User Interface Validation using Mode Confusion Detection

Jayaprakash Suraj Nandiganahalli, Sangjin Lee and Inseok Hwang

Purdue University, West Lafayette, IN, 47907, USA

Bong-Jun Yang

Optimal Synthesis Inc., 95 First Street, Suite 240, Los Altos, CA 94022-4622, U.S.A.

Due to the rapid advancements in the digital avionics systems, the aircraft operation is getting more automated. However, this does not completely exclude the operations carried out by the pilot via interaction with the automation through the User Interface (UI) (or more commonly known as flight deck instruments). At times, this interaction raises situational awareness issues to the pilot about the behavior of the automation. One such critical issue is mode confusion where a mismatch occurs between the behavior of the automation expected by the pilot and the actual behavior of the automation. This paper examines the mode confusion detection problem from the UI design perspective by taking a control-theoretic perspective on dynamical systems. The automation is modeled as a hybrid system and the pilot is modeled as an intent-based discrete-event system for the given UI which is modeled as a filter of flight information. By inferring and comparing the intents of automation and pilot using a host of information such as the aircraft’s continuous states and flight modes, and automation’s and pilot’s inputs, this paper attempts to validate the given UI via mode confusion detection. The causes responsible for the detected mode confusion are used to suggest improvements to the existing UI design, thereby improving the situational awareness of the pilot and aircraft safety. The framework is tested on two representative mode confusion incidents/accidents and the associated UIs are validated.

I. Introduction

Automated control systems are known to provide tremendous benefits to aircraft operation in terms of accuracy and efficiency. However, increased reliance on automation raises potential safety problems such as lack of situational awareness for users about the current and future status and behavior (referred to as mode) of the automation, leading to automation/situation induced mode confusion or mode awareness incidents/accidents. For example, modes of the automation of an aircraft could be either mode control panel (MCP) related such as Vertical Speed (V/S), Heading (HDG), Glide Slope (G/S), etc. or flight management system (FMS) related such as Vertical Navigation (VNAV) and Lateral Navigation (LNAV). This paper only considers MCP related modes which govern short-term tactical aircraft behaviors.

Mode confusion refers to the automation-human interaction situation where there exists a divergence between the real state of the physical system (e.g., actual behavior of the automation (aircraft)) and the human’s (e.g., pilot’s) perception of the automation as observed via its interface (e.g., flight deck). This issue is prevalent in most automation-human interactions, but more predominant in complex safety-critical systems such as commercial aircraft where most flying tasks are handled by autopilots and autothrottles (collectively known as automation). In this setting, pilots spend much of their time as a supervisor (e.g.,
providing inputs through the MCP or FMS to interact with the automation) and an observer (e.g., reading Flight Mode Announcer (FMA) display modes). Therefore, the performance of the aircraft operation is highly dependent on the successful collaboration among the following three components: the automation, the user interface (UI) and the pilot. Thus, appropriate modeling of each component (as shown in Figure 1) is crucial for understanding their interacting behaviors to successfully operate an aircraft.

Figure 1. Schematic for Automation-Pilot Interaction

This paper assumes that the automation logic is valid and correct, however the UI that is critically important for the pilot decision making, is considered responsible for creating “confusion or surprise” due to inaccurate representation of what is going on underneath in the automation. This is mainly due to the many-to-one mapping of several modes to the lesser number of UI displayed information (e.g., on different sections of the FMA). Thus, either under or over represented UI could pose a potential risk for the pilots to lose real-time situational awareness as reported in many NASA Aviation Safety Reporting System (ASRS) reports and other incidents. Through literature survey, we have found this is a global problem prevalent in various types of aircraft and occurs in critical phases of flight especially during departure and landing. The question we ask is: Can we detect and further resolve this issue, so that automation-pilot interaction can be made safer?

A. Literature Survey

The mode confusion detection problem has been addressed by many researchers along different lines in the literature. Degani et al. consider only the discrete model for the automation (using discrete flight mode information) and neglect the continuous dynamics aspect of the automation, while Oishi et al. use a hybrid model of automation for autopilot verification purposes only and do not address mode confusion in particular. Miller et al. model only the automation without modeling the pilot and detect mode confusion by connecting it to a simulation of the flight deck reviewed by pilots and experts. Bass et al. consider a range of automation models and apply formal method techniques for a basic pilot model. Their method is a bottom-up approach involving iterative addition of system constraints to a very simple automation model to address spurious anomalies. We believe though this is an interesting work, repetitive manual corrections are unavoidable unless some learning methods are employed. Additionally, Degani et al. do not distinguish the design of the UI from the pilot model during the design and validation phases and assume the pilot always behaves according to the user manual which is an abstraction of the automation model. Distinguishing the UI model from the pilot model and automation model (a modular approach) is important because the pilot does not necessarily behave as in the user manual in all situations. This is also important because it is usually difficult to know how a pilot will react when operational conditions and task environment changes, especially in abnormal conditions.

Some other researches have posed the mode confusion detection problem as anomaly detection problem purely based on data-driven techniques such as cluster-based anomaly detection by Das et al., multiple kernel anomaly detection by Li et al., etc. However, these approaches base their decisions of anomalies
on data itself and not on the models that generate the data. Taking a model-driven detection approach is computationally beneficial because detecting deviations of the actual data from the true model is much efficient.

Finally, it can be observed that the mode confusion detection and the associated UI validation problems have not been approached from the control-theoretic perspective on dynamical systems as is the case with our paper. By separately developing automation, pilot and UI models, our paper attempts to descriptively and efficiently capture the behavior of the automation-pilot interaction by keeping track of different automation and pilot related parameters as filtered by the given UI and other operational data such as flight plan and air traffic control (ATC) clearance information.

The four incidents/accidents that we identified from literature and analyzed for the UI validation and mode confusion detection, across the flight spectrum (i.e., from Boeing to Airbus, from autopilot-only confusion to autopilot and autothrottle confusion, from departure to landing phase of flight) are given below. However, due to space limitations we describe the results for two mode confusion incidents/accidents only i.e., Speed Protection incident and Bangalore accident (a more complex mode confusion example).

Mode confusion incidents:

- **Kill the Capture Incident (1989)**
  - Memphis Center incident
  - Cause: Autonomous mode transition from Vertical Speed (V/S) to Capture mode and from Capture to V/S Free Climb mode.

- **Speed Protection incident (1994)**
  - Paris: Tarom incident
  - Cause: Autonomous mode transition from V/S to OP CLB (Open Climb) mode

- **Bangalore accident (1990)** (eerie similarities with Asiana B-737 SFO (2013) accident)
  - Autopilot and Autothrottle confusion
  - Cause: Autonomous mode transition from V/S + SPD (Speed) mode to OP DES (Open Descent) + THR IDLE (Idle Thrust) mode and continuation of OP DES + THR IDLE mode (one Flight Director (FD) OFF).

- **Roll Confusion accident (2004)**
  - Egypt accident: Flash Airlines accident
  - Cause: Autonomous mode transition (from HDG SEL (Heading Select) mode to CWS-R (Control Wheel Steering-Roll) mode.

### B. Proposed Approach

The aim of the proposed research is to implement a general framework as initially proposed by Lee et al., for UI validation purposes using mode confusion detection (by validating on various mode confusion incidents/accidents) from the control-theoretic perspective. An intent-based framework for mode confusion detection is proposed because the mode confusion is the divergence in expectations or intents of the automation and the pilot about the actual behavior of the aircraft. Intent describes automation’s/pilot’s expectations in flying the aircraft and can abstractly capture the aircraft motion. The proposed approach develops modular, yet interacting models of the automation, the pilot and the UI, unlike existing works. The automation which has interacting logical behavior (e.g., flight modes transitions) and physical behavior (e.g., speed change or altitude change) is initially modeled as a hybrid system and then is abstracted into an intent-state transition model using automation intent inference. The UI on the other hand is modeled as a simple filter of flight information (e.g., flight mode and (guard) system condition), whose role is to basically either display or not display a particular flight information. The pilot is modeled as the discrete-event system with the pilot’s intents as its states to represent the pilot’s behavior using pilot intent inference.
intent inference (for automation and pilot) is performed using a host of information such as the continuous state (e.g., altitude, speed) and discrete flight mode (i.e., V/S, ALT HLD) of the automation (aircraft), automation’s and pilot’s inputs and flight plan information. The inferred automation’s and pilot’s intents are then compared to see if there is any mismatch to detect a mode confusion. The framework described here is intended to complement the existing works in the field of system design, human factors and aviation psychology which also seek to better understand the automation-pilot interaction issues.

To summarize, our approach is unique in that it can be used for either offline or on-board mode confusion detection in a descriptive and efficient manner. The automation-pilot interaction is continuously kept track of by monitoring different automation and pilot related parameters filtered shown by the given UI, in addition to the aircraft operational data such as ATC clearance, flight plan and weather related information. This facilitates development of an on-board pilot alert system for mode confusion detection. Also, the causes identified for the mode confusion can be further used to validate the given UI and suggest improvements to it’s design (either off-line or on-board through context-sensitive displays) to help increase the pilot’s situational awareness, thus increasing aircraft safety.

In this paper, we address UI validation using mode confusion detection for the tactical aircraft behaviors only. For example, the current algorithm considers the primitive (tactical level) intents, such as “climb” or “constant heading” or “decelerate”, etc. However, a series of such tactical level intents can make a higher level intent such as “go to the next waypoint” or “approach to an airport”. Thus, our mode confusion detection framework can be easily extendable to capture and address much wider class of automation surprise problems.

This paper is organized as follows: Section II explains the basic framework of mode confusion detection. Section III analyzes mode confusion detection and discusses about the associated UI validation issues on two mode confusion incidents/accidents. Section IV discusses the conclusions and future work.

II. Mode Confusion Detection Framework

In this section, the overall framework of our algorithm for UI validation is presented and each component is described in detail. The architecture for UI validation using mode confusion detection is shown in Figure 2 inspired by our previous work. Our approach develops modular models of automation (hybrid system) and pilot (discrete-event system), and models the UI as a filter of flight information, unlike existing works.

Mode confusion could happen when the actual behavior of the automation does not match with the expected goal of the pilot. Thus, the proposed way to detect mode confusion is to compare the pilot’s expectation (or intent) of the automation to that of the actual behavior of the automation. To understand the behavior of the automation-pilot interaction system, we need to model the respective components i.e., automation, pilot and the UI, which will be discussed in the following subsections.

It is to be noted that since the aircraft’s continuous states (e.g., acceleration, altitude, etc.) are generally corrupted by measurement noise (due to sensors such as accelerometer, altimeter, etc.), a Kalman filter is implemented for each of the modes of the automation to extract the filtered continuous states. In the case that the flight modes (e.g., V/S, ALT HLD, etc.) are unobserved, both the continuous state and flight mode are estimated by the hybrid estimation algorithm which has a bank of Kalman filters. Similarly, the continuous inputs of pilot (e.g., control yoke commands) are normally subjected to noise. Therefore, a simple (Finite Impulse Response) FIR filter is implemented to extract the filtered continuous inputs of the pilot. The resulting automation (aircraft) behavior (i.e., filtered continuous states and flight modes) along with the pilot’s inputs to the MCP (e.g., turn down the ALT knob) are used to infer the automation’s intent. Similarly, the pilot behaviors (i.e., filtered pilot inputs) along with the user interface displaying the aircraft’s continuous states and discrete flight modes, flight plan information are used to infer the pilot’s intent. Then, by comparing the intents of automation and pilot and looking for a mismatch, mode confusion is detected. By understanding the cause for mode confusion, the UI is validated along with the recommendations to improve it’s design through context-sensitive feedback of critical information to facilitate complete situational awareness for the pilot.

A. Automation Model

The aircraft motion can be modeled by decomposing it’s behavior into a sequence of discrete flight modes along with the mode-specific continuous dynamics (to describe aircraft behavior) as shown in Figure 3. The
transition between flight modes can be described using guard conditions which may be either autonomous or pilot triggered. Thus, the automation (aircraft) can be naturally modeled using the hybrid system which captures interacting physical behavior of the aircraft (using continuous dynamics) and the logical behavior of the automation (using discrete dynamics) controlled by either automation or pilot or disturbance governed by preset guard conditions. For example, due to the rapid descent profile of the aircraft, if the current airspeed exceeded the maximum airspeed in the particular configuration, it can trigger an autonomous mode transition from V/S + SPD mode to OP DES + THR IDLE mode.\(^5\)

Figure 3. Automation (Aircraft) Motion as a Hybrid System

More formally, a hybrid system can be described by a 7-tuple: \( H = (Q, X, R, f, G, \Sigma, U) \) where,

- Set of discrete modes (e.g., flight modes of automation): \( Q \)
- Domain of the continuous states (e.g., altitude of an aircraft): \( X \)
- Transition function (representing mode changes): \( R \) i.e., \( q_{k+1} = R(q_k, x_k, \sigma_k, u_k) \), where \( q_k \) is the discrete state, \( x_k \) is the continuous state, \( \sigma_k \) is the discrete input, and \( u_k \) is the continuous input at
time $k$.

- Continuous dynamics (representing aircraft behavior): $f$ i.e., $x_{k+1} = f_q(x_k, u_k)$
- Guard (values for $x$ and $q$ enabling the transition from $q(i)$ to $q(j)$): $G : Q \times Q \to 2^I$, where $\Omega = X \times Q$ represents the invariant domain
- Set of discrete inputs: $\Sigma = \Sigma_a \times \Sigma_p \times \Sigma_d$ (Automation-controlled ($\Sigma_a$), pilot-controlled ($\Sigma_p$), disturbance ($\Sigma_d$) discrete inputs)
- Set of continuous inputs: $U = U_a \times U_p \times U_d$ (Automation-controlled ($U_a$), pilot-controlled ($U_p$), disturbance ($U_d$) continuous inputs)
- $\sigma = (\sigma_a, \sigma_p, \sigma_d) \in \Sigma$
- $u = (u_a, u_p, u_d) \in U$

Then, the automation’s (aircraft’s) hybrid states such as the continuous state corrupted by process and measurement noise and discrete mode of the automation can be estimated (i.e., $\hat{x}_a(k)$ and $\hat{q}_a(k)$ respectively) using a hybrid estimation algorithm with a bank of Kalman filters (one for each discrete mode for the mode-specific continuous dynamics) under the multiple model adaptive estimation (MMAE) framework.

B. User Interface Model

The main goal of this paper is to validate the UI (as pertaining to mode confusion) and assess the sufficiency of information displayed on the UI (delivered by the automation) for aircraft safety. A properly designed UI must enable the pilot to gain complete real-time situational awareness about the state of the dynamical system (e.g., aircraft) so as to correctly predict the trend of his/her actions (i.e., inputs or commands). To this end, the user interface is modeled as a filter whose role is to filter out by either displaying or not displaying the automation flight modes and guard conditions before displaying it to the pilot. Thus, the UI either under or over represents the information (e.g., discrete flight modes, etc.) of the automation to the pilot. For example, a UI design may filter out by not displaying “V/S free” mode of the automation. Note that “V/S free” mode is an altitude unconstrained version of “V/S” mode.

C. Pilot Model

On the other hand, the behavior of the pilot can be understood in terms of his/her intentions or intents about the actual aircraft behavior. Intent describes pilot’s expectations in flying the aircraft and can abstractly capture the aircraft motion. Thus, the pilot behavior can be modeled using the discrete-event system with intent as its states. This is a reasonable model of pilot, since for the purposes of mode confusion detection, we are interested in knowing what the pilot wants the automation to do, which can be expressed using his/her intents. It is to be noted that our interest is not to model the psychological details about the pilot, rather to understand or infer the cause (e.g., climb intent) which makes the pilot to issue control commands (e.g., turn up the MCP ALT knob) to the automation using the information from the UI. A mathematical description is given below.

We denote the set of pilot’s intents as $I = \{I^1, I^2, \cdots, I^N\}$. Note that the intent set includes “do nothing i.e., $\emptyset$” too. Mathematically, a pilot discrete-event system with intent as it’s states can be defined as:

$I_p(k+1) = f_p(I_p(k), x_p(k), q_p(k), \hat{u}_p(k), \sigma_p(k), \Gamma), I_p \in I$

where $f_p$ denotes the state transition function which maps the pilot’s intent at time $k$ to that at time $k+1$. Here, subscript $p$ stands for the pilot, $I_p(k)$ is the pilot intent at time $k$, $x_p(k), q_p(k)$ are the aircraft’s continuous states and discrete flight modes displayed by the UI to the pilot respectively, $\hat{u}_p(k), \sigma_p(k)$ are the filtered continuous and discrete pilot inputs respectively and $\Gamma$ is the flight plan information. The above intent dynamics for the pilot allows to predict the intent of the pilot at the next time step $k+1$ as $I_p(k+1)$. 

American Institute of Aeronautics and Astronautics
D. Construction of Intent Set

Evidently, an important relevant task to describe the intent dynamics is initially to build a suitable intent set. Here, we have focused on building short term or tactical level intents to represent the aircraft’s behaviors, since we are interested in mode confusion problems operating at the MCP level. The construction of intents is motivated by the available flight modes of the automation, i.e., for autothrottle (governs speed behavior) and autopilot (governs lateral and vertical behavior). Thus, a 3-tuple tactical intent that consists of the speed intent (accelerate, decelerate, constant speed), lateral intent (turn left, turn right, constant heading) and vertical intent (climb, descent, constant altitude) are constructed. Using these 3-tuple intents, tactical aircraft’s behavior such as climb with constant speed and constant heading, etc can be described. Mathematically, the tactical level intent set can be defined as
\[ I = \{ I_1, I_2, \cdots, I_{27} \} \]
representing a set of 27 3-tuple tactical intents (generated compositionally under the check of feasibility), with each \( I_i = (I_{iS}, I_{iL}, I_{iV}) \), \( i \in \{1, 2, \cdots, 27\} \) and \( I_{iS} \in \{\text{Accelerate, Decelerate, Constant Speed}\}, I_{iL} \in \{\text{Turn Left, Turn Right, Constant Heading}\}, I_{iV} \in \{\text{Climb, Descent, Constant Altitude}\} \). Thus, the aircraft’s behavior can be described using transitions between the tactical intents as shown in Figure 4, where each specific automation’s (aircraft’s) behavior (corresponding to a flight mode shown in red) can be equivalently described using 3-tuple tactical intent transitions (shown in purple). Similarly, the pilot’s behavior can also be described using the above 3-tuple tactical intent transitions.

![Figure 4. Automation Behavior described using 3-tuple Tactical Intent Transitions](image)

E. Intent Inference

As discussed before, a way to detect mode confusion is to compare the automation behavior with the pilot behavior under the characteristics of the given UI. Since, the automation is modeled as a hybrid system and the pilot is modeled as a discrete-event system with intent as its states, the automation’s states cannot be directly compared with the pilot’s intent states. Based on the fact that the actual aircraft motion can be abstractly described by intent, we abstract the automation hybrid model to an intent model through automation intent inference. Generally, the automation’s and pilot’s intents are inferred using a host of information such as the continuous states and discrete flight modes of the automation, automation and pilot inputs and flight plan information. This step is formally referred to as intent inference. Mathematically, the automation (subscript \( a \)) intent and pilot (subscript \( p \)) intent can be inferred as below:

1. Automation’s intent at time \( k + 1 \) is inferred as \( I_a(k + 1) \) given by:
   \[
   I_a(k + 1) = f_a(I_a(k), \dot{x}_a(k), \dot{q}_a(k), u_a(k), \sigma_a(k), \dot{u}_p(k), \sigma_p(k)), I_a \in I
   \]

2. Pilot’s intent at time \( k + 1 \) is inferred as \( I_p(k + 1) \) given by:
   \[
   I_p(k + 1) = f_p(I_p(k), x_p(k), q_p(k), \dot{u}_p(k), \sigma_p(k), \Gamma), I_p \in I
   \]

The above equations indicate that intent inference for the automation is critically dependent on the knowledge of the \( \dot{x}_a(k), \dot{q}_a(k) \) (i.e., estimated continuous state and discrete flight mode respectively) of the automation, \([u_a(k), \sigma_a(k)], [\dot{u}_p(k), \sigma_p(k)]\) (i.e., automation’s inputs and pilot’s estimated continuous and discrete inputs respectively). Similarly, the intent inference for the pilot is critically dependent on the
knowledge of the $x_p(k), q_p(k)$ (i.e., UI displayed continuous state and discrete flight mode respectively) of the automation, $\hat{u}_p(k), \sigma_p(k)$ (i.e., pilot’s filtered continuous and discrete inputs), and $\Gamma$ flight plan information. Finally, $I_a(k+1), I_p(k+1)$ refer to the inferred automation’s and pilot’s intents at time $k+1$ respectively. It is to be noted that the the predicted intent of the pilot at the next time step $k+1$ i.e., $I_p(k+1)$ corresponds to the inferred pilot intent.

F. Mode Confusion Detection and UI Validation

The mode confusion can then be detected by comparing the inferred automation’s and pilot’s intents $I_a(k)$ vs. $I_p(k)$. That is, if there is a mismatch between the inferred intents of the automation and pilot, the algorithm detects it as a mode confusion.

Additionally, we have identified the following as the main causes for mode confusion (as connected to the UI design issue):

1. The critical modes involved in autonomous mode transitions are not displayed on the UI.
2. The associated guard conditions for autonomous mode transitions are not indicated on the UI.
3. When mode confusion occurs, the pilot usually is surprised and confused about what caused the issue and does not know which information on the UI to pay more attention to. Thus, highlighting the critical information on the UI is important.

Thus, improving the UI design in the above areas should help bring the pilot’s attention to the cause of mode confusion (especially important in time-critical incidents) and improve his/her situational awareness to effectively resolve it.

III. Analysis and Validation of UI via Mode Confusion Detection on Two Incidents/Accidents

In this section, two mode confusion examples are initially presented in detail. Then, the methodology and results for mode confusion detection and UI validation on these two examples as discussed in Section II are explained.

A. Tarom Flight 381 Speed Protection Autopilot Mode Confusion Incident

The Airbus-310’s “Speed Protection” incident of 1994 is an autopilot mode confusion incident as depicted in Figure 5. The actual aircraft trajectory is shown in blue, while the pilot expected trajectory is shown in grey.

This type of mode confusion was prevalent in different productions of Airbus A310/A320/A330/A340. This incident is based on the autonomous speed protection in A-310 that occurred on Tarom Flight 381 from Bucharest to Paris Orly on 24th September 1994. This infamous incident occurred during the descent phase. When ATC allowed the aircraft to perform descent, the aircraft was located above the reference glide slope and was in the process of performing a steep descent maneuver to intercept the normal descent path. Following the Airbus’s recommendation, the pilot had used V/S mode to descend by setting the FCU (Flight Control Unit, same as Boeing’s MCP) altitude to the missed approach altitude. The unfortunate thing was the timing of this target altitude setting. The pilot had undertaken this action when the aircraft was already below the missed approach altitude and the maximum allowable airspeed (in that configuration) had reduced significantly, resulting in the current airspeed exceeding the maximum allowable airspeed. The above two (guard) conditions were just enough to trigger the autonomous speed protection to prevent airspeed reaching dangerous levels by automatically substituting V/S mode with OP CLB mode. This caused the aircraft to climb and capture the missed approach altitude that was initially set, instead of the pilot expected descent, thus causing a confusion.

1. Autopilot Model

The flight modes relevant in the “Speed Protection” incident are the vertical modes, namely V/S mode, OP CLB mode and Altitude Hold (ALT HLD) mode. OP mode has climb or descent mode that has no specific
vertical speed target and gives priority to airspeed. The selected sub-mode of OP (i.e., OP CLB or OP DES) depends on whether the target altitude set in the FCU is above or below (respectively) the aircraft’s current altitude. Figure 6 depicts the inner-workings of the automation (i.e., autopilot-logic) with a hybrid system model representing different relevant flight modes and their respective continuous dynamics. The discrete mode transitions are either autonomous (broken arrows) upon satisfaction of a preset guard condition or pilot triggered (solid arrows). For example, in Figure 6 there is an autonomous flight mode transition from V/S climb mode to OP CLB mode when the aircraft’s current altitude is less than the missed-approach altitude and the current airspeed is greater than the maximum airspeed in the configuration.

The continuous dynamics of the hybrid system corresponding to each discrete state (or flight mode) is represented as a discrete-time linear dynamical model with altitude \( h \) and altitude change rate \( \dot{h} \) as the continuous states, while the flight modes V/S, OP CLB and ALT HLD are the discrete states of the hybrid system. As discussed in the previous section, using the estimated continuous state and flight mode of the aircraft (automation) and the automation’s and pilot’s discrete and (estimated) continuous input information, an automation intent model is constructed. For the “Speed Protection” incident, the automation (here, vertical dimension of autopilot logic is only relevant) intent map is shown in Table 1.

<table>
<thead>
<tr>
<th>Flight Mode</th>
<th>Automation Intent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hold Altitude mode</td>
<td>Constant Altitude</td>
</tr>
<tr>
<td>Vertical Speed mode</td>
<td>Climb/Descent</td>
</tr>
<tr>
<td>OP mode</td>
<td>Climb/Descent</td>
</tr>
</tbody>
</table>

Table 1. Automation Intent Map according to Flight Modes

In Table 1, the relevant vertical flight modes are shown corresponding to respective intents of the automation. The ambiguity in uniquely determining the automation’s intent (e.g., OP mode has two intents (either climb or descent)) can be resolved by either considering the annunciated FMA sub-mode information (e.g., OP CLB implies climb intent and OP DES implies descent intent) or by considering the actual continuous state information (i.e., if the altitude change rate is positive, it denotes climb intent).

2. **Pilot Model**

The pilot intent model for the “Speed Protection” incident is constructed using the continuous state and the discrete mode of the automation displayed in the UI along with the pilot’s input information. Here, only discrete pilot input is relevant.
In the “Speed Protection” incident, the current intent of the pilot was to descend as determined from the UI displayed vertical speed (e.g., negative vertical speed rate change) and current mode (e.g., V/S mode) information. From Table 2, a single flight mode (e.g., OP mode) corresponds to two pilot’s intents (either climb or descent intent). This ambiguity can be resolved by considering the pilot’s input information. For example, in the “Speed Protection” incident, during the descent motion of the aircraft, the pilot did not do anything (over some time period). This indicates the pilot’s intent for the aircraft is to descend (i.e., intent is unchanged).

### Mode Confusion Detection

Mode confusion is detected by comparing the inferred intents of the automation and pilot. Figure 7 shows the intended altitude trajectories of the autopilot and pilot along with the actual flight modes (with autonomous transitions shown as the broken red arrows). The fork in the intended trajectory indicates the mismatching expectations between the aircraft’s behaviors intended by the autopilot and the pilot.

Figure 8 shows the inferred intents of the autopilot and pilot over the entire time interval. It can be observed that after 10 time steps and 13 time steps, there exist two divergences of the autopilot’s and pilot’s intents (climb vs. descent and fly constant altitude at the missed-approach altitude vs. maintain constant altitude upon landing.) The lower plot of Figure 8 shows mode confusion detection by the our algorithm.
4. The Results of User Interface Validation

Careful observation of the Airbus-310 “Speed Protection” incident reveals the followings as important causes for the mode confusion incident:

1. Autonomous mode transition from V/S to OP DES mode.

2. Even though all the critical flight modes were displayed on the UI (FMA), when mode confusion occurred, the pilot was surprised and confused as to what caused the issue. The pilot couldn’t discern as to which information on the UI he/she had to pay more attention to.

3. Additionally, the guard conditions responsible for the autonomous mode transitions seem to be unknown to the pilot.

Thus, the analysis suggests that upon detection of on-board mode confusion, by highlighting the critical information such as the mode involved in the autonomous mode transition (e.g., OP CLB mode) and the associated guard condition (e.g., current airspeed exceeds maximum airspeed) through feedback in a context-sensitive UI display, the pilot’s attention can be drawn to the cause of mode confusion, thereby improving his/her situational awareness.
B. Indian Airlines Flight 605 Autothrottle and Autopilot Mode Confusion Incident

The Airbus-320's Bangalore accident of 1990 involving Indian Airlines Flight 605\textsuperscript{5} is an interesting and complex mode confusion accident due to both “Autothrottle and Autopilot” confusion as depicted in Figure 9. The actual trajectory is shown in blue (with red aircraft), while the pilot expected trajectory is shown in grey (with grey aircraft).

The investigation report quoted: “The entire crash is the result of what the pilots did not do between 295 to 320 seconds during 25 seconds (i.e., less than half a minute) and not what they did”. On February 14th, 1990, the crew of Indian Airlines Flight 605 flying from Mumbai to Bangalore, India decided to perform visual manual landing, since the route was very familiar. The aircraft prepared its manual descent maneuver to capture the 3 degree Glide Slope using V/S + SPD (autopilot and autothrottle mode respectively) mode after manually transitioning from OP DES + THR IDLE mode initially. During this period, the copilot mistakenly input a target altitude lower than the current altitude. Unfortunately, at this very same time, due to the rapid descent profile of the aircraft, the current airspeed exceeded the maximum airspeed in the particular configuration, triggering an autonomous mode transition from V/S + SPD mode to OP DES + THR IDLE mode. This transition went unnoticed by the crew with the aircraft quickly losing airspeed and descended much lower than the desired 3 degree Glide Slope. Lately realizing his mistake, the captain turned off his Flight Director, expecting an autonomous transition to SPD mode to regain engine power to maintain the approach speed and perform a climb maneuver and keep aircraft afloat. However, since only one Flight Director was OFF (note that there are two Flight Director switches and both of them must be off to achieve the pilot intent), the autothrottle continued to provide idle thrust (by remaining in THR IDLE mode). The subsequent alpha-protection activation (providing Go Around thrust) ran short of time to fully
Figure 9. Bangalore Mode Confusion Accident (Actual aircraft motion is shown in black dotted layout filled in red color, while aircraft on continuous layout filled in grey color denotes pilot intened aircraft position along the vertical dimension)

kick-in, crashing the aircraft.

This is a complex mode confusion accident involving the interplay of both autopilot and autothrottle (i.e., automation) modes and associated confusions. Our mode confusion detection framework works seamlessly for this case too as discussed below.

1. Automation Model

The different flight modes relevant in this Bangalore accident are the autopilot and autothrottle modes (collectively called the automation modes). The relevant vertical modes of the autopilot are V/S mode and OP DES mode, while the relevant speed modes of the autothrottle are SPD mode and THR IDLE mode. In OP DES autopilot mode, the autothrottle runs in THR IDLE mode providing idle engine power, resulting in loss of airspeed. However, the SPD mode of the autothrottle holds the engine thrust to maintain an appropriate airspeed. The default autothrottle mode in V/S autopilot mode is SPD mode. Figure 10 depicts the relevant automation-logic with a hybrid system model representing relevant flight modes and their respective continuous dynamics. The discrete mode transitions are either autonomous or pilot triggered upon satisfaction of a guard condition. For example, in Figure 10 there is an autonomous flight mode transition from V/S + SPD mode to OP DES + THR IDLE mode due to the mentioned guard condition (i.e., current altitude higher than the missed-approach altitude and current airspeed greater than or equal to the maximum allowable airspeed).

The hybrid system model of the automation has a discrete-time linear dynamical model for each flight mode with altitude $h$, airspeed $v$, altitude change rate $\dot{h}$ and airspeed rate $\dot{v}$ as the continuous states, while the flight modes V/S + SPD and OP DES + THR IDLE are the discrete states. Using the estimated continuous state and flight mode of the automation along with the automation’s and pilot’s discrete and (estimated) continuous input information, an automation intent model is constructed. The automation intent map is shown in Table 3.

Table 3 shows both the autopilot (V/S and OP DES) and autothrottle (SPD and THR IDLE) flight modes with their respective intents. The purpose of SPD mode is to maintain a set speed on the FCU. If the speed is set higher than the current airspeed, the engine gives extra power to capture the target airspeed and maintain it. As before, the ambiguity in uniquely determining the automation’s intent (e.g., OP mode has two intents (either climb or descent)) can be resolved by either considering the annunciated FMA sub-mode information (e.g., OP CLB implies climb intent and OP DES implies descent intent) or by considering the actual continuous state information (i.e., if the altitude change rate is positive, it denotes climb intent).
2. Pilot Model

The pilot intent model for Bangalore accident is constructed similarly using the continuous states and discrete modes of the automation along with the pilot’s input information. The pilot intent map is shown in Table 4.

From Table 4, as before, a single flight mode (e.g., OP mode) corresponds to two pilot’s intents (either climb or descent intent). This ambiguity can be resolved by considering the pilot’s input information. For example, suppose that the autopilot is switched on and the current intent of the pilot is a decelerating descent (e.g., OP DES + THR IDLE). Now the pilot switches off the FD in the FCU, expecting the mode to transition to V/S + SPD mode. This indicates that the pilot’s next intent for the aircraft is to regain the engine power and perform a climb maneuver by accelerating to achieve the desired final approach speed. The intent transitions of the automation and pilot are put together to cause the divergence of the automation and pilot intents in Figure 11. It is clearly observable that the mismatches in the pilot’s expected behavior of the automation and actual aircraft’s behavior are captured by using (tactical) intent transitions.

3. Mode Confusion Detection

The mode confusion in the Bangalore accident is detected by comparing the inferred intents of the automation and pilot. Figure 12 shows the intended altitude trajectories by the autopilot and pilot and the intended airspeed trajectories by the autothrottle and pilot along with the actual flight modes of the autopilot and
<table>
<thead>
<tr>
<th>UI Displayed Mode</th>
<th>Pilot Intent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vertical Speed mode</td>
<td>Climb/Descent</td>
</tr>
<tr>
<td>OP mode</td>
<td>Climb/Descent</td>
</tr>
<tr>
<td>SPD mode</td>
<td>Constant Speed</td>
</tr>
<tr>
<td>THR IDLE mode</td>
<td>Decelerate</td>
</tr>
</tbody>
</table>

Table 4. Pilot Intent Map

Figure 11. Divergence of Inferred Automation and Pilot Intents alongside Actual Aircraft’s Behavior and Pilot’s Expected Behavior of the Automation along the Vertical Dimension (Actual aircraft is shown in black dotted layout filled in red color, while continuous layout aircraft filled in grey color denotes pilot intented aircraft position)

4. The Results of User Interface Validation

Careful observation of the Airbus-320 Bangalore accident reveals the followings as important causes for the mode confusion:

1. Autonomous mode transition from V/S + SPD to OP DES + THR IDLE mode.
2. Only one of the 2 Flight Directors were switched-off. This kept the autothrottle in THR IDLE mode instead of the pilot intended transition to SPD mode.
3. Even though all the critical flight modes were displayed on UI (FMA), when mode confusion occurred,
the pilot was surprised and confused as to what caused the issue. It seems the pilot was unaware as to which information on UI he/she had to pay more attention to.

4. Additionally, the guard conditions responsible for autonomous mode transitions seem to be unknown to the pilot.

Thus, the analysis suggests that, in this autopilot and autothrottle involved accident, upon detection of mode confusion, it is highly important that the critical information such as the modes involved in the autonomous mode transition (e.g., V/S + SPD mode) and the associated guard condition (e.g., one FD off only) are highlighted in the context-sensitive UI through feedback. This should help improve the pilot’s situational awareness, thus increasing safety.

IV. Conclusions and Future Work

This paper attempts to demonstrate on two illustrative mode confusion incidents/accidents (e.g., speed protection autopilot mode confusion incident and Bangalore autothrottle and autopilot mode confusion accident) that the proposed algorithm can be used to effectively detect mode confusion and validate the associated User Interface (UI) (also called as flight deck instruments) through an intent-based mode confusion detection framework. Our approach has developed modular, yet interacting models for automation (as a hybrid system) and pilot (as a discrete-event system) for a given UI (as a filter of flight information). Then, the mode confusion detection is performed by comparing the automation’s inferred intent with pilot’s inferred intent. The validation of the UI has been done by identifying specific UI design issues such as: either critical modes and guard conditions involved in autonomous mode transitions were not displayed, or even though all the critical flight modes were displayed on the UI (e.g., Flight Mode Annunciator (FMA)), when mode confusion occurred, the pilot couldn’t discern which information on the UI he/she had to pay more attention to and thus was surprised. This can happen when the pilot’s intent does not match with that
of the automation, and the UI which is expected to act as a buffer to compensate for the missing critical information from the pilot model, is poorly designed. Thus, we recommend that upon detection of mode confusion, an on-board pilot-alert system activates on the UI providing missing critical information (e.g., associated modes and guards) through feedback to create real-time situational awareness to the pilot and thus for aircraft safety. This is possible through context-sensitive UI design. In the future, we will attempt to extend the framework to address mode confusions involving more complex (e.g., capture an airway, go to airport) and time-varying aircraft (e.g., faster descent, slower right turn) behaviors through appropriate construction of intent sets.

V. Acknowledgement

This work is supported by the 2012 NASA SBIR & TT Topic A1-03.

References


Das, S., Matthews, B. L., Srivastava, A. N., and Oza, N. C. “Multiple kernel learning for heterogeneous anomaly detection: algorithm and aviation safety case study,” In the Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2010.


