Statistical Analysis and Runtime Monitoring for an AI-based Autonomous Centerline Tracking System

Yuning He[1], Johann Schumann[2]

[1] *NASA Ames Research Center*
yuning.he@nasa.gov
[2] *KBR/Wyle, NASA ARC*
johann.m.schumann@nasa.gov

ABSTRACT

Autonomous Centerline Tracking (ACT) enables an uninhabited aircraft system (UAS) to be guided down the center of the runway, using a camera-based Deep Neural Network (DNN). ACT is safety-critical. Guidelines by the European Union Aviation Safety Agency (EASA) for machine-learning based systems list numerous assurance objectives that must be met toward Verification and Validation (V&V), and certification. We extend our analysis framework SYSAI (System Analysis using Statistical AI) to support meeting assurance objectives for a system with AI/ML (Artificial Intelligence / Machine Learning) components and describe a combination with a runtime monitoring architecture that also supports advanced risk mitigation to support safety assurance of a complex AI-based aerospace system.

1. INTRODUCTION

In recent years, applications of Machine Learning (ML) and, in particular Deep Neural Networks (DNNs) have demonstrated a performance that can substantially increase operational capabilities of autonomous Unmanned Aerial Systems (UASs). For example, DNNs can be used to visually detect obstacles on the runway, to identify runways during approach and landing, or to support taxi of a UAS at an airport.

All these applications have in common that they are safety-critical. This means that failures can lead to mission failure, cause damage to the UAS or on the ground, or even lead to injuries or loss of human life. Therefore the system must perform safely and with good performance in a multitude of nominal and off-nominal situations.

Because of the safety-criticality, such systems must be carefully designed and analyzed and certification is necessary to demonstrate safety and performance of such a system. However, current AC safety standards (e.g., DO-178C) only applies to "traditional" flight software. Approaches that are based upon Machine Learning are not covered and techniques for V&V are still under development.

The European Union Aviation Safety Agency (EASA) published guidelines on how to approach V&V and certification of autonomous systems with ML-based components; (European Aviation Safety Agency, 2021) is focusing on Level 1 systems. As defined by the SAE (Society of Automotive Engineers (SAE), 2021), Level 1 is the lowest of 6 levels of autonomy and concerns "driving assistance" systems. The EASA guidelines breaks down the certification process into numerous subtasks and objectives and provides thoughts and initial guidelines on how to meet these objectives. The document also contains a detailed use case, a Neural Network (NN) based visual landing guidance system.

The EASA document shows numerous certification objectives, which span system design, design of DNN and learning algorithm, acquisition and management of training/test data, testing, and deployment. Many of these certification objects are not independent of each other; therefore they should not be studied in isolation. In this paper, we investigate, how our framework and tool SYSAI (System Analysis using Statistical AI, (He & Schumann, 2020)) can be used to perform in-depth statistical safety- and performance analysis in a high-dimensional space spanned by system and environmental parameters.

SYSAI can model regions of safe and high performance, and can characterize boundaries between these regions. In this paper, we will discuss, how SYSAI can be used and extended to enable unified safety and performance analyses on the individual AI component and the entire system for multiple environmental and failure scenarios. Such analysis can be performed for several DNN variants and can provide feedback to the system designer. It also can provide essential information for Run Time Assurance and Performance Monitors. We will present a runtime monitoring architecture, which uses the in-

https://doi.org/10.36001/IJPHM.2024.v15i3.3860

Table 1. Relevant certification objectives (see Case Study in (European Aviation Safety Agency, 2021), p. 77)

Obj	Description
CO-02	The applicant should define the AI-based (sub)system taking into account domain-specific definitions of 'system'.
CO-04	The applicant (APPL) should perform a functional analysis of the (sub)system.
CO-03	APPL should define and document the ConOps for all AI-based (sub)systems.
CL-01	APPL should classify the AI-based (sub)system, based on the levels presented in the EASA AI typology and definitions.
SA-01	APPL should define metrics to evaluate the AI/ML component performance and reliability.
SA-02	APPL should perform a system safety assessment for all AI-based (sub)systems.
DM-03	To enable the data collection step, APPL should identify explicitly and record the input space and the operating parameters that drive the selection of the training, validation and test data sets.
DM-04	Once data sources are collected, APPL should make sure that the data set is correctly annotated or labelled.
DM-10	APPL should ensure V&V of the data all along the data management process so that the DQRs are addressed.
LM-01	APPL should describe the AI/ML components and model architecture.
LM-03	APPL should document the credit taken from the training environment and qualify the environment accordingly.
LM-04	APPL should provide quantifiable generalization guarantees.
LM-05	APPL should document the result of the model training.
LM-06	APPL should document any model optimization that may affect the model behavior (e.g. pruning, quantization) and assess their impact on the model behavior or performance.
LM-07	APPL should estimate bias and variance ... should provide evidence of the reproducibility of the training process.
LM-08	APPL should ensure ... meet the associated learning process management requirements.
LM-09	APPL should perform an evaluation of the performance of the trained model based on the test data set and document the result of the model verification.
SRM-01	... APPL should determine whether the coverage of the objectives associated with the explainability and learning assurance building blocks is sufficient or if ... safety risk mitigation (SRM), would be necessary ...
SRM-02	APPL should establish SRM means as identified in Objective SRM-01.

formation produced by SYSAI to dynamically select a well performing AI component based upon the current situation or to switch to an fall-back component to sustain safety in case the AI components cannot perform adequately.

We will illustrate, how our smart SYSAI framework can perform these tasks and thus support performance/safety evaluation, performance improvement, and revalidation.

The rest of the paper is structured as follows: in Section 2 we discuss the range of certification objectives set up by the EASA guidelines. Section 3 provides an overview of the Autonomous Centerline Tracking system. Section 4 presents SYSAI and its extension toward multi-objective analysis and the synergistic integration with runtime monitoring. Section 5 discusses related work, and Section 6 summarizes and concludes.

This paper is an extended version of (He & Schumann, 2023), which was presented at the PHMAP 2023 conference.

2. BACKGROUND: AUTONOMY CERTIFICATION

Autonomous capabilities for aircraft are usually safety critical: a malfunction can lead to loss of the UAS and the payload, it can even endanger human life in other aircraft or on the ground. Therefore, safety of operations must be established by certification. A first usable guide paper on autonomy certification and verification has recently been published by the EASA (European Aviation Safety Agency, 2021). For the important trustworthiness of the autonomous system, numerous verification objectives are set up that must be met. Our approach has been inspired by the EASA Guide Paper, Appendix F (European Aviation Safety Agency, 2021; EASA & Daedalean, 2021), where a relevant AI-based aerospace system, a vision-based landing system, is analyzed as a case study.

Table 1 lists the main objectives, which span five groups, ranging from ConOps (CO-*), safety assessment (SA-*), data collection and management (DM-*), learning management (LM-*), and safety risk mitigation (SRM-*). Obviously objectives in all categories interact with each other. For a complete certification, all objectives need to be met.

3. AUTONOMOUS CENTERLINE TRACKING

As a case study, we use the ACT (Autonomous Center Line Tracking) system, which enables autonomous taxiing, one of the most important ground operations for Unmanned Aerial Systems. The core component of ACT is a Deep Neural Network (DNN) that takes images as inputs from cameras mounted on the aircraft's starboard wing (Figure 1). The DNN component is running onboard the aircraft and continuously estimates the position and orientation of the aircraft with respect to the runway center line. These values are the cross-track error cte in meters, and the heading error he in degrees, respectively.

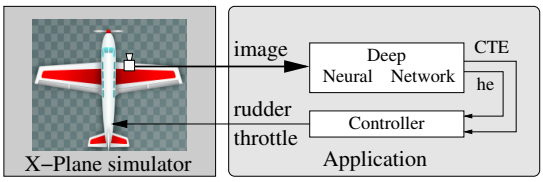

Figure 1. Autonomous Centerline Tracking (ACT) system

A simple fixed-gain controller uses this information to produce control signals to steer the aircraft left and right, while the aircraft is rolling at a constant, low speed. For our experiments, the X-Plane Flight Simulator (`www.xplane.com`) was used as simulation environment. For this case study, we were provided with a finalized prototype implementation and fully trained DNNs.

4. SYSAI ANALYSIS AND RUNTIME ASSURANCE

In this paper, we present our extended AI-component and system analysis with SYSAI. It can be used to address most of the objectives in Table 1. The analysis can provide feedback to the designers, and generate information to be used by our extended Runtime Assurance Architecture (RTA, Section 4.3) to enforce safety and risk mitigation throughout operations while ensuring adequate performance.

4.1. The Smart Analysis Framework SYSAI

SYSAI (System Analysis using Statistical AI) (He & Schumann, 2020) is a flexible statistical learning framework for V&V and the analysis of complex and high-dimensional cyber-physical systems with AI components. Figure 2 shows the high-level architecture of SYSAI analysis framework. On the left-hand side, we have the "system under test" (SuT), which in our case is the ACT system and the XPlane simulator, as described in the previous section. The SuT is executed given a set of parameters provided by the statistical learning model of SYSAI. The result of the test run is then used to incrementally construct the statistical model.

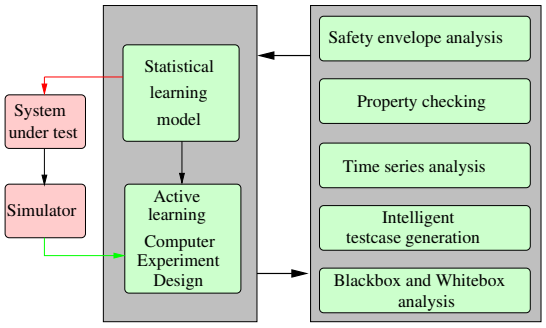

Figure 2. SYSAI architecture

For the representation and construction of the statistical model, SYSAI uses Dynamic Regression Trees (DynaTrees (Taddy, Gramacy, & Polson, 2011; Gramacy & Polson, 2011)), a dynamic Gaussian process model based upon Particle Filters. DynaTrees are regression and classification learning models with complicated response surfaces in on-line application settings. DynaTrees create a sequential tree model whose state changes over time with the accumulation of new data, and provide particle learning algorithms that allow for the efficient on-line posterior filtering of tree-states. A major advantage of DynaTrees is that they allow for the use of simple models within each partition. The models also facilitate a natural division in sequential particle-based inference: tree dynamics are defined through a few potential changes that are local to each newly arrived observation, while global uncertainty is captured by the ensemble of particles.

This surrogate model is initialized with available training data and incrementally refined using candidate data points that are produced by our active learning module. It evaluates the current surrogate model using a customized active-learning heuristics and suggests candidate data points that provide most information for model refinement. For these candidate points, the ground truth is obtained by executing the SuT.

SYSAI features customizable heuristics that allow the active learning to focus on particular characteristics of the model. Classical algorithms like ALM (MacKay, 1992) or ALC (Cohn, 1996) focus on under-explored regions in general of the domain space. Inspired by (Jones, Schonlau, & Welch, 1998) and work on contour finding algorithms, we loosely follow (Ranjan, Bingham, & Michailidis, 2008) and define our boundary-aware metric boundary-EI (He, 2015, 2012) that puts the focus of the search into "interesting" and potentially "troublesome" areas near safety boundaries. Here, the surrogate model therefore exhibits substantially more details than in other areas that are not of interest. This exploration is guided by the selected active learning heuristics and is able to cover the entire input space with a low number of data points. The SYSAI framework and the underlying models and algorithms are described in detail in (He & Schumann, 2020).

SYSAI, in general, supports the following important analysis tasks (Figure 2):

- *Safety-envelope analysis:* our framework can perform automatic analysis of the safety-envelope, which indicates under which operational conditions the system is behaving safely or not. Geometric shape modeling does not only identify but also characterizes regions with similar behavior and describes those regions in easy to understand geometrical terms. SYSAI thus helps to make the system more explainable.
- *Property checking:* our tool supports the automatic checking and analysis of safety and performance requirements.
- *Time series analysis:* SYSAI can perform advanced time-series analysis in a high-dimensional parameter and state space. This analysis provides a deeper understanding of the system behavior and its dynamics. The tool also supports event prediction.
- *Intelligent test-case generation:* SYSAI can efficiently generate relevant test cases in high-dimensional spaces.
- *Blackbox and Whitebox analysis:* with our SYSAI tool, system-wide and component-based analyses can be carried out. The SYSAI interface allows to access data in-

side of components, thus enabling Whitebox analysis.

4.2. Analysis with SYSAI

Traditional approaches for DNN performance analysis and improvement are usually restricted to individual characteristics of the neural network, e.g., architecture, learning parameters, or data sets. As Table 1 shows, however, V&V objectives span multiple dimensions, including concepts of operations, network architecture, learning, data sets, and risk mitigation. Both, the DNN as well as the entire system, which uses the DNN component have to be considered.

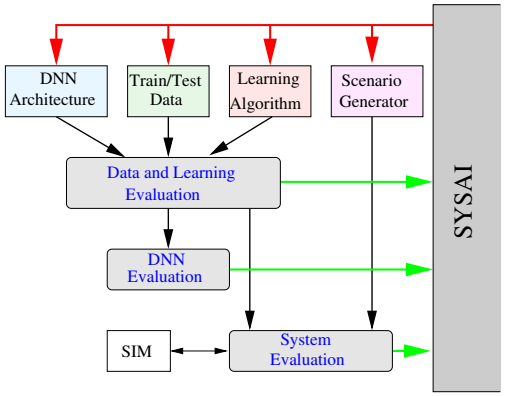

Figure 3. Extended SYSAI interface

To enable such a multi-faceted analysis, we have extended the interface from SYSAI to the system under test (Figure 3) to allow SYSAI not only to control the execution of the system with the AI component, but also to execute automatic experiments using different DNN architectures, learning algorithms, and scenarios. In principle, our architecture is capable of performing analyses, which can include automatic generation of data sets and DNN training. However, the execution times for such runs can be very long. In this paper, we therefore focus on the analysis of the ACT with two different trained DNNs, which have been provided by the designers. Even so, the execution of SYSAI runs can take substantial time because for each run, the full scenario of taxiing down the runway has to be simulated in real time, which takes about 3 minutes each.

After the run of SYSAI, customized plots are produced that can provide feedback to the designer regarding safety and performance. The resulting data are also used by our runtime assurance architecture, which will be discussed in Section 4.3.

Figure 4A shows the probability distribution of the actual DNN output cte versus the ground truth cte_{gt}. Two different DNNs, given to us were considered in this case. The ideal DNN should have sharp peaks along the diagonal (shown in red). Whereas DNN 1 has an overall good behavior but a small bias for negative values of cte, DNN 2 behaves better in that area but has substantial deviations for larger positive values. These results are based upon the DNN only analysis. Incorporated into the ACT system, both DNNs perform satisfactory, due to robustness of the controller. Here, SYSAI caused the execution of an entire run down the runway with a random initial cte.

Figure 4B shows the analysis of a training set that had been manually generated. Given different starting locations, the aircraft drives down the runway in a typical manner without violating any safety constraint. Such a training set obviously covers the space of nominal operations; however scenarios, where the AC enters in the middle of the runway, would not be covered by this data set. Here, the designers need to decide if the operational envelope is sufficiently covered, or additional training data sets need to be generated and analyzed, a task that SYSAI can perform.

4.3. Runtime Assurance Architecture

Many safety requirements of a complex system cannot be totally verified during design time. In order to overcome this gap, ASTM has developed and published the standard F-3269 (ASTM, 2017) for an assured Runtime Assurance Architecture (RTA). Its underlying principles of operations are as follows (Nagarajan, Kannan, Torens, Vukas, & Wilber, 2021): since the AI component cannot be V&V'ed or trusted, its output signals are considered to be "unassured" even if the inputs are assured. In order to prevent faulty AI outputs from propagating through the system, the behavior of the AI component is continuously monitored *during flight*. This is accomplished by the RTA monitor, a traditional piece of software certified separately. Therefore, an assured signal is available at all times judging if the AI component can be trusted or not. In case the AI component cannot be trusted, the RTA switch changes the signal routing from the unreliable AI component to an assured fallback component, which takes over system operations, albeit with some restrictions.

In previous work (He, Schumann, & Yu, 2022), we have instantiated the RTA to use a powerful temporal reasoning engine and use parameters and data provided by SYSAI to populate the temporal properties. In this paper, we describe an extension of our RTA, which does not only allow continuous safety checks, but can also use the monitors to mitigate loss of performance during run time. Figure 5 shows the extended architecture. Like the AI component of the ACT system, our RTA is executed on board the aircraft. The additional computational resources needed for runtime monitor and the RTA switch are minimal, compared to the execution of the DNN. The core of the runtime monitor is the R2U2 (Realizable,Responsive, Unobtrusive Unit) (Rozier & Schumann, 2017), which performs signal processing, evaluation of temporal logic formulas, and calculation of posterior probabilities for discrete Bayesian networks. If necessary, the R2U2

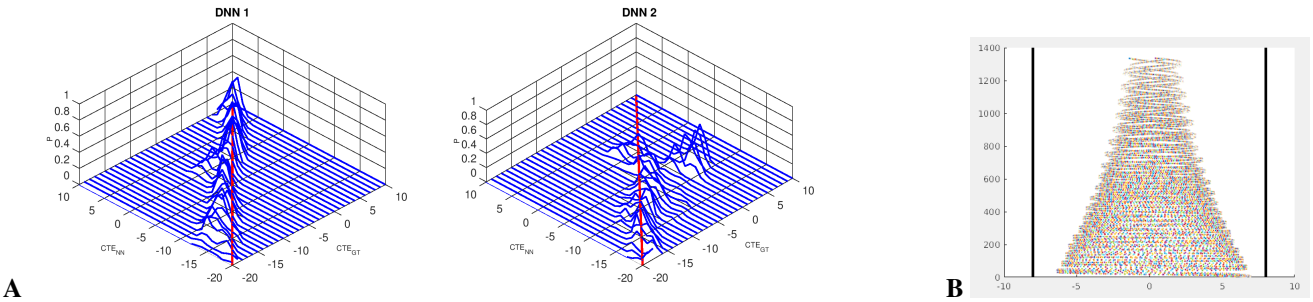

Figure 4. **A:** performance analysis of two different DNN architectures. **B:** Camera positions for training set on the runway. The runway is oriented vertical and the taxi trajectories start at the bottom of the graph.

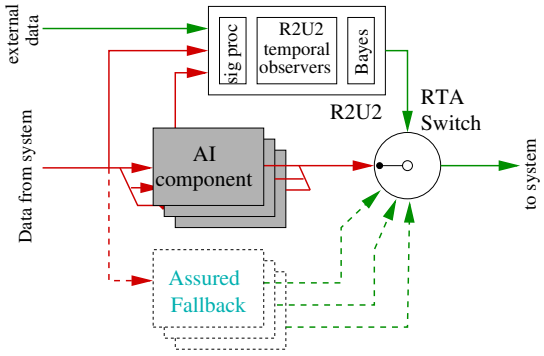

Figure 5. R2U2 runtime monitoring architecture (inspired by (Nagarajan et al., 2021), Fig. 1)

can be implemented as an FPGA configuration, again lowering the computational load of the flight computer.

In addition to the runtime monitor, the RTA switch, and the assured fallback components, the system can have *multiple AI components*, which are connected to the RTA switch. These are designed to perform the same function, but might have differences in performance for certain conditions, operational conditions, or failures. They also might have a different computational footprint.

In our ACT case study, we use two different DNNs. The SYSAI analysis reveals, which of the AI components perform better and more reliable in which region of the state space, under which failures and scenarios, etc. This spatial and temporal information is then encoded using R2U2 (temporal) formulas, which then control the RTA switch. Specifically, the switching is extended with respect to the original RTA: if the currently active AI component fails to meet safety or performance requirements, the R2U2 monitor will check if one of the other AI components can do the job. If this is possible, the RTA switch will activate that component. Note that in this case, the R2U2 provides the assured result that the activated component performs safely (although it is an unassured component). This switching between the two trained networks can be done with very little overhead.

In case, no AI component meet the necessary requirements, the R2U2 will switch to a fallback position to assure continued system safety, exactly as in the original F-3269 RTA.

The switch must be designed in such a way that (a) no big transients occur, which might damage the physical system, and (b) no repeated switching or bouncing between the different fallbacks can happen. For (a), our architecture supports a soft RTA switch, which allows phasing a phasing out period to avoid harsh transitions. Rules about switching are formulated in temporal logic and are checked and enforced by R2U2.

4.4. Monitoring Requirements

Based upon the relevant objectives shown in Table 1, we develop requirements that are checked by R2U2 during system operation. These requirements can be grouped into (1) safety requirements (for the entire ACT system), (2) system performance requirements, (3) performance requirements for the AI component (the ACT DNN), (4) environmental requirements and requirements based on system ConOps, and (5) failure mode based requirements. These requirements are formulated as a set of temporal logic formulas, which can be efficiently checked by R2U2. As described in (Reinbacher, Rozier, & Schumann, 2014; He et al., 2022), R2U2 uses future and past time observers for Linear Temporal Logic; for details of operation and the logics see these papers.

The relevant monitoring requirements use signals produced by the ACT system, the DNN, the NAV (navigation system), and the aircraft. Signal names and descriptions are shown in Table 2. Note that these signals are produced by the system itself and cannot contain ground truth values. In order to improve the quality of the R2U2, we include a navigational component (e.g., GPS). However, measurements of this component are not considered as verified. The signals in Table 2 are produced by the ACT system, the DNN, the NAV system, and the AC (e.g., the flight control system). Except for the DNN output, traditional non-ML software is used to calculate these values. They are read with a suitable rate (here: 1Hz).

Table 2. Signals used for R2U2

Signal	Source	Description
T	AC	local time
T^{act}	ACT	elapsed time
act	AC	ACT is active (Bool)
cte, he	DNN	DNN output
$\dot{cte}, \dot{he}$	DNN	derivatives of DNN output
cte^0, he^0	NAV	initial position on Rwy
$RwyID$	AC	runway ID
v	AC	AC velocity (front-wheel)
v^{NAV}	NAV	AC velocity down Rwy (from NAV)
s^{cam}	ACT	camera status (categorical)
b^{cam}	ACT	image brightness
f^{cam}	ACT	fuzziness of camera image
$o^{cam}, o_x^{cam}, o_y^{cam}$	ACT	obstructions in image
A_{DNNi}	ACT	DNN number i is active

The requirements are listed in Table 3. For a more compact representation we used:

- All requirements (except PS-02) are only valid while the ACT system is active, i.e., for a requirement R we get: $act \rightarrow R$.
- Short, infrequent dropouts of the requirements can be tolerated. Otherwise, too many false alarms occur. For a requirement R we require $\neg\Diamond_{I_d}\neg R$ holds,
- all thresholds for signal comparisons are shown as θ (despite the fact that they have different values),
- $SR(\theta)$ denotes a geometric internal representation of a shape as characterized by SYSAI. Typical shapes include rectangles, triangles, (hyper-)spheres or ellipsoids.

The requirements shown in Table 3 list the relevant requirements for checking by R2U2. Many requirements can be expressed by Boolean expressions, past time temporal logic operators are used to express (a) the immediate beginning of the center-line tracking (e.g., SS-02) or (b) some time thereafter (e.g., SS-03, SS-04). In each formula, the condition that the tracking is off (i.e., $\neg act$) is used as a trigger. Here, $\circ$ means the temporal previous operator. Other temporal formulas show a typical pattern: $\neg\Box_J\neg p$ for a Boolean formula p means that p cannot be false for a longer period than interval J. For example, SS-06 $\neg\Box_{10s}(cte > \theta \wedge he > 0)$ says, when the AC is already on the right side of the runway $cte > \theta$ and the AC is pointing to the right—away from the center line, then this situation should not occur for more than 10 consecutive seconds (the interval J), because the AC might leave the runway otherwise.

The safety-condition SS-08 is a result of a SYSAI analysis, where a safety region with respect to the initial cte^0, he^0 is characterized. For a given threshold θ, the red line in Figure 6 depicts to the safety boundary; all starting conditions outside this region can lead to ACT failure. R2U2 uses estimated shapes (e.g., an ellipse) for efficient checking.

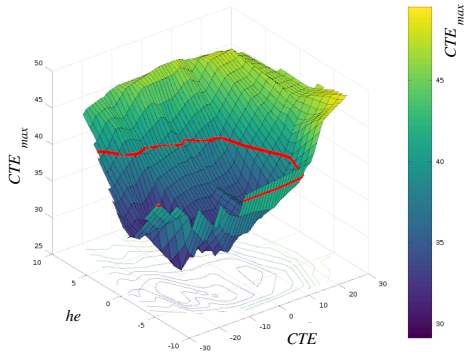

Figure 6. Safety-envelope: surface shows estimated maximal CTE value during a run over initial values cte^0 and he^0. The safety boundary at a given threshold of 40ft is shown as a red line.

Obviously, many of the requirements concern values that need to be checked against given thresholds in a high dimensional space.

The coverage of training data is an important issue affecting safety of the DNN-based system. E.g., SS-02 checks if the actual situation is consistent with training data (Figure 4B). If the AC is outside the area, on which the DNN has been trained, probability of ACT failure is higher. In this case, the DNN has not been trained well for larger values of cte further down the runway. If such a situation is detected (SS-02), it might lead to bad performance of ACT and could possibly lead the AC off the runway.

Several of the component performance conditions (PC) describe constraints on the dynamic behavior of the DNN itself. Typically, such conditions express that, while operating, the output of the DNN should change, but should not "jump around" wildly. R2U2 can also detect oscillations of the system (PS-03) using Fast Fourier Transforms of the relevant signals. That way, it can be avoided that the ACT produces vibrations (high frequency oscillations in the front-wheel steering) as well as low-frequency oscillations where the AC "tumbles" along the runway like drunk.

In most of the requirements, values for the thresholds θ and the intervals are based upon results of the SYSAI analysis. For example, PC-04 – PC-06 encode conditions on when a switch between the two DNNs should take place. These formulas are a direct result of the analysis shown in Figure 4A.

Obviously, many of the requirements concern values that need to be checked against given thresholds in a high dimensional space. Since SYSAI is capable of performing geometric estimation of boundaries, the runtime monitor is not restricted to hyperplane thresholds. Richer geometric shapes (including those based on circles, ellipses, parallelograms) enable fine-tuning of the runtime properties.

Table 3. Requirements for R2U2

ID	Formula	Description
SS-01	$RwyId = RwyId_{target}$	the AC shall be on the correct runway
SS-02	$\circ\neg act \rightarrow v = 0$	when ACT starts, the AC should be stationary
SS-03	$\Diamond_J \neg act \wedge v > \theta$	the AC should move with a measured minimum speed (measured at front wheel)
SS-04	$\Diamond_J \neg act \wedge v \leq \theta$	the AC should move with a measured minimum speed (measured at front wheel)
SS-05	$\circ\neg act \rightarrow \lvert cte - cte^0\rvert < \theta \wedge \lvert he - he^0\rvert < \theta$	when ACT starts, the AC should be close to the defined starting point on the runway
SS-06	$\neg\Box_J(cte > \theta \wedge he > 0)$	when AC is near the right border of the runway, the AC shall not move further to the right ($he > 0$) for an extended time
SS-07	$\neg\Box_J(cte < -\theta \wedge he < 0)$	symmetric to SS-06
SS-08	$(cte^0, he^0) \in SR^0(\theta)$	The starting point on the runway shall lie within the ACT safety region (Figure 6)
SC-01	$\lvert cte\rvert < \theta \wedge \lvert he\rvert < \theta$	the DNN outputs shall be reasonably limited
SC-02	coverage	the DNN should operate within trained domain (see Fig 4A)
PS-01	$v^{NAV} > \theta$	AC shall move with a minimum velocity down the runway
PS-02	$\circ act \wedge act \rightarrow T_{act} < \theta$	the AC shall reach the end of the runway (end of act) within a reasonable time
PS-03	¬Osc	the ACT system should not cause low frequency oscillations (tumbling down the runway) or high frequency oscillations (vibrations). To be measured using $FFT(cte)$
PC-01	$\neg\Box_J(\dot{cte} = 0 \wedge \dot{he} = 0)$	the DNN outputs should not be stationary for a longer time
PC-02	$\lvert\dot{cte}\rvert < \theta \wedge \lvert\dot{he}\rvert < \theta$	the DNN outputs should not "jump around"
PC-03	$\neg\Box_J\lvert he\rvert > \theta$	the AC should be well aligned to the runway; longer times with larger he should be avoided
PC-04	$cte > \theta \rightarrow A_{DNN1}$	when $cte > \theta$, ACT should use DNN1 for most of the time
PC-05	$cte < 0 \rightarrow A_{DNN2}$	when $cte < 0$, ACT should use DNN2 for most of the time
PC-06	$\neg(A_{DNN2} \wedge cte < 0 \wedge \Box_J\dot{cte} > 0)$	when $cte < 0$ with DNN1 active, but cte is growing, ACT shall switch over to DNN1
ES-01	$T > 0900 \wedge T < 1430$	ACT shall only operate between 9AM and 2:30PM local
ES-02	$RwyId \in \{\text{allowable-Runway-IDs}\}$	ACT shall only operate on "allowable" runways
ES-03	$f^{cam} < \theta$	ACT shall only operate if the vision is not to bad (foggy)
FS-01	$\neg\Box_J\neg s^{cam}$	the camera shall be operational for most of the time
FS-02	$\theta_{low} \leq b^{cam} \leq \theta_{high}$	the brightness of the the camera image shall be always within a suitable range
FS-03	$f^{cam} \leq \theta$	the fuzziness of the the camera image shall be always limited
FS-04	$\neg\Box_J\neg O^{cam} \in SR(\theta)$	a camera obstruction (no signal in certain area of image sensor, Figure 9) shall not show up for an extended time
FS-05	$\circ A_{DNN1} \wedge O^{cam} \in SR(\theta) \rightarrow A_{DNN2}$	when DNN1 was active and an obstruction shows up in certain areas (Figure 10), DNN2 shall be activated
RTA-01	$\neg(\Box_J S_{rta}(i,j) \vee \Box_J S_{rta}(j,i))$	a switch between component/fallback i and j must not happen too often

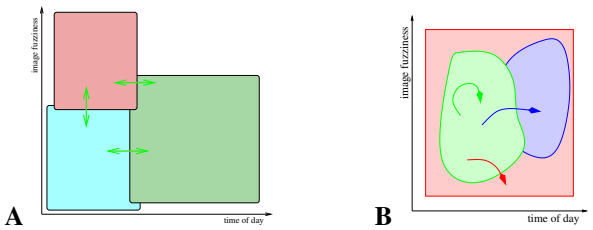

Figure 7. Performance areas for different components wrt. two variables

Figure 7A shows a 2D projection of a simple scenario: we have 3 different AI components available that are performing particularly well in different regions of the space, spanned by the parameter of image fuzziness, and time of the day (which governs light conditions). These regions are abstracted as colored rectangles.

Requirements then can be formulated to specifically enforce or forbid switching between the different AI components, in our example, e.g., PC-04 – PC-06 or FS-06.

Environmental conditions, like lighting or visibility can substantially influence the behavior of ACT. With SYSAI, we were analyzing the performance at different times of the day. Figure 8A shows that the success rate of ACT strongly differs: whereas the performance is satisfying during the morning hours, the performance increases during noon hours. However, after 2PM local time, the performance of ACT drops.

The reason for this behavior becomes clear that lighting of the scene is substantially different during different times of the day and the DNN, which had been analyzed had only been trained with data obtained at 9AM (Figure 8). Images taken earlier in the morning are darker (A) and thus reduce DNN performance. During afternoon hours, however, the AC casts a shadow on the runway (Figure 8C), which obviously confuses the DNN and ACT effectively stops working. Our R2U2 uses ES-01 (derived from SYSAI analysis) to avoid such situations.

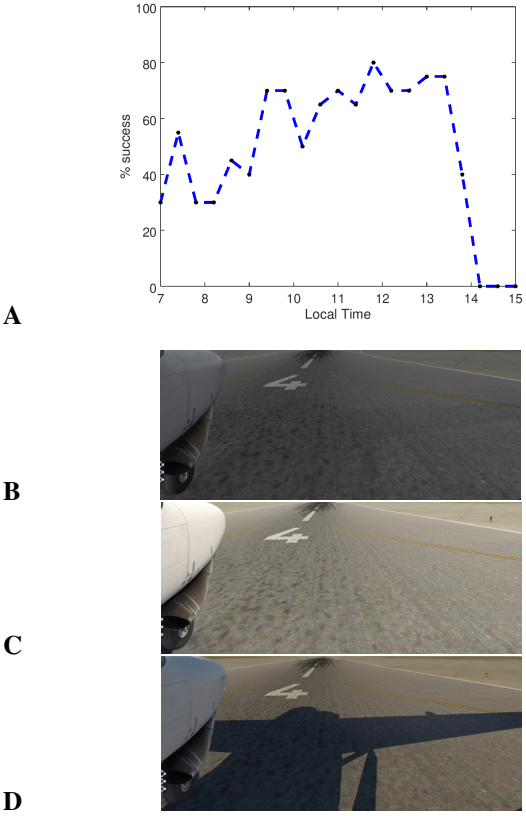

Figure 8. A: Success rate (in %) for different times of the day. Threshold for CTE is 40ft. Camera images taken from runs at 8AM (B), 11AM (C), and 3PM (D).

In a real application, ACT has to be able to safely operate under a host of possible failure cases. We have analyzed a set of typical, camera-related failure cases with SYSAI: camera images being too dark or too bright, foggy camera lenses, or dirt specks on the camera lens blocking part of the image. Figure 9 shows a typical example when ACT has to operate under failures. In our case, a part of the camera image is blocked by a piece of dirt on the camera lens. Depending on the (x,y) position of this dirt patch, its influence on the overall performance can vary substantially. Figure 10A shows the system behavior for one DNN 1. Darker colors mean better performance. Although the overall performance is very good, there are some critical locations, where the dirt pro-

Figure 9. ACT camera image with "dirt" spot (black rectangle) and superimposed boundary lines for high risk regions

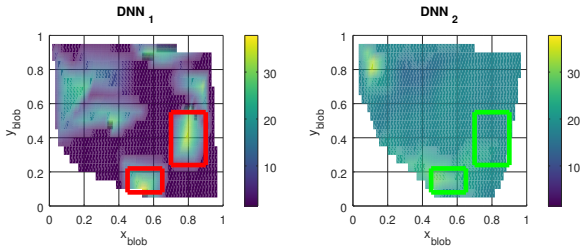

Figure 10. Performance of ACT under dirt on the camera lens.

hibits successful operation (bright yellow areas). The analysis of a different DNN 2 reveals that, albeit poorer overall performance, DNN 2 is not that sensitive to dirt patches (Figure 10B). Therefore the RTA will switch from DNN1 to DNN2 when a dirt patch is detected in areas bounded in red.

Finally, the RTA monitors its own operations: E.g., RTA-01 constrains how often a switch between the AI component and a specific fallback can happen and also controls switching back. Such requirements are necessary to avoid bouncing between different components and fallbacks.

In general, these requirements are produced in accordance with objectives SRM-01 and SRM-02 in Table 1. A detailed safety risk analysis provides the basis for the system and component safety properties. Performance and environmental properties are derived from system and design requirements, and requirements concerning failures (FS-*) can be taken from the results of a Fault Tree analysis (FTA) and an FMEA (Failure Mode and Effects Analysis).

5. RELATED WORK

Several AI-based aerospace systems have been manually analyzed for safety using the EASA Guidelines (European Aviation Safety Agency, 2021), e.g., (EASA & Daedalean, 2021; FAA, 2021). For the performance analysis and improvement of AI components, in particular, DNN-based components nu-

merous approaches and tools exist. (Yu & Zhu, 2020) provides a good overview of this area. Most of these approaches focus on network architecture and other hyper parameters of the DNN, but do not analyze the performance of the entire system. Furthermore, analysis of architecture, training/testing data sets, and training algorithms are usually performed in isolation.

Runtime assurance architectures have been subjected to a standard in the ASTM F-3269 (Nagarajan et al., 2021) for safety certification purposes. Numerous approaches for runtime monitoring exists, e.g., coPilot (Pike, Goodloe, Morisset, & Niller, 2010).

6. CONCLUSIONS

Certification procedures and guidelines for systems with ML and AI components have numerous V&V objectives. In this paper, we present an extension of our SYSAI framework, which enables simultaneous analysis of objectives on multiple levels. The analysis provides feedback to the designer and generates information for an advanced runtime assurance architecture, which does not only continuously monitors the system for safety violations, but also can switch between different AI components to yield best possible performance.

As complex systems with advanced AI/ML components are increasingly used in many areas, our SYSAI approach can be used in many other areas, like in the automotive domain. Albeit similar safety and performance requirements apply for the automotive domain, RTA is still in its infancy there. Other application areas for SYSAI include non-visual ML-based subsystems, e.g., those used for prognostics and health management.

For this case study, the ACT system including the DNNs were given to us for analysis and V&V. Valuable feedback can be provided to the designers as a result of SYSAI analysis. This not only concerns threshold values or operational constraints, as we discussed the use of two different DNNs, but SYSAI allows to quickly, and without too much effort test and evaluate substantially different architectures. Examples could include the use of two cameras, mounted at different positions on the aircraft, or a DNN that ingests a number of consecutive frame before estimating cte, he. SYSAI can provide substantial support during an early, highly iterative design process.

Future work will include improvements of our RTA with respect to potentially harmful transients while switching components and the integration of prognostics algorithms to predict when an AI component's performance declines or the system becomes unsafe.

We will also work on extending SYSAI applications toward other safety-relevant domains (e.g., automotive), Prognostics and Health Management (PHM) components (He & Schumann, 2024), and toward integrating SYSAI into the design process for AI/ML components in complex systems.

NOMENCLATURE

AC	Aircraft
ACT	Autonomous Centerline Tracking
AI	Artificial Intelligence
APPL	Applicant
cte	centerline tracking error
DNN	Deep Neural Network
DQR	Data Quality Review
EASA	European Union Aviation Safety Agency
FTA	Fault Tree Analysis
FMEA	Fault Mode & Effects Analysis
GPS	Global Positioning System
he	heading error
ML	machine learning
NAV	navigation
NN	Neural Network
PC	Performance Condition
ReLU	Rectified Linear Unit
RTA	Runtime assurance
SAE	Society of Automotive Engineers
SRM	Safety Risk Mitigation
SuT	System under Test
SYSAI	System Analysis using Statistical AI
UASs	Unmanned Aerial Systems
V&V	Verification and Validation

REFERENCES

ASTM. (2017). *ASTM F3269 - 17 standard practice for methods to safely bound flight behavior of unmanned aircraft systems containing complex functions.*

Cohn, D. A. (1996). Neural network exploration using optimal experimental design. *Advances in Neural Information Processing Systems*, *6*(9), 679–686.

EASA, & Daedalean. (2021). *Concepts of design assurance for neural networks II* (Tech. Rep.).

EASA Concept Paper: First usable guidance for Level 1 machine learning applications (Tech. Rep.). (2021). European Aviation Safety Agency.

Gramacy, R., & Polson, N. (2011). Particle learning of Gaussian process models for sequential design and optimization. *Journal of Computational and Graphical Statistics*, *20*(1), 467–478.

He, Y. (2012). *Variable-length functional output prediction and boundary detection for an adaptive flight control simulator* (Unpublished doctoral dissertation). University of California at Santa Cruz.

He, Y. (2015). Online Detection and Modeling of

safety Boundaries for Aerospace Applications using active learning and Bayesian statistics. In *2015 International Joint Conference on Neural Networks, IJCNN, 2015* (pp. 1–8). IEEE. Retrieved from http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=7256526 doi: 10.1109/IJCNN.2015.7280595

He, Y., & Schumann, J. (2020). A framework for the analysis of Deep Neural Networks in aerospace applications using Bayesian statistics..

He, Y., & Schumann, J. (2023). Statistical analysis and runtime monitoring for an AI-based autonomous Centerline Tracking System. In *Proc. phmap.*

He, Y., & Schumann, J. (2024). SYSAI for System Health Management - a statistical framework for the analysis of diagnosis systems. In *Proc. phm 2024.*

He, Y., Schumann, J., & Yu, H. (2022). Toward runtime assurance of complex systems with AI components. In *Proc. phme 2022.*

Jones, D., Schonlau, M., & Welch, W. J. (1998). Efficient global optimization of expensive black box functions. *Journal of Global Optimization*, *13*, 455–492.

MacKay, D. J. C. (1992). Information–based objective functions for active data selection. *Neural Computation*, *4*(4), 589–603.

Nagarajan, P., Kannan, S. K., Torens, C., Vukas, M. E., & Wilber, G. F. (2021). ASTM F3269 - an industry standard on run time assurance for aircraft systems. In *Aiaa scitech 2021 forum.* Retrieved from https://arc.aiaa.org/doi/abs/10.2514/6.2021-0525 doi: 10.2514/6.2021-0525

Pike, L., Goodloe, A., Morisset, R., & Niller, S. (2010, November). Copilot: A hard real-time runtime monitor. In *Proceedings of the 1st Intl. Conference on Runtime Verification.* Springer. (Preprint available at https://leepike.github.io/pub_pages/rv2010.html)

Ranjan, P., Bingham, D., & Michailidis, G. (2008). Sequential Experiment Design for Contour Estimation from complex Computer Codes. *Technometrics*, *50*(4), 527–541.

Reinbacher, T., Rozier, K. Y., & Schumann, J. (2014). Temporal-Logic Based Runtime Observer Pairs for System Health Management of Real-Time Systems. In *Tools and Algorithms for the Construction and Analysis of Systems - 20th International Conference, TACAS* (Vol. 8413, pp. 357–372). Springer.

Rozier, K. Y., & Schumann, J. (2017). R2U2: tool overview. In *Proceedings rv-cubes 2017* (pp. 138–156).

Society of Automotive Engineers (SAE). (2021). *Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles* (Tech. Rep. No. J3016_202104). SAE.

Taddy, M. A., Gramacy, R. B., & Polson, N. G. (2011). Dynamic Trees for Learning and Design. *Journal of the American Statistical Association*, *106*(493), 109-123.

Yu, T., & Zhu, H. (2020). *Hyper-parameter optimization: A review of algorithms and applications.* Retrieved from https://arxiv.org/abs/2003.05689