4th Asia Pacific Conference of the Prognostics and Health Management,
Tokyo, Japan, September 11 – 14, 2023 R02-06

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*
johann.schumann@us.kbr.com

ABSTRACT

Autonomous Centerline Tracking (ACT) enables an unmanned aircraft to be guided down the center of the runway, using a camera-based Deep Neural Network (DNN). ACT is safety-critical. The EASA Guidelines for machine-learning based systems list numerous assurance objectives that must be met toward certification and V&V. We extend our analysis framework SYSAI to support meeting assurance objectives for a system with AI 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. Recently, the EASA (European Union Aviation Safety Agency) published guidelines (European Aviation Safety Agency, 2021) on how to deal with autonomous Level 1 systems that contain ML-based components like a DNN. It 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 information 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.

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 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 starboard wing (Figure 1). 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.

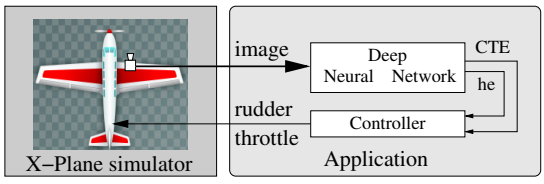

Figure 1. Autonomous Centerline Tracking (ACT) system

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.

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.

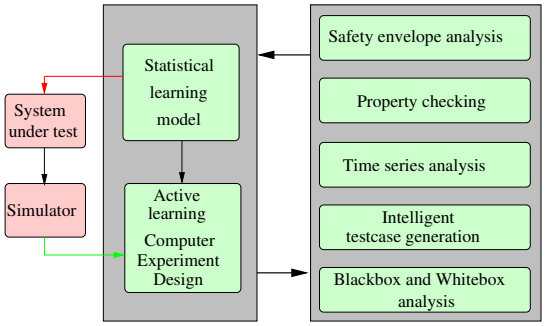

Figure 2. SYSAI architecture

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.

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.

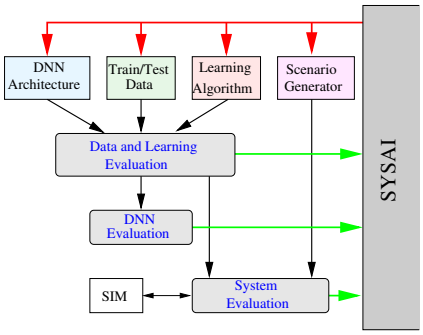

Figure 3. Extended SYSAI interface

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.

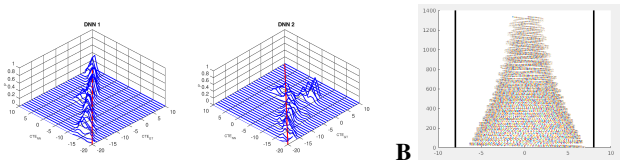

Figure 4. **A:** performance analysis of two different DNN architectures. **B:** Camera positions for training set on the runway. The runway is oriented vertical and the taxi trajectories start at the bottom of the graph.

Figure 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, the RTA has developed and published the standard F-3269 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.

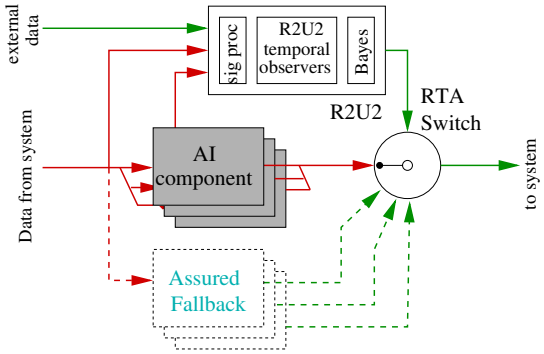

Figure 5. R2U2 runtime monitoring architecture (inspired by (Nagarajan et al., 2021), Fig. 1)

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. 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.

Based upon systems and safety requirements and information obtained from the SYSAI analysis, a set of temporal logic formulas are developed, 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.

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.

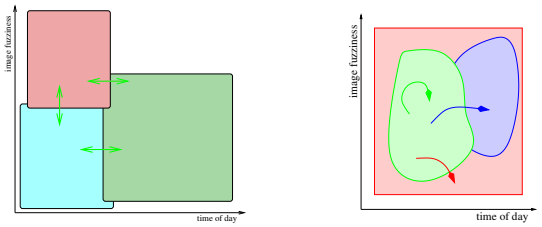

Figure 6. Performance areas for different components wrt. two variables

Figure 6A 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. For example, the red NN performs well during early morning and foggy conditions; others perform well for bright sunlight or afternoon situations. The RTA can extract the image fuzziness using simple and trustworthy code. The R2U2 formula for switching would be:

$$SW : \begin{cases} ACT \wedge H_{[20s]}(\neg act(DNN_2) \wedge \\ (T > 1100(local)) \wedge H_{[10s]} Fz(img) > \Theta^{fz} \end{cases}$$

A switch to DNN_2 can happen if (a) the autonomous centerline control is active, (b) this network has not been active within the last 20 seconds, (c) the time of the day is later than the given threshold, and (d) the fuzziness of the image is larger than some threshold. In order to avoid quick switching and oscillation, the temporal operator $H_{[10s]}$ has been added. The corresponding subformula means that the bad image quality has to persist for at least 10 seconds.

Figure 7 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 7A 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 prohibits successful operation (bright yellow areas). The analysis of a different DNN 2 reveals that, albeit poorer overall performance,

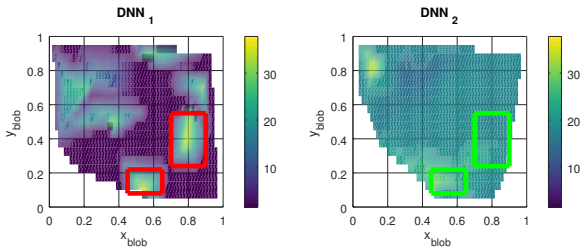

Figure 7. Performance of ACT under dirt on the camera lens.

DNN 2 is not that sensitive to dirt patches (Figure 7B). Therefore the RTA will switch from DNN 1 to DNN 2 whenever a dirt patch is detected in areas bounded in red.

During operations, the R2U2 continuously checks these properties (usually with numerous other safety properties) to come up with a verdict on (a) which of the AI component is working safely and best under the current circumstances, and (b) if the overall system safety is upheld. In the latter case, the R2U2 switches from any AI component to a safe and assured fallback component. System safety and performance is not only influenced by the current system state, but also how the system state develops over time. In Figure 6B we show safe operational regions of two AI components (green and blue) in a projection of two parameters or input signals. At each point in time, the system state is represented as an (x,y) position in this figure; development of the states, trajectories, are shown as arrows. The green trajectory in Figure 6B corresponds to a benign development of the system space as the system remains in the green area. In contrast, the blue trajectory will lead from the green to the blue area, requesting the RTA monitor to switch between the green and blue component. Finally, the red arrow is an example, where good and safe performance leads to an unsafe state; here a switch to an assured fallback component is necessary. In our RTA, such checks can be done easily by combining derivatives of the available state signals, Kalman filters, and temporal logic.

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 numerous 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.

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.

References

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.

FAA. (2021). *Neural Network based Runway Landing Guidance for General Aviation Autoland* (Tech. Rep. No. DOT/FAA/TC-21/48).

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., & Schumann, J. (2020). A Framework for the Analysis of Deep Neural Networks in Aerospace Applications using Bayesian Statistics. In *Proc. Int. Joint Conf. for Neural Networks (IJCNN), WCCI* .

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

Nagarajan, P., Kannan, S. K., Torens, C., Vukas, M. E., & Wilber, G. F. (2021). ASTM F3269 - An Industry Standard on Run Time Assurance for Air-

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.

craft 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.

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.

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