of a hazard, but no explicit prediction of a hazard event is required for triggering an alert. As shown, the normal region is defined so as to be mutually exclusive of the hazard, even though the hazard is not explicitly modeled in the final algorithm. In PRM, for example, as long as both aircraft remain outside the NTZ, the hazard cannot occur. An aircraft entering the NTZ will trigger an alert whether or not it actually threatens another aircraft.

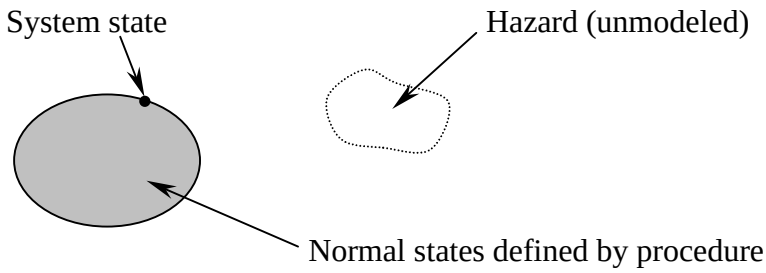


Figure B.1: Trajectory Conformance Alerting

A deviation (“blunder”) from the normal procedure is a necessary precursor to a hazard event, so it can be argued that an observed deviation from normal is sufficient reason for an alert and corrective action, provided such a policy does not result in a high rate of

alerts occurring without a blunder. Frequent unnecessary alerts during normal system operation would come to be perceived as incorrect by operators, and might in time cause operators to ignore or delay responding to alerts.

In addition to establishing that the non-blunder alert rate is acceptably low, it should be shown that when a blunder does occur there will be an evasive maneuver having an adequate likelihood of success. Such an analysis typically involves a reference dynamic model of the system, and iterative adjustment of the threshold. Because of the dependence of the threshold on the operational procedure, it may be necessary to adjust the procedure itself to achieve performance goals. For example, it was concluded that PRM could be used with parallel runways spaced no less than 3400 feet apart because below this spacing the likelihood of safe resolution of a blunder was too low in simulation studies.

B.2 Nominal Trajectory Hazard Prediction (Unnecessary Alert Prevention)

This alerting strategy involves continuous checking for a particular hazard through explicit prediction of the non-alert, or nominal, system trajectory (figure B.2). For an alert to occur, the hazard event must be predicted. Under this philosophy, the logic avoids alerts that are not clearly justified with respect to the hazard. The hazard is

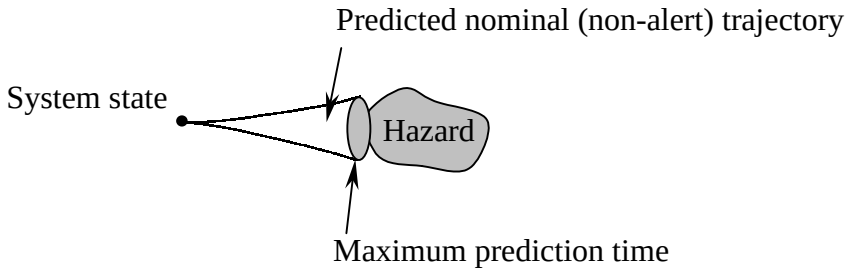


Figure B.2: Nominal Trajectory Prediction Alerting

described in terms of a set of state variables composing a state space. The trajectory model, which might be probabilistic, worst case, or a single predicted trajectory, is

propagated forward in this state space from the current, measured location. In the figure B.2 example, the trajectory model is a worst case model, and is predicting that a hazard may be encountered in the future.

Due to uncertainty in prediction and the consequent possibility of unnecessary alerts, it is insufficient to define the alerting rule as "alert when a hazard is predicted." Typically an additional metric or metrics are required for threshold definition. Possible metrics include the degree of certainty in occurrence of a predicted event (for a probabilistic trajectory model), or the predicted time-to-collision (for worst case or single-trajectory models). The values of threshold parameters must be chosen to satisfy both safety and unnecessary alert goals. In the illustrated example, collision prediction time is the metric and a particular value of this must be selected to define the threshold. Using a reference model of the behavior of the entire human-controlled system (able to describe its dynamics both before and after an alert occurs, and covering all possible initial conditions in state space), optimal threshold parameters are determined, typically through repeated Monte Carlo simulation and adjustment (Yang & Kuchar, 2000).

Whether a hazard is imminent for the nominal trajectory is not a direct indication of whether an evasion maneuver will be safe. In this type of logic, the justification of alerts is inherently stressed over the safety of the alerting decision.

In the course of analysis, the complexity of the logic may increase to cover special cases that were not initially foreseen. This is likely when few state variables are available for measurement or the actual system dynamics are not well understood. An example is the development of TCAS logic for midair collision prevention. This logic began with a simple range rate and time-to-collision prediction model (with adjustable parameters for the threshold prediction time and miss distance) characterized by large trajectory errors, and was eventually augmented with conditional statements and new parameters in order to handle problem scenarios (Drumm, 1996). For example, a situation where two aircraft unknowingly fly parallel at the same speed may be unacceptable, yet trigger no alerts when using a time-to-collision criterion only. To cover such problem scenarios additional checks were added to the logic.

Other examples of logics that use explicit incident prediction as the basis of thresholds are GPWS and AILS.

B.3 Existence of Safe Escape Options (Safety Monitoring)

In general there may be specific completion conditions that must be met in order for a potential incident to be considered resolved, and it is possible to make deferral of alerts conditional on the predicted attainment of such conditions. For example, the MIT logic issues alerts based on knowledge that a collision will probably not occur within a certain period of time following the alert.

This type of logic is superficially similar to the nominal trajectory hazard checking method described in section B.2 in that it involves a trajectory model. As illustrated in figure B.3, a hazard event is once again defined in terms of measurable state variables. A trajectory model is used to propagate the system state, but this time under

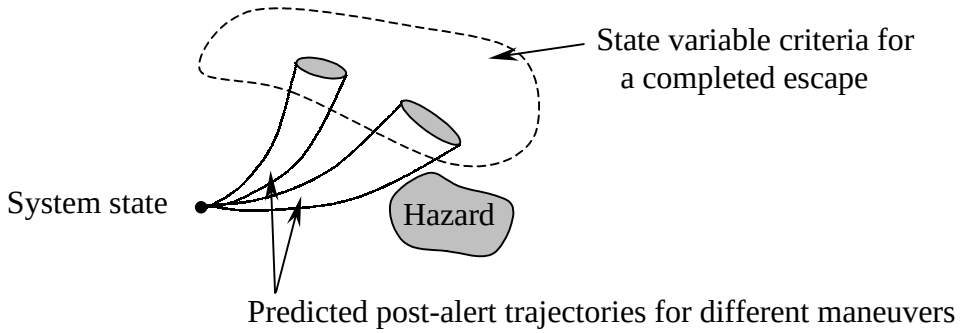


Figure B.3: Ensuring that Safe Options Exist

the assumption that an alert has occurred or will occur at a particular time, resulting in escape maneuvers. In general there may be multiple maneuver options (represented by evolving state envelopes—each resembling a horn—in figure B.3), corresponding to different warning inputs that can be issued to operators. Completion conditions are defined in terms of the evasion trajectory and state variables. As shown, completion conditions may require that the system reach a specific region in state space. In addition, it may be required that the system reach the completion state set within a particular time

interval. Finally, if the hazard is some catastrophe, then the completion state set cannot intersect with the set of hazard states and be considered as part of a desirable alerting outcome.

If a probabilistic trajectory model is to be used, then alerting decisions will be based on the probability of reaching the completion state set within the required time interval. Therefore an additional component of completion is a threshold probability, such that above this an alerting decision is considered safe.

If the trajectory model is worst case or a single trajectory, safety requires that all trajectories for an alerting option reach the completion set within a given time interval. Safety is marginal if any one of the trajectories reaches a boundary value of the completion state set or allowed time interval.

According to this philosophy an alert may be deferred as long as an available alerting option is safe. An alert can no longer be deferred when safety becomes marginal. In other words, an alert is considered justified when there may be no safe option remaining at the next alerting opportunity.

In this method safety is fixed at the threshold, resulting in a loss of direct control over unnecessary alerts. This is because whether a post-alert maneuver is safe is not a direct indication of whether the nominal system trajectory is safe (i.e. whether an alert will be an unnecessary alert). For example, it may be possible for an evasion option to become marginally unsafe, triggering an alert, even when no hazard would be encountered on the nominal trajectory.

References

Drumm, A. C. (1996). *Lincoln Laboratory Evaluation of TCAS II Logic Version 6.04a. Volume I.* Lincoln Laboratory, MIT. Lexington, MA.

Yang, Lee C. & James K. Kuchar. (2000). *Aircraft Conflict Analysis and Real-Time Conflict Probing Using Probabilistic Trajectory Modeling*. MIT, International Center for Air Transportation. Report No. ICAT-2000-2. May.