Received 13 August 2025, accepted 1 September 2025, date of publication 4 September 2025, date of current version 10 September 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3605878

RESEARCH ARTICLE

AsaFG: A Human-in-the-Loop Integration Module for Air Combat Simulations

SAMARA R. SILVA[1,2], VITOR C. F. GOMES[1], ALESSANDRO O. ARANTES[1], ANDRE F. M. CAETANO[1], VICTOR L. D. B. COSTA[1], ADRISSON R. SAMERSLA[1], FELIPE L. L. MEDEIROS[1], YURI D. FERREIRA[1], MARCIA R. C. AQUINO[1], AND JOAO P. A. DANTAS[1,2]

[1]Instituto de Estudos Avançados, São José dos Campos, São Paulo 12228-001, Brazil
[2]Instituto Tecnológico de Aeronáutica, São José dos Campos, São Paulo 12288-900, Brazil

Corresponding author: Joao P. A. Dantas (jpdantas@ita.br)

This work was supported in part by the Coordination for the Improvement of Higher Education Personnel (CAPES), Brazil, under Research Organization Registry (ROR) Identifier: 00X0ma614; and in part by the Funding Authority for Studies and Projects (FINEP) under Finance Code 01.20.0196.00/2824/20.

ABSTRACT This work presents AsaFG, a human-in-the-loop integration module that connects the Aerospace Simulation Environment (ASA), a constructive simulation environment for military scenarios, with the FlightGear flight simulator. The module enables real-time interaction between human pilots and simulated entities in air combat scenarios, allowing pilots to assume roles within simulated missions and interact with autonomous agents. This integration enables direct interaction between constructive and virtual components, contributing to more realistic and immersive air combat simulations. The work also discusses key challenges during the integration process, including data exchange, control interfaces, and missile modeling differences between the platforms. A specific contribution involves adjusting the missile model in FlightGear to closely match ASA's implementation, enabling consistent simulation behavior across systems. Validation includes performance metrics and experiments with Brazilian Air Force pilots across three operational scenarios. The results demonstrate the module's applicability for training, operational analysis, and studies on human-autonomy teaming in simulated air combat scenarios.

INDEX TERMS Autonomous systems, constructive simulation, virtual simulation, human-in-the-loop simulation, simulation interoperability.

I. INTRODUCTION

Simulation is an essential tool in modern military planning, analysis, and training because it allows the evaluation of systems and missions without the risks and costs of real-world operations [1]. In areas such as air combat, simulation makes it possible to test mission strategies, analyze different engagement situations, and support pilot preparation in safe and controlled conditions [2].

As military operations become more complex and involve the use of autonomous systems, it becomes increasingly important to understand how humans and machines work together in operational environments [3]. Human-in-the-loop simulation is helpful in this context because it places a real person inside the simulation, making it possible to evaluate how humans interact with systems in realistic missions [4]. This kind of simulation helps researchers and developers study human decision-making, workload, and cooperation with autonomous-based systems.

Several constructive simulation systems have been created to represent missions with multiple military units, including vehicles, communication structures, and sensors [5]. These systems are powerful for modeling high-level behavior, but they usually do not allow direct interaction with a human

The associate editor coordinating the review of this manuscript and approving it for publication was Rongbo Zhu.

during the simulation. On the other hand, there are very good virtual flight simulators, which offer detailed flight models and visual interfaces [6], and are available as open-source tools. Despite the use of interoperability protocols intended to facilitate integration, achieving real-time interaction between constructive and virtual simulations with human involvement continues to be a significant technical challenge [7].

To address this challenge, this work presents the integration of the Aerospace Simulation Environment (ASA) [8], a constructive simulation environment developed for military applications, with FlightGear [9], an open-source flight simulator widely used for research and training. The integration, referred to as AsaFG, enables human pilots to participate in constructive mission simulations by controlling virtual aircraft in real time and interacting with simulated mission elements.

The main contribution of this work is to demonstrate the integration between a constructive simulation platform and a virtual flight simulator, highlighting the technical challenges involved. We also present a validation process based on computational performance and human-in-the-loop experiments. The proposed approach was tested through a series of experiments with Brazilian Air Force (FAB) pilots in three distinct air combat scenarios. The results were used to assess the realism, responsiveness, and overall feasibility of the integrated simulation.

The remainder of this work is organized as follows: Section II presents background on constructive and virtual simulation, as well as human-in-the-loop concepts. Section III reviews related work on simulation architectures and human-machine collaboration. Section IV describes the proposed integration methodology, including the validation process. Section V presents the results obtained from the validation experiments. Finally, Section VI concludes the work and presents directions for future research.

II. BACKGROUND

This section provides an overview of the main simulation tools and environments used in this work. It begins with the constructive simulation frameworks used in this study, which enable the modeling of large-scale military operations involving multiple agents. Then, it presents the flight simulators employed for developing and evaluating autonomous agents in air combat scenarios, highlighting their features and integration capabilities.

A. CONSTRUCTIVE SIMULATION FRAMEWORKS

Constructive simulations are important tools for studying complex military missions that involve several units working together, such as air combat operations [10]. These simulations make it possible to test how aircraft, sensors, communication systems, and decision-making processes work, without the risks or costs of real missions. They are now widely used in planning, training, and evaluating military strategies [11].

One example is the Mixed Reality Simulation Platform (MIXR) [12], a flexible and modular system originally created for defense applications. It supports both constructive and virtual simulations and makes it possible to connect and simulate different elements, such as aircraft, weapons, and control systems. MIXR is often used for planning missions, creating wargame scenarios, and testing new ideas for military operations.

ASA, as mentioned in the introduction, is a simulation environment developed by FAB to serve as a simulation service for improving decision-making processes in operational contexts [13]. It is based on the MIXR platform but extends its capabilities with additional tools focused on research and analysis of air combat scenarios. ASA includes specific models for aircraft and missiles, tools to run large batches of simulations, and a Python library, AsaPy [14], to analyze results, calculate performance metrics, and test autonomous agent models. The environment has been especially useful in studies involving autonomous agents and decision support systems in aerial missions [15], [16], [17], [18].

B. FLIGHT SIMULATORS

Flight simulators are integrated systems composed of software and hardware components designed to emulate aircraft and replicate realistic flight conditions. These systems offer immersive experiences through precise environmental rendering, intuitive cockpit interfaces, and realistic flight dynamics. They are used in flight safety enhancement, aircraft development, defense training, academic research, and education [1].

While many high-fidelity simulators are used in certified training environments, the demand for accessible flight simulation software has grown within the research and enthusiast communities. Notable platforms include the already mentioned FlightGear, X-Plane [19], and Microsoft Flight Simulator [20]. Among these, FlightGear is particularly notable for its open-source architecture, extensive documentation, and flexibility, making it well-suited for integration with autonomous systems and simulation-based research.

The FlightGear Flight Simulator (commonly abbreviated as FlightGear or FGFS) is a cross-platform flight simulation software used in research, education, industry, and for personal use. As a free and open-source project, FlightGear allows for extensive customization and continuous development. It is compatible with major operating systems, including Windows, macOS, and Linux, and features a wide selection of aircraft models. Its global scenery is automatically updated, enhancing immersion and realism [9].

A key component of FlightGear is its default flight dynamics engine, JSBSim [6], [21]. JSBSim is a data-driven model that utilizes aircraft configuration files, which define geometry, propulsion, control systems, and other parameters, to simulate aerodynamics in real-time. Its lightweight, flexible design has enabled its use beyond FlightGear, including in applications such as the DARPA AlphaDogfight Trials, where autonomous agents performed simulated dogfights against human pilots [22].

FlightGear also supports multiplayer functionality through the FlightGear Multiplayer Server (FGMS), enabling multiple aircraft to coexist in a shared virtual airspace. This feature is essential for testing cooperative and adversarial scenarios involving both human and autonomous agents.

X-Plane 12 was also investigated due to its broad adoption in research and training applications. Developed by Laminar Research, it is a proprietary simulator known for its high-fidelity flight model, based on Blade Element Theory, which calculates aerodynamic forces in real-time from aircraft geometry [19], [23]. X-Plane supports multiplayer functionality for up to 20 users over a local network, using a host-client architecture in which one machine acts as the server, redistributing aircraft state data among all connected clients [24].

Additionally, the simulator provides access to internal variables via DataRefs and the Data Input/Output system, allowing for real-time reading and writing through the User Datagram Protocol (UDP). This makes it particularly suitable for integration with external hardware and sensors [23], [24]. X-Plane also includes features that satisfy certification requirements established by the U.S. Federal Aviation Administration (FAA), allowing its use in FAA-approved training devices [25]. However, such certification depends on the specific combination of hardware and software. In this work, X-Plane is used as a reference for comparison with the open-source simulator FlightGear, which serves as the primary simulation platform.

III. RELATED WORK

Combining constructive and virtual simulations has become important for training and testing human-AI collaboration in air combat scenarios. The concept behind Live–Virtual–Constructive (LVC) simulations is to integrate real pilots, virtual simulators, and computer-generated forces. This allows for more complete training, helping pilots and AI systems practice missions under realistic conditions [26].

Programs like the Air Combat Evolution (ACE), led by DARPA, have worked to integrate autonomous agents into air combat. In the AlphaDogfight Trials, autonomous agents piloted F-16 simulators and defeated human pilots in simulated dogfights, showing how AI can support air combat operations [27].

Other research explores the use of autonomous Uncrewed Aerial Vehicles (UAVs) to fly alongside piloted aircraft. These "loyal wingman" projects aim to create teams where human pilots and AI-controlled aircraft work together in combat [28]. For instance, [29] proposed a hierarchical reinforcement learning approach to enhance the decision-making capabilities of loyal wingmen in complex aerial combat scenarios, demonstrating improved coordination between human pilots and autonomous agents.

More recently, new tools have made it easier to connect flight simulators with machine learning environments. Reference [30] created GymFG, a framework that connects the FlightGear simulator with the OpenAI Gym interface, making it easier to train autonomous agents in realistic flight conditions. Reference [31] developed another framework using JSBSim and Gymnasium to train reinforcement learning models for air combat simulation, facilitating the testing of new strategies and designs.

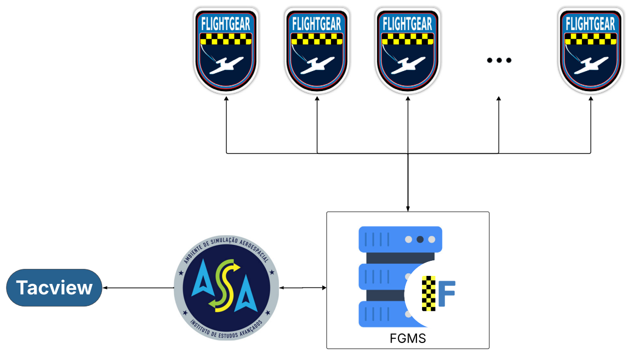


FIGURE 1. AsaFG integration architecture.

While previous studies have focused on integrating autonomous agents into virtual simulations or improving decision-making in loyal wingman scenarios, our work presents an integration of a constructive simulation environment (ASA) with a virtual flight simulator (FlightGear), allowing real-time interaction between human pilots and autonomous systems in air combat scenarios. This integration is important not only for better pilot training but also for helping human pilots and autonomous systems operate together during air combat missions, especially in modern aerial warfare, where radar systems and advanced weapons are key to mission success.

IV. METHODOLOGY

This section describes the methodological steps adopted to integrate the ASA with FlightGear and to validate the resulting system. The approach involved the design and implementation of a communication architecture, adjustments to existing simulation models, and a structured validation strategy that includes technical performance metrics, behavioral consistency across systems, and feedback from operational users. Each of these components is detailed in the following subsections.

A. ARCHITECTURE

The objective of integrating ASA with the flight simulator is to ensure behavioral consistency between both environments. Within the ASA, the aircraft piloted by a human should exhibit the same characteristics as ASA aircraft. Similarly, within the flight simulator environment, the aircraft simulated by the ASA must exhibit behavior that is indistinguishable from that of the simulator's reference aircraft. In essence, the simulator-controlled aircraft will emulate an ASA-native aircraft in the ASA, while ASA-controlled entities will emulate locally simulated aircraft within the flight simulator.

The integration of ASA with X-Plane began as a promising approach. Although X-Plane is not open-source, its use in other virtual simulators within FAB suggests that future integration could be more straightforward. However, specific limitations in X-Plane's multiplayer protocol prevented successful integration. The simulator allows control via DataRefs and the Input/Output system only for the primary simulated aircraft or AI-generated aircraft. These AI aircraft lack DataRefs for weapons systems and do not appear to other participants in multiplayer sessions. Due to these restrictions, the project shifted to using FlightGear. As an open-source platform, FlightGear grants full access to its data packet structure, making it possible to translate ASA data into the FlightGear communication protocol.

Figure 1 presents the architecture for integrating ASA with FlightGear. FGMS coordinates communication between the ASA system and the *n* instances of FlightGear by receiving and distributing data packets. Upon receiving a packet, FGMS forwards it to all connected instances except the sender. Additionally, ASA transmits simulation data to Tacview, a software platform that supports the recording, playback, and analysis of flight data from both simulated environments [32].

B. COMMUNICATION PROTOCOL

Integrating ASA with FlightGear requires clearly defined rules and standards for exchanging data between the two systems and the FGMS server. These rules collectively form the communication protocol.

The FlightGear multiplayer protocol encodes messages using the External Data Representation (XDR) format and transmits them over the network via UDP protocol. It organizes the message into three components: header, position message, and properties [33].

The first two parts of the protocol are well-defined, and the variables within each field remain consistent regardless of the aircraft model used. The main challenge lies in interpreting the values of each property in the third part of the protocol. Since each aircraft model may define a different set of properties, and the FlightGear multiplayer protocol is sufficiently generic to allow developers significant flexibility in choosing which properties to use for transmitting variable values, identifying the variable corresponding to each value can be difficult, especially when the chosen aircraft model lacks proper documentation.

Some properties are clearly defined in the protocol, such as the left aileron position (surface-positions/left-aileron-pos-norm) and the landing gear position (gear/gear[0]/position-norm), among others. However, for many properties, it is not possible to directly determine the variable or subsystem associated with the property's value. One example is sim/multiplay/generic/string[i], where *i* denotes the index of an array. This ambiguity also applies to variables of types such as `float`, `int`, `short`, and `bool` [33].

Unlike FlightGear, which uses a proprietary communication protocol, ASA adopts the Distributed Interactive Simulation (DIS) protocol. DIS provides a standardized infrastructure for integrating heterogeneous simulations, enabling the creation of sophisticated virtual environments for interactive operations. The protocol supports interoperability across systems, technologies, products, and platforms from multiple vendors, encompassing computer-controlled virtual entities, human operators, live systems, and automated simulations. DIS originated from DARPA's SIMNET program, which was developed for real-time distributed combat simulation, and evolved through a series of biannual workshops initiated in 1989 that established industry-wide interoperability standards. These efforts culminated in the publication of multiple IEEE standards for distributed interactive simulation [34].

Since the implementation of the necessary architecture to support the DIS protocol in FlightGear has not been completed [35], ASA handles protocol compatibility during the reception and transmission of messages by adapting them to the FlightGear protocol format. The flowchart illustrating this compatibility process is shown in Figure 2.

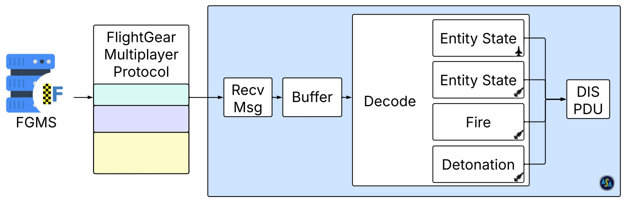


FIGURE 2. Decoding workflow of incoming FlightGear messages.

ASA employs two separate threads to manage protocol compatibility. Since the message transmission frequency in FlightGear may differ from ASA's processing frequency, the first thread is responsible for receiving messages and storing the data in a buffer. The second thread handles message processing, which includes data decoding and the construction of the corresponding Protocol Data Units (PDUs). It is important to note that the encoding and decoding procedures follow the implementation found in the FlightGear source code, as referenced in [36].

The FlightGear multiplayer protocol handles missile data differently from the ASA protocol. Unlike ASA, it does not treat the missile as an independent entity and instead encodes missile-related data as properties of the aircraft. Figure 3 illustrates how ASA transforms each part of the received message. Upon receiving a message from FlightGear, ASA processes the data and may generate up to three types of Protocol Data Units (PDUs). It always generates an Entity State PDU for the aircraft. If the message includes missile data for the first time, ASA creates a Fire PDU. For subsequent messages about the same missile, it generates an Entity State PDU for the missile. If the message indicates a successful hit, ASA issues a Detonation PDU. Due to the

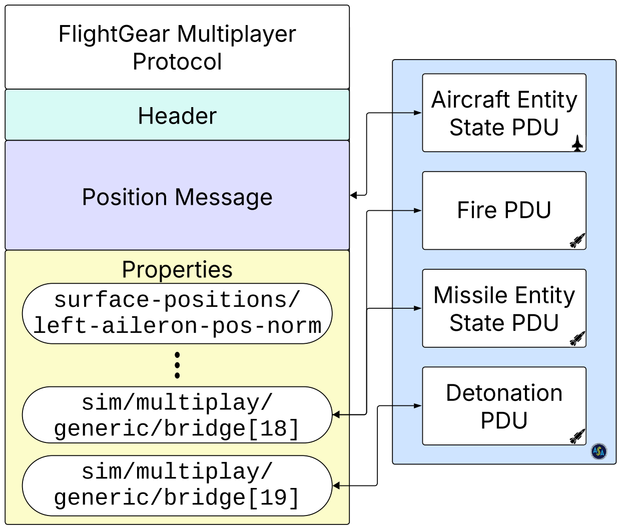

FlightGear Multiplayer Protocol
Header
Position Message
Properties
surface-positions/ left-aileron-pos-norm
sim/multiplay/ generic/bridge[18]
sim/multiplay/ generic/bridge[19]
Aircraft Entity State PDU
Fire PDU
Missile Entity State PDU
Detonation PDU

FIGURE 3. Protocol translation between FlightGear and DIS.

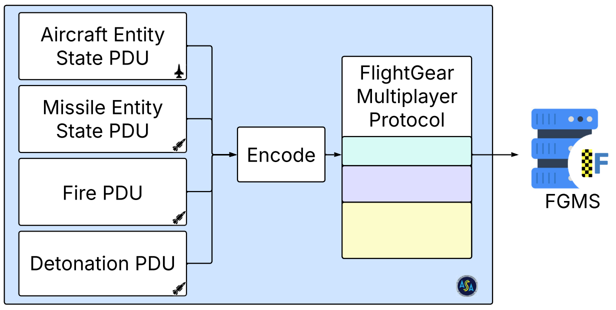


FIGURE 4. ASA-to-FlightGear message encoding process.

arbitrary structure of the selected F-16 model [37], the system transmits missile-in-flight and hit data using the properties sim/multiplay/generic/bridge[i], where *i* equals 18 and 19, respectively.

To avoid losing critical missile-related information, the system stores the most recent aircraft position messages in the buffer shown in Figure 2, along with messages that contain in-flight weapon data or missile hit data for each missile already launched in the simulation. During processing, the system always selects the latest aircraft position message and combines it with the relevant property sections of buffered messages that include missile data. The system does not need to process every position message from the aircraft and missile, since both ASA and FlightGear use dead-reckoning algorithms. These algorithms estimate an entity's location based on known or estimated velocity and course direction over time [38].

Figure 4 illustrates the conversion of PDUs to the FlightGear multiplayer protocol. To generate a message for FlightGear, the process uses data from one to three PDUs. Although it does not transmit the Fire PDU, the system uses its information during the encoding process to organize missile identifiers, source aircraft, and target aircraft. Missile-related messages are stored and appended to the next aircraft position message. Once the message packet is formatted according to the FlightGear multiplayer protocol, it is sent over the network via UDP to the FGMS.

Tables 1, 2 and 3 present the structure of the PDUs populated with data received from FlightGear for integration into the ASA simulation. In most cases, position, orientation, velocity, acceleration, and callsign information is preserved as received or undergoes only a conversion process. Table 1 shows the PDU Header, which contains values either generated within ASA or selected to meet ASA's requirements, since these fields are not provided by the FlightGear protocol.

TABLE 1. Mapping PDU header fields with the FlightGear protocol.

Variable	Value
Protocol Version	Always version 7
Exercise Identifier	ASA exercise value
PDU Type	Assigned by ASA according to Figure 3
Protocol Family	Entity info or warfare
Timestamp	ASA timestamp
Length	Standard length of the PDU
Status	0
Padding	0

TABLE 2. Mapping entity state PDU fields with the FlightGear protocol.

Variable	Value
Entity ID	ID generated by ASA based on the unique FlightGear CallSign
Force ID	Filled by ASA according to the Call-Sign
N of Articulation Parameters	0
Entity Type	[1 2 225 1 3 9 0] — Entity type corresponding to ASA custom aircraft
Alternative Type	Same value as Entity Type
Entity Linear Velocity	Conversion of the value received from FlightGear (body to ECEF)
Entity Location	Value received from FlightGear
Entity Orientation	Value computed by SimGear based on the received message
Appearance	0
Dead Reckoning Algorithm	Default MIXR algorithm FVW_DRM (World, no rotation, 2nd order linear)
Other Parameters [i]	0
Dead Reckoning Entity Linear Acceleration	Value received from FlightGear
Dead Reckoning Entity Angular Velocity	Value received from FlightGear
Entity Marking	CallSign
Capabilities	0

This approach is also applied to fill in missing fields in other PDU types. Some parameters are assigned zero because they are not relevant to the integration's objective, which is to maintain the flight dynamics and enable interaction between

TABLE 3. Mapping detonation PDU fields with the FlightGear protocol.

Variable	Value
Firing Entity ID	ID generated by ASA based on the aircraft's unique FlightGear CallSign
Target Entity ID	ID generated by ASA based on the target aircraft's unique FlightGear CallSign
Munition ID	ID generated by ASA based on the aircraft's unique FlightGear CallSign appended with the missile ID received from FlightGear
Event ID	Generated by ASA
Velocity	Value received from FlightGear
Location	Value received from FlightGear
Burst	[2 9 225 1 14 99 0] — Entity type corresponding to AIM-120 missile used in ASA
Location in Entity Coordinates	Calculated using distance, bearing, and relative altitude received from FlightGear
Detonation Result	Entity impact
N of Articulation Parameters	0
Padding	0

TABLE 4. Mapping of FlightGear multiplayer protocol header fields populated with ASA data.

Variable	Value
Magic	Always 0x46474653 ("FGFS")
Version	2
Msg ID	7 — standard value for Position Data ID
Msg Length	Length of the data
Requested Range (Nm)	0–100
Reply Port	0 — deprecated and ignored
CallSign	Aircraft call sign — maximum length of 8 characters

TABLE 5. Mapping of aircraft position message fields in the FlightGear multiplayer protocol populated with ASA data.

Variable	Value
Model Name	"Aircraft/f16/Models/F-16.xml"
Time	Time of ASA simulation execution
Lag	0.1 — standard value
Position	X, Y, and Z coordinates
Orientation	X, Y, and Z orientation
Velocity	Body velocity
Angular Velocity	Body angular velocity
Linear Acceleration	Constant value of 0 in both incoming and outgoing data
Angular Acceleration	Constant value of 0 in both incoming and outgoing data
Pad	0x1FACE002

entities. It is worth noting that the Fire PDU contains data very similar to the Detonation PDU; therefore, no separate table was created to describe it.

TABLE 6. Mapping of missile position message fields in the FlightGear multiplayer protocol populated with ASA data.

Variable	Value
Position	Missile position
Kind	2 — standard value indicating that the missile is in motion
Secondary Kind	73 — ID of the AIM-120 missile model in FlightGear
Body Velocity (u)	Body velocity along the *u* axis
Heading	Heading of the missile
Pitch	Pitch of the missile
Target CallSign	CallSign of the target aircraft
Flag	1 if the seeker is active, 2 if in acceleration phase, and 3 if both conditions are met
Unique ID	Missile ID

TABLE 7. Mapping of hit message fields in the FlightGear multiplayer protocol populated with ASA data.

Variable	Value
Kind	4 — standard value indicating impact
Secondary Kind	73 — ID of the AIM-120 missile model in FlightGear
Relative Altitude	0.5
Distance	0.0
Bearing	0.0
Target CallSign	CallSign of the target aircraft
Unique ID	Missile ID

In the encoding process illustrated in Figure 4, ASA populates the FlightGear protocol for transmission with the data specified in Tables 4, 5, 6 and 7. The Header (Table 4) contains fixed values obtained either from the FlightGear protocol documentation or from previously received messages, ensuring consistency with the communication standard. Within the position message (Table 5), only the model name, lag, and pad fields remain fixed, while all other fields are dynamically extracted from the ASA simulation. For the missile and hit messages (Tables 6 and 7), the kind and secondary kind fields are configured according to the F-16 model's code documentation. Additionally, the hit message always transmits fixed values for relative altitude, distance, and bearing. This choice reflects the decision to let ASA determine when total damage occurs, while still sending values that ensure the FlightGear model processes the event as complete destruction.

Finally, regarding the remaining FlightGear protocol properties, these are used for visualization purposes or for systems that fall outside the scope of this integration. For such cases, fixed values obtained from a test-phase FlightGear packet are sent. Nevertheless, even when fixed, these values must remain consistent with the flight phase. In this integration, all ASA flights start in cruise phase, which means, for example, that the landing gear must be retracted. Consequently, anyone reusing this code should consider the intended application and review the properties that were not modified in this work.

C. MODELS ADJUSTMENTS

The selected F-16 model [37] includes both combat-related equipment and sensors, as well as missiles configured for short-range combat, specifically within visual range (WVR), commonly known in aviation as dogfighting. In contrast, ASA focuses on beyond visual range (BVR) combat. BVR engagements take place beyond the pilot's line of sight, where decision-making depends on data provided by available sensors and onboard systems [10], [39], [40]. In this scenario, the effectiveness of these systems, combined with the pilot's experience and skills, becomes critical for mission success. For the integration process, the radar, radar warning receiver (RWR), and missile were identified as key components influencing combat outcomes. Therefore, adapting and aligning these models is essential.

Implementing this compatibility brings about several challenges, as the simulation flow in the two software platforms differs significantly. An initial proposal involved simulating these systems on only one platform and transmitting the data to the other. However, this approach risked generating excessive message traffic and would have required changes to the FlightGear protocol. Another alternative explored the possibility of modeling the systems on both platforms to ensure identical behavior, but this approach also proved complex and required considerable effort. Ultimately, the adopted solution focused on making targeted modifications to the existing models to approximate system behavior as closely as possible, while accepting certain differences. The ASA repository [41] provides documentation of all modifications applied to the model.

In this context, several modifications were implemented, including increasing the radar's maximum range from 10 nm to 35 nm and the RWR's detection distance from 2.5 nm to 4.8 nm in FlightGear.

The missile's detonation distance relative to the target aircraft was modified to establish consistent thresholds for total or partial damage across both platforms. Additionally, parameters related to the missile's flight dynamics were adjusted in FlightGear. The FlightGear missile uses a two-stage propulsion model, while the ASA model uses only one stage. To harmonize the models, the second-stage duration was set to 0 s, and the first-stage duration was reduced from 10 s to 8 s, matching the value used in the ASA model. Furthermore, the first-stage thrust (`thrust-lbf-stage-1`) in FlightGear was reduced from 2,700 lbf to 2,200 lbf, a value determined through successive evaluations until the peak velocities of both models were aligned. Most of these changes were necessary because the F-16 model used [37] was originally designed for short-range combat, in contrast to ASA, which focuses on BVR engagements.

D. VALIDATION PROCESS

The validation process was structured to verify both the technical correctness and operational applicability of the proposed integration between ASA and FlightGear. Three complementary strategies were adopted: (i) performance evaluation of the communication mechanism, (ii) comparative testing of missile model behavior across systems, and (iii) user-based assessment involving real fighter pilots. This multi-level validation aimed to ensure that the integrated system is not only functionally accurate but also suitable for practical use in realistic air combat simulation scenarios.

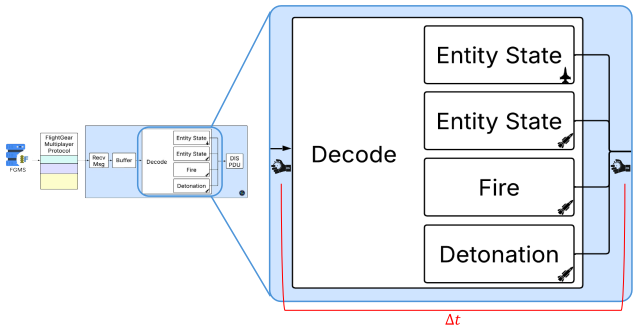


FIGURE 5. **Time measurement of the decoding process.**

1) COMPUTER INTEGRATION METRICS

In addition to evaluating visualization, interaction capabilities, and behavior consistency, the validation process examined the impact of communication latency, focusing on the data encoding and decoding stages. The measurement focused on quantifying the additional latency introduced by the integration during message exchange. This involved calculating the time from when a message in FlightGear format leaves the buffer to when it is sent to ASA in PDU format. This interval represents the time required to perform the decoding steps illustrated in Figure 5. The encoding process was also monitored to evaluate its impact on the overall communication latency.

We focused exclusively on latency as a performance metric because the integration code does not introduce computational complexity that would raise concerns about CPU usage. Memory consumption is also not a limiting factor, since the buffer stores only a single message per simulated aircraft in FlightGear, with each message having a maximum size of 1200 bytes.

2) MISSILE VALIDATION

The adjustments made to the missile model in FlightGear were validated to verify whether the modifications applied to the original FlightGear model (FlightGear-Base) resulted in an adjusted model (FlightGear-Adjusted) that is compatible with the ASA missile model, which has already been validated in several studies [42], [43], [44], [45].

A test suite was established for this validation. Two scenarios were defined for the tests. In the first scenario (SC-1), an aircraft simulated in FlightGear launches missiles toward an aircraft simulated in the ASA. In the second scenario (SC-2), the roles are reversed: the aircraft simulated in the ASA launches missiles against the aircraft simulated in FlightGear. In both scenarios, at the moment of missile launch, the aircraft was flying at an average altitude of 20,000 feet and an average speed of Mach 0.8.

Figure 6 illustrates these scenarios, in which a red aircraft can be observed launching a missile towards a blue aircraft.

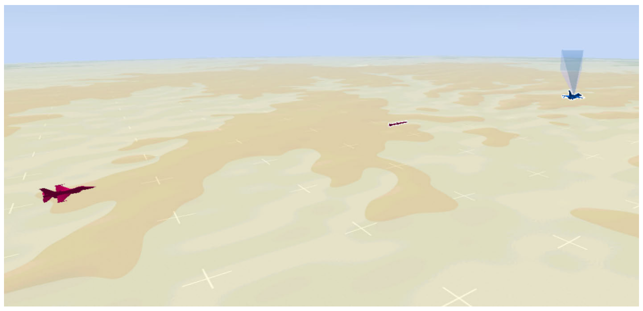


FIGURE 6. Example of a missile launch conducted to validate the adjustments made to the missile model in FlightGear.

The SC-1 scenario was used to collect data from the `FlightGear-Base` and `FlightGear-Adjusted` models. For data collection of the ASA missile model, the SC-2 scenario was employed.

In each of these situations, 10 launches were conducted, and data such as velocity (Mach), roll, and pitch were recorded.

3) USER EVALUATION

To evaluate the proposed integration from a user-centered perspective, a validation process was conducted with five experienced fighter pilots from FAB. The objective was to assess the robustness and usability of the integration module (AsaFG) between ASA and FlightGear during operationally relevant tasks.

Three distinct scenarios were designed to represent key mission profiles:

- **Attack (Q1–Q5):** The pilot must intercept and neutralize an intruding aircraft threatening a designated friendly airspace zone. The scenario simulates defensive counter-air operations, requiring rapid target acquisition and engagement.
- **Evasion (Q6–Q10):** The pilot defends a high-value asset while evading multiple incoming missiles. This scenario highlights the importance of utilizing onboard defensive systems, maintaining situational awareness, and executing evasive maneuvers to survive in a contested environment.
- **Formation (Q11–Q15):** The pilot participates in a low-altitude formation flight composed of four aircraft. The mission involves terrain-following navigation to reduce radar detectability and reach a designated target area for strike operations. Success depends on maintaining formation integrity and synchronized maneuvering.

Figure 7 presents visual representations of the three mission scenarios discussed above.

After each scenario, participants answered five questions evaluating the integration's effectiveness, realism, and usability. At the end of all three missions, an additional set of five general questions (Q16–Q20) was presented to gather broader feedback regarding the integration of ASA and FlightGear. All responses were collected using a five-point Likert-type scale [46], totaling 20 items per participant.

Table 8 presents the full list of questions grouped by category. In the results section, we present the mean and standard deviation of responses for each question, followed by an aggregate analysis per category (Attack, Evasion, Formation, and General). For each group, we also compute a 95% confidence interval (CI) using the t-distribution, which is appropriate when the sample size is small and the population standard deviation is unknown, as recommended in standard statistical analysis practices [47].

This qualitative evaluation complements the technical validation stages and provides practical insights into the operational reliability and user perception of the integrated system. Feedback from the pilots serves as a key indicator of the system's readiness for use in human-in-the-loop simulations and mission planning applications.

V. RESULTS

This section presents the results obtained from the three validation strategies described in Section IV. The first part focuses on the performance of the integration code, analyzing encoding and decoding times between ASA and FlightGear. The second part evaluates the behavior of the missile models to determine the degree of similarity between ASA and FlightGear after adjustments. Finally, the third part provides the results of the user evaluation with FAB fighter pilots, highlighting their perceptions of system integration, realism, and usability across different mission scenarios.

A. COMPUTER INTEGRATION METRICS

Table 9 shows consistently low execution times, particularly for the encoding process involved in sending messages from ASA to FlightGear. This can be attributed to the fact that, unlike decoding (FlightGear to ASA), the encoding step does not involve variable transformations. Instead, it directly maps ASA variables into the FlightGear protocol format, which demands significantly less processing time.

Decoding requires multiple variable conversions to correctly instantiate PDUs with values in the appropriate units, which increases processing time. This added computational effort accounts for the higher average execution times observed in Table 10. Additionally, decoding exhibited a larger standard deviation, likely due to timing inconsistencies in the queuing mechanism. Depending on when a message

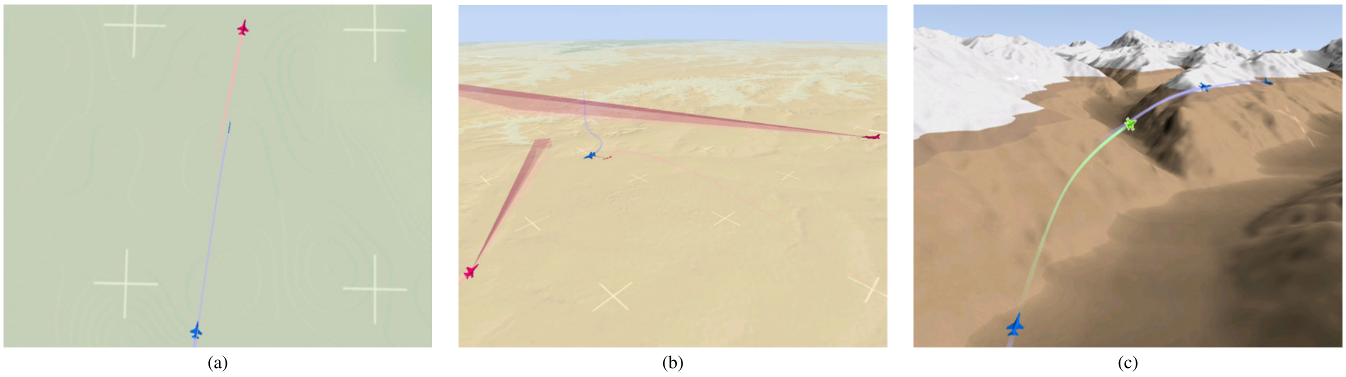


FIGURE 7. Representative frames of the evaluation scenarios: (a) Attack mission involving interception and engagement of an enemy aircraft, (b) Evasion mission with the pilot avoiding multiple threats while radar coverage is illustrated by cones extending forward from each aircraft, and (c) Formation flight at low altitude through mountainous terrain with the human pilot highlighted in green.

enters the queue, it may be processed immediately or delayed until the next processing cycle, introducing variability in execution times.

It is also important to consider that system clock requests can introduce variability into the timing measurements, potentially affecting the accuracy of recorded execution times. Each time a program requests the current timestamp, the call passes through multiple abstraction layers, from user space to kernel space, before reaching the system timer or hardware clock. This process adds overhead that is generally insignificant for long-duration tasks but becomes increasingly relevant when measuring events in the millisecond or microsecond range [48]. Even minimal delays caused by context switching, interrupt handling, or resource contention can affect timing precision and introduce noise into performance measurements.

Tables 9 and 10 show that the execution time for both encoding and decoding processes does not correlate directly with the number of aircraft simulated in either ASA or FlightGear. Despite measurement uncertainties, all observed values remain within acceptable bounds and do not compromise the successful integration of the AsaFG. Since the system transmits data at a frequency of 10 *Hz*, each message is processed with a delay of less than one frame. This level of latency does not affect the simulation's consistency or the user's perception, and it satisfies the performance criteria defined for this integration scenario.

B. MISSILE METRICS

Based on data collected from 30 missile launches using the FlightGear-Base, FlightGear-Adjusted, and ASA models, three variables were selected for the analysis: Mach number, Roll, and Pitch. A graph was generated for each variable. In these graphs, the solid line represents the mean value, whereas the shaded region indicates variance. The horizontal axis represents the time in seconds from the moment of missile launch. In all graphs, the data obtained using the ASA model are shown in blue, FlightGear-Adjusted in red, and FlightGear-Base in yellow.

Figure 8 shows a graph of the Mach number. It can be observed that during the first 8 seconds, the curves exhibited similar behaviors. However, from that point onward, the FlightGear-Base data diverge from those of the other two models. This discrepancy is attributed to the presence of a second propulsion stage in the FlightGear-Base missile model, which is not present in the ASA model. By disabling this stage and calibrating the timing and thrust values of the first stage, the FlightGear-Adjusted curve was obtained, which closely resembled the reference model, ASA.

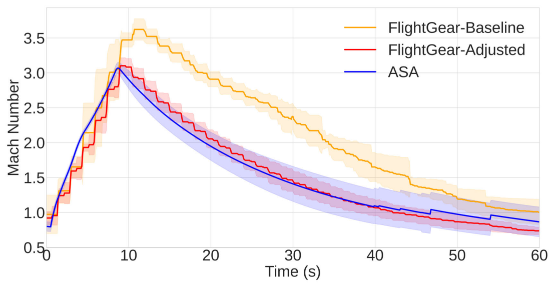


FIGURE 8. Average mach values for missile launches using FlightGear-Baseline (yellow), FlightGear-Adjusted (red) and the ASA model (blue).

Although the FlightGear-Adjusted and ASA curves exhibit similar behaviors, it is noted that in the initial 8 seconds, the velocity of the ASA model is slightly higher than that of the FlightGear-Adjusted model. However, starting from 9 seconds, the adjusted model displays a higher average velocity, which persists until approximately 37 seconds into launch. From that point on, the ASA model once again demonstrated a higher velocity than the adjusted model. The integrals of these curves were calculated to assess the impact of these differences on the distance traveled by the missiles in each model. The values obtained for FlightGear-Base, FlightGear-Adjusted, and ASA were 23.29nm, 17.09nm, and 17.32nm, respectively. The difference in the average distance

TABLE 8. **Evaluation questions answered by pilots, grouped by mission category.**

	Attack
Q1	The cockpit interface (HUD, radar, etc.) provided adequate information for target detection and engagement.
Q2	It was possible to clearly identify the moment of missile launch and its trajectory toward the target.
Q3	During the attack mission, the simulator responded appropriately to control inputs and weapon systems.
Q4	The aircraft's behavior during maneuvers and weapon deployment was consistent with real mission expectations.
Q5	The threat could be easily identified based on the information presented in the simulation environment.
	Evasion
Q6	Visual and auditory alerts were triggered at the expected time for missile evasion.
Q7	I had sufficient data to make timely decisions regarding evasive maneuvers.
Q8	During evasive maneuvers, the aircraft's response was adequate to pilot inputs.
Q9	The adversary missile behavior in the environment appeared realistic and consistent.
Q10	The movement of autonomous aircraft was predictable and coherent with operational logic.
	Formation
Q11	It was easy to visually identify your position and that of other formation members.
Q12	The applied control inputs were sufficient to adjust your position relative to the formation.
Q13	The level of difficulty to maintain position in the formation was compatible with operational simulation expectations.
Q14	The simulated formation flight closely reflected the dynamics of a real operational mission.
Q15	The initial time allocated for briefing on the environment and scenarios was sufficient for me to feel comfortable executing the proposed activities.
	General
Q16	During the three scenarios, the integrated systems (ASA + FlightGear) operated stably, without technical failures or interruptions.
Q17	The visual and interface elements were sufficient to understand and monitor each flight situation.
Q18	The response of the controls and onboard systems was consistent across all scenarios.
Q19	The ASA + FlightGear integration shows potential for use in pilot-involved testing and validation.
Q20	I would recommend the use of this integrated system for future simulation-based studies involving operational training or system evaluation.

TABLE 9. **Encode and decode execution time vs. number of simulated aircraft in ASA.**

ASA Aircraft	Encode ASA to FG (ms)	Decode FG to ASA (ms)
1	$0,31 \pm 0,01$	$46,68 \pm 14,73$
2	$0,29 \pm 0,01$	$47,69 \pm 16,00$
3	$0,29 \pm 0,01$	$55,21 \pm 15,46$
4	$0,29 \pm 0,02$	$46,39 \pm 15,54$

traveled between the FlightGear-Adjusted missile and ASA missile was approximately 1.31%.

TABLE 10. **Encode and decode execution time vs. number of FlightGear aircraft.**

FG Aircraft	Encode ASA to FG (ms)	Decode FG to ASA (ms)
1	$0,31 \pm 0,01$	$46,68 \pm 14,73$
2	$0,31 \pm 0,01$	$52,16 \pm 17,89$
3	$0,31 \pm 0,01$	$51,83 \pm 16,09$
4	$0,32 \pm 0,01$	$49,10 \pm 17,25$

Figure 9 shows the data obtained for the Roll variable. It is noteworthy that the FlightGear-Base model exhibited significant roll variation, particularly during the second propulsion stage (t > 8s). After adjustments to this model, such variation was no longer observed, resulting in the FlightGear-Adjusted model being very similar to the ASA model. Additionally, the scale of the data is worth highlighting: the roll values remain very close to zero, even in the original FlightGear model.

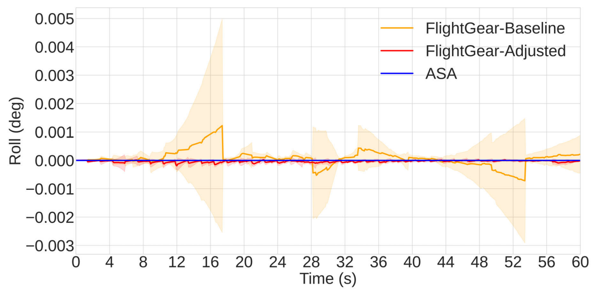


FIGURE 9. **Average roll values for missile launches using FlightGear-Baseline (yellow), FlightGear-Adjusted (red) and the ASA model (blue).**

The Pitch data are shown in Figure 10. This angle represents the upward and downward motions (lofting) of the missile. In the ASA model, this movement was more pronounced than in the FlightGear models. Nevertheless, the overall behavior was similar among models, with the main differences in the intensity and variability of the values observed.

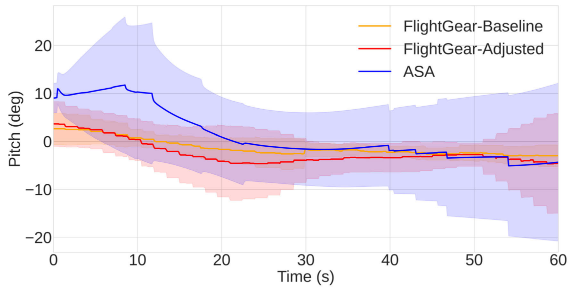


FIGURE 10. **Average pitch values for missile launches using FlightGear-Baseline (yellow), FlightGear-Adjusted (red) and the ASA model (blue).**

C. PILOT EVALUATION METRICS

This subsection presents the evaluation of the ASA and FlightGear integration based on the responses of five experienced fighter pilots from FAB. The evaluation was conducted

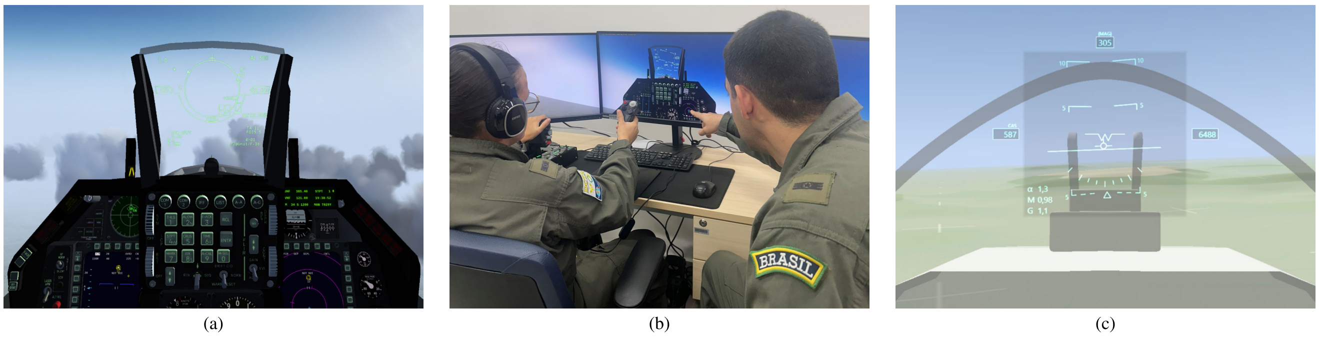


FIGURE 11. Captured views during the user evaluation: (a) Inside the FlightGear cockpit from the pilot's perspective, (b) analyst interaction during system testing, and (c) External TacView view, as seen by the pilot, showing the scenario in real-time.

through hands-on testing and post-scenario questionnaires. Figure 11 illustrates key perspectives from the user evaluation sessions, including the pilot's viewpoint in FlightGear, interaction with an analyst during the test, and external visualization in TacView.

The questionnaire consisted of 20 Likert-scale questions divided into four categories: Attack (Q1–Q5), Evasion (Q6–Q10), Formation (Q11–Q15), and General (Q16–Q20). Table 11 displays the average score and standard deviation for each question based on the five pilot responses. These statistics offer a fine-grained perspective on the perceived performance of the integration across various aspects of simulation realism and interface usability.

TABLE 11. Mean score and standard deviation for each evaluation question answered by the pilots.

Question	Mean Score	Standard Deviation
Q1	4.40	1.14
Q2	3.60	1.14
Q3	4.20	1.10
Q4	4.80	0.45
Q5	4.40	0.55
Q6	3.60	0.89
Q7	4.20	1.10
Q8	2.80	1.10
Q9	4.60	1.14
Q10	4.20	1.10
Q11	4.40	1.14
Q12	4.40	1.14
Q13	4.60	1.14
Q14	4.00	1.00
Q15	4.60	0.55
Q16	4.40	1.14
Q17	4.40	1.14
Q18	4.00	1.00
Q19	4.20	0.45
Q20	5.00	0.00

To summarize the results more concisely, we aggregated the responses by category and computed the mean score, standard deviation, and the 95% CI using the Student's *t*-distribution with four degrees of freedom. This interval indicates the range in which the true average score is expected to lie with 95% confidence.

Regarding the operational scenarios, the highest-rated category was Formation, with an average score of 4.40,

TABLE 12. Summary of pilot evaluations by mission category. Each value is based on five questions answered by five fighter pilots.

Category	Mean Score	Standard Deviation	95% CI
Attack	4.28	0.44	[3.74, 4.82]
Evasion	3.92	0.69	[3.07, 4.77]
Formation	4.40	0.24	[4.10, 4.70]
General	4.40	0.37	[3.94, 4.86]

suggesting that the integration was particularly effective in supporting coordinated flight operations. Pilots consistently reported that maintaining position and interpreting spatial references in formation was easy, indicating strong visual fidelity and consistency in flight dynamics for this scenario.

In the Attack category, the average score reached 4.28, reinforcing that the interface elements and aircraft behavior were generally appropriate for engagement missions. Some variability in responses was observed, possibly due to individual pilot preferences regarding missile guidance or radar behavior.

The Evasion category received the lowest average score, 3.92, and exhibited the widest confidence interval. This indicates a relatively more diverse perception among pilots. Notably, question Q8 — related to aircraft response during evasive maneuvers — had the lowest individual score, averaging 2.80, suggesting that further refinement in evasive control realism or feedback timing could improve user confidence in this context.

The General Evaluation category also reached an average score of 4.40, with relatively low variability across responses. This reflects a stable and positive overall perception of the integrated system's usability, reliability, and potential applicability in pilot-involved experiments.

Overall, the results demonstrate that the ASA-FlightGear integration delivered a robust and credible simulation experience, particularly in formation and general usage contexts, while also highlighting specific areas, such as evasive response fidelity, that could benefit from targeted enhancements.

VI. CONCLUSION AND FUTURE WORK

This work presented AsaFG, a human-in-the-loop integration module that connects the constructive framework ASA with the FlightGear virtual flight simulator. The proposed architecture enables real-time interaction between human pilots and simulated agents in air combat missions, allowing for the evaluation of human-autonomy teaming within realistic operational contexts.

The validation process, which included performance analysis, missile model comparison, and a pilot-centered evaluation, confirmed the functional viability and operational relevance of the integration. The system maintained low latency throughout message processing, and the adjusted missile model showed behavior closely aligned with ASA's native implementation. Pilot feedback was especially positive in formation flight and general integration aspects, supporting the system's use for simulation-based training and operational studies.

Despite these encouraging results, the current version of AsaFG presents limitations that must be addressed to improve realism and applicability. These include the absence of Identification Friend or Foe (IFF) mechanisms, the lack of a communication link between ASA and FlightGear (e.g., data link), and the unavailability of countermeasure systems such as chaff and flare. The system also lacks a web-based interface to support broader accessibility and mission control.

Future work will address these gaps through the implementation of IFF and countermeasure modeling, integration of a data link to enhance information flow between constructive and virtual elements, and support for bomb deployment in FlightGear with corresponding effects on ASA entities, such as damage to buildings or ground targets. Usability will also be improved with the development of a web interface and additional data handling features, such as support for player-generated inputs in FlightGear.

Beyond advancing AsaFG itself, we expect that the methodology presented in this work can serve as a reference for integrating other constructive frameworks with FlightGear. By making the integration code publicly available, we aim to support future initiatives that seek to combine open-source virtual simulators with military simulation environments for research, development, and training applications.

SOURCE CODES

The adjusted FlightGear F-16 model used in this study is publicly available at https://github.com/ASA-Simulation/f16. This version includes refinements to flight dynamics, system behavior, and adjustments to radar, RWR, and missile parameters for BVR combat.

The AsaFG integration module is also publicly available at https://github.com/ASA-Simulation/AsaFG. This repository contains communication scripts, protocol translation logic, and setup instructions, facilitating reproducibility and further development. The module was developed by extending MIXR classes to implement the conversion from the DIS protocol to the proprietary protocol employed by FlightGear. Because it operates independently from ASA classes, it can be applied to other applications utilizing the open-source MIXR framework with little or no adaptation.

REFERENCES

[1] D. J. Allerton, "The impact of flight simulation in aerospace," *Aeronaut. J.*, vol. 114, no. 1162, pp. 747–756, Dec. 2010.

[2] H. Zhang, Y. Wei, H. Zhou, and C. Huang, "Maneuver decision-making for autonomous air combat based on final reward estimation and proximal policy optimization," *Appl. Sci.*, vol. 12, no. 20, p. 10230, 2022.

[3] J. Y. C. Chen and M. J. Barnes, "Human–agent teaming for multirobot control: A review of human factors issues," *IEEE Trans. Hum.-Mach. Syst.*, vol. 44, no. 1, pp. 13–29, Feb. 2014.

[4] P. A. Hancock, D. R. Billings, K. E. Schaefer, J. Y. C. Chen, E. J. de Visser, and R. Parasuraman, "A meta-analysis of factors affecting trust in human–robot interaction," *Human Factors, J. Human Factors Ergonom. Soc.*, vol. 53, no. 5, pp. 517–527, Oct. 2011.

[5] D. D. Hodson, "Military simulation: A ubiquitous future," in *Proc. Winter Simul. Conf. (WSC)*, Dec. 2017, pp. 4024–4025.

[6] J. Berndt, "JSBSim: An open source flight dynamics model in C++," in *Proc. AIAA Model. Simul. Technol. Conf. Exhib.*, Aug. 2004, p. 4923.

[7] W. J. Bezdek, J. Maleport, and R. Z. Olshan, "Live, virtual & constructive simulation for real time rapid prototyping, experimentation and testing using network centric operations," in *Proc. AIAA Model. Simul. Technol. Conf. Exhibit*, Honolulu, HI, USA, 2008, Paper AIAA-2008-7090, doi: 10.2514/6.2008-7090.

[8] J. P. A. Dantas, A. N. Costa, V. C. F. Gomes, A. R. Kuroswiski, F. L. L. Medeiros, and D. Geraldo, "ASA: A simulation environment for evaluating military operational scenarios," 2022, *arXiv:2207.12084*.

[9] FlightGear. (2025). *FlightGear 2020.3*. Accessed: Feb. 12, 2025. [Online]. Available: https://www.flightgear.org/about/policy/

[10] A. N. Costa, J. P. A. Dantas, E. Scukins, F. L. L. Medeiros, and P. Ögren, "Simulation and machine learning in beyond visual range air combat: A survey," *IEEE Access*, vol. 13, pp. 76755–76774, 2025.

[11] P. Sabin. (2012). *Simulating War: Studying Conflict Through Simulation Games*. [Online]. Available: https://www.bloomsbury.com/us/simulating-war-9781472533913/

[12] D. D. Hodson and D. P. Gehl. (2018). *The Mixed Reality Simulation Platform (MIXR)*. [Online]. Available: https://www.mixr.dev/assets/pages/docs/the-mixed-reality-simulation-platform-csc-2018.pdf

[13] J. P. A. Dantas, D. Geraldo, A. N. Costa, M. R. O. A. Máximo, and T. Yoneyama, "ASA-SimaaS: Advancing digital transformation through simulation services in the Brazilian air force," in *Proc. Simpósio de Aplicações Operacionais em Areas de Defesa*, 2023, pp. 1–6. [Online]. Available: https://www.sige.ita.br/edicoes-anteriores/2023/st/235455_1.pdf

[14] J. P. A. Dantas, S. R. Silva, V. C. F. Gomes, A. N. Costa, A. R. Samersla, D. Geraldo, M. R. O. A. Maximo, and T. Yoneyama, "AsaPy: A Python library for aerospace simulation analysis," in *Proc. 38th ACM SIGSIM Conf. Princ. Adv. Discrete Simul.*, New York, NY, USA, Jun. 2024, pp. 15–24, doi: 10.1145/3615979.3656063.

[15] J. P. A. Dantas, M. R. O. A. Maximo, and T. Yoneyama, "Autonomous agent for beyond visual range air combat: A deep reinforcement learning approach," in *Proc. ACM SIGSIM Conf. Princ. Adv. Discrete Simul.*, New York, NY, USA, Jun. 2023, pp. 48–49, doi: 10.1145/3573900.3593631.

[16] J. P. A. Dantas, M. R. O. A. Maximo, and T. Yoneyama, "Autonomous aircraft tactical pop-up attack using imitation and generative learning," *IEEE Access*, vol. 13, pp. 81204–81217, 2025.

[17] J. P. A. Dantas, F. L. L. Medeiros, A. R. Samersla, P. L. R. Botelho, V. C. F. Gomes, S. R. Silva, Y. D. Ferreira, A. O. Arantes, M. R. C. Aquino, and M. R. O. A. Maximo, "Deep reinforcement learning agents with collective situational awareness for beyond visual range air combat," *IEEE Access*, vol. 13, pp. 143052–143069, 2025.

[18] J. P. A. Dantas, M. R. O. A. Maximo, A. N. Costa, D. Geraldo, and T. Yoneyama, "Machine learning to improve situational awareness in beyond visual range air combat," *IEEE Latin Amer. Trans.*, vol. 20, no. 8, pp. 2039–2045, Aug. 2022. [Online]. Available: https://latamt.ieeer9.org/index.php/transactions/article/view/6530

[19] X-Plane. (2025). *X-Plane 12*. Accessed: Apr. 16, 2025. [Online]. Available: https://www.x-plane.com/

[20] Microsoft Corporation and Asobo Studio. (2020). *Microsoft Flight Simulator*. [Online]. Available: https://www.flightsimulator.com

[21] M. Basler et al. *The FlightGear manual*. 2024. Accessed: Feb. 12, 2025. [Online]. Available: https://www.flightgear.org/support/manual/

[22] FlightGear. (2019). *DARPA Picks Teams for Virtual Air Combat Competition*. Accessed: Feb. 13, 2025. [Online]. Available: https://www.darpa.mil/news/2019/virtual-air-combat-competition

[23] A. Bittar, H. V. Figuereido, P. A. Guimaraes, and A. C. Mendes, "Guidance software-in-the-loop simulation using X-plane and simulink for UAVs," in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, May 2014, pp. 993–1002.

[24] X-Plane. (2024). *X-Plane 12 Desktop Manual*. Accessed: Apr. 16, 2025. [Online]. Available: https://www.x-plane.com/manuals/desktop/

[25] X-Plane. (2025). *FAA-Certified X-Plane*. Accessed: Apr. 16, 2025. [Online]. Available: https://www.x-plane.com/pro/certified/

[26] C. Best and B. Rice, "Science and technology enablers of live virtual constructive training in the air domain," *Air Space Power J.*, vol. 32, no. 4, pp. 59–73, 2018. [Online]. Available: https://www.airuniversity.af.edu/ASPJ/Display/Article/1669937/science-and-technology-enablers-of-live-virtual-constructive-training-in-the-air/

[27] C. R. DeMay, E. L. White, W. D. Dunham, and J. A. Pino, "Alphadogfight trials: Bringing autonomy to air combat," *Johns Hopkins APL Tech. Dig.*, vol. 36, no. 2, pp. 154–163, 2022.

[28] J. P. A. Dantas, M. R. O. A. Maximo, and T. Yoneyama, "Loyal wingman assessment: Social navigation for human-autonomous collaboration in simulated air combat," in *Proc. 38th ACM SIGSIM Conf. Princ. Adv. Discrete Simul.*, New York, NY, USA, Jun. 2024, pp. 61–62, doi: 10.1145/3615979.3662149.

[29] J. Zhang, D. Wang, Q. Yang, S. Zhuoyong, J. Longmeng, S. Guoqing, and W. Yong, "Loyal wingman task execution for future aerial combat: A hierarchical prior-based reinforcement learning approach," *Chin. J. Aeronaut.*, vol. 37, no. 5, pp. 462–475, May 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1000936124000839

[30] A. Wood, A. Sydney, P. Chin, B. Thapa, and R. Ross, "GymFG: A framework with a gym interface for FlightGear," 2020, *arXiv:2004.12481*.

[31] A. Salhi, J. E. Jabour, T. L. Arnolds, J. E. Ross, and H. R. Dozier, "Leveraging JSBSim and gymnasium: A reinforcement learning approach for air combat simulation," in *Communications in Computer and Information Science*, L. Zhou, F.-Y. Wang, and A. M. Madni, Eds., Cham, Switzerland: Springer, 2025, pp. 271–283.

[32] Tacview. (2025). *The Universal Flight Data Analysis Tool*. Accessed: Mar. 5, 2025. [Online]. Available: https://www.tacview.net/

[33] FlightGear. (2022). *Multiplayer Protocol*. Accessed: Feb. 18, 2025. [Online]. Available: https://wiki.flightgear.org/Multiplayer_protocol

[34] *IEEE Standard for Distributed Interactive Simulation (DIS)—Communication Services and Profiles*, IEEE Standard 1278.2-2015, 2015, pp. 1–42.

[35] FlightGear. (2014). *Distributed Interactive Simulation*. Accessed: Feb. 19, 2025. [Online]. Available: https://wiki.flightgear.org/Distributed_Interactive_Simulation

[36] FlightGear. (2025). *FlightGear Source Code*. Accessed: Feb. 20, 2025. [Online]. Available: https://sourceforge.net/p/flightgear/flightgear/ci/next/tree/src/MultiPlayer/tiny_xdr.cxx

[37] E. Hofman and N. Chr. (2025). *F-16 Fighting Falcon*. Accessed: Feb. 20, 2025. [Online]. Available: https://github.com/NikolaiVChr/f16

[38] G. Oguntala, R. Abd-Alhameed, S. Jones, J. Noras, M. Patwary, and J. Rodriguez, "Indoor location identification technologies for real-time IoT-based applications: An inclusive survey," *Comput. Sci. Rev.*, vol. 30, pp. 55–79, Nov. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574013718301163

[39] J. P. A. Dantas, A. N. Costa, D. Geraldo, M. R. O. A. Maximo, and T. Yoneyama, "Engagement decision support for beyond visual range air combat," in *Proc. Latin Amer. Robot. Symp. (LARS), Brazilian Symp. Robot. (SBR), Workshop Robot. Educ. (WRE)*, Natal, Brazil, Oct. 2021, pp. 96–101.

[40] J. P. A. Dantas, A. N. Costa, F. L. L. Medeiros, D. Geraldo, M. R. O. A. Maximo, and T. Yoneyama, "Supervised machine learning for effective missile launch based on beyond visual range air combat simulations," in *Proc. Winter Simul. Conf. (WSC)*, Singapore, Dec. 2022, pp. 1990–2001.

[41] (2025). *Ambiente De Simulação Aeroespacial (ASA)*. Accessed: Feb. 20, 2025. [Online]. Available: https://github.com/ASA-Simulation/f16.git

[42] J. P. Dantas, A. N. Costa, D. Geraldo, M. R. Maximo, and T. Yoneyama, "PoKER: A probability of kill estimation rate model for air-to-air missiles using machine learning on stochastic targets," *J. Defense Model. Simul.*, Jan. 2025, Art. no. 15485129241309675, doi: 10.1177/15485129241309675.

[43] A. R. Kuroswiski, A. S. Wu, and A. Passaro, "Optimized prediction of weapon effectiveness in BVR air combat scenarios using enhanced regression models," *IEEE Access*, vol. 13, pp. 21759–21772, 2025.

[44] J. P. A. Dantas, D. Geraldo, F. L. L. Medeiros, M. R. O. A. Maximo, and T. Yoneyama, "Real-time surface-to-air missile engagement zone prediction using simulation and machine learning," in *Proc. Interservice/Ind. Training, Simul. Educ. Conf. (I/ITSEC)*. Orlando, FL, USA: National Training and Simulation Association (NTSA), Nov. 2023.

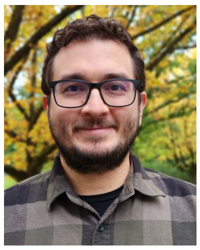
[45] J. P. A. Dantas, A. N. Costa, D. Geraldo, M. R. O. A. Maximo, and T. Yoneyama, "Weapon engagement zone maximum launch range estimation using a deep neural network," in *Intelligent Systems*. Cham, Switzerland: Springer, 2021, pp. 193–207.

[46] R. Likert, "A technique for the measurement of attitudes," *Arch. Psychol.*, vol. 140, pp. 1–55, May 1932.

[47] D. C. Montgomery and G. C. Runger, *Applied Statistics and Probability for Engineers*, 7th ed., Hoboken, NJ, USA: Wiley, 2020.

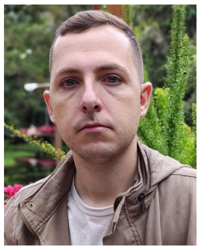
[48] D. Tsafrir, Y. Etsion, D. G. Feitelson, and S. Kirkpatrick, "System noise, OS clock ticks, and fine-grained parallel applications," in *Proc. 19th Annu. Int. Conf. Supercomputing*, Jun. 2005, pp. 303–312. [Online]. Available: https://dl.acm.org/doi/10.1145/1088149.1088190

SAMARA R. SILVA received the B.Sc. degree in aeronautical science from Brazilian Air Force Academy (AFA), in 2013, and the degree in computer engineering from the Aeronautics Institute of Technology (ITA), Brazil, in 2022, where she is currently pursuing the M.Sc. degree. She is a Researcher at the Institute for Advanced Studies, Brazilian Air Force. Her research interests include data science, artificial intelligence, machine learning, and simulation.

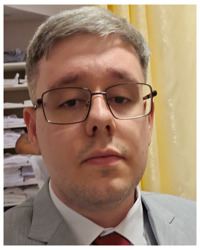


VITOR C. F. GOMES received the B.Sc. degree in computer science and the M.Sc. and Ph.D. degrees in applied computing from the National Institute for Space Research (INPE), Brazil, in 2009, 2012, and 2023, respectively. Currently, he is a Researcher with the Institute for Advanced Studies, Brazilian Air Force. His research interests include distributed computing, geospatial big data, and data science.

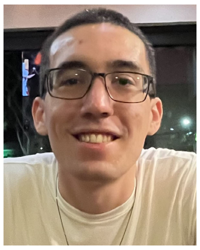
ALESSANDRO O. ARANTES received the B.Sc. degree in computer science and the M.Sc. and Ph.D. degrees in applied computing from the National Institute for Space Research (INPE), Brazil, in 2000, 2008, and 2016, respectively. Currently, he is a Researcher with the Institute for Advanced Studies (IEAv), Brazilian Air Force. His research interests include applied computing, software testing, and data science.



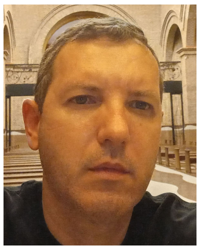
ANDRE F. M. CAETANO received the B.Sc. and M.Sc. degrees in computer science from São Paulo State University (UNESP), Brazil, in 2013 and 2017, respectively, and the D.Sc. degree in electrical and computer engineering from the Aeronautics Institute of Technology (ITA), Brazil, in 2023. In 2024, he was a Postdoctoral Researcher at the University of Campinas (UNICAMP), focusing on digital twins for photovoltaic systems. In 2025, he joined the Institute for Advanced Studies (IEAv) as a Researcher. He is currently with Brazilian Air Force. His research interests include cloud computing, big data, distributed systems, and modeling and simulation.



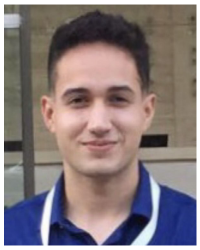
VICTOR L. D. B. COSTA received the B.Sc. degree in mechanical-aeronautical engineering from the Aeronautics Institute of Technology (ITA), Brazil, in 2024. During his undergraduate studies, he worked in the areas of image-based navigation with AI and tactical simulation analysis at the Institute for Advanced Studies, supporting projects related to Brazilian Air Force. He is currently a Research Engineer with the Institute for Advanced Studies. His research interests include simulation, finite elements, and parallel processing.



ADRISSON R. SAMERSLA received the B.Sc. degree in computer engineering from the Aeronautics Institute of Technology (ITA), Brazil, in 2021, and the M.Sc. degree from the Graduate Program in Electronic and Computer Engineering, ITA, in 2022. He is currently a Researcher with the Institute for Advanced Studies (IEAv), Brazilian Air Force. His research interests include high-performance computing, machine learning, robotics, and simulation.

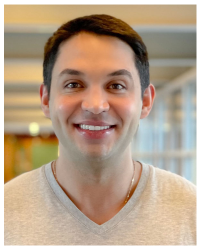


FELIPE L. L. MEDEIROS received the B.Sc. degree in computer science from the Federal University of Ouro Preto, Ouro Preto, Brazil, in 1999, and the M.Sc. and Ph.D. degrees in applied computing from the National Institute for Space Research, São José dos Campos, Brazil, in 2002 and 2012, respectively. He has been working in the field of artificial intelligence, with an emphasis on metaheuristics, autonomous agents, and simulation.



YURI D. FERREIRA received the B.Sc. degree in aerospace engineering from the Aeronautics Institute of Technology (ITA), Brazil, in 2021. Currently, he is a Researcher with the Institute for Advanced Studies, Brazilian Air Force. His research interests include reinforcement learning and its applications.

MARCIA R. C. AQUINO received the B.Sc. degree in computer science from the Federal University of Juiz de Fora, Brazil, in 1997, and the M.Sc. degree in applied computing from the National Institute for Space Research, São José dos Campos, Brazil, in 2005. She is currently a Researcher with the Institute for Advanced Studies (IEAv), Brazilian Air Force. Her research interests include data science, artificial intelligence, machine learning, simulation, and project management.



JOAO P. A. DANTAS received the B.Sc. degree in mechanical-aeronautical engineering from the Aeronautics Institute of Technology (ITA), Brazil, in 2015, and the M.Sc. degree from the Graduate Program in Electronic and Computer Engineering, ITA, in 2019, where he is currently pursuing the Ph.D. degree. In 2015, he participated in a year-long exchange program at Stony Brook University, USA. In 2022, he was a Visiting Researcher at the AirLab, Robotics Institute, Carnegie Mellon University. He is a Researcher with the Institute for Advanced Studies, Brazilian Air Force. His research interests include machine learning, robotics, and simulation.

Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) - ROR identifier: 00x0ma614