Toward Runtime Assurance of Complex Systems with AI Components

Yuning He[1], Johann Schumann[2], Huafeng Yu[3]

[1] *NASA, NASA Ames Research Center, Moffett Field, CA, 94035, USA*
Yuning.He@nasa.gov

[2] *KBR/Wyle, NASA Ames Research Center, Moffett Field, CA, 94035, USA*
Johann.M.Schumann@nasa.gov

[3] *Boeing Research & Technology, Huntsville, AL 35808, USA*
huafeng.yu@boeing.com

ABSTRACT

AI components (e.g., Deep Neural Networks) are increasingly used in safety-relevant aerospace applications. Rigorous Verification and Validation (V&V) is mandatory for such components, yet V&V techniques for DNNs are still in their infancy and can often only provide relatively weak guarantees. In this paper, we will present a runtime-monitoring architecture, which combines the advanced statistical analysis framework SYSAI (System Analysis using Statistical AI) with temporal and probabilistic runtime monitoring carried out by R2U2 (Realizable, Responsive, and Unobtrusive Unit). We will present initial results of our tool set and architecture on a case study, a DNN-based autonomous centerline tracking system (ACT).

1. INTRODUCTION

Artificial Intelligence (AI) components such as Deep Neural Networks (DNNs) have found their way into many complex systems in the aerospace and automotive domain. Such use of AI exhibits tremendous benefits, but most of the applications are safety-critical, and failures might lead to loss of vehicle and mission or even to loss of life. Certification standards for safety-critical systems (e.g., DO-178C or ISO 26262)require processes with rigorous Verification and Validation (V&V) goals. However, techniques for V&V for AI components are still in their infancy. Certification standards for safety-critical components, which are based upon AI and machine learning are still under development (e.g., (EASA, 2021; He, Yu, Brat, & Davies, 2022)).

Yuning He et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 United States License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

The intended target applications for AI and machine learning systems also require that such systems need to operate properly under a wide variety of different operational and environmental conditions, as well as under failures. This is, in particular, true for the area of autonomous vehicles, where AI components are taking over tasks of perception and decision making.

These tasks also require that failures, abnormal environmental conditions, and other hazards can be detected in real time, are properly diagnosed, and potential mitigation actions are proposed to the decision-making layers.

In order to facilitate a safe operation of the complex system and potentially support certification, advanced runtime monitoring is essential. Inspired by the ASTM-F3269 (ASTM, Nov 2021) standard, which defines a runtime assurance architecture, we propose a runtime architecture, which

- uses efficient and advanced runtime monitoring techniques (temporal logic, Bayesian probabilistic reasoning) provided by the R2U2 system, and synergistically combine it with the
- SYSAI analysis framework for complex systems with AI components.

In the architecture, which will be presented in this paper, the R2U2 system dynamically monitors numerous system signals and information originating from the AI component. Temporal logic observers for past and mission time temporal logic make it possible to check a multitude of complex properties, which need to be fulfilled when the complex system is working properly. If the AI system is not working as expected or failures occur, the R2U2 monitors and reasoners provide diagnostic information, which can be used to operate a "runtime assurance switch", which causes to activate safe (and potentially verified) fall-back components.

A complex (AI) system requires complex properties and parameters to be checked; simple thresholding is, in most cases, not sufficient. But how can those complex properties and parameters be obtained?

For this task, we use SYSAI (System Analysis using Statistical AI), our flexible statistical learning framework for V&V and analysis of complex and high-dimensional cyber-physical systems with AI components. SYSAI provides algorithms to efficiently create statistical models, perform safety-envelope analysis, characterize safety boundaries, and carry out time series analysis. SYSAI is used during design and V&V time of the system development process. Learned statistical models of the complex system and its AI components, which are produced by SYSAI during V&V provide the detailed information that is necessary to enable the R2U2 runtime monitor to efficiently perform advanced safety and performance checks for nominal and off-nominal conditions. These checks are expressed as temporal properties and also include Bayesian statistical reasoning.

In this paper, we propose a draft of a process that uses SYSAI for system analysis and feedback to the designer during development time and that transfers essential information to be used by the R2U2 observers in our runtime monitoring architecture.

We will demonstrate our approach with a case study on a DNN-based autonomous centerline tracking system (ACT). This ACT system uses a vision-based deep neural network to guide an aircraft down the runway during taxi. We will illustrate the capabilities of this architecture using safety-boundary monitoring and handling of a class of camera-related failures.

The rest of the paper is structured as follows: Sections 2 and 3 present background about our SYSAI statistical framework and the R2U2 tool, respectively. In Section 4, we will in detail describe our monitoring architecture, which is based upon R2U2 and define a process, on how SYSAI can provide system model data and parameters, which are needed for the monitor. Section 5 focuses on our case study on autonomous centerline tracking (ACT). We first describe experiments on monitoring the system under nominal operating conditions and then illustrate the capabilities of our architecture on a selected failure case: partial obstruction of the camera by dirt or an insect on the camera lens. Section 6 presents related work and Section 7 concludes and discusses future work.

2. Background: The Statistical Analysis Framework SYSAI

SYSAI (System Analysis using Statistical AI) is a flexible statistical learning framework for V&V and analysis of complex and high-dimensional cyber-physical systems with AI components. Figure 1 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 ATC system and the XPlane simulator, as described in the previous section. The SuT is executed given a set of parameters and initial conditions provided by the statistical learning model of SYSAI. The result of the test run, which could be a binary safe/not-safe information, a single value (e.g., cte_{max}), or an entire time series is provided back to SYSAI. These data are then used to incrementally construct our statistical model.

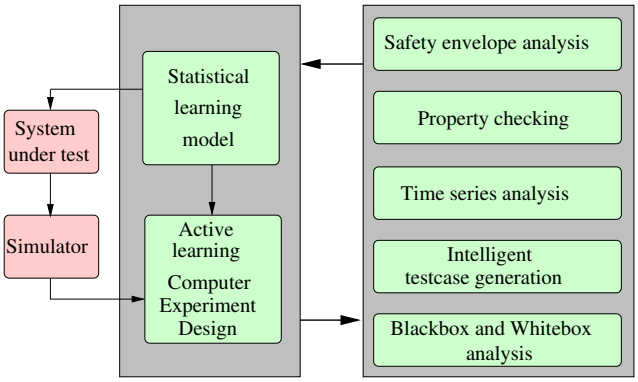

Figure 1. SYSAI architecture

The interface between SYSAI and the SuT is designed to be very small and generic, so that systems implemented in R, Matlab, Java, or Python can be connected easily. For the representation and construction of the statistical SYSAI model, we are using 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 very 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, our 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 has been used for the analysis of several complex and safety-critical aerospace systems (He, 2015; He et al., 2022; He, Yu, Brat, & Davies, 2021).

3. BACKGROUND: R2U2

The R2U2 (Realizable, Responsive, and Unobtrusive Unit) (Rozier & Schumann, 2017; Reinbacher, Rozier, & Schumann, 2014; Geist, Rozier, & Schumann, 2014) is an onboard monitoring system to continuously monitor system and safety properties of a cyber-physical system or its components. Health models within this framework (Schumann, Rozier, et al., 2015) are defined using Metric Temporal Logic (MTL) and Mission-time Linear Temporal Logic (LTL) (Reinbacher et al., 2014) for expressing temporal properties as well as Bayesian Networks (BN) for probabilistic and diagnostic reasoning. A signal processing unit reads in continuous sensor signals or information from the prognostics unit and performs filtering and discretization operations. Figure 2 shows the high-level architecture of R2U2.

A large number of safety and performance properties for ACT can be formulated using temporal logic. Some properties directly monitor the DNN component, e.g., a simple range check for the DNN outputs, e.g. $\square(|cte_{NN}| < 30)$. With these instantaneous properties, which have no temporal component, the current behavior of the DNN as well as the aircraft (e.g., the commanded steering angle for the front-wheel shall be limited). Proper temporal formulas are used to suppress short dropouts and deviations of the DNN output, as they will be counter-acted by the ACT controller. Of more interest are temporal properties, which limit the number of bad outputs per minute, or classification of longer-duration problems. For example,

$$\square((|he| > 10^\circ)\mathcal{U}_{[0\,\mathrm{s},9\,\mathrm{s}]}(|he| \leq 8^\circ)) \quad (1)$$

raises an alarm, if a large heading error persists for more than 10 seconds, indicating a possibly unbounded movement to the edge of the runway. In our application for the dynamic comparison of DNN performance, which will be described below, we use temporal properties to analyze a short temporal trace of the DNN and to analyze closed-loop behavior of the ATC.

On the system level, R2U2 can be used, for example, to continuously check for oscillations occurring in ATC, as they might cause poor performance or can lead to unsafe situations. For the definition of all temporal operators and more examples see (Rozier & Schumann, 2017; Schumann, Roychoudhury, & Kulkarni, 2015).

R2U2 can also perform efficient Bayesian reasoning and has a built-in model-based prognostics engine, which will be helpful for monitoring an AI-based system, but these capabilities have not yet been used for this paper.

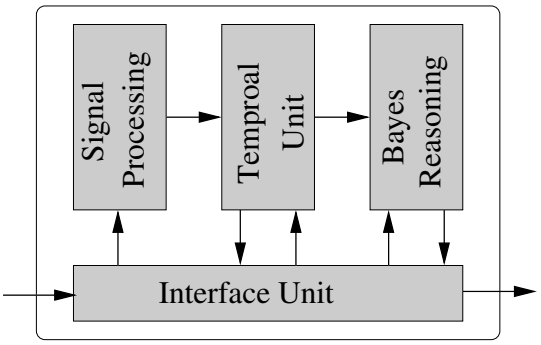

Figure 2. R2U2 architecture with major components for signal processing including prognostics, and temporal and Bayesian reasoning

4. MONITORING AND ASSURANCE ARCHITECTURE

In this section, we present our architecture for monitoring of the complex AI component, using the R2U2 runtime monitor. Important information for the R2U2 properties are produced by SYSAI during statistical analysis of the system at design and V&V time. For a synergistic combination of both tools, we propose a draft of a process.

The main goal of this architecture is to provide a framework for the monitoring of a complex AI system, e.g., a Deep Neural Network, during runtime. Future work (see Section 7) will refine that architecture into a runtime assertion framework suitable for certification purposes.

4.1. Monitoring Architecture

The R2U2 system dynamically monitors numerous signals and information provided by the system or its components. It can provide Boolean results on any violation, but can also perform Bayesian reasoning, returning probabilities and confidence values.

For the continuous monitoring of an AI component in a potentially safety-critical system, we have designed, inspired by the ASTM F-3269 RTA (ASTM, Nov 2021), an architecture

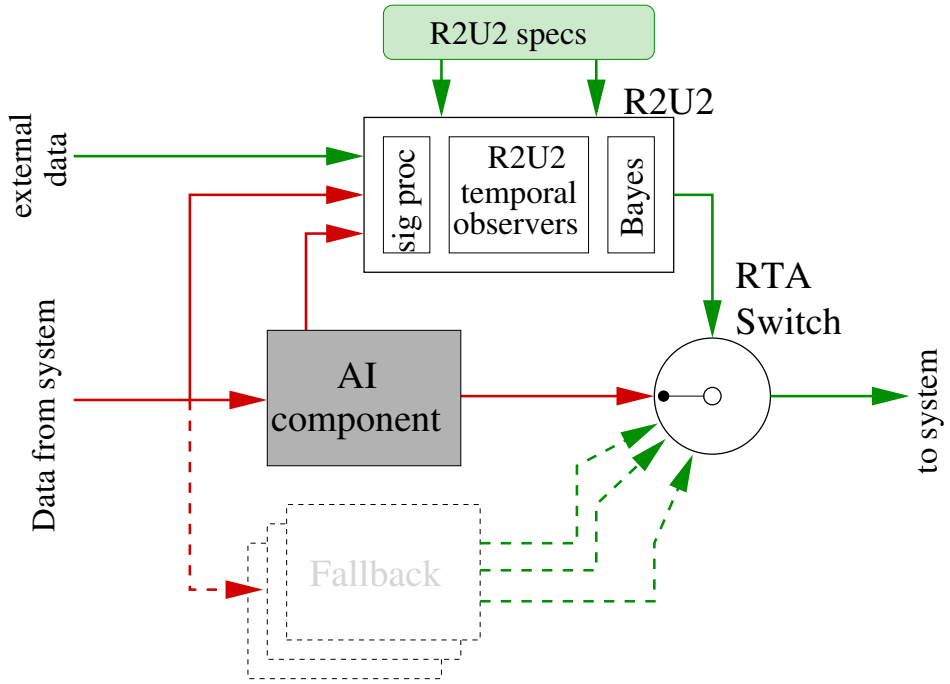

Figure 3. R2U2 runtime monitoring architecture (inspired by (Nagarajan et al., 2021), Fig. 1)

as shown in Figure 3.

The complex AI component, e.g., a Deep Neural Network, is shown as a gray box in the center of the figure. It receives inputs from the system, e.g., camera inputs or sensor signals and processes them. The results (e.g., estimated position of the AC on the runway) are then passed through the RTA switch back to the system, e.g., the aircraft controller. In nominal operations, the RTA switch is set to route the signals from the AI component to the system.

In parallel, R2U2 receives the system signals, as well as signals from the AI component. The latter signals can be, for example, output of the neural network, confidence values for the output, or internal values. The latter, for example, would be important in applications, where the Neural Network is trained or adapted during operation.

In addition to these signals, R2U2 can receive external data of high integrity (e.g., pilot input, redundant sensors, etc). The properties for R2U2 and their parameters have been designed and augmented with results from the SYSAI analysis as discussed below. R2U2 is operating on inputs and specifications and produces an updated result every time step. Typically, R2U2 is operated with a rate of 1Hz or 10Hz. The R2U2 output is used to control the RTA switch: in case, R2U2 detects a violation of important safety/performance properties, the RTA switch can be turned to use a fallback component instead of the AI component to retain system safety and (at least limited) performance. Multiple fallback methods might be provided, ranging from algorithmic components (e.g., simple dead reckoning) to entering a fail-safe mode, stopping the AC, and contact a remote operator.

4.2. Development Process

Figure 4 illustrates the overall development and monitoring process. Based upon detailed system requirements, the system with AI components is developed and the DNN(s) are trained using training data. At this V&V stage, SYSAI can be used for analysis of training data, characterization of safety regions in a high-dimensional state space, as well as analysis of the system's behavior under failures (He et al., 2022, 2021). Analysis results also provide feedback to the designer.

After system development and testing, the system is being deployed. At this stage, the R2U2 runtime monitoring is active while the system is in operation. Without affecting the overall system behavior (unobtrusiveness), a multitude of temporal and probabilistic properties can be checked and warning signals or alarms be generated. The statistical models and results, produced by SYSAI, are used to define and customize properties to be checked by R2U2 (vertical red arrow). The information passed can range from simple threshold parameters, whose values have been be determined by SYSAI's safety-boundary characterization. In that case, SYSAI's advanced capabilities for the geometric characterization of safety boundaries can be used for setting up efficient R2U2 property checking.

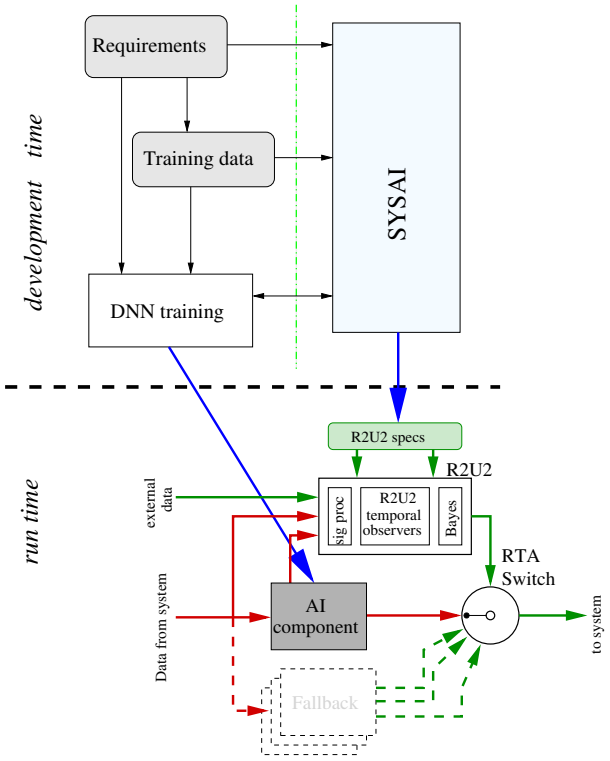

Figure 4. Tool chain and process for the combination of SYSAI and R2U2

5. Case Study

5.1. Autonomous Centerline Tracking

As a case study for our approach, 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 wings (Figure 5). The DNN component 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.

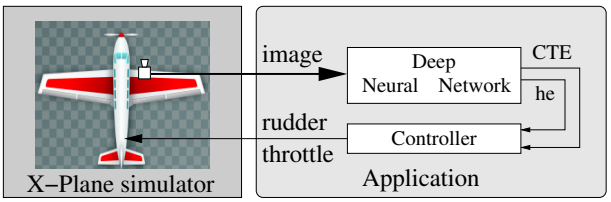

Figure 5. Architectural Overview of ATC

A simple fixed-gain proportional controller uses this information to produce control signals to steer the aircraft left and right. A separate controller keeps the aircraft is rolling with a constant, low speed.

For our experiments, the X-Plane Flight Simulator[1] is used as simulation environment. A simulated camera takes information from the simulator display; the control signals for throttle and rudder are sent to the X-Plane simulator using the programmatic interface NASA XPlane Connect.[2]

The DNN is a multi-layer feed-forward network with ReLU nodes. The DNNs are implemented using the TensorFlow framework[3] and have been trained on data that have been obtained with the simulated aircraft within the X-Plane simulator. Note that in this application, DNN is not learning any time-series data. Rather the DNN is learning a mapping between the input image (showing a part of the runway) and the corresponding cte and he values.

5.2. Nominal Safety Regions

For the setup of the R2U2 specifications for nominal operation, it is, among others, important to establish reasonable safety thresholds for the neural network outputs using SYSAI. As described above, the ATC DNN produces two outputs, the cross-track error CTE and the heading error he. SYSAI can perform a simultaneous analysis for both parameter, but for this paper we focus on CTE to simplify the presentation of results. During a ATC-guided run, the value of CTE must not surpass the safety threshold θ_{CTE}, i.e., our safety condition is $CTE < \theta_{CTE}$. Obviously, if θ_{CTE} is very small, only few runs will be successful and most runs will violate our threshold safety property.

On the other hand, a large threshold would allow almost all runs to succeed, but the aircraft might veer off the runway proper, which is an unsafe situation. We also have to assume that the AC does not always start exactly at the beginning of the runway precisely on the center line and is perfectly assigned to the center line. Rather, the initial conditions imply non-zero initial cross track error CTE_0 and he_0. With out SYSAI analysis, we want to find out (a) what are the success rates for a given threshold, and within which geometric boundaries of the AC initial position and heading, a good success rate can be accomplished. In this experiment, we therefore allow SYSAI to vary the initial parameters CTE_0 and he_0.

Figure 6 shows how, for a given threshold, the starting position and heading of the AC influences the success of a run. Each dot in each panel indicates the starting position of the AC, the protruding line shows the initial AC heading. In each panel, the initial part of the runway is shown, going from the lower left to the upper right. If the threshold is very low, almost no runs are successful (Figure 6A). A somewhat larger

[1] www.xplane.com
[2] https://github.com/nasa/XPlaneConnect
[3] https://www.tensorflow.org

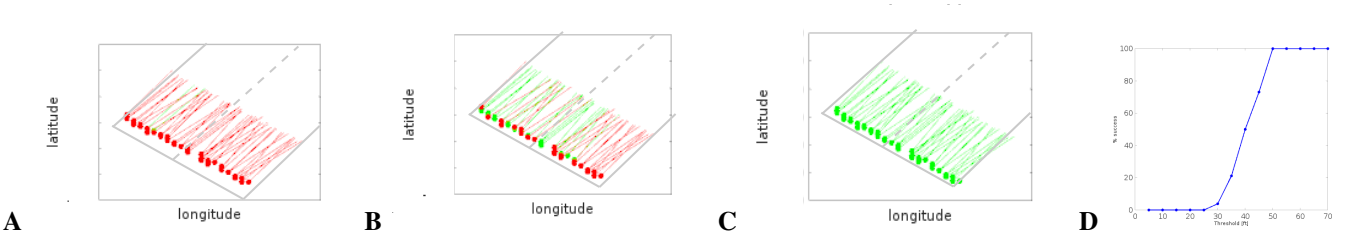

Figure 6. Threshold analysis: success (green) or failure (red) for different initial positions and headings of the AC at the beginning of the runway for different thresholds (A: 30ft, B: 40ft, and C: 50ft). The dots mark the initial position, the lines indicate the heading of the AC. D: success rate (in %) over threshold.

threshold (Figure 6B) shows that around half of the runs are successful. Here it can be seen that the starting position actually makes a difference: starting positions to the left of the center line tend to be much more successful than when starting on the right of the center line. This result can be a basis for further analysis of the coverage of training data, camera placement, or the controller design. Finally, when the threshold is very large, all runs succeed. Figure 6D shows the success rate (in %) for different thresholds.

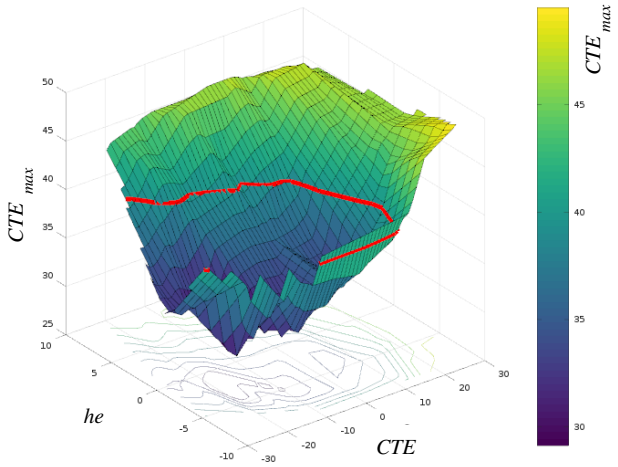

Figure 7. Safety-envelope: surface shows estimated maximal CTE value during a run over initial position CTE and heading of the aircraft he. The safety envelope at a given threshold of 40ft is shown as a red line.

Figure 7 shows how the safety envelope for ATC under different initial conditions CTE_0 and he_0 develop. For a small threshold, only runs with initial values close to $CTE_0 = 0$ and $he_0 = 0$ are successful, i.e., that our safety conditions is never violated during a run. This safety envelope becomes larger as the value for θ increases. The red line in Figure 7 shows its boundary for $\theta = 40ft$. SYSAI has been used to effectively create a model of this surface; a geometric characterization of the boundary can be obtained from SYSAI.

So far, all experiments were carried out in clear conditions and 0900 local time. We then extended the experiment to include the time-of-the-day as an additional parameter. Obviously, ATC will not perform well in darkness, but it is important to know if ATC is performing differently at different times during the day.

Figure 8A shows the overall success rate in percent for a safety threshold of 40ft. The success rate varies tremendously during different times of the day and is only satisfactory between around 9AM and 1PM local time. Outside this time window, the performance of ATC is dropping sharply. A closer look at the images captured by the camera reveals the reason: Figure 8 shows typical images for a run at 9AM, 11AM, and 3PM, respectively. Compared to the 11AM run (middle panel), the early morning image is much darker. Since our version of ATC only has been trained with brighter images only, it is obvious that ATC performs not well in the earlier morning hours. The image on the right, taken during a 3PM run shows that the shadow of the aircraft is clearly visible and thus dramatically changing the overall image. Unless ATC had been trained on images like that, its performance is likely to be strongly diminished. Similarly, additional environmental parameters, like a wet runway, snow, or a cloudy sky can be modeled and analyzed with SYSAI.

The information obtained during the SYSAI analysis is then used to set up the R2U2 properties and monitors. We can distinguish between three different categories of R2U2 properties: (a) universal properties, (b) temporal properties, and (c) probabilistic properties. Universal properties are supposed to be valid throughout the entire operation and are necessary to define many safety properties. For example, $v_w < 5m/s \wedge 0 \leq v_w$ makes sure that the speed of the aircraft is always limited and that the aircraft never rolls backward. Within the R2U2 monitor, such properties are usually linked to conditions or system modes. In our example, this condition is only to be checked if the ATC system is on and the AC in taxi mode M^{AC}. This will yield:

$$\square((\text{ATC}_{on} \wedge M^{AC} = \text{TAXI}) \rightarrow (v_w < 5\frac{m}{s} \wedge 0 \leq v_w)) \quad (2)$$

Temporal R2U2 properties can be used to specify

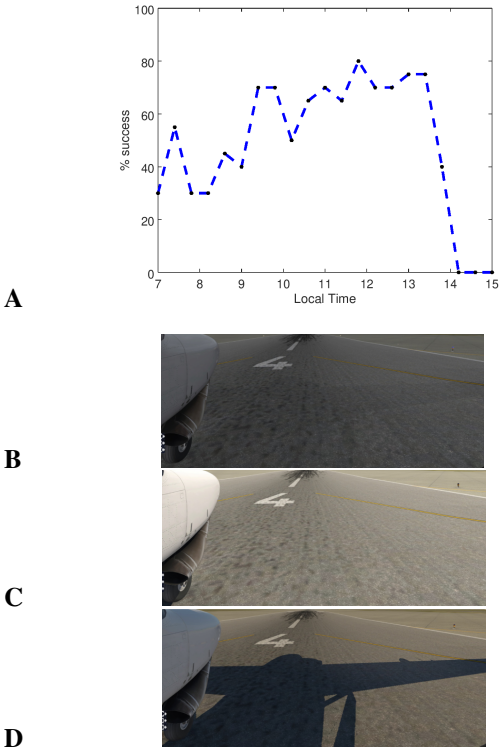

Figure 8. A: Success rate (in %) for different times of the day. Threshold for CTE is 40ft. Camera images taken from runs at 8AM (A), 11AM (B), and 3PM (C).

- overall performance properties, e.g., the end of the runway should be reached within 4-5 minutes:

$$M^{AC} = \text{TAXI}) \rightarrow \Diamond_{[4,5min]}(d_{rwy} > 0.9 * L_{rwy}) \quad (3)$$

- filtering of transients. For example, the outputs of the DNN should always lie in a certain range, e.g., $he \in [-10, 10]$. However, transients, yielding values outside that range should be tolerated if they are short enough, for example, less than 2 seconds:

$$(-10^\circ \leq he \leq 10^\circ) \vee \neg H_{[2s]}(he < -10^\circ \vee he > 10^\circ) \quad (4)$$

- limiting the number of occurrences of events. For example, it can be specified that no more than 3 transients occur within a period of 20 seconds. Such a property can also be seen as a discrete form of specifying error rates.

Such properties can be defined using the original signals (e.g., cte, he), or results of signal processing. In our case study, we use:

- signal rates, as approximation of signal derivatives are used to help monitor the system dynamics,
- sum or integration is used to check for biases over time,
- fast Fourier Transformation of signals are helpful in detection of oscillations. Such effects, similar to pilot-induced oscillations can lead to dangerous situations that need to be avoided.
- Kalman filtering can be used for sensor fusion or to check the behavior of a signal against a given dynamical model. In this case study, Kalman filters have not been used.
- prognostics algorithms can be used to estimate the state of important components, e.g., the battery in electrical AC. R2U2 can, for example, check that there is always enough battery to taxi down the full runway. (not used in this case study)

For our case study, we used the signals from ACT and the aircraft as shown in Table 1. A more realistic case study would include numerous additional signals and sensor outputs (e.g., GPS, runway maps, etc).

Table 1. Signals used for R2U2 in the ACT case study. The column S indicates if signal processing is used. Signals in the lower part are used for failure monitoring (see below)

Name	S	Description
cte	•	DNN output cross-track error
he	•	DNN output heading error
v_w	•	AC front wheel speed
d_{rwy}	•	distance on runway (est)
α	•	steering angle
ATC_{on}		Boolean: ATC system on
AC_{mode}		AC mode (e.g., taxi, takeoff)
T_{UTC}		current on-board time
I_{bright}	•	image brightness
I_{contr}	•	image contrast
I_{block}	•	image blockage

5.3. Monitoring of Failure Conditions

As demonstrated above, it is important to monitor the behavior of the AI component in nominal operating conditions. Equally important, if not more important, however, is the monitoring of the complex AI function in case of failures. Failures can be the result of an unexpected environmental condition, e.g., fog or snow, or problems and faults with the sensors and actuators.

For traditional systems, fault detection and diagnosis systems are used to detect, isolate, and react upon the fault. In many cases, such systems are model-based and rely on the detailed knowledge about the system behavior in the failure case.

AI systems, on the other hand, are often considered black-box, i.e., they cannot explain or describe their behavior while in operation. This, well-known problem of explainability of AI and the fact that AI systems often have to operate in a huge, high-dimensional state space makes it impossible to perform coverage testing during V&V time.

In our architecture, we use information from SYSAI on failure analyses, to derive powerful runtime monitors that can be checked with R2U2.

5.3.1. Camera Failures

As a motivating example, let us consider a failure in the camera system: a piece of dirt or an insect on the lens is obstructing a part of the image. Image data in the obstructed region are consistently dark. Obviously, the ACT DNN has some robustness against such situations.

However, for improved system safety, we need to dynamically monitor, if the obstruction may lead to situations, where ACT fails. An analysis of this failure type revealed that the impact of an obstructing piece of dirt on the behavior of ACT is far from trivial: there are many regions of the image, where such an obstruction does not pose any restriction, which means that the DNN robustness is taking care of that situation.

However, SYSAI detected certain regions, where an obstruction can have notable and even severe consequences. Figure 9 shows a typical ACT camera image. The ”dirt” is modeled in this case as a black square. Super-imposed on this image are the boundaries of the sensitivity regions as detected by SYSAI. Inside the green boundary, a considerable risk of ACT failure exists; inside the region defined by the red boundaries, a high risk is imminent.

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

These regions obviously have to do with the way, the DNN has been trained to perceive the visual situation. As expected, areas near the horizon are of concern. The second, high risk area is close to the front wheel and it is suspected that the DNN uses this region to detect the runway center line or other optical markings. However, these regions are not obviously explainable and strongly depend on how the DNN has been trained.

With our R2U2 architecture, we are now able to use this information to produce an effective and powerful runtime monitor to detect such situations. By using SYSAI analysis data, we can avoid over-conservative monitors that would shut down the AI component as soon as a spot of a certain size is detected, causing numerous false alarms. Still the detailed SYSAI analysis provides the necessary confidence in the AI behavior that allows the R2U2 monitor to behave safely.

More specifically, our R2U2 monitor consists of the following R2U2 components and specifications

- a simple, traditional detection algorithm for camera obstructions. This algorithm returns a Boolean array, indicating the locations of obstructions.
- an R2U2 matcher that matches obstruction regions with a heat-map produced by SYSAI (with boundaries similar to Figure 9). This code is also traditional.
- an obstruction-risk value is calculated, using a weighted sum of the obstructions with the boundaries, and fed, after thresholding into the R2U2 temporal reasoner
- temporal formulas now check this signal, trying to weed out transient signals, checking persistence of the obstruction, and correlating with potential other failures or situations. E.g., taxiing after dark should not trigger the camera-obstruction monitor.
- the resulting signal is merged with results from the other R2U2 monitors to produce a final verdict to be sent to the RTA switch. In this case study, we are using Boolean conditions for that; a more elaborate monitoring variant would feed these monitoring results into a Bayesian network for probabilistic reasoning and calculation of confidence levels.

6. Related Work

Runtime monitoring and runtime verification is mainly focusing on checking safety or security properties while the system is in operation (see e.g., (Havelund, Reger, & Rosu, 2019) for an overview). Violations usually cause alarms and can lead to drastic mitigation actions. Furthermore, most runtime monitoring systems are only concerned with model-based or (temporal) logic-based property checking. R2U2 also features efficient Bayesian reasoning, which seems to be a major help in the analysis of the, by nature, probabilistic DNNs.

In contrast to most related work, which aims at supporting property checking for V&V and safety purposes, we use R2U2 to switch between the AI component and different fall-back components in order to dynamically select a safe and suitable one. Our architecture is somewhat inspired by the ASTM3269 Runtime Assurance (RTA) architecture (ASTM, Nov 2021; Nagarajan et al., 2021), where a safety-monitor can switch from a complex, unassured component (e.g., a DNN) to some assured fall-back function, but aims to fulfill a different purpose.

7. Conclusions

In this paper, we have presented an advanced architecture to monitor the safety and performance of a complex AI component (e.g., a DNN) within an aerospace system. Inspired

by the ASTM RTA, we are using the R2U2 runtime monitoring system to dynamically check numerous properties, using temporal logic observers, Bayesian reasoners, and signal processing.

Our SYSAI statistical analysis framework can provide models, parameters, and other information to R2U2 to enable the definition of complex, yet justified properties that go ways beyond traditional range and rate checking monitors.

Future work will include the use of dynamic statistical reasoners and prognostic engines to extend this architecture into a fully statistical monitoring system, which can reason and decide with probabilities and confidence levels—a prerequisite for monitoring systems like Deep Neural Networks. We are also planning to work toward the use of this architecture and process in certification and risk management.

REFERENCES

ASTM. (Nov 2021). *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. (2021). *Easa concept paper: First usable guidance for level 1 machine learning applications* (Tech. Rep.). European Aviation Safety Agency.

Geist, J., Rozier, K. Y., & Schumann, J. (2014). Runtime Observer Pairs and Bayesian Network Reasoners On-board FPGAs: Flight-Certifiable System Health Management for Embedded Systems. In *Proceedings Runtime Verification (RV14)* (pp. 215–230). Springer.

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.

Havelund, K., Reger, G., & Rosu, G. (2019). Runtime verification past experiences and future projections. In B. Steffen & G. J. Woeginger (Eds.), *Computing and software science - state of the art and perspectives* (Vol. 10000, pp. 532–562). Springer. doi: 10.1007/978-3-319-91908-9_25

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, killarney, ireland, july 12-17, 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., Yu, H., Brat, G., & Davies, M. (2021). Statistical learning framework for safety and failure analysis of a DNN-based autonomous aircraft system. IEEE.

He, Y., Yu, H., Brat, G., & Davies, M. (2022). System and safety analysis for autonomous center line tracking with sysai..

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.* doi: 10.2514/6.2021-0525

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).

Schumann, J., Roychoudhury, I., & Kulkarni, C. (2015). Diagnostic reasoning using prognostic information for unmanned aerial systems. In *PHM15.*

Schumann, J., Rozier, K. Y., Reinbacher, T., Mengshoel, O. J., Mbaya, T., & Ippolito, C. (2015). Towards Real-time, On-board, Hardware-supported Sensor and Software Health Management for Unmanned Aerial Systems. *International Journal of Prognostics and Health Management*.

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.