Dual Pilot Active Sidestick Demonstrator 2PASD
Flexible rapid prototyping platform for research flight simulation

Alexej Dikarew
Research Associate

Mario Müllhäuser
Research Associate

Tanja Martini
Research Associate

German Aerospace Center (DLR)
Braunschweig, Germany

ABSTRACT

The application of flight simulators is a standard practice in aerospace research. At the German Aerospace Center (DLR), these tools are employed to validate systems prior to their integration into the research helicopter ACT/FHS. Recently, the development toolchain for the experimental system of ACT/FHS was augmented by the addition of the Dual Pilot Active Sidestick Demonstrator (2PASD) simulator. Originally designed for research on active inceptors, 2PASD was expanded to replicate the experimental system of ACT/FHS. The integration of a MATLAB/Simulink development environment with 2PASD's real-time simulation framework enables the simulator to serve as a rapid prototyping platform for flight control systems development. Furthermore, 2PASD's visual system was enhanced by the incorporation of Virtual Reality and Mixed Reality environments, thereby broadening its applicability in new research contexts. Additionally, latency measurements were conducted using a non-invasive setup to identify sources of time delay between pilot inputs and the response of the visual system. The results provide a baseline for optimizing the simulator's performance.

NOTATION

Acronyms

2D	two-dimensional
2PASD	Dual Pilot Active Sidestick Demonstrator
3D	three-dimensional
ACS	Active Control System
ACT/FHS	Active Control Technology/Flying Helicopter Simulator
AVES	Air VEhicle Simulator
CDF	Common Data Format
CDU	Control Display Unit
CLS	Control Loading System
COS	Core System
CS-FSTD(H)	Certification Specification for Helicopter Flight Simulation Training Devices
DLR	German Aerospace Center
EC	Experiment Computer
ECC	Experiment Co-Computer
EKF	Extended Kalman Filter
EP	Experimental Pilot
FCS	Flight Control System
FLI	First Limit Indicator
FOV	Field Of View
GC	Graphics Computer
GUI	Graphical User Interface
IC	Interface Computer
ICD	Interface Control Document
LCD	Liquid Crystal Display
MR	Mixed Reality
MRC	Mixed Reality Computer
NAV	Navigation Control System
PFD	Primary Flight Display
ROS	Robot Operating System
ROSC	Robot Operating System Computer
SCM	System Control Module
SDK	Software Development Kit
TCP	Transport Control Protocol
UDP	User Datagram Protocol
VE	Virtual Environment
VR	Virtual Reality

INTRODUCTION

Flight simulators are a common tool used by researchers for studying various aspects related to flight. These simulators replicate the experience of flying an aircraft, allowing for controlled experiments to investigate pilot performance under specific tasks or conditions. Moreover, flight simulators can also be employed to test new technologies, for example pilot assistance systems, in a safe and controlled environment. For research facilities that conduct experiments with real aircraft, flight simulators provide a valuable tool to evaluate new sys-

tems before testing them in real flight.

At the German Aerospace Center (DLR), the main platform for research on helicopters is the Active Control Technology/Flying Helicopter Simulator (ACT/FHS) (Ref. 1). The aircraft is a strongly modified EC135 helicopter equipped with a fly-by-wire system that allows to evaluate experimental software with extensive authority to the flight control system. Research conducted with the ACT/FHS covers a wide range of topics, including flight control and pilot assistance, such as basic stabilization, autopilot functions and path planning, as well as the investigation of active control inceptor systems. Therefore, in addition to conventional control inceptors, it is equipped with active sidesticks on the cyclic and collective control axes.

Preparation of flight tests with the aircraft as well as simulator evaluations in general are carried out at the Air VEhicle Simulator (AVES) facility (Ref. 2). This simulator features a dome projection system, motion platform and multiple cockpit mockups that resemble DLR's research aircraft such as the Airbus A320 and Airbus Helicopters EC135. The AVES also includes a replication of the flight experimental system of the ACT/FHS with identical hardware to replicate the behavior of the research aircraft together with the experimental software during simulation.

The software for the experimental flight control functions is being developed in MATLAB/Simulink in a desktop environment. To run the software in AVES, a code generation toolchain is used that translates the code to C/C++ and then compiles binaries for the target hardware. While this provides hardware-in-the-loop and software-in-the-loop simulation capabilities, it has limited possibilities for software debugging compared to the desktop simulation. On the other hand, the desktop simulation environment lacks in terms of interacting with the software during execution as it does not provide control inceptors or a visual environment of the simulation.

The Dual Pilot Active Sidestick Demonstrator (2PASD) (Ref. 3) was added to the development toolchain to bridge the gap between desktop simulation and real-time simulation. 2PASD's original use case was in developing algorithms for active control inceptors, such as tactile cueing (Refs. 4, 5). However, its capabilities were recently expanded by combining the flexibility of modifying software in MATLAB/Simulink with the possibility of running a real-time simulation with the simulator mockup. To achieve this, the existing 2PASD simulation framework was significantly modified and extended to allow for direct integration and debugging of software within the MATLAB/Simulink development environment.

Another new application for 2PASD is the usage of Virtual Reality (VR) and Mixed Reality (MR) headsets as an alternative to the original visual system. New interfaces were created to integrate such hardware into the simulation framework. As the VR/MR headsets are being used both in 2PASD and AVES, the interfaces were designed to provide compatibility between both simulators.

This paper presents the expansion of the 2PASD simulator towards a flexible rapid prototyping platform for research flight simulations. It provides an overview of the general architecture and software framework, followed by the integration of a replicated experimental system from the ACT/FHS, the development of a debugging environment for tactile cueing functions, and the creation of a VR/MR simulation environment. The simulator's real-time performance is evaluated by measuring latency between pilot inputs and the visual response. The paper concludes with a summary of the findings.

SIMULATOR OVERVIEW

2PASD is a fixed base simulator featuring two pilot stations, each equipped with cyclic and collective sidesticks and pedal inceptors, see Figure 1. Five TV screens are used as the primary vision system, supplemented by additional monitors displaying cockpit instruments. The operator station for configuration and simulation control is located behind the simulator. The pilot stations allow for flexible arrangement due to their separate bases. Besides the typical side-by-side setup, for example single pilot or tandem configurations are possible. The following sections detail the hardware setup, specifically focusing on the Control Loading System (CLS) and the visual system. Additionally, the simulation framework used to operate the 2PASD simulator is discussed in depth.

Figure 1: 2PASD simulator

Control Loading System

Each of the two pilot stations is outfitted with two sidesticks and one set of pedals, as depicted in Figure 2. Both sidesticks provide two degrees of freedom for longitudinal and lateral stick displacement. The righthand sidestick serves for cyclic control, while the lefthand sidestick is used for collective control. It offers an additional degree of freedom, enabling yaw control as an alternative to the pedals. Overall, the CLS provides ten control inceptor axes, five per pilot station.

2PASD implements active inceptors that are provided by the manufacturer Wittenstein motion control GmbH. Their main mechanical characteristics are listed in Table 1. These

Figure 2: Active inceptors

Figure 3: View into the cockpit

active inceptors generate control forces through integrated electromechanical actuators and enable arbitrary force-feel-characteristics to be defined via user and data interfaces. For instance, the global force-deflection curve can be defined via lookup tables. Additionally, detents, gates, or stick shakers can be specified or the friction and damping of the inceptor can be modified. This allows for replicating the force-feel-characteristics of various aircraft. Also, pairwise coupling of control axes is possible, to replicate a physical connection between them, known as "electronic rod". This was demonstrated by previous research carried out with 2PASD (Refs. 6, 7). Furthermore, the active inceptors can be utilized for tactile cueing (Ref. 8), which is an ongoing research topic at DLR. Development and evaluation of tactile cueing for helicopters was the driver for creating 2PASD and is one main application of the simulator, see Section Tactile Cueing Functions Development.

Parameter	Sidesticks	Pedals
Travel range	±18.5°	±18°
Nominal force	155 N	535 N
	(@ grip ref. 170 mm)	
Deflection rate	121 °/s	126 °/s

Table 1: CLS mechanical characteristics

Both sidestick grips are identical to those used at the Experimental Pilot (EP) seat of the ACT/FHS. The grip of the right hand sidestick replicates the original EC135 cyclic grip, while the left hand grip was specifically manufactured for the ACT/FHS active sidestick. Both grips are equipped with various switches such as force trim release, beep trim and switches that can be used arbitrarily.

Visual System

2PASD's primary vision system consists of five Liquid Crystal Display (LCD) TV screens with a screen diagonal size of 55 inch (139 cm) each, operating at a resolution of 1920 × 1080 with a image refresh rate of 60 Hz. These screens are arranged in an arc around the pilot stations, see Figure 3.

The center screen is placed in landscape orientation while the four outer screens are placed in portrait orientation. This arrangement covers a Field Of View (FOV) of 150° × 50° (horizontal × vertical) and is plotted in Figure 4.

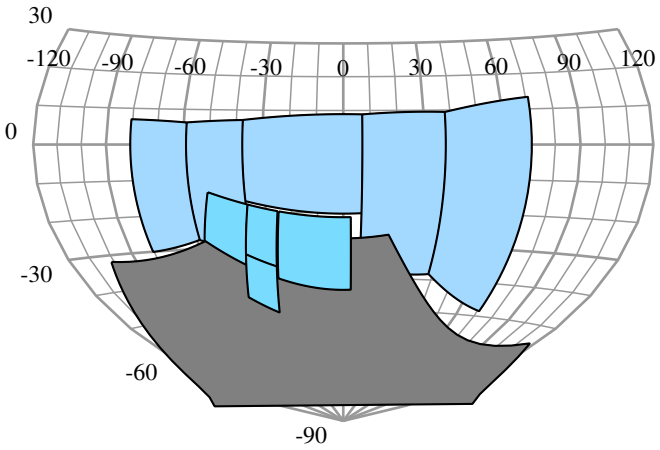


Figure 4: Field of view from right seat

To display flight instruments, a set of screens is mounted on the pilot stations in front of the seats. A 17 inch (34 cm) screen with a resolution of 1280×1024 at 60 Hz is placed in front of each pilot seat, typically displaying a Primary Flight Display (PFD). A smaller 10 inch (25 cm) screen with a resolution of 768 × 1024 at 60 Hz is placed between the two larger screens and displays the First Limit Indicator (FLI), i.e. engine limit gauge. A second 10 inch screen is placed underneath the FLI display and provides indicators and touch buttons to monitor and control the state of the simulation. This allows the user to start and stop the simulation, trim the control inceptors and enable or disable pairwise coupling between the pilot stations directly in the cockpit.

Simulation Framework

The 2PASD simulation framework connects the CLS and visual system hardware with the software modules required to run the flight simulation, such as the flight dynamics model. This framework is implemented as a distributed system with

multiple computers, see Figure 5. 2PASD is based on the simulation framework 2Simulate (Ref. 9), which was initially developed for AVES. Using the same simulation framework enables the exchange of software modules between both simulators.

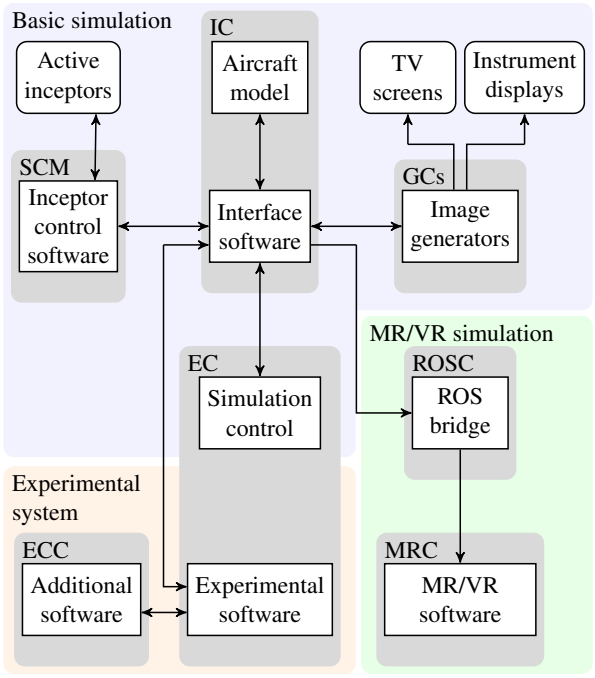


Figure 5: 2PASD system architecture

The 2Simulate framework provides a software library for real-time scheduling of simulation tasks and inter-process communication. A central task, called the interface software, connects various software modules running on different computers. In 2PASD, the interface software is executed on the Interface Computer (IC), which runs on the real-time operating system QNX Neutrino 7, see Fig. 5. The IC software defines different simulation tasks that are executed at configurable frame rates. For the communication with other modules, each task can establish a network connection using an User Datagram Protocol (UDP) driver provided by the 2Simulate library.

For the basic simulation setup, the IC is connected to the System Control Module (SCM), which runs the control software for the active inceptors. The stick forces and displacements provided by the pilots are processed on the SCM and transferred to the IC. Besides the interface software, the IC also runs the aircraft model, implemented with the 2Simulate framework. This model receives pilot commands from the interface software and returns the aircraft response. The model is implemented with C++ and resembles the dynamics of the ACT/FHS. It is the same model being used in AVES and includes models for the main rotor with dynamic inflow, tail rotor, fuselage, engine and flight control actuators (Ref. 10). The aircraft response from the model is fed to the interface software and sent from the IC to three Graphics Computers (GCs), that render the simulation view and drive the visual system of the simulator. One GC supplies the center TV screen and instrument displays while the other two GCs each supply two TV screens.

The user may configure the simulation with the simulation control software, which is executed in a desktop environment on the Experiment Computer (EC) behind the simulator cockpit, see Figure 1. This software includes a Graphical User Interface (GUI) that allows for remotely powering the computers needed for simulation and starting different software modules on the remote computers. A build history of software modules is kept available to allow the user to recreate previous simulation setups.

In addition to the standard visual system, 2PASD allows users to use MR/VR headsets to render the visual environment by configuring the interface software to send the aircraft model's output to the Robot Operating System Computer (ROSC), which translates received UDP datagrams to Robot Operating System (ROS) topics. The ROS messages are then transferred to the Mixed Reality Computer (MRC) running the MR/VR software toolchain. Details of the MR/VR setup and toolchain are described in Section Mixed and Virtual Reality Setup .

To include the ACT/FHS experimental system in the simulation, the experimental software is executed on the EC and the interface software is configured to send inceptor signals from the SCM to the experimental software. The flight control algorithms in the experimental software calculate actuator commands and send them to the interface software, which feeds them to the aircraft model. Additional software for the experimental system, such as path planning algorithms for automatic guidance, is executed on the Experiment Co-Computer (ECC). The architecture and implementation of the experimental system are described in detail in the next section.

EXPERIMENTAL SYSTEM

The ACT/FHS flight control system comprises two main components: a base system, referred to as the Core System (COS) and an experimental system. The COS includes the pilot control inceptors, a redundant fly-by-wire architecture and electromechanical actuators at the main and tail rotors. This system allows pilots to fly the bare airframe helicopter without any experimental flight control functions. Furthermore, it incorporates a switch that facilitates communication between the COS and the experimental system, allowing for the transmission of pilot commands to the experimental system and receipt of actuator commands from it.
The experimental system consists of a set of computers integrated into the aircraft and connected to the COS. It allows for implementation of experimental flight control software (Ref. 11).

The ACT/FHS experimental system software is divided into two distinct components: system software and user software. The system software, implemented in C, handles scheduling, data acquisition and communication with the COS. In contrast, the user software, developed using MATLAB/Simulink, implements the experimental flight control algorithms. A

code generation toolchain translates the user software into C code and integrates it with the system software for execution.

The user software consists of multiple tasks that run in parallel to each other, including those responsible for basic flight control (Flight Control System - FCS), path following control (Navigation Control System - NAV), and control of active sidesticks (Active Control System - ACS).

To evaluate user software code in real-time simulation, a replication of the ACT/FHS experimental system is integrated into AVES. This configuration utilizes identical hardware and the same code generation toolchain as the original system to facilitate hardware-in-the-loop simulation.

An additional replication of the ACT/FHS experimental system has been integrated into 2PASD. To complement the abilities of the AVES simulation, this setup allows for low-effort and rapid changes to the user software. Instead of translating the MATLAB/Simulink user software with the code generation toolchain, this implementation enables the user to directly modify and debug their software within the MATLAB/Simulink editor. As such, the experimental system software replication for 2PASD was implemented in MATLAB/Simulink and integrated with the simulator.

MATLAB/Simulink Implementation

The 2PASD replication of the experimental system consists of a Simulink model running on the EC that implements the core functionality of the original system. The model's main components and signal flow are depicted in Figure 6. This section elaborates on the functions of these components and details their implementation.

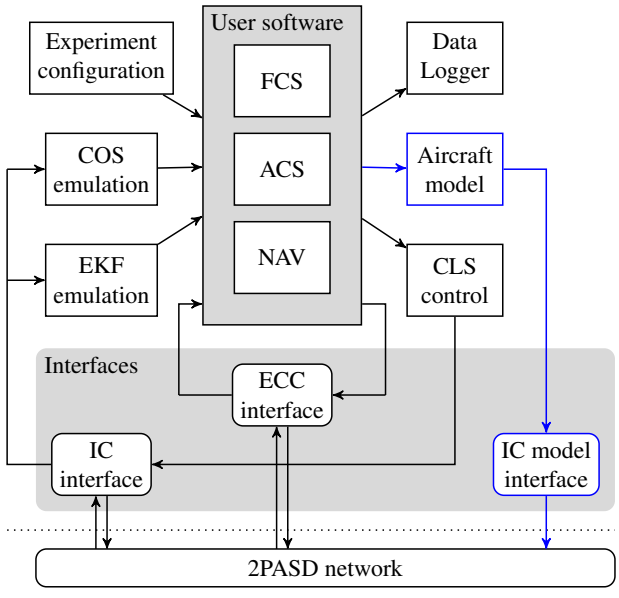


Figure 6: 2PASD experimental system

Experiment Configuration

The user can configure the tasks of the user software utilizing the experiment configuration block, which replicates the Control Display Unit (CDU) used in the aircraft with a set of Simulink masks providing a simple GUI.

COS Emulation

In contrast to the original experimental system, the 2PASD framework does not include a COS. Instead, a COS emulation block is added to the experimental system implementation, replicating the basic functions for mode handling and stick trimming.

EKF Emulation

The original experimental system relies on an Extended Kalman Filter (EKF) to estimate the aircraft state from filtered and fused sensor data. In 2PASD, as there are no sensor data available, the aircraft state is provided by the aircraft model, thereby necessitating an EKF emulation block to generate outputs identical to those of the original EKF for the user software.

Data Logging

To store flight test results, the original experimental system implements a data logging function utilizing the CDU. In 2PASD, the data logging function is implemented using a MATLAB function embedded within a Simulink block. The user can select from two data format options: Common Data Format (CDF) or MATLAB's binary format (MAT). The signal labels correspond to those of the original experimental system, enabling users to process data generated in 2PASD with the same tools used for analyzing data from ACT/FHS or AVES.

CLS control

The CLS exhibits significant differences compared to the ACT/FHS. In the aircraft, the primary control inceptors do not provide active functions, but rather a set of active sidesticks is added to the cockpit when necessary. Conversely, 2PASD provides a single set of control inceptors with active functions available on all axes. The CLS control block accounts for these different inceptor setups and abstracts the 2PASD CLS for the user software providing interfaces identical to those in the original experimental system. For utilization of active functions, the block includes a translation layer that converts generic tactile cueing elements into hardware specific commands, see also Section Tactile Cueing Functions Development.

Interfaces

The ACT/FHS experimental system supports to run software on other computers, for example the ECC, in addition to the experimental tasks that are executed on the EC, see Figure 5. This additional software typically covers tasks like path planning, processing of optical sensor data, implementing extra displays, etc. The experimental system utilizes UDP connections for communication with the additional computers.

To replicate network communication in 2PASD, the UDP driver from the 2Simulate framework (Section Simulation Framework) is implemented. Simulink's code wrapper, called S-Function (Ref. 12), is used to include the driver, which is written in C++. The S-Function provides a Simulink block that calls the compiled C/C++ code at runtime. A tool was developed that automatically generates the Simulink block for a specific UDP connection based on a Interface Control Document (ICD). Besides handling communication with the ECC, a Simulink-UDP block is also used to provide the experimental system's main interface with the IC, see Figure 5.

Model Referencing

The different tasks of the ACT/FHS experimental user software (FCS, ACS, NAV) were originally implemented as stand-alone Simulink models. In 2PASD, they are included into the top-level experimental system model as Simulink reference models. Reference models act as links to other models, allowing for code reuse (Ref. 13). By including the user software tasks as reference models, 2PASD uses the same code base as the desktop development environment and the code generation toolchain. This enables any modifications made to the user software 2PASD to be directly transferred to AVES or ACT/FHS.

Real-Time Execution

In its default configuration, Simulink will execute a model as fast as possible, i.e. when it has finished the calculation for one time step, it will immediately start with the next one. However, this is not feasible for real-time simulation. Simulink offers a function to slow down the simulation, this is called pacing (Ref. 14). For real-time execution in 2PASD, the pacing is configured such that 1 s of simulation time matches 1 s of elapsed time on the machine running Simulink. This setup ensures that the model execution rate remains synchronized with the system's clock, thereby enabling real-time execution. If the computational resources available on the machine running Simulink exceed the simulation demands, the real-time requirement will be met. For the 2PASD experimental system it was found possible to execute the model in real-time on a recent work station computer with Windows operating system; for the machine's specification see Table 2. Experience showed the setup to be feasible in the context of research flight simulation even in the absence of a dedicated real-time operating system.

Component	Value
Processor	Intel i7-9700 @ 3 GHz
Memory	16 GB DDR4
Graphics card	NVIDIA GeForce GT 730
Operating system	Microsoft Windows 10 Enterprise
MATLAB version	R2020a

Table 2: Specification of the Experiment Computer running the 2PASD experimental system

To decrease a model's execution time, Simulink provides accelerated modes (Ref. 15). Instead of executing interpreted code as in normal mode, the accelerator mode compiles the code before execution. This speeds up the simulation but increases initialization time when editing the software, as the code needs to be compiled every time it is changed. Also, signal values cannot be viewed during simulation when executing in accelerator mode. The simulation mode (normal vs. accelerated) can be set for an entire model or separately for referenced models. For the 2PASD experimental system, this allows the user to set the mode separately for the individual user software tasks and benefit from extended debugging tools in normal mode and utilize higher performance in accelerator mode. Typically, a user will modify one software task at a time and will therefore run this task in normal mode. By executing the remaining tasks in accelerator mode, the overall model initialization time is decreased and more computational resources are available for debugging the task of interest, e.g. viewing signal time histories.

Aircraft Modelling

In addition to replicating features of the ACT/FHS experimental system, the 2PASD experimental system implements the ability to execute the aircraft flight dynamics model directly in Simulink. For this use case, the standard aircraft model is disabled (see Figure 5) and the interface software is configured to receive the flight state from an additional UDP interface in the experimental system, as depicted by the highlighted blocks and signals in Figure 6. This allows for development of flight dynamics models directly in Simulink and provides the user to debug and evaluate models in hands-on flight.

Summary

The replication of the ACT/FHS experimental system with a Simulink implementation significantly extends the usability of the 2PASD simulator. It provides core functionalities of the original implementation and enables users to develop experimental flight control software in the simulator environment. Utilization of Simulink features like model referencing for code reuse and accelerator mode for performance increase together with plug-and-play network interfaces make it a flexible and robust platform for simulator software development. The new features strongly extend the debugging capabilities compared to the existing desktop environment and AVES simulation, complementing the existing toolchain.

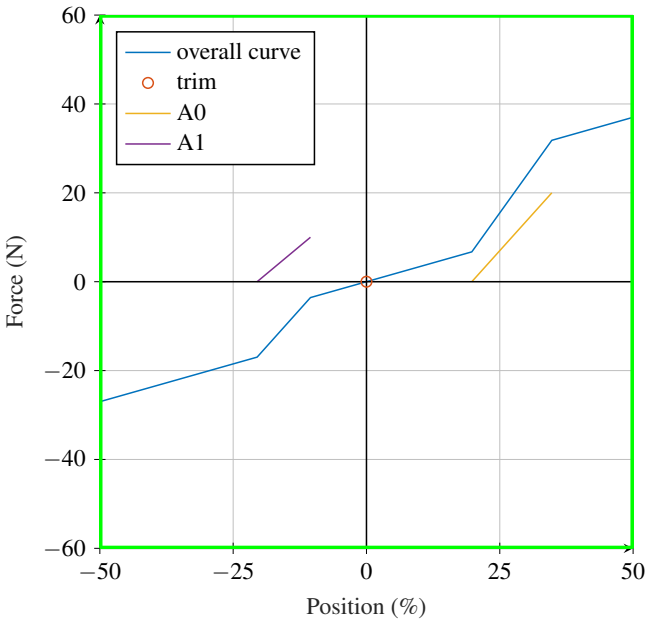
TACTILE CUEING FUNCTIONS DEVELOPMENT

One of the main applications of 2PASD is the development of tactile cueing functions. This involves the algorithms required for calculation of inceptor limit positions as well as the design of tactile cueing force patterns.

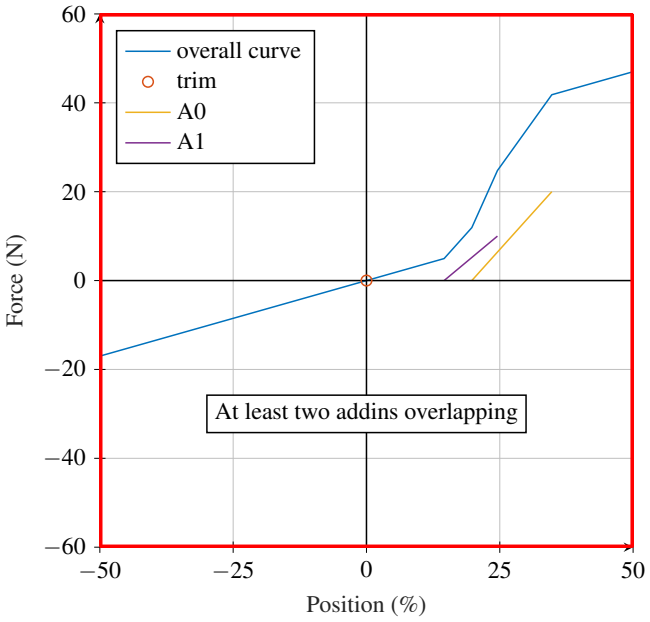
To effectively realize tactile cueing design concepts, it is essential to translate them into suitable active inceptor configurations. This process typically introduces a trial-and-error approach, as it requires to translate a desired feel into abstract parameters. Also, several parameters might need to be combined for the active inceptor to behave in the desired way. Moreover, specific hardware constraints must be considered to ensure correct operation. In some cases, users may struggle to distinguish between undesirable behavior resulting from inadequate abstraction or undetected constraint violations.

To facilitate the translation of tactile cueing patterns into hardware-specific parameters, an abstraction layer was implemented into the CLS control module, see also Figure 6. This layer enables the mapping of common cues, such as softstops, onto hardware-specific configurations. A softstop is a local force gradient used to cue a specific stick position, that e.g. corresponds to a vehicle limit. Here, the generation of a softstop at a specific position with desired intensity and width requires the selection of appropriate parameters. For the 2PASD CLS, this is achieved through the utilization of so called addins. An addin is a manufacturer specific solution to realize local and temporary modifications of the main curve. It replaces a part of the main curve as an inset. Its position, shape, force intensity and width can be configured freely.

To simplify the debugging process and accelerate design iterations, an emulation of the 2PASD inceptor hardware was implemented using Simulink. This emulation is based on documented hardware constraints and observed responses to various parameter inputs. The emulator instantaneously generates a force deflection curve that would be realized by the inceptor for a given parameter set, while also performing sanity checks to detect incorrect parameter inputs. Figure 7 illustrates examples of correct and incorrect input configurations. In this example, two addins A0 and A1 are used to define a negative and a positive softstop. The system does not allow for configuration of addins overlapping each other, therefore the inceptor control software ignores any overlapping addins provided by the user. In the upper graph, the configuration is correct and the resulting overall curve would be accepted and reflected by the real CLS. The lower graph shows a test case with overlapping addins. The error is detected by the sanity check, turning the plot's frame to red and displaying an error message on the canvas. The CLS emulation can be used standalone or in combination with the real CLS hardware.



(a) CLS emulation: correct



(b) CLS emulation: failure

Figure 7: CLS emulation output: commanded elements and resulting overall curve

MIXED AND VIRTUAL REALITY SETUP

The training of helicopter pilots is essential to ensure safe helicopter operations. In addition to training with real aircraft, simulation is already an integral part of helicopter pilot training. The availability of VR and MR headsets has opened up new possibilities in aviation training, making it possible to recreate real-life scenarios in a controlled, safe and cost-effective environment (Figure 8). Unlike full flight simulators with dome projection systems, VR/MR simulators have

the potential to be small, require less maintenance and may be mounted on mobile platforms for easy displacement.

Virtual Reality and Mixed Reality are defined according to (Ref. 16) within the Reality-Virtuality Continuum. In this context, VR, respectively Virtual Environment (VE), refers to a synthetic environment in which a user is fully immersed, whereas the real world is limited by physical laws. MR, on the other hand, blends elements from both the real and virtual worlds, allowing for various combinations and interactions.

Mixed Reality Headset

For the 2PASD MR setup, the Varjo XR-3 Focal Edition (Ref. 17), monitor-based headset with a FOV of 115° and 95° (horizontally $\times$ vertically), is used. The focal distance ranges from 30 cm to 80 cm, mirroring the typical distance between a pilot's eyes and the flight instruments situated within the cockpit. The image refresh rate is 90 Hz. The utilization of the Varjo XR-3 headset relies on the use of the Varjo Base Software, a proprietary application provided by the manufacturer.

In 2PASD, chroma key is used to create an immersive MR experience with the Varjo XR-3 headset. Chroma key, also known as green screen techniques, is a visual-effect method for layering two separately created images together based on color hues.

Figure 8: 2PASD Mixed Reality setup

Hardware Setup

Due to 2PASD's flexible cockpit arrangement, either single or dual pilot cockpits can be utilized for the MR experience. In Figure 9 an example of a MR setup is shown for a single pilot setup. The pilot's seat is surrounded on all sides by the green screen. Other structures, such as the beams for the flight instrument displays, are covered with additional green fabric so that they are not displayed in the VE. For the head tracking, two HTC Vive Basis Tracking stations (Ref. 18) are used and are mounted on top of the green screen providing an unobstructed view towards the pilot seat. An additional light source was installed to provide bright illumination.

Figure 9: Green screen for Mixed Reality setup

Rendering Software

In order to generate the VE, two softwares were utilized: Unity and Prepar3D.

Unity is a game engine capable of rendering both two-dimensional (2D) and three-dimensional (3D) games for various platforms, making it a popular choice among game developers, architects, and designers. Within Unity, the virtual environment can be created as required, allowing for the customization of 3D helicopter cockpits, an example is illustrated in Figure 10.

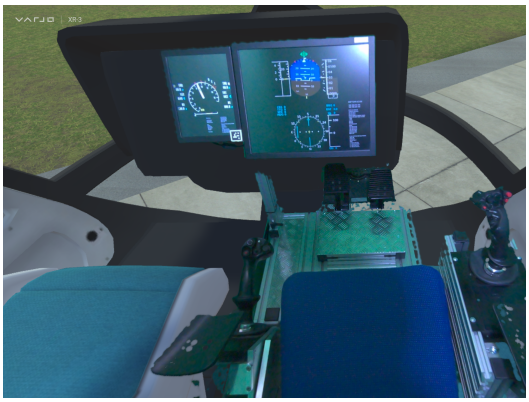


Figure 10: Cockpit view with MR setup

Prepar3D is a visual flight simulator software. It simulates a wide range of aircraft in various environments, including airports, mountains, and coastal areas. Additionally, it features realistic weather conditions, atmospheric effects, and other environmental factors. Prepar3D implements the SimConnect Software Development Kit (SDK) allowing for modifications of 3D objects and data exchange with external software such as the 2PASD simulation framework.

While both Unity and Prepar3D can be run independently, each providing its own VE, it is also possible to combine them. For instance, an adapted helicopter cockpit can be rendered in Unity and displayed as an overlay within the rendered scenery of Prepar3D. The combination of these features has the potential to create highly immersive and realistic VR experiences that simulate real-world environments and scenarios.

Data Flow for Displaying Mixed Reality

In order to enable the dynamic rendering of VR imagery in response to pilot inputs, it is necessary to obtain and integrate flight state data from 2PASD's simulation framework (Figure 5).

To enable communication between 2PASD and the MRC rendering the image on the headset, the Robot Operating System is being used (see Figure 11). Initially developed for the purpose of creating robotics applications, ROS offers a comprehensive suite of software libraries and tools. ROS uses a message-based communication system, where software modules called nodes exchange messages with each other. Nodes are small programs performing specific tasks. In this case, ROS is employed as a communications middleware between the simulator and the MRC to transmit simulation data, i.e. the helicopter state vector calculated by the aircraft dynamics model.

The ROS bridge receives the helicopter flight state from the interface software and translates it to a ROS message. The ROSC transmits the message to the MRC where it is processed by the ROS Transport Control Protocol (TCP) Connector, which is integrated with Unity. The ROS TCP Connector enables sending and receiving ROS messages within Unity and therefore, it is possible to render the helicopter position and orientation within Unity accordingly to the flight state data. Furthermore, Varjo provides a dedicated plugin for Unity, specifically designed for integration with Varjo's technology. The execution of a Unity application is managed through Varjo Base Software, which enables the rendering of the VR content on the MR headset.

Prepar3D also receives the flight state data via the same ROS message. A receiver node is integrated into the SimConnect simulation framework of Prepar3D as an Addon ROS subscriber, which enables the provision of the flight state data as input for the helicopter position and orientation within the virtual environment (VE). When initiating the Prepar3D application with VR settings, the MR experience is launched through Varjo Base Software.

The Varjo Base Software incorporates the multi-application feature enabling the concurrent execution of multiple applications, thereby allowing for the simultaneous display of, for instance, a Unity-based helicopter cockpit as an overlay within the Prepar3D scenery.

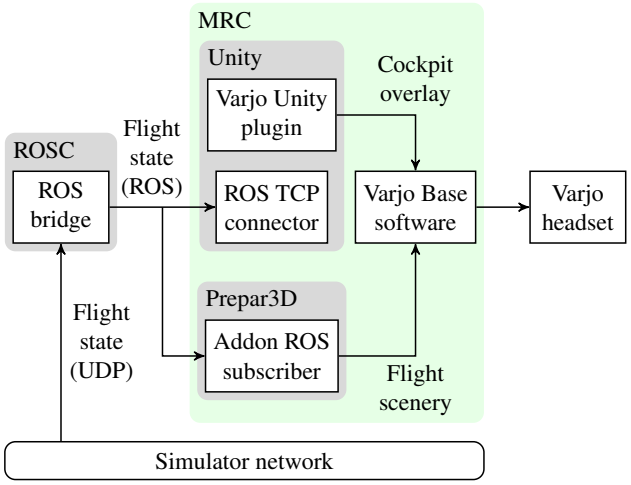


Figure 11: MR/VR system architecture

2PASD advantages and disadvantages for MR/VR

The development of MR/VR applications with 2PASD has several advantages in contrast to a full flight simulator like AVES:

- The cockpit's flexible arrangement enables experimentation with diverse pilot configurations, such as single or dual pilot scenarios.
- Clear space around the cockpit allows for setting up ample green screen environments, thereby ensuring optimal head tracker functionality and minimizing shadows cast by lighting systems.
- The setup provides a reproducible test environment for experiments or pilot campaigns, characterized by uniform illumination conditions and the absence of reflective surfaces.
- The facility allows for exploring alternative spatial arrangements for head tracking and lighting.
- The ability of rapidly entering and exiting the cockpit and the possibility of implementing experiments directly from a desktop-based setup facilitates efficient testing.

Nevertheless, the flexible simulation setup of 2PASD introduces some limitations:

- The absence of a fully equipped cockpit with physical switches and buttons reduces the possibilities for haptic pilot interaction and thus limits the experience in terms of realism and immersion.

- There is no motion platform available for examining the interaction between the MR headset's visual functionality and motion cues.
- The 2PASD currently features only a simple 3D model. For a more detailed cockpit an advanced model would be necessary.

AVES advantages and disadvantages for MR/VR

In contrast to 2PASD, the AVES simulator setup offers following advantages:

- The fully equipped EC135 cockpit offers significant advantages in terms of pilot interaction with the cockpit, providing a more realistic and immersive experience.
- The motion platform featured in AVES may enhance the overall immersion of the flight simulation, making it feel even more like actual flying.
- It is not required to create a 3D model of a helicopter cockpit for rendering within the VE.

However, the following challenges are associated with AVES:

- Forced proximity of the head tracking stations to the headset due to the compact cockpit hinders optimal performance, as e.g. line of sight between base stations may be obscured.
- The cockpit window glass may interfere with the base station's signals, introducing another challenge in installing head tracking technology.
- Additional light sources can introduce unintended reflections caused by the windows within the cockpit, compromising the overall immersion and realism of the simulation.

Summary

The 2PASD simulator exhibits dual functionality, serving as both an effective development platform and a helicopter simulator suitable for piloted campaigns. Furthermore, the outputs generated through implementation on the 2PASD system can be directly applied to the AVES system, thereby reducing overall development time and allowing for the utilization of advantages inherent in both simulators.

LATENCY MEASUREMENT

One key parameter for evaluating the real-time performance of a flight simulator is the latency, i.e. the time delay between a pilot input and the output reaction from visual, auditory, or motion systems. The Certification Specification for Helicopter Flight Simulation Training Devices (CS-FSTD(H)) defines an acceptable latency for full-flight simulators to be less than 100 ms (Ref. 19). The latency of a flight simulator can be attributed to multiple sources. Firstly, the computational time consumed by software components, such as processing pilot input, calculating aircraft response, and rendering the virtual environment generates time delay. Additionally, network communication transport delays and asynchronous software execution on distributed machines contribute to the overall latency. To assess the performance of 2PASD, its end-to-end time delay was measured.

Measurement method

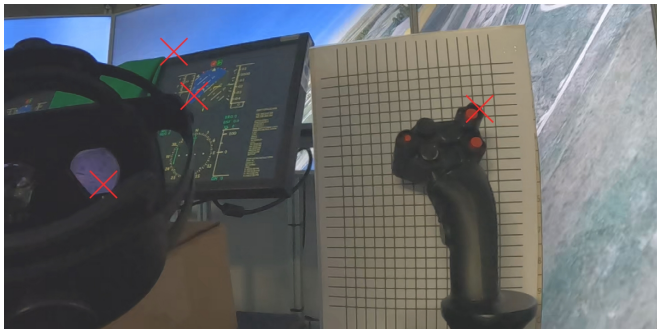
An end-to-end time delay measurement was conducted between control inceptor inputs and the visual system's reaction. A non-invasive setup was employed to minimize software modifications, allowing for the evaluation of both the standard visual system utilizing TV screens and the setup incorporating the VR/MR headset. A high-speed camera was utilized to capture images of control inceptor movements and visual system responses.

To generate reproducible inputs, the cyclic sidestick was configured to generate lateral step inputs. To obtain a distinct visual response to this input, not influenced by aircraft model dynamics, the model response was overwritten by a step signal-like response in the roll axis. The output roll angle ϕ responds to a change of the stick position δ_y with

$$\phi = \begin{cases} -45^\circ & \text{for } \delta_y < -0.1\,\% \\ 0^\circ & \text{for } -0.1\,\% < \delta_y < 0.1\,\% \\ 45^\circ & \text{for } \delta_y > 0.1\,\% \,. \end{cases} \tag{1}$$

By applying this relation, the visual system generates a distinct and reproducible response for small stick displacements, easily identifiable in the recorded video images.

To determine the time delay for individual vision system components from the image stream, an analysis tool was developed using MATLAB. The user selects points in the image exhibiting strong changes in color or brightness during the step response. Figure 12 displays camera images recorded during the measurement, with the upper image (12a) showing the idle state and the lower image (12b) showing the state after a step input. The cyclic stick is visible in the right half of the image, exhibiting a lateral displacement to the left in the lower image due to the step input. In the left portion of the images, the VR/MR headset can be observed, oriented such that its displays are facing the camera. The headset is configured to display a simple horizon, showing either blue color when above the horizon or orange color when below. In the background, the PFD display and the TV screens are visible. One can clearly identify the distinct rotations against the horizon in both the simulated environment and the instrument display. The red markers indicate four characteristic points selected by the user for analysis. The image processing tool operates on a frame-by-frame basis, capturing timestamps when the brightness of the marker on the stick increases significantly, indicative of the stick's movement to the left. The tool then proceeds to step through the video, capturing individual timestamps and



(a) Initial state

(b) State after step response

Figure 12: Time delay analysis

calculating time delays for the other markers when the pixel color changes significantly, i.e. when the step response is detected.

The camera utilized for data acquisition operates at a frame rate of $f = 240\,\text{Hz}$, resulting in a temporal resolution of the measurement of $t_r = 1/f = 4.2\,\text{ms}$. Due to the possibility that both the stick's initial movement and the visual system's step response may occur at any time between consecutive frames, the worst-case accuracy is determined to be $t_a = 2t_r = 8.4\,\text{ms}$.

Setup and results

The time delay measurement was carried out to identify the time delay associated with different simulation setups. The experiments were performed using the two simulation modes: the basic simulation mode and the simulation with experimental system enabled. Additionally, the measurement included both operating modes of the VR/MR headset. Each simulation setup was evaluated by recording at least 50 step responses to allow for statistical analysis and robustness evaluation. The signal flow differed between the basic simulation mode and the experimental system mode. Figure 13 illustrates the signal routing for the basic simulation mode.

In this configuration, the step input generated by the inceptor is processed by the SCM and sent to the IC. The interface software routes the signal to the aircraft model which responds with the step response defined in (1). The step response is then fed back to the interface software, which distributes it to the various components of the visual system, as indicated by color highlighting in the plot. The signal is transmitted to the image generators, which render both the images for the instrument displays and the simulated environment displayed on the TV screens. Furthermore, the image is sent to the ROSC, which translates UDP datagrams into ROS messages and sends them to the the MRC for rendering the image for the VR/MR headset.

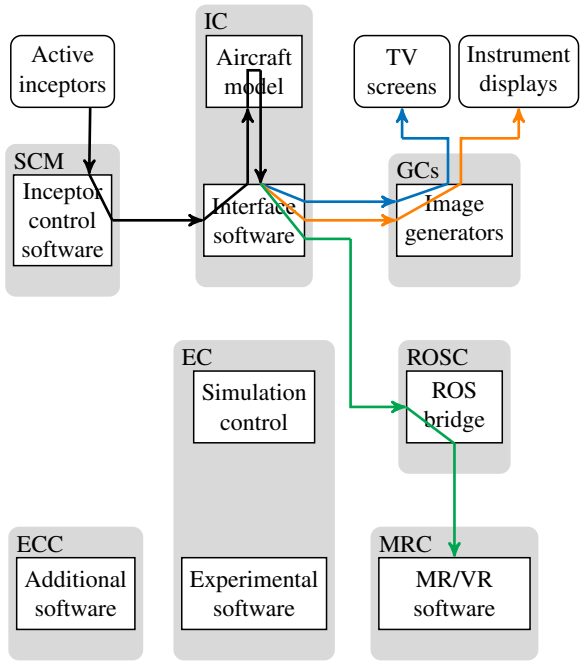


Figure 13: Step signal flow in basic simulation setup

This basic simulation setup was evaluated for both the MR and the VR configurations of the headset, yielding four end-to-end time delay values. These results are depicted in Figure 14. As observed in the plot, the latency for the TV screens exhibits the highest value, with a median of 154 ms. In contrast, the instrument display and MR headset configuration demonstrate significantly shorter time delays of 92 ms, followed by VR configuration with 100 ms.

Notably, the comparison between the TV screen latency and instrument latency suggests that rendering the simulated environment requires at least 50 ms, assuming both systems receive the step input simultaneously. This finding is reasonable, considering the signal path is identical for both systems. In contrast to the TV screens, the VR/MR headset presents the image significantly faster, despite a longer signal path. This difference in time delay may be caused by different display technologies used in the LCD TV screens and the VR/MR headset and/or longer processing time of the TV screens image generator software compared to the VR/MR rendering software. To minimize time delay introduced by the TV screens, they are set to "gaming mode", a configuration provided by many TVs disabling image enhancement algorithms in favor of increased responsiveness. The GC's hardware used for image rendering is strongly outdated compared to the MRC's hardware. An upgrade to recent components may reduce the overall latency of the TV screen setup.

The second measurement configuration incorporates the experimental system into the simulation, as depicted in Fig-

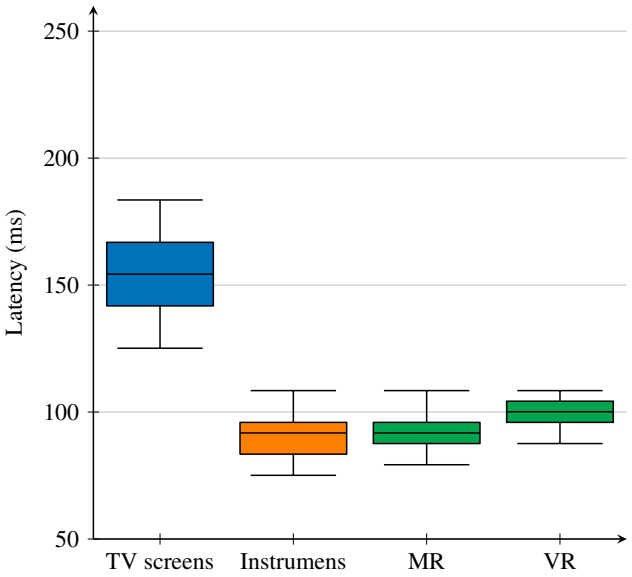


Figure 14: Time delays in basic simulation setup

ure 15. Again, input from the inceptor is processed by the SCM and sent to the IC. The interface software then routes the signal to the EC, where it is received by the experimental system implemented in Simulink (Section MATLAB/Simulink Implementation). The experimental system, typically executing a flight controller algorithm, is configured to return the signal without modification, thereby preserving the original shape of the step input. The signal is then sent back to the IC and routed to the aircraft model. Subsequently, the signal flow follows the same path as in the previous measurement setup: the aircraft model calculates a step response according to (1), and the interface software distributes the response to the various components of the visual system.

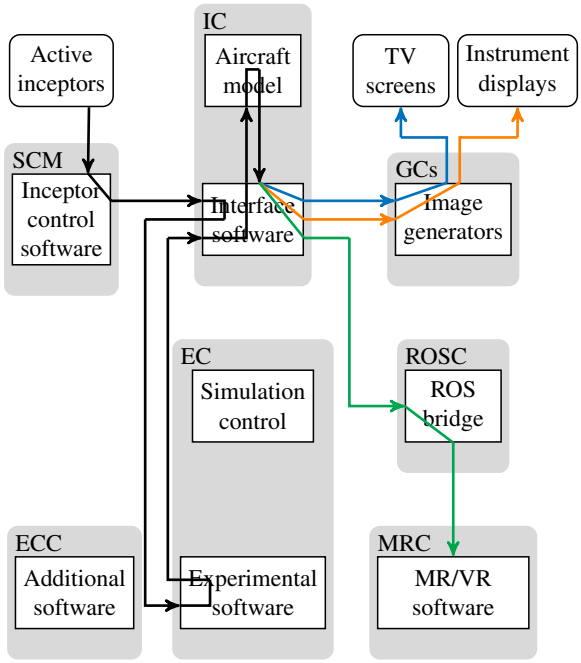


Figure 15: Step signal flow with experimental system enabled

The results of the time delay measurements conducted with the experimental system enabled are depicted in Figure 16. As before, this measurement was performed for both VR and MR configurations. Again, the time delay of the TV screens exhibit the highest median time delay of 219 ms, while the instrument display introduces a delay of 154 ms. The VR headset configuration shows a time delay of 167 ms, whereas the VR headset configuration demonstrates a time delay of 158 ms.

Notably, the end-to-end latency for all four signal paths is increased by 58 ms to 75 ms compared to the first setup. Given that the simulator configuration remains identical to the first measurement, except for the addition of the experimental system, one can infer that the time delay introduced by the experimental system itself is approximately 67 ms.

Opposed to the first measurement with experimental system disabled, the time delay for the MR configuration is slightly higher than the VR configuration. As the signal route for VR and MR is the same and is not altered by the experimental system, this indicates that the analysis tool used for evaluating the camera images introduces some variation to the results.

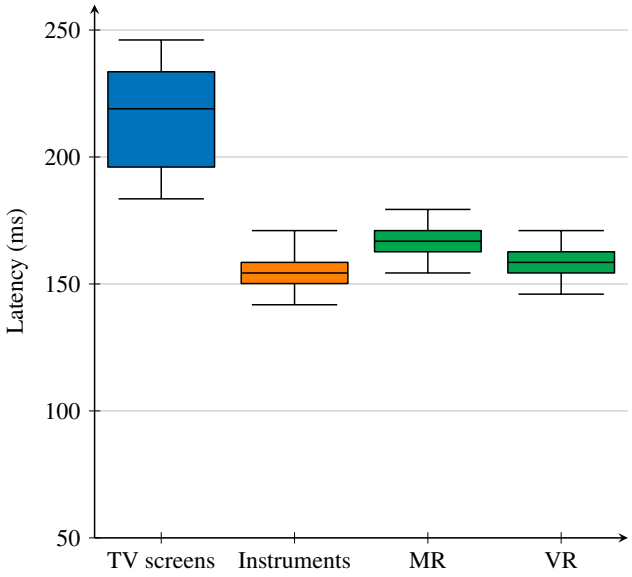


Figure 16: Time delays with experimental system enabled

Discussion

The video-based delay measurement setup offers a straightforward means to assess the end-to-end latency of the visual system components within 2PASD. The results indicate that there is room for improvement with respect to the time delay requirements defined by the CS-FSTD(H). Although the 2PASD is intended for research and development purposes rather than pilot training, it is advisable to address these delays.

Specifically, the image generators and experimental system introduce significant delays that require optimization. To address this issue, the hardware components of the image generators are being replaced, which may reduce the associated

delay.
For the measurements, the experimental system was executed in normal execution mode. However, it is hypothesized that running the model in accelerator mode (Section MATLAB/Simulink Implementation) could potentially reduce the delay. This aspect has yet to be investigated.

It is important to note that the end-to-end measurement only provides relative delay times. To gather more detailed information on a module-level basis, the video-based setup is insufficient. One potential approach to further investigate delays involves modifying the software of the simulation components. Due to the centralized structure, it would be sufficient to modify the interface software to measure the elapsed time between sending a data packet to another module and receiving the response. This modification would facilitate identifying time delays of individual simulation modules.

The conducted measurements provide a valid baseline for optimizations. Various measures can now be applied to reduce latency, and their effects on the overall latency can be measured.

CONCLUSIONS

This paper provides an overview of progress made with the Dual Pilot Active Sidestick Demonstrator (2PASD). The simulator's capabilities have been strongly extended towards a flexible platform for rapid prototyping. A replication of DLR's research helicopter's experimental system has been added to the simulator. Unlike the original system, it is implemented with MATLAB/Simulink and enables users to develop experimental flight control software in the simulator environment. 2PASD's capabilities regarding tactile cueing development were further extended by implementing an emulation of the Control Loading System hardware, supporting users during development of active functions. Furthermore, the simulator's portfolio was extended by implementing a Virtual Reality/Mixed Reality headset as an alternative to the standard TV screen visual system. In addition, the simulators performance was investigated by analyzing the end-to-end latency between control inceptor inputs and the visual system's response. A minimal invasive video-based measurement system was established for that purpose.

The most relevant conclusions derived from this paper are:

1. Flexible flight simulator configurations such as 2PASD offer a versatile environment for experimental flight software development, thereby complementing the capabilities of full-flight simulators.

2. MATLAB/Simulink is a suitable tool for development of experimental flight control software embedded in real-time simulation frameworks.

3. Emulation of Control Loading System hardware characteristics increases efficiency of the development process for tactile cueing functions.

4. Open and flexible simulator arrangements provide several benefits for setting up VR/MR simulation environments compared to full-flight simulators.

5. Latency measurement with a video-based, non-invasive setup provides valuable insights for identifying sources of time delay.

Author contact:
alexej.dikarew@dlr.de
mario.muellhauser@dlr.de
tanja.martini@dlr.de

REFERENCES

1. Kaletka, J., Kurscheid, H., and Butter, U., "FHS, the new research helicopter: Ready for service," *Aerospace Science and Technology*, Vol. 9, (5), July 2005, pp. 456–467. DOI: 10.1016/j.ast.2005.02.003.

2. Duda, H., Advani, S. K., and Potter, M., "Design of the DLR AVES Research Flight Simulator," AIAA Modeling and Simulation Technologies (MST) Conference, August 2013. DOI: 10.2514/6.2013-4737.

3. Dullo, C. M. U., Müllhäuser, M. R., and dos Santos Sampaio, R., "The Dual Pilot Active Sidestick Demonstrator (2PASD) for Development and Evaluation of Tactile Cueing Functions in a Multicrew Cockpit," Deutscher Luft- und Raumfahrtkongress 2019, October 2019.

4. Walko, C., and Müllhäuser, M., "Design of a Haptic Obstacle Avoidance for Low Speed Helicopter Operations Using Active Sidesticks," 47th European Rotorcraft Forum, September 2021.

5. Müllhäuser, M., "Effect of a Haptic Torque Protection with an Active Sidestick on Pilot's Eyes-Out Capability During a Helicopter Takeoff," Proceedings of the 49th European Rotorcraft Forum, 2023-09-05/2023-09-07.

6. dos Santos Sampaio, R., "Electronic Coupled Active Sidesticks in Dual Pilot Helicopters for Instructional Flights," 43rd European Rotorcraft Forum, 2017.

7. dos Santos Sampaio, R., *Influence of Variable Inceptor Coupling on Dual Pilot Helicopters Using Active Sidesticks*, Ph.D. thesis, Technische Universität Carolo-Wilhelmina zu Braunschweig, November 2018.

8. Taylor, A., Greenfield, A., and Sahasrabudhe, V., "The Development of Active Inceptor Systems and the Scope and Design Issues of Tactile Cueing Systems," American Helicopter Society 64th Annual Forum, 2008.

9. Gotschlich, J., Gerlach, T., and Durak, U., "2Simulate: A Distributed Real-Time Simulation Framework," *ASIM Mitteilung 149*, February 2014.

10. Gotschlich, J., and Jones, M., "Online Trimming of Flight Dynamics Models Using the 2Simulate Realtime Simulation Framework," AIAA Scitech 2018 Forum, January 2018.

11. Gestwa, M., Höfinger, M., Lantzsch, R., and Alvermann, K., "The Software Development Environment of the DLR Research Rotorcraft ACT/FHS," Deutscher Luft- und Raumfahrtkongress 2012, September 2012.

12. MathWorks, "S-Function - Include S-function in model," https://de.mathworks.com/help/simulink/slref/sfunction.html, Accessed 2024-07-11.

13. MathWorks, "Model References - Reuse models as blocks in other models," https://www.mathworks.com/help/simulink/model-reference.html, Accessed 2024-07-11.

14. MathWorks, "Simulation Pacing Options - Slow simulation to a specified ratio of simulation time to wall clock time," https://www.mathworks.com/help/simulink/slref/simulationpacingoptions.html, Accessed 2024-07-11.

15. MathWorks, "Acceleration - Improve simulation speed with accelerator and rapid accelerator modes," https://www.mathworks.com/help/simulink/acceleration.html, Accessed 2024-07-11.

16. Milgram, P., Takemura, H., Utsumi, A., and Kishino, F., "Augmented reality: a class of displays on the reality-virtuality continuum," Proc. SPIE 2351, Telemanipulator and Telepresence Technologies, edited by H. Das, December 1995. DOI: 10.1117/12.197321.

17. Varjo, "Varjo XR-3, the first true mixed reality headset," https://varjo.com/products/varjo-xr-3/, Accessed 2024-03-20.

18. VIVE, "SteamVR Basisstation 2.0," https://www.vive.com/de/accessory/base-station2/, Accessed 2024-03-20.

19. European Aviation Safety Agency., *Certification Specifications for Helicopter Flight Simulation Training Devices*, Publications Office, June 2012.