

SUPPORTING PILOT SITUATIONAL AWARENESS BY EXTRACTING INSIGHTS FROM AN AI EXPERT

Cynthia Koopman[1] & David Zammit Mangion[1,2]

[1]University of Malta, Msida MSD 2080, Malta

Abstract

Loss of control in commercial aviation occurs due to various factors even within the safe flight envelope, where the safety-critical risk worsens in case of non-linear areas of the flight envelope or when experiencing failures. The underlying cause of various loss of control events can be summarized by a human-automation gap due to the increase in automation systems in the cockpit, where the mental model of pilots does not correspond to the state of the aircraft systems. This paper builds on the integration of an AI expert, trained with reinforcement learning, that has an understanding of the aircraft environment, to aid pilots in loss of control. This paper explores how knowledge can be extracted from the AI system to enhance pilot situational awareness. Three distinct areas are addressed to support situational awareness of the flight crew with a focus on stall recovery scenarios: to enhance awareness of failures and dependent systems, to provide pilots with more control over recovery maneuvers, and to enhance awareness on the dynamic safe flight envelope by estimating safe states over time. These methods were implemented by either adopting changes to the neural network architecture or to the reward function. The results show potential for the proposed methods and indicate that AI solutions can be modified in different ways to take advantage of the knowledge that is developed during the training of the AI system.

Keywords: AI, Safety, Situational awareness, Upset recovery, human-centered

1. Introduction

Pilots in commercial aviation can experience loss of control due to various factors involving both their situational awareness and the state of automation systems. Although failures in automation systems and non-linear areas of the flight envelope exacerbate the risk of loss of control, this can become critical even inside the safe flight envelope and even when the aircraft is airworthy. The majority of loss of control conditions may therefore be summarized as a result of a human-automation gap, created by the rising complexity in the cockpit due to the increasing introduction of automation systems. This becomes evident in, for example, the Air France 447 accident [1] where confusion among the pilots contributed to an unrecoverable stall scenario. But also in the more recent accidents involving the Boeing 737 MAX [2, 3], where two accidents resulted from automation failure in combination with pilots lacking the required knowledge (human-automation gap) to respond in a correct and timely manner.

Progress in mitigating loss of control events has been challenging largely because prior work typically accounts for a broad range of contributing factors for loss of control [4] and, as a result, tends to address only a limited subset of these factors within one system. With such an approach, many, often separate, systems address the various factors leading to a potential loss of control event and create the need for advanced methods for system integration to mitigate potential undetected vulnerabilities (which can increase the human-automation gap) [5, 6, 7]. Such an subsystems approach also leaves combinations of, for example, sensor failures unresolved.

Addressing the human-automation gap in the cockpit of commercial aircraft leads to a more holistic insight which can have significant effect on the mitigation of loss of control for pilots, by improving the situational awareness of the pilots and therefore narrowing the gap. In such an approach, the pilot would remain in authority, but be equipped with additional insights regarding aspects that the pilot is unable to fully comprehend, whether that is due to the complexity or stressful circumstances or both. The provision of such an additional knowledge has previously been proposed by the authors to be addressed by an AI expert [8, 9, 10], which would be capable of understanding the complex environment created by the interaction of the systems on the aircraft and their response in non-linear areas of the flight envelope or when faced with failures. It is the task of the AI expert to comprehend the complexity inherent in the cockpit environment and to guide human pilots by providing critical information and insights. It would have the potential to allow pilots to perform their responsibilities in a more informed manner while being able to make use of the advantages of the automation systems on board, without unrealistically requiring them to have the knowledge of the designers of these automation systems in case of emergency. The previously trained AI expert [8, 9, 10] is adopted in this paper and explored further.

Given the safety-critical nature of this application, reliability is central in the development of the AI expert. Therefore, the previously trained AI expert [8, 9, 10] was designed with the aim of developing a deep understanding of the environment such that its response becomes reliable even in unforeseen conditions. For this, the knowledge about the environment was created by having the AI expert control the aircraft using Reinforcement Learning (RL), where the AI expert developed relationships between the information from the systems and the response of the aircraft [10]. The approach resulted in such a deep understanding of this aircraft environment that the AI expert showed reliability by being able to perform stall recovery without being explicitly trained for such conditions.

In this paper, we investigate how this internal knowledge of the previously trained AI expert system can be extracted and repurposed towards pilot-oriented support functions rather than only providing optimal control actions. Three distinct areas are addressed to support situational awareness of the flight crew with a focus on stall recovery scenarios: to enhance awareness of failures and dependent systems, to provide pilots with more control over recovery maneuvers, and to enhance awareness on the dynamic safe flight envelope by estimating safe states over time.

2. Implementation of the AI Expert System

The fundamental implementation of the AI system discussed in this paper follows the previous implementation of the AI expert system by the authors [8, 9, 10]. A brief explanation follows below.

The AI system was trained in the JSBsim [11] simulation software using the standard Airbus A320 commercial aircraft model with the aim of developing a deep comprehensive knowledge of the environment of the aircraft, which includes the controls, the surroundings and the systems on board of the aircraft.

The AI expert system was trained using RL (hereafter also referred to as RL agent), which allows learning via interaction with the aircraft. With this approach, the AI agent must develop relationships between the information received from the systems on the aircraft and the response of the aircraft, resulting in a comprehensive understanding of the environment.

The control task during training was defined by a combination of an initial altitude and airspeed with a target altitude and airspeed, which the AI agent was required to optimize using elevator and thrust control. This definition of the control task results in the RL agent learning from flying towards simple combinations of altitude and airspeed targets, while being abstract enough to use in testing by, for example, placing the RL agent in an upset condition and requiring it to fly towards defined target altitude and airspeed.

During training, the combinations of initial and target states were randomly sampled spanning a large area of the flight envelope: 15,000 to 30,000 ft initial altitude, 248 to 456 knots (420 to 770 ft/s) initial true airspeed (TAS), -1,000 to +1,000 target change in altitude, -100 to +100 ft/s target change in calibrated airspeed (CAS). The true airspeed for initial states was used due to the nature of initialization functions inherent to JSBsim, whereas the target was implemented using calibrated airspeed to reflect the cockpit environment.

The input variables to the RL agent were a subset of the entire state such that it only included variables that are measured on the aircraft during flight. This included parameters relating to aircraft altitude, speed and attitude; ambient atmospheric conditions; aircraft control inputs and target parameters.

The RL agent was trained using Proximal Policy Optimization [12] implemented by an actor-critic architecture, where the actor network (the policy) provides the control actions and optimizes using the estimates made by the critic network on how advantageous it is to be in a particular state. The actor and critic networks were implemented using neural networks with an identical architecture: a Long-Short-term memory (LSTM) layer followed by two fully connected layers each of 256 neurons. Parameters were not shared between actor and critic, allowing both networks to fully optimize towards their respective tasks; and in case of data failures, for the critic to observe the state without these data failures.

The optimization of the RL agent is based on a reward function, which quantifies the progression towards the altitude and airspeed targets. The reward function was defined as a weighted sum over three reward components according to Eq. 1:

$$R = \frac{1}{3}\sum w_1 r_{target} + w_2 r_{limits} + w_3 r_{actions} \tag{1}$$

where the three reward components are weighted by w_1, w_2, and w_3. The individual reward components are a target reward r_{target}, a reward for adhering to limits r_{limits}, and a reward to stabilize action $r_{actions}$.

The target reward r_{target} is presented in Eq. 2:

$$r_{target} = sigmoid(r_{\Delta_{altitude}}) + sigmoid(r_{\Delta_{CAS}}), \tag{2}$$

where $sigmoid(x) = \frac{1}{1+e^{-x}}$. The sigmoid function was used as it allows for large reward changes when far from the target which leads to a faster convergence and requires the RL agent to be more precise when close to the target.

A small reward was also included to discourage large rates of change of the outputs (large changes between actions in the previous and current step), in the form of Eq 3:

$$r_{actions} = r_{\Delta_{elevator}} + r_{\Delta_{thrust}}, \tag{3}$$

where $r_{\Delta_{elevator}}$ and $r_{\Delta_{thrust}}$ are the negative of the action errors between two time steps.

Remaining within the limits of vertical speed, vertical acceleration, and angle of attack was encouraged by including the reward component r_{limits}, which follows Eq 4:

$$r_{limits} = (1 - r_{\dot{h}}) + (1 - r_{a_y}) + (1 - r_{aoa}), \tag{4}$$

where each bracketed group is equal to 1 when within defined limits and 0 otherwise.

As mentioned earlier, the training of the RL agent entailed flying tasks towards simple combinations of target altitude and airspeed. During this training, it was previously shown [10] by the authors of this paper that the RL agent was able to generalize and thus gain a deep understanding of the environment such that it was able to perform stall recovery in a dynamic manner (for various altitudes, aircraft weights and positions of center of gravity). This stall recovery response is the main focus of the next sections, where modifications to the described AI agent are centered around this response.

Stall scenarios were created by trimming the aircraft near its stall speed and subsequently reducing thrust while increasing pitch to induce the stall. The RL agent was then activated and tasked to fly the aircraft towards defined altitude and airspeed targets once the angle of attack exceeded the critical angle of attack in combination with a decreasing vertical speed.

Although the RL agent was trained to provide optimal control actions to fly towards the altitude and airspeed targets, it will have gained useful knowledge about the environment in the process. The next sections discuss three methods of how this knowledge can be extracted and used to create functions towards supporting the situational awareness of pilots.

3. Supporting Pilot Situational Awareness of Failures and System Reliability

Certain failures in automation systems, such as data errors, can cause confusion for pilots, especially when it becomes unclear what effect such failures may have on other systems. Such confusion or misunderstanding can therefore lead to, for example, incorrect pilot action or automation surprise, both of which indicate a currently incorrect understanding of the aircraft environment by the pilots.

An RL agent could be explicitly trained to perform stall recovery maneuvers even in the presence of data errors. Whereas the approach adopted by the authors in this work, where the RL agent implicitly learns how to recover from the stall, allows it to have deeper learning. In the context of data errors, deeper learning leads the RL agent to learn which data are erroneous and this would be advantageous and of value to the pilots.

During such training with failures, the RL agent would internally develop relationships to understand not only which sensor readings remain reliable, but may also learn to triangulate the correct sensor readings to provide optimal control actions. This knowledge about the identification and reconstruction of data errors could be highly valuable to pilots and would therefore be of interest to extract.

In the current architecture of the RL agent, this information is inaccessible and can only be observed by the ability of being able to provide optimal control whilst experiencing data errors. To access this knowledge gained by the RL agent, adaptations were made to its neural network architecture, namely, additional output units were added, one for each input unit. These additional output units represent the correct input data. This configuration forms an additional task for the RL agent, where in addition to the optimal control, it has to output the correct input data, which can be seen as a denoising autoencoder that is regularized by the optimal control task.

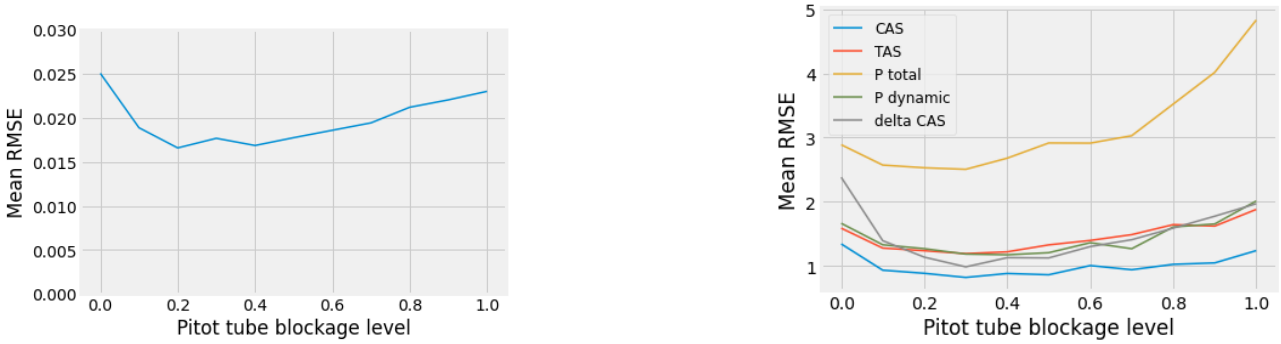
The addition of this task is unlikely to distract the RL agent from its primary task of outputting the optimal control actions. This is because previously the RL agent exhibited skills of triangulating and possibly reconstructing correct data, such skills will now be reinforced by the additional task. The effect on the internal representation of the RL agent is thus that triangulation and reconstruction of correct data are learned more explicitly and carried further into the last neural network layers (as it is an output). This can therefore also have a positive effect on the optimal control outputs, as forcing the RL agent to learn these skills provides a helpful direction for optimization.

The RL agent was trained using the adapted neural network architecture, where the additional output units that provide the reconstructions of the faulty sensor readings can easily be optimized during training using the correct readings available in simulation (via a mean-squared-error loss). Two different types of data errors were tested: pitot tube blockages and incorrect control readings. The pitot tube blockages affect multiple data inputs for airspeeds and pressure measurements and thus require the RL agent to rely on data that remain correct and the response of the aircraft itself. For the incorrect control readings, the current position of the elevator control and thrust setting were manipulated. This removes reference points for the current position of the actuators for which the RL agent must learn to optimize using the response of the aircraft over time.

3.1 Results

Figure 1 shows the average performance of the RL agent reconstruction task while experiencing different severity of pitot static blockages, where the results are shown using the root-mean-squared-error (RMSE) for all in Fig. 1a and unnormalized RMSE for selected variables in Fig. 1b. These blockages affect several sensor readings such as airspeeds and pressure readings. Overall, the RMSE remains low, where the calibrated airspeed has a mean RMSE below 1.5 ft/s and true airspeed below 2 ft/s. These low errors show that the RL agent can accurately reconstruct the data errors even when all related data is lost with a fully blocked pitot tube.

An example in stall recovery is shown in Fig. 2, where the RL agent was activated in a stall at 30,000 ft and required to regain initial altitude and increase airspeed. In this example, the pitot tube was blocked at 250 episode steps until the end of the episode. The severity of the blockage was randomized until episode step 750 and kept at a constant offset thereafter. The RL agent was able to successfully recover from the stall and approach target altitude and airspeed. The figure shows the data as it is seen by the RL agent (denoted by 'input' in the legend), the reconstruction output of the RL agent (denoted by 'reconstructed') and the actual data from the simulation (denoted by



(a) All variables normalized.

(b) Selected variables unnormalized.

Figure 1 – Reconstruction error shown by mean RMSE over 100 randomly sampled episodes for different severity in pitot tube blockage, where 0 is a fully blocked pitot tube and 1 is no blockage. Results shown for all variables normalized and selected variables unnormalized.

'real'). The results for the affected variables shown (the calibrated airspeed, true airspeed, dynamic pressure, total pressure and change in calibrated airspeed) all show precise reconstruction of the correct input data. Such precise results are observed despite the blockage being uncorrelated with the real input (between episode steps 250 and 750), which shows that the RL agent is capable of using the remaining reliable data as well as the response of the aircraft to accurately reconstruct the correct data and provide optimal control actions. The precise reconstruction during the constant offset that follows after 750 steps further shows that the RL agent is not overfitting and is able to correctly identify failures and reconstruct the data.

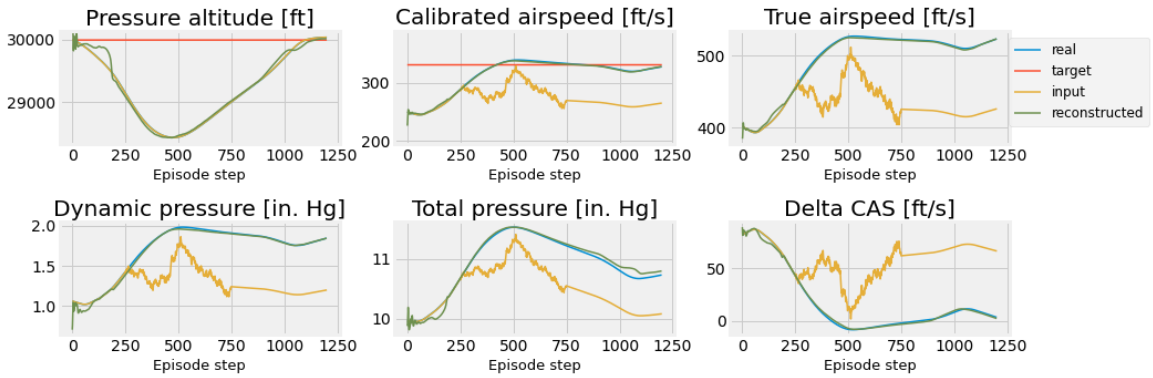


Figure 2 – Example of a stall recovery performed by the RL agent while experiencing a pitot tube blockage starting at 250 episode steps.

Figure 3 shows a stall recovery example at 25,000 ft for the RL agent trained to reconstruct the current position of elevator control and thrust setting. In this example, the current position of the elevator and thrust controls are shown, together with the angle of attack and vertical speed. The RL agent successfully recovers from the stall by reducing the angle of attack and subsequently approaches the target altitude and airspeed in a controlled manner, which can be observed by the stable descent and climb with a comfortable vertical speed below an advised limit of 33 ft/s. The current position of both the elevator and thrust control was manipulated by a constant offset during the entire episode, with a magnitude that previously left the RL agent (trained without such failures) unable to recover. The figure shows that the RL agent was able to reconstruct the elevator position and thrust setting with adequate accuracy, where the thrust setting reconstruction exhibited the larger error of the two.
These results showed that the adaptation to the neural network architecture presents potential for extracting useful information about failures from the knowledge gained by the RL agent. The pitot tube blockages and the control position errors required the RL agent to focus on different aspects of the environment, where the pitot tube blockages required more focus on the triangulation and the

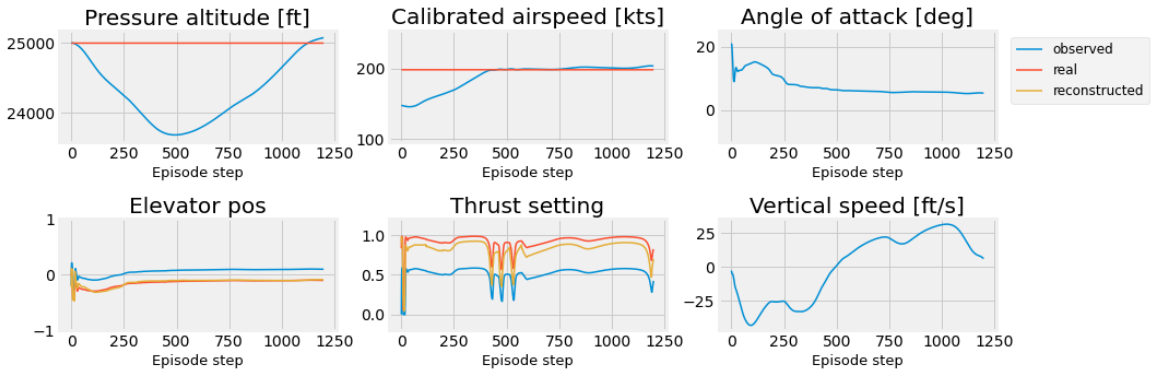


Figure 3 – Example of a stall recovery performed by the RL agent while experiencing manipulations to the data reflecting the current control positions.

manipulations of the control position required more focus on the aircraft response over time. The results indicate that the proposed architecture is appropriate for both such data errors. The additional insights which result from this neural network adaptation can be used as a basis to provide pilots with both detection of failures and estimates of failure reconstruction in addition to the existing optimal control commands. This can result in an improvement in situational awareness as pilots would have the ability to not only understand which values are no longer correct but also which other systems are no longer reliable due to their interdependency.

4. Supporting Pilots by Improving Control over Automatic Stall Recovery

The actions required to recover from a stall can seem contradictory as it requires a loss of altitude to decrease the angle of attack and increase airspeed before increasing pitch to mitigate a secondary stall. The ability of the RL agent to perform stall recovery despite not being trained explicitly was found to be a result of the ability to generalize, which was developed by training in a large training space with a relatively sparsely defined reward function [10]. As the RL agent was trained to approach the target altitude and airspeed, its stall recovery is actually optimized with respect to the given targets. This approach follows naturally from the training setup but in a context such as stall recovery, becomes significantly different to the sequential approach adopted by human pilots, where stall is typically recovered before considering the next task. It is therefore of interest to observe how the RL agent changes its stall recovery to approach different altitude and airspeed targets.

An example that illustrates such different approaches by the same RL agent is shown in Fig. 4. Two stall recovery maneuvers are shown, where the response in red was required to climb an additional 1,000 ft compared to the response shown in blue. In order to climb within the given time frame, it is shown in the figure that the RL agent adopts a greater and steeper loss of altitude.

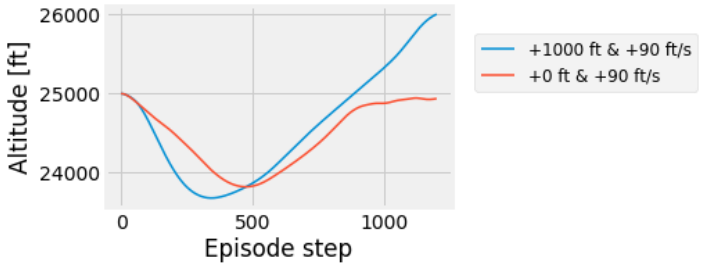


Figure 4 – Example of a vertical profile showing different climb strategies of a single RL agent during stall recovery. The target airspeed and altitude settings are shown in the legend, which are relative to the initialization values.

The ability to adapt the stall recovery strategy indicates not only that the RL agent indeed takes the targets into account while recovering, but also that the RL agent has gained knowledge to distinguish between stall recovery strategies. The responses shown in Fig. 4 therefore suggest that the RL agent

developed knowledge about different climb strategies.
For pilots, this knowledge about the use of different climb strategies in combination with stall recovery can be useful when using such automatic recovery techniques, and this is therefore further investigated in this section. Accordingly, the RL agent was required to perform the same stall recovery in different ways, providing pilots with more control over the RL strategy. For example, stall recovery while having a mountain below or in front of the aircraft requires different strategies, where one minimizes altitude loss while the other maximizes vertical gain.
To extract this knowledge, the RL agent was required to learn multiple ways to approach the altitude and airspeed targets by making changes in the reward function and informing the RL agent which strategy should be optimized for as 'modes' in the reward function. Changes in the reward function were made with the aim towards distinguishing best climb gradient and best rate of climb. Best climb gradient is needed when obstacles need to be cleared whilst best rate of climb is useful to acquire the target altitude at the earliest possible when no obstacle exists. Several different approaches with a focus on the stall recovery response are discussed below.

4.1 Results

Currently, the RL agent switches between the best climb gradient and the best rate of climb based on the given targets. To gain control over these strategies, the RL agent should optimize the trade-off between them, and this trade-off should therefore not be explicitly defined in the reward function. Instead, a more indirect approach is of interest.
In the current implementation, the RL agent is defined by the altitude and airspeed targets, which does not allow for a fully optimized airspeed as this is according to the given target. Therefore, approaches were tested where less importance was put on reaching the airspeed target. Results of two approaches that aim to incorporate reward based on horizontal distance and vertical gain are briefly discussed below, which is followed by an even more indirect reward approach for which more detailed examples are shown.

4.1.1 Reward approaches with horizontal distance and vertical gain

Figures 5 show an example of stall recovery in which the RL agent was trained with an additional mode, where the airspeed target was reduced in importance (shown in red). This was implemented by significantly reducing the weight on the reward component for the airspeed target. The figures show that the RL agent has learned two distinct strategies, where the airspeed target in Fig. 5b is strictly adhered to for the blue response and significantly less for the response shown in red, which corresponds to the lower weight on this target. The response in altitude in Fig. 5a shows that by allowing the airspeed to change (e.i. less reward is gained) a faster climbing strategy was found. However, these results do not show that the RL agent actually optimized for the best climb gradient or the best rate of climb, as for the example shown, the altitude response shown in red reaches the target altitude with less time and horizontal distance. In addition, the response in airspeed does not suggest that the RL agent has found the optimal airspeed as a more optimal speed would have been higher and requiring a larger loss in altitude, but instead the RL agent chooses to gain reward rather by minimizing altitude loss. Therefore, this interaction does not suggest control over the desired climbing strategies.
Another approach was tested with the aim to emphasize the distinct climbing strategies in addition to the previous approach. An RL agent was trained with 3 different modes: (1) optimizing for both airspeed and altitude targets together; (2) optimizing only for the altitude target achieving minimum possible time; and (3) optimizing only for the altitude target achieving minimal possible horizontal distance. These last two modes do not define any reference airspeed with the aim of giving the RL agent the freedom to optimize the airspeed without being penalized. The mode for optimizing over both targets was included to not lose functionality and to force the RL agent to learn to control airspeed as the other 2 modes do not have this incentive.
The response in altitude and airspeed is shown in an example of a stall recovery in Fig. 6, showing the three different modes. The altitude response in Fig. 6a shows three different approaches towards the target altitude, each with a larger increase in altitude loss and an increase in horizontal distance, indicating a successful distinction between modes. However, the airspeed response in Fig. 6b shows



(a) (b)

Figure 5 – Vertical profile for an example of a stall recovery by an RL agent for optimizing both targets equally (denoted by '0' shown in blue) and for optimizing with less weight in the airspeed target (denoted by '1' shown in red). Vertical profile is shown both against horizontal distance and time.

a significantly low speed for each mode, suggesting suboptimal convergence. This may be due to a lack of exploration, where the RL agents never observes how increasing airspeed results in significantly more reward and therefore attempts to create different responses within the experience it has seen. The similarity between modes in airspeed, also suggests that the RL agent is influenced by the different modes in an undesired manner. Such behavioral uncertainties are undesirable for the current aim of enhancing control over the RL strategy.



(a) (b)

Figure 6 – Vertical profile for an example of a stall recovery by an RL agent for optimizing 3 modes: both targets equally (denoted by 'h&v' in blue), only altitude target with respect to horizontal distance (denoted by 'Vx' in red) and only altitude target with respect to time (denoted by 'Vy' in yellow). Vertical profile is shown both against horizontal distance and time.

The results of both approaches show that an reward approach where best climb gradient and best climb angle are implemented by minimizing the horizontal distance or minimizing the time over reaching an altitude target and thus by not defining an airspeed target, do not lead to optimization of the airspeed. Moreover, this approach entails additional limitations such as the absence of the airspeed target for the RL agent as the functionality of being able to control the RL agent via both altitude and airspeed targets is lost. Furthermore, due to an unavoidable time constraint, the RL agent struggled to converge to solutions with shorter horizontal distances, as due to interactions with reward components, a shorter amount of time resulted in a shorter horizontal distance.

4.1.2 Reward based on target priorities

The most effective method for regulating the climbing strategies of the RL agent was found to be the modification of the reward function, assigning higher priority to achieving the prescribed altitude or airspeed target. Given the observation that a secondary target influences the primary target of the RL agent, the reward for the secondary target was given only if the primary target was reached. For example, the distance to the airspeed target was only rewarded if the RL agent reached the target altitude. This implementation allows the RL agent to optimize the airspeed freely as long as it is close to the target airspeed when the altitude target is reached, which provides both freedom for

airspeed optimization to reach the altitude and facilitates exploration. Having these 'priorities' on the targets results not in a response that is fully optimized for best climb gradient or best rate of climb but instead takes the multi-objective optimization into account preserving the functionality of controlling the RL agent with both targets and training with a time limit. Moreover, leading not to exact best climb gradient or best rate of climb but offering functionalities more appropriate to gain control over the RL agent.
This reward function was implemented by adapting the target reward r_{target} previously shown in Eq. 2, where conditions were added to the existing target rewards. The changes in target reward are summarized in Eq. 5 and Eq. 6:

$$r_{target_v} = (m_v * sigmoid(r_{\Delta_{altitude}})) + (2 * sigmoid(r_{\Delta_{CAS}})), \tag{5}$$

$$r_{target_h} = (2 * sigmoid(r_{\Delta_{altitude}})) + (m_h * sigmoid(r_{\Delta_{CAS}})), \tag{6}$$

where r_{target_v} and r_{target_h} are the airspeed and altitude priority target reward components, and m_v and m_h are binary variables indicating if the aircraft is close to the first target priority and engages the reward on the secondary target priority.
Figures 7 and 8 show examples of different stall recovery responses that can be achieved with this RL agent. Figure 7 compares stall recovery responses in which the RL agent prioritizes the altitude target for increasing magnitudes in climb relative to the initial altitude; 0 ft, +500ft and +1,000ft. The three recoveries show distinct strategies while all minimize altitude loss. The distinction of strategies is visible most prominently in altitude, airspeed, angle of attack and elevator control. A comparison in the altitude loss shows that the RL agent adapts its strategy to accommodate the increase in the climb of 1,000 ft (shown in yellow) by losing more altitude to gain more airspeed.
Figure 8 compares stall recovery responses in which the RL agent prioritizes the airspeed target for increasing magnitudes in increase of airspeed relative to the initial airspeed; 0 ft/s, +35ft/s and +70ft/s. This again shows different recovery strategies except that instead of minimizing altitude loss (as shown in Fig. 7) these strategies optimize the target airspeed and approach the target altitude by increasing the altitude loss. The distinction of strategies is visible in all variables shown. An inadvertent effect of prioritizing the airspeed target can be observed where the RL agent reaches the target altitude earlier despite losing significantly more altitude (>70 seconds and 1,000 ft comparing the blue and yellow in Fig. 8).

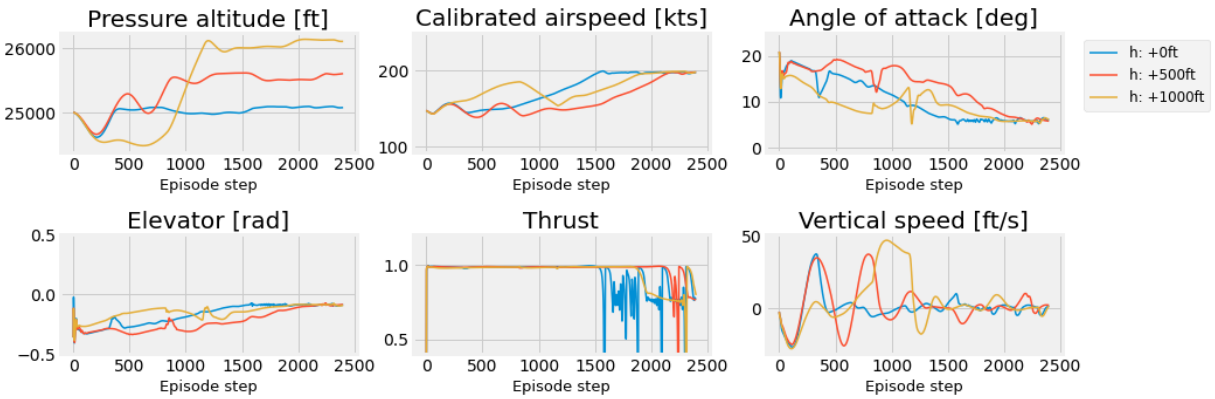


Figure 7 – Example of a stall recovery performed by the RL agent showing the response in altitude priority (denoted by 'h') for three different target settings described in the legend relative to the initial values.

The stall recovery strategies previously shown in Figs. 7 and 8 show that a single RL agent can perform stall recovery with varying climb strategies, six of which were shown; three in altitude target priority mode and three in airspeed target priority mode. More importantly, one can have control over the RL strategy by selecting a priority mode and appropriate altitude and airspeed targets. In either mode, the RL agent optimizes the recovery for a given target (e.g. an increase in altitude loss despite prioritizing altitude) which provides confidence in the RL agent. This approach enhanced the

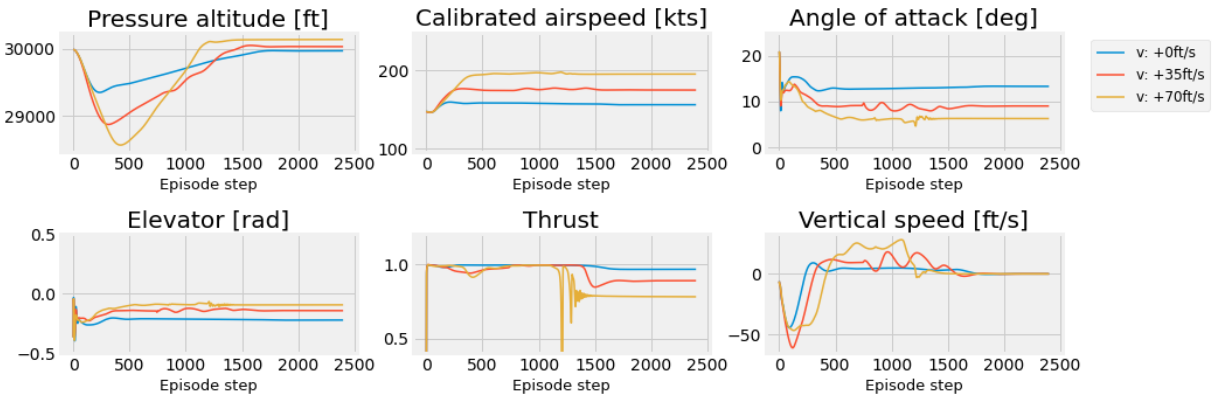


Figure 8 – Example of a stall recovery performed by the RL agent showing the response in airspeed priority (denoted by 'v') for three different target settings described in the legend relative to the initial values.

initial observation of the RL agent having gained knowledge on different climbing strategies and in a manner in which one can have control over such strategies. This capability adds significantly to the previous optimal control actions and can be used towards improving automatic upset recovery. Moreover, the capability of providing automatic stall recovery while having control over the recovery strategy can provide pilots with more authority over the recovery flight path, allowing them to be more in the loop thereby enhancing human-AI collaboration.

5. Supporting Pilot Situational Awareness through Anticipation of Safe Aircraft States

The high performance of the RL agent, where it is able to perform stall recovery dynamically via training on simple flying tasks, suggests internal development of knowledge regarding the long term effect of its actions. More precisely, the RL agent achieves high-level airspeed and altitude targets while only receiving current state information, which requires planning and understanding of the maneuverable areas of the flight envelope. This knowledge, developed by the RL agent during training, would be of interest to extract and could provide insight into the planning and safety of a proposed RL recovery. For pilots, it would be highly valuable to have authority over the recovery strategy and insights into a maneuver proposed by an AI system. Not only can this enhance the trust into the AI system, but it has potential to provide a functionality to understand what is possible from the current state of aircraft. For example, as the AI expert has an understanding of the environment and what the optimal control actions would be for a given pair of altitude and airspeed targets, it can be used to establish what can be achieved from a given state (or point in time). In the stall recovery context, a pilot may be interested in how quickly the aircraft will be recovered or how much altitude loss could be anticipated, given the current state of the aircraft (and the proposed altitude and airspeed targets). The RL agent has previously demonstrated to adapt its stall recovery strategy significantly, for example, in the previous section where the climb strategy was adapted from the beginning of the episode. Therefore, this section investigates a framework towards extracting future predictions of states in real-time and presents initial experimentation using the stall recovery context.

Initial experimentation was performed by using the angle of attack, where the RL agent has to provide future prediction on when the angle of attack would be below 15 degrees. These future predictions were implemented by probabilities of the angle of attack being below 15 degrees at increasing steps into the future. For this, the neural network architecture, specifically the output layer, was changed by the addition of output neurons representing the probabilities in the future. To clarify, Fig. 9 illustrates the neural network adaptation on the left, where the current state and targets remain the input to the RL agent, followed by the hidden layers, and only an addition is made of multiple output units. These output units, representing probabilities, each output values between zero and one indicating if at the future time step the angle of attack will be below 15 degrees. For example, the first output provides the prediction for the current timestep (zero steps into the future), and the 10th output unit provides the prediction for 10 time steps into the future. In total, 120 output units were added each

representing 1 second more into the future (the episode length previously used is 1200 steps at 10 Hz). These multiple output units can be combined, as illustrated in the graph on the right of Fig. 9, forming a prediction over time. These 120 output units are optimized during training by using the captured values available in the simulation after the episode ended, which is illustrated by the blue graph in Fig. 9, where the 120 output units would approximate this and resemble a probability curve as illustrated by the red graph in Fig. 9. The RL agent was trained with the adapted neural network architecture optimizing for prediction of angle of attack being below 15 degrees up to a 120 seconds into the future.

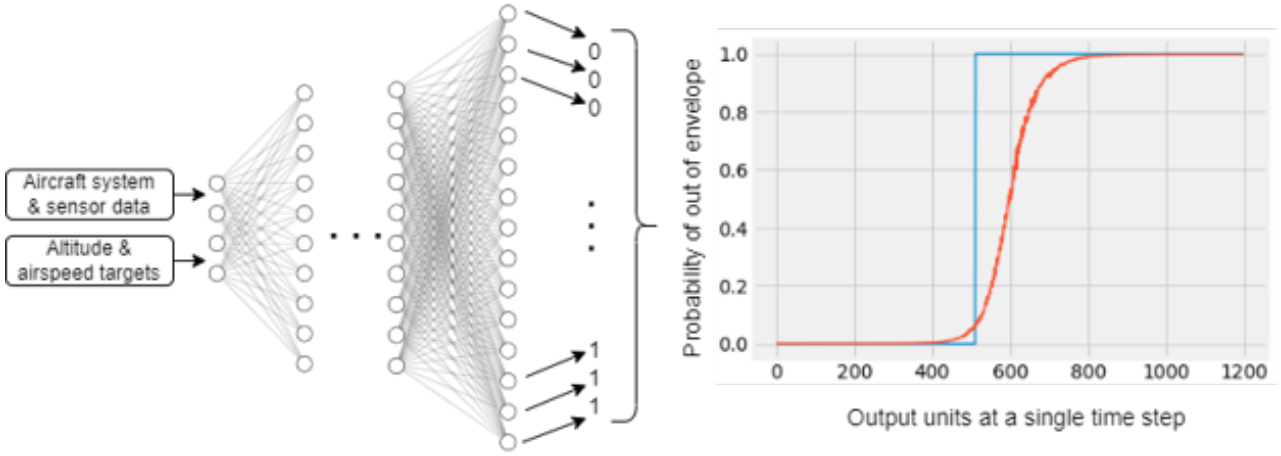


Figure 9 – Neural network architecture showing multiple output units (left) and an example of the consolidation of the output values (red) against their label (blue) showing the process of obtaining probabilities over a time horizon (right).

5.1 Results

Figures 10 and 11 show an example of stall recovery and the corresponding future predictions, respectively. This stall recovery example was chosen specifically to illustrate the prediction framework as the recovery of the RL agent does not immediately reduce the angle of attack, where Fig. 10 only provides a reference to the corresponding stall recovery for this example. The output of all neural network output units is shown in Fig. 11a for selected steps in the episode (the output units are on the x-axis). This illustrates how the probabilities from 8 seconds after the episode started onward are always equal to one as the angle of attack is predicted to remain below 15 degrees from this point forward, whereas the output in the first four seconds is not always equal to one. Figure 11c shows where the angle of attack was relative to 15 degrees over time, indicating that the angle of attack remained below 15 degrees after approximately 10 seconds. Figure 11b shows the probabilities outputted by all 120 neural network outputs at the start of the episode in blue (this is the same graph as previously shown in blue in Fig. 11a), together with the reference of where the angle of attack was actually below 15 degrees shown in red. This prediction (shown in blue), shows that the RL agent correctly approximates the future for the angle of attack; the prediction starts to increase from zero exactly where the angle of attack decreases below 15 degrees. This initial increase in probability could be, for example, taken as the predicted time to lower the angle of attack. It could, however, also be used to say that, for example, the angle of attack will be below 15 degrees in 20 seconds with a probability of 80 percent or in 25 seconds with a probability of 99 percent.

Another example is shown in Fig. 12, which illustrates a different stall recovery. The corresponding angle of attack is shown in Fig. 12c. For this example, the prediction at the first initial time step of the RL agent, shown in Fig. 12b is different to the previous example, where the probability is already larger than zero for a shorter prediction horizon, reflecting the oscillations below the 15 degrees of angle of attack. Therefore, in this example, according to the RL agent, the probability of the angle of attack being below 15 degrees is at 40 percent for one second into the future, and at, for example, 80 percent at 10 seconds (following from Fig. 12b).

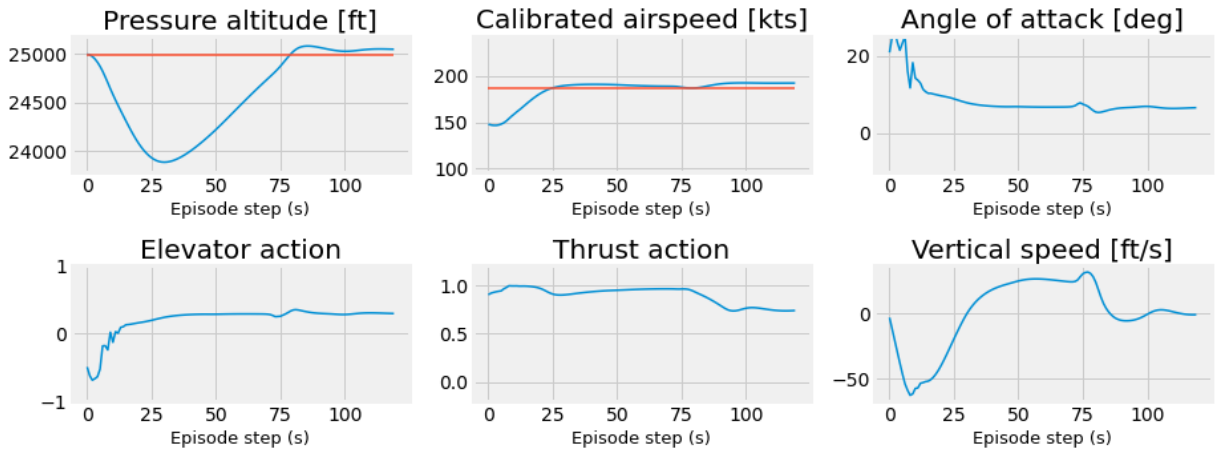
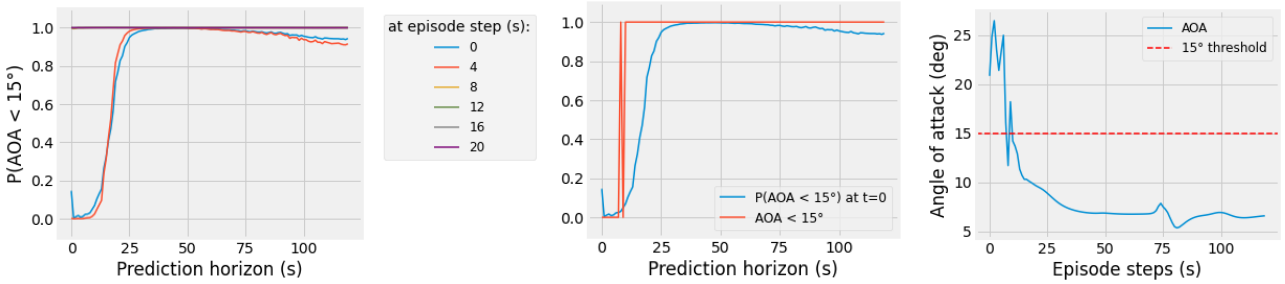
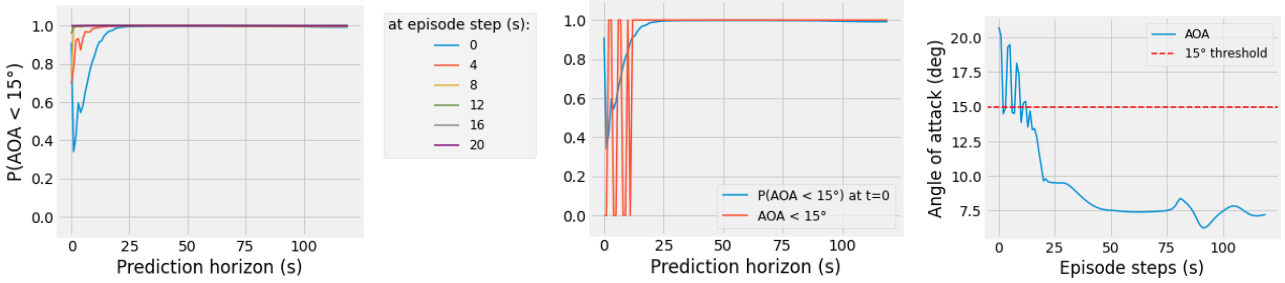


Figure 10 – Example of a stall recovery by the RL agent, corresponding to the data shown in Fig. 11.



(a) (b) (c)

Figure 11 – Probabilities of having an angle of attack below 15 degrees over time for a stall recovery example (Fig. 10): (a) Output of all output units at various episode steps.(b) Output of all output units at the first episode step (blue) and its ground truth (red). (c) Angle of attack during the episode shown with the 15 degrees mark.



(a) (b) (c)

Figure 12 – Probabilities of having an angle of attack below 15 degrees over time for a stall recovery example: (a) Output of all output units at various episode steps.(b) Output of all output units at the first episode step (blue) and its ground truth (red). (c) Angle of attack during the episode shown with the 15 degrees mark.

The implementation in this section serves only as an initial experimentation with a novel framework to extract the planning of the RL agent and therefore only looked at a short future time horizon. Given that in stall recovery the reduction of angle of attack is of high importance, the RL agent has successfully learned to do this in a short amount of time and therefore the future predictions beyond 20 seconds could not be showcased. Nevertheless, the RL agent approximated the future time predictions with adequate accuracy, showing value in experimentation on examples that may showcase a larger prediction horizon. The proposed framework offers not only time estimates but also probabilities attached to them which aims to provide transparency. This may have potential to improve the situational awareness of pilots, as it allows them to conclude if their actions (or the actions of the RL agent) would result in risk that should be avoided, but it can also allow for a way of anticipating the strategy of control of the RL agent.

6. Conclusion

This paper presented three distinct methods which provide additional pilot centered functionalities by extracting knowledge from the same baseline RL agent, focused on stall recovery. These methods either made changes to the neural network architecture or the reward function. The first method added the reconstruction of data errors to the optimal control actions, which can be useful for failure detection as well as estimation of correct data. The second method presented an implementation to gain control over different climb strategies of the RL agent during stall recovery, which provides pilots with more control over the RL strategy in terms of altitude loss and airspeed gain. The third method investigated a framework to provide estimates into the future of the RL response, which can be used to anticipate certain behavior such as the time required to decrease angle of attack. The results shown in this paper demonstrate potential for the proposed methods and indicate that AI solutions can be modified in very different ways to take advantage of the knowledge that is developed by the AI system. All of the presented methods work towards solutions that mitigate the human-automation gap by addressing the situational awareness of pilots and encourage a human-AI collaboration where the pilot remains in authority and the AI expert system provides advise and crucial insights into the complex cockpit environment. For future work, it would be of interest to investigate further methods to create new functionalities and new insights as well as to improve the maturity of these solutions.

7. Contact Author Email Address

Cynthia Koopman can be contacted at the following email address: cynthia.koopman@um.edu.mt

8. Copyright Statement

The authors confirm that they, and/or their company or organization, hold copyright on all of the original material included in this paper. The authors also confirm that they have obtained permission, from the copyright holder of any third party material included in this paper, to publish it as part of their paper. The authors confirm that they give permission, or have obtained permission from the copyright holder of this paper, for the publication and distribution of this paper as part of the ICAS proceedings or as individual off-prints from the proceedings.

References

[1] Anonymous. Final Report On the accident on 1st June 2009 to the Airbus A330-203 registered F-GZCP operated by Air France flight AF 447 Rio de Janeiro - Paris. Bureau d'Enquêtes et d'Analyses pour la sécurité de l'aviation civile, July 2012.

[2] Anonymous. Interim Investigation Report on Accident to the B737-8 (MAX) Registered ET-AVJ operated by Ethiopian Airlines. The Federal Democratic Republic of Ethiopia Ministry of Transport Aircraft Accident Investigation Bureau, March 2020.

[3] Anonymous. Final Aircraft Accident Investigation Report PT. Lion Mentari Airlines Boeing 737-8 (MAX). Komite Nasional Keselamatan Transportasi Republic of Indonesia, October 2019.

[4] Christine M. Belcastro. Aircraft loss-of-control: Problem analysis and research toward a holistic solution. Technical Report NASA/TM-2011-217158, NASA Langley Research Center, 2011.

[5] Christine Belcastro and Steven Jacobson. Future integrated systems concept for preventing aircraft loss-of-control accidents. In *AIAA Guidance, Navigation, and Control Conference*, 2010.

[6] Ming-Jui Yu, Kevin McDonough, Dennis S. Bernstein, and Ilya Kolmanovsky. Retrospective cost model refinement for aircraft fault signature detection. In *2014 American Control Conference*, pages 2486–2491, 2014.

[7] Murray Kerr. Reconfiguration of control in flight for integral global upset recovery (reconfigure): Final publishable summary report. Technical report, Deimos Space Soceidad Limitada Unipersonal, 2016.

[8] Cynthia Koopman and David Zammit-Mangion. Towards safe aircraft control with ai: Reinforcement learning with guardian angels and mischievous gremlins. In *AIAA AVIATION FORUM AND ASCEND 2024*, 2024.

[9] Cynthia Koopman and David Zammit-Mangion. Enhanced safety through ai: An assessment of adversarial reinforcement learning for control of complex aircraft. In *34th Congress of the International Council of the Aeronautical Sciences*, 2024.

[10] Cynthia Koopman and David Zammit-Mangion. Understanding training components in shaping ai for loss of control recovery in unforeseen flight conditions. In *AIAA AVIATION FORUM AND ASCEND 2025*, page 3337, 2025.

[11] Jon Berndt. Jsbsim: An open source flight dynamics model in c++. In *AIAA Modeling and Simulation Technologies Conference and Exhibit*, page 4923, 2004.

[12] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.