PSAA
AerospaceEurope Conference2021
WARSAW, POLAND | 23 -26 NOVEMBER 2021
CEAS
Council of European Aerospace Societies

VR ENVIRONMENT FOR UAV PILOTS TRAINING WITH AUTOMATED FLIGHT ASSESSMENT SYSTEM

Ph.D. Antoni Kopyt[1], B.Sc. Dominik Tokarz[1] & B.Sc. Stanisław Gajek[1]

[1]Warsaw University of Technology, Pl. Politechniki 1, 00-661 Warsaw, Poland

Abstract

Recent legal changes performed by European Commission visualise both private and public markets' demand for safe and reliable unmanned aircraft systems. Although the development of automatic, or even autonomous control systems has moved strain of manoeuvring off the pilot, it is still a necessity for a human to take responsibility for drone's actions. In face of such needs, an adequate, new approach to pilot training must be taken, in order to ensure the effectiveness of such a course.
The core of the simulator is based on open flight controller software (such as ArduPilot or PX4) which is easily integrated with real RC controller, thus increasing the quality of training by easily switching between simulated and real flight conditions by using identical control equipment. This approach also provides open identification of dynamic models and their modifiability. With this solution, it is possible to implement new dynamics models for training or software testing purposes in the SITL(Software-in-the-loop) approach. The architecture also allows devices to be tested in a HITL(Hardware-in-the-loop) approach by replacing a flight controller simulated on a PC with a real unit. The environment is to function in virtual reality (VR) technology, with the possibility of using augmented reality (AR) functions for the creation of artificial obstacles. The instructor panel will be an extension of the simulator, from which it will be possible to modify environmental parameters, change mission settings or react to the trainee's behaviour and decisions during training. These functions will be coupled with an automatic function of pilot skill assessment based on selected flight characteristics.
The paper describes development of a simulated environment, hasting advancements in trainee's flight skills. The main objectives of this project are building a custom flight environment, providing a tool for creating uncharacteristic mission scenarios and aiding instructors with automatic flight analysis module. The simulator should enable practicing in harsh and often unrepeatable in real life mission settings, that could be prepared both before an exercise, or induced during one with a specifically assembled instructor's panel. Data gathered during the activity is subjected to thorough dissection by a set of algorithms, resulting in breakdown of subject's performance in time, as well as its overall grading. The results may be directed to the instructor, providing them with an overview on pilot's progress and helping with preparing the optimal guidance during the rest of the course.
The paper concludes with a summary of milestones achieved during recent project development, as well as a prediction of steps required to further advance with the project. Given there are a brief depiction of program's performance optimisation methods, graphical user interface adjustments and consideration of possible integration with a biofeedback system, developed in parallel to described simulator.

Keywords: UAV, AR, training, assessment, simulator

VR environment for UAV pilots training with automated flight assessment system

Ph.D. Antoni Kopyt[1], B.Sc. Dominik Tokarz[1] & B.Sc. Stanisław Gajek[1]

[1]Warsaw University of Technology, Pl. Politechniki 1, 00-661 Warsaw, Poland

October 23, 2021

1. Introduction

The increase in the number of airspace users, legal changes in the use of UAVs, the unification of laws within the European Union, new classifications of drones and market demand for services provided using UAVs make the need for highly qualified drone pilots greater than ever.[1] This also indirectly affects the tools and methods of their training and licensing.

In line with the above factors, one may expect a growing need in the market for simulator solutions for drone operator training, often portraying very specific scenarios and conditions. Practicing in a simulator provides repetitiveness and control over weather conditions, mission scenarios, and (in an automated and objective manner) enables a pilot capability evaluation based on accepted assessment criteria and methods.

The development of the equipment, and in particular the virtual or augmented reality simulation headset, creates a new niche for UAV simulation flight applications that is worth exploiting for the development of today's simulation techniques.[2] Until now, simulation during UAVO courses has not been very popular mainly due to low resolution of the head-mounted displays interfering with parallax distance estimation and was not a viable alternative to the classic flight outside the simulation environment, at most a complement to it.

The paper is a proposal for a new simulation environment using dependencies already existing on the market, augmented reality headset from the Microsoft HoloLens series, and proprietary methods and criteria for evaluating the piloting abilities of trainees. The developed project is based on the concept of Ph.D. Antoni Kopyt's 3-step training approach utilising advatages of virtual, augmented and mixed reality.

2. Design goals

The three main distinguishable tasks to be performed by the developed project are:

(a) providing a platform for effective and affordable training of UAV pilots,

(b) research and development (R&D) for hardware/software testing purposes,

(c) evaluating new technologies.

Transferring the burden of learning to fly a UAV from real equipment to the simulation environment gives the freedom to create environmental conditions, any air traffic to study the pilot's reactions and improve the pilot's skills in previously prepared mission profiles (photo missions, reconnaissance missions, etc.).

The proposed UAV flight learning environment will enable a smooth and more natural transition for the trainee between virtual reality and piloting in real conditions by implementing First Person View (FPV), Line of Sight (LOS) view and aerial view for practising all types of flights performed with drones. This may be achieved by an application of VR and AR technology visualising artificially generated obstacles. Combined with a manual transmitter should aid improvement of visual-motor skills and overall confidence over a flight with a real drone.

Simulation station will be augmented with an instructor panel connected to the simulation environment, whose role will be to react to the student's behavior, modify environmental parameters or adjust mission settings during training. Users will be also provided with an autonomous environment to assess pilot skills based on selected flight characteristics. Resulting report should be easily available and readable both for beginning and advanced users.

3. Architecture

3.1 General structure

In the system architecture of the proposed solution, three main subsystems can be distinguished that communicate with each other and exchange the necessary data for operation:

1. Simulation Station
2. Management Station
3. Visualisation Station

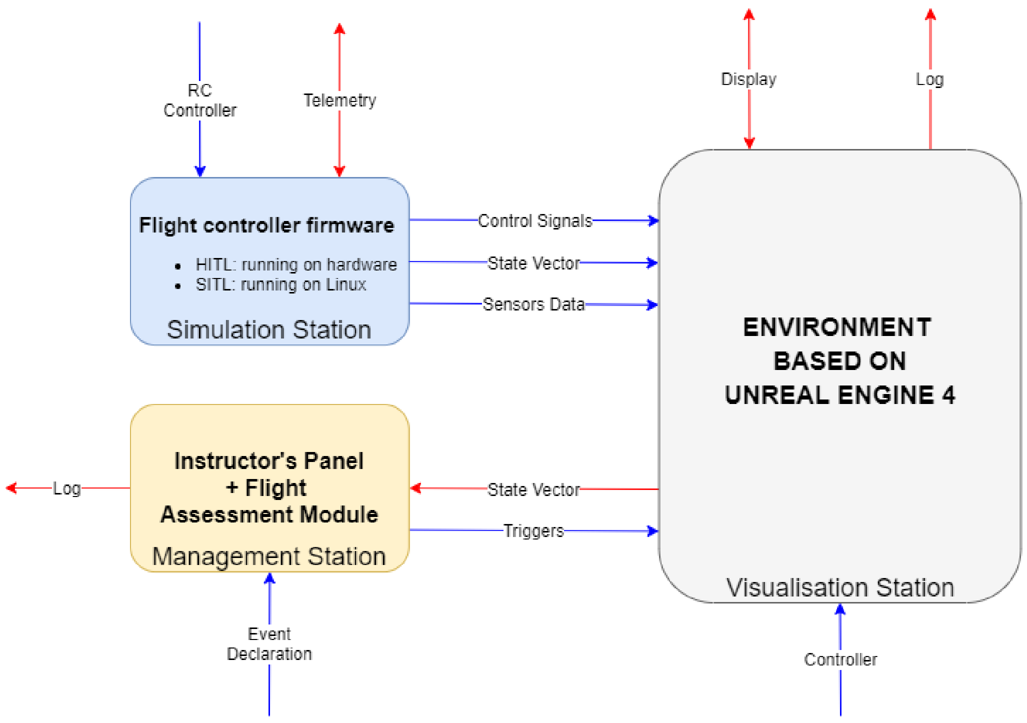


Figure 1 – Simplified architecture structure of the proposed simulation platform

The Simulation Station (1) consists of flight controller software running on a hardware plane (in the HITL approach) or emulated on a Linux platform (SITL approach). The input to the system is telemetry and RC Controller while the output to the UAV are commands. Outputs to the Visualization Station(3) are: state vector, sensors data and control signals. For ease of tracking the direction of information flow shown in figure 1, all inputs are marked in blue, outputs and bidirectional exchanges in red.

The Management Station (2) consists of the Instructor Panel and Flight Assessment Module. The module receives a state vector from the simulation environment to evaluate flight parameters and generate results. By declaring events, it returns data in the form of triggers to the Visualization Station(3).

The core connecting the whole system architecture is the Visualization Station(3) which connects all peripheral devices and is responsible for calculation and visualization of the drone position in the virtual environment. It receives control signals, sensors data and the state vector from the Simulation Station(1). It then processes the information, performs visualisation and returns the state vector to the Management Station(2). Input data, such as triggers or control information is gathered from Management Station(2) or other control sources in form of additional controllers as well as interactive displays (like AR goggles). It is possible to log data from the simulator as output.

3.2 Simulation Station

The Simulation Station can function in the SITL approach when the flight controller software is emulated on a virtual machine (in the considered case - on Linux) or in the HITL approach where all signals are processed by the Flight Controller itself. The possibility of reconfiguration of the simulation station setup opens up opportunities for testing and validation of software and hardware using the proposed solution.

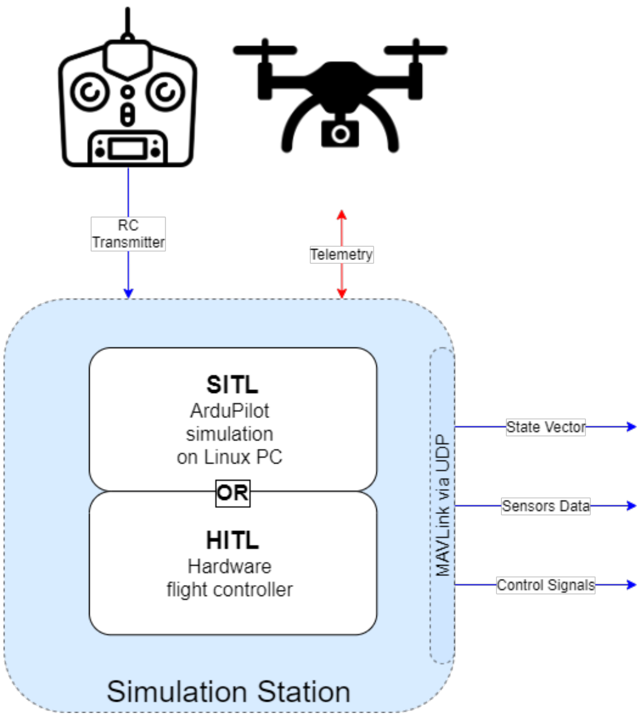


Figure 2 – Internal design of Simulation Station

The MAVLink protocol [3] is used to transmit state vector, sensor and control signal data. The peripherals are an RC transmitter sending control signals from the pilot and an UAV exchanging telemetry and commands.

3.3 Management Station

The Management Station consists of two subsystems: the Instructor's Panel and the Flight Assessment Module (FAM).

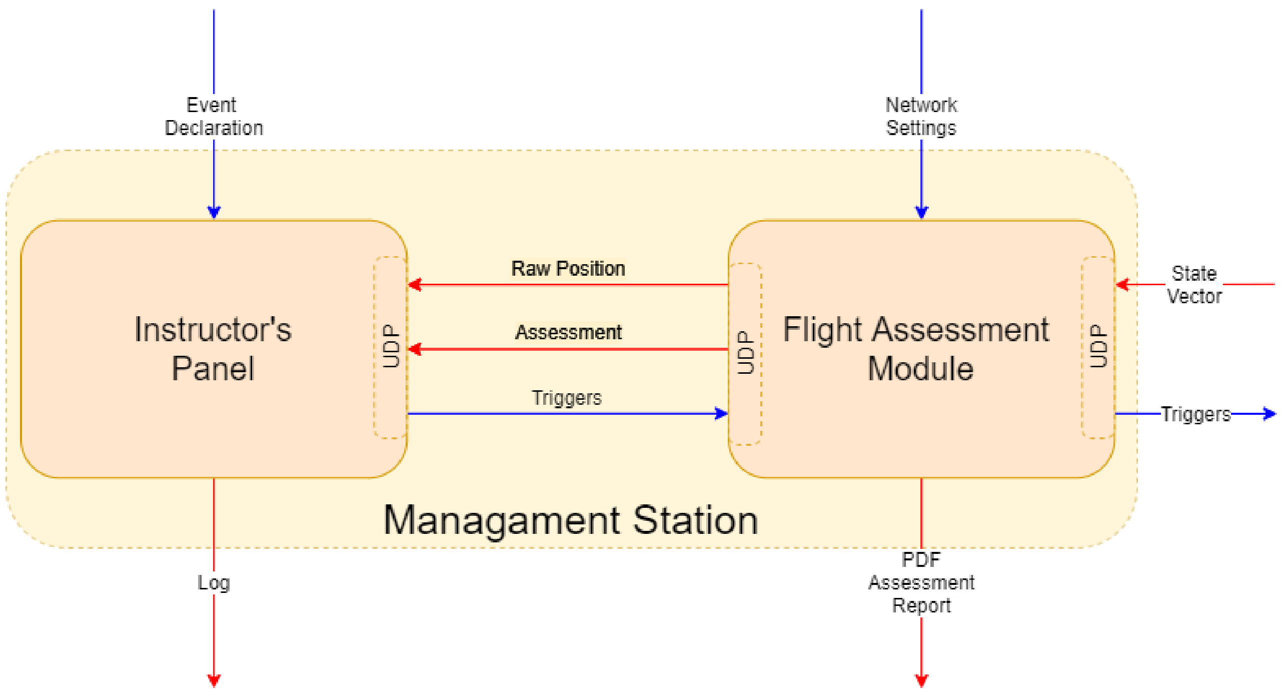


Figure 3 – Internal design of Management Station

The FAM module receives information regarding the state vector of the drone simulated in the virtual environment from the Visualization Station's API via a UDP interface. This data is considered being the input to the internal FAM evaluation algorithm, which generates a PDF report after a training conclusion. As an another data source to the FAM should be considered the Network Settings file, which modifies the FAM's communication behaviour. The FAM module, also using the UDP protocol, transmits obtained raw position and generated assessment values down to the Instructor's Panel.

The instructor from the panel level is able to observe the results of the conducted course, log them and declare events modifying the course of the exercise or simulator settings. Event declarations are turned into triggers, which are sent as an input to the visualization station through the FAM subsystem.

3.4 Visualisation Station

The visualization station is the core module of the overall architecture, linking together the peripherals, the evaluation module and the simulation module.

The API is responsible for communication of the Visualization Station with all other modules and their components. It receives the state vector from the Simulation Station and then, after processing in the simulation environment, returns it to the Management Station where it is treated as the input of the flight characteristics assessment algorithm. Triggers are the input that modifies the environmental operation of the Visualization Station. Sensor readings and control data also flow from the Simulation Station, which are inputs to the module components.

The display are the Microsoft HoloLens2 series headset, which is a translucent head-mounted display device that presents in AR the results of visualization. It also performs as an additional computer that can make modifications to the system (functionality of modifying the simulation environment from the goggles user level is envisaged at a later stage of project development). The goggles are connected via Bluetooth or USB port. Other peripheral devices are different types of controllers (mouse, keyboard, joystick, game-pad) that aid and accelerate development and debugging of simulation environment. Data from the Visualization Station can be logged for debugging, testing, validation, or architecture expansion purposes.

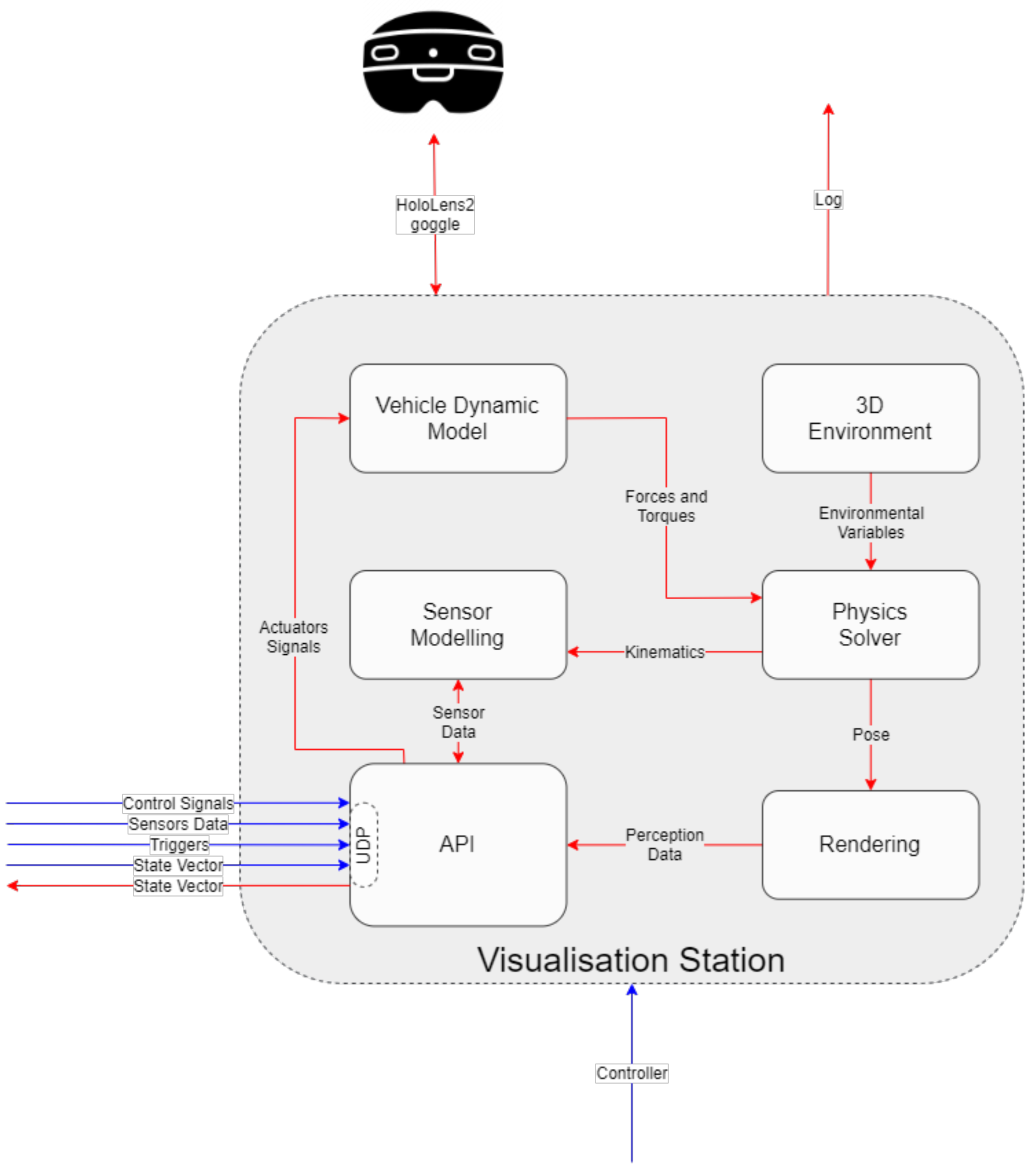


Figure 4 – Visualization Station components

The API passes the actuator state data to the vehicle dynamics model, from which we get information about forces and moments in the space of the simulation environment. The 3D environment itself generates environmental variables (weather conditions, physical settings, etc.). The force and environment data are recalculated by the engine/physics solver where the pose in the virtual world and its kinematics are generated. The kinematics data is provided to the sensor modeling segment, which also obtains real readings from the Visualization Station API. This allows the system to mirror the performance of the real apparatus in the simulation environment, but it is also possible to simulate the performance of the sample apparatus for testing purposes. All the data processed by the previous segments is used to generate a render frame creating an image of the simulation.

3.5 Complex architecture

The architecture following all modules assembly is shown as a flowchart in figure 5.

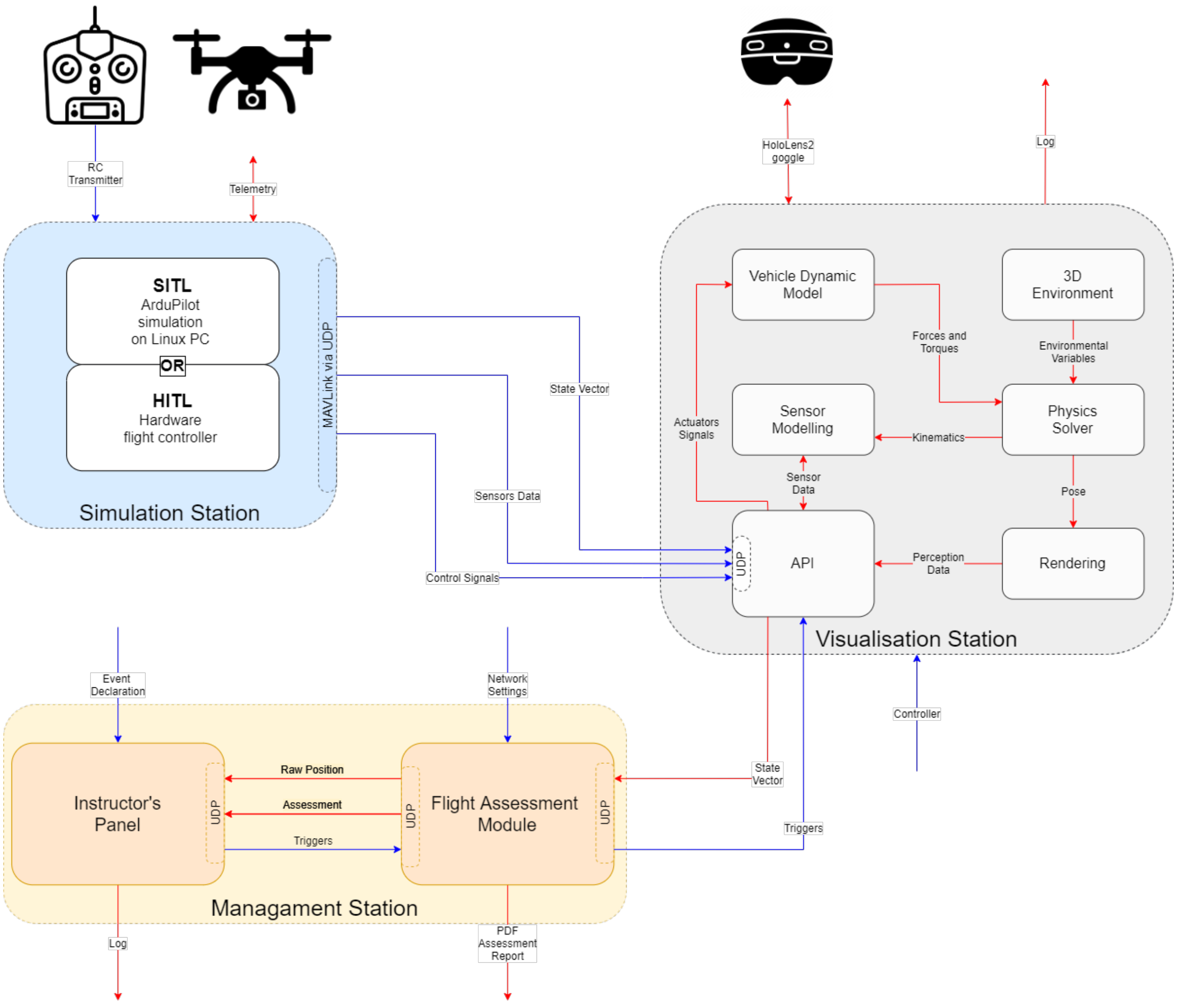


Figure 5 – Composition of all subsystems

The number of connections has been kept to a minimum. As a reminder, the blue shows all the input signals to the system, the red shows the output signals. Thanks to the well laid out API of the modules, the location of the system functions and the consideration of what information is needed for the functioning of the modules and smaller components, it was possible to achieve a very clear architecture that has the potential for further expansion.

4. Data flow

4.1 Flight Assessment Module communication frame

In order to exchange data between system components, a message frame structure was prepared. Communication algorithm is performed between federates and Flight Assessment Module (FAM) as follows:

1. a federate sends a [SND] flag,
2. FAM answers with a [LST] flag,
3. the federate sends a [FMSG] message,
4. FAM answers with [AMSG] message.

The [SND] flag consists of 5 characters message informing FAM about the type of message the federate is going to provide. Possible flags are:

- POSIT - state vector indication,
- ERROR - error indication,
- SIMST - beginning of federate's simulation,
- SIMED - end of federate's simulation,
- ACTIO - IPn action start confirmation,
- EJECO - action removal confirmation,
- EJECU - user removal confirmation.

Answering federate's call, the FAM responds with 5 characters [LST] flag in which one of three types of instructions may be provided to the federate:

- NOSIG - no new commands from IPn,
- IPSIG - new control commands from IPn,
- EVSIG - new event trigger.

Every flag is followed within next communication period by a [FMSG] or [AMSG] message consisting of even characters number. Within each message all positional data are coded with 12 characters, error identification numbers with 7 characters, angular orientation data and federate identification code with 5 characters and control input with 4. Other data is coded by a specific standard described in software documentation (such as weather information coded by modified METAR standard).

4.2 Assessment Interface for Instructors and Trainees

According to chapter 2.*Design goals*, the developed system requires to have an interface allowing users (both pilots and instructors) to freely review the training progress. This means, that for both programs separate human-software interfaces must be provided.

Due to the assumption of computing most of assessment parameters away from Instructors Panel (IPn), it is necessary to enable exporting the gathered data directly from Flight Assessment Module (FAM) in form of a easily readable document containing the most important information, such as:

- overall grading,
- two or three dimensional flight path visualisations,
- temporal score graphs of all analysed flight parameters,
- a short text comment on score (optional).

All of the above may be presented as a PDF document generated with a python script and saved on a local or remote drive, allowing easy access to the report. Such an action is possible using for instance PDFDocument python package [4]. Overall grading, as well as temporal score analysis will be performed as described in chapter 5*Flight assessment methods* and visualised with use of matplotlib package [5]. A short text comment on score will be chosen based on specified threshold levels and from a prefabricated list of strings, stored in a separate text file, allowing possible future translations to other languages depending on training requirements.

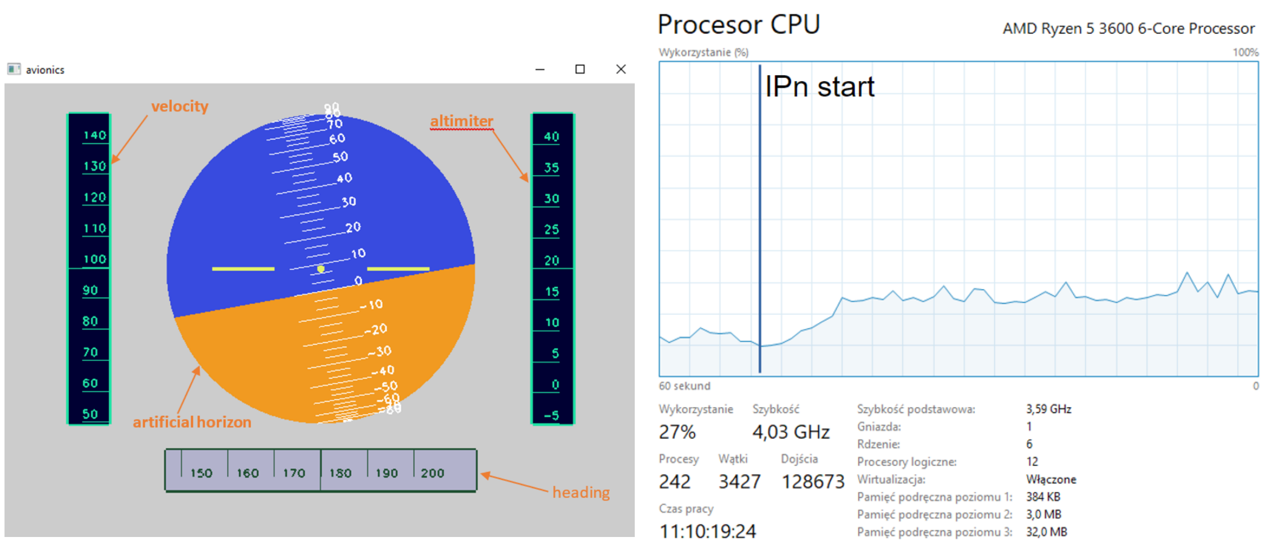
With the above resolutions it is no longer necessary to provide a Graphical User Interface (GUI) to the FAM. If needed, any configuration may be performed with an external file storing settings such as communication ports, maximal allowable user count, connection time-out, type of data forwarded to federates, etc. It is, however, of great importance to provide solutions as stable as possible to limit the need of configuration file edits to the absolute minimum and to prepare an in-depth documentation on editable options, clarifying the results of available changes.

On the other hand, in terms of Instructor Panel a GUI is required. In order to bring a meaningful advice on flight performance or trigger a non-predefined event, instructor must be provided with a real time summary of flight characteristics, path and environment. To reduce computational strain on instructor's machine, a more modest approach in comparison to the simulator itself was chosen to visualise aforementioned parameters. Built using an open-source OpenGL Utility Toolkit alternative freeglut [6], the IPn is divided into 4 panels:

- flight path visualisation box,
- instrument panel,
- real-time assessment panel,
- event trigger panel.

The panels are positioned in four directions (west, north, east and south respectively) in order to effectively use 3D rendering capabilities of OpenGL. Ultimately all of listed panels should be also displayable as separate windows, however the former approach enables a possibility of future porting to VR or AR. In order to achive that, however, it would be necessary to rewrite the GUI using more modern API like Microsoft® DirectX12® or AMD Vulcan™.

Elements such as graphs and instrument faces are rendered as two dimensional polygons, positioned in front of the virtual camera in three dimensional space. Simulation event triggers (weather, action etc.) are controlled via set of virtual switches, activated with mouse or keyboard. Every trigger may be started with a keyboard shortcut defined in a configuration text file provided during software installation. It is worth noting, that trigger will be executed from any panel by sending a special flagged message to FAM without further confirmation to ensure precise event activation.



Procesor CPU AMD Ryzen 5 3600 6-Core Processor

Wykorzystanie (%)

Wykorzystanie 27% Szybkość 4,03 GHz

Procesy 242 Wątki 3427 Dojścia 128673

Czas pracy 11:10:19:24

Szybkość podstawowa:	3,59 GHz
Gniazda:	1
Rdzenie:	6
Procesory logiczne:	12
Wirtualizacja:	Włączone
Pamięć podręczna poziomu 1:	384 KB
Pamięć podręczna poziomu 2:	3,0 MB
Pamięć podręczna poziomu 3:	32,0 MB

(a) IPn instruments panel GUI

(b) CPU usage after IPn's instruments rendering activation

Figure 6 – Instruments Panel

While graphics are calculated by GPU, indicators' positioning is performed by CPU. To retain highest possible smoothness of instruments' motion, other IPn tasks like constant UAS state logging or input listening (both from FAM and user) are performed by separate threads, using standard "Thread" C++ library [7]. While with modern hardware such an approach may be an overstate, especially with single core clock frequencies reaching above 4GHz, multi-threading allows for more sustainable high frame counts, improving user experience. Moreover, as shown in figure 6b, the positioning process in especially rapidly evolving conditions may take up to 18% of additional CPU power, compared to idle state.

5. Flight assessment methods

As the developed system requires an unambiguous way of pilot's performance assessment, a set of quantifiable flight parameters must be defined. The approach taken in this project enables an accurate, real-time, four dimensional positioning (space and time), as well as steering and other data to be rapidly shared between nodes other than simulator itself. This effectively allows a combination of flight path and pilot's reactions analysis, aiding instructor in finding potentially dangerous in-flight behaviour or enabling highlighting unsafe decisions directly to the pilot.

Referring to chapter 3. *Architecture*, from networking side of view, the system is comprised of two types of federates: Simulators (SIM) and Instructor Panels (IPn). In order to maintain communication, but also to enable independent operations between said nodes, a run-time infrastructure was developed in form of Flight Assessment Module (FAM). Being a [8], FAM also performs an automated assessment of gathered data, as trainee should have access to their performance log without IPn being a part of the federation.

Flight evaluation in the FAM is performed based on following parameters:

- Distance to waypoint centre,
- Path segment deviation,
- Path segment mean velocity,
- Input jerk,
- Danger reaction time (if a preconditioned event is triggered).

Several methods found, inter alia, in "*A study on objective evaluation method for steering quality taking into consideration ride comfort*" [9] were tested with the addition of in-house made equations on grading aforementioned flight parameters. The comparisons were made using theoretical flight path deviations and simulated flight paths with artificially induced errors scenarios. For each run, a set of marks was generated in range of $T_i \in [0,1]$, with i being a chosen category. Additionally, a a_i parameter was defined, marking the maximal accepted error value for each parameter. If the subject was to maintain a constant error of the value specified in a_i parameter, it would be graded $T_i = 0.5$, which is further considered a minimal acceptable score.

5.1 Position based parameters assessment

Grades for position based parameters were calculated using:

1. absolute error value $g_{p1}(\delta) = 1 - \frac{0.5}{a_i} \cdot |\delta|$,
2. squared error value $g_{p2}(\delta) = 1 - \frac{0.5}{a_i^2} \cdot \delta^2$,
3. exponential assessment method $g_{p3}(\delta) = \exp\left(\frac{\ln(0.5)}{a_i^2} \cdot \delta^2\right)$

where $\delta = |\vec{x}_{cor} - \vec{x}_{real}|$, $\vec{x}_{real}$ is a current drone position and $\vec{x}_{cor}$ is a position expected by exercise. In order to prevent calculated values from exceeding assumed range, a condition of $g_i(\delta) \in [0,1]$ was hard-implemented as

$$g_i(\delta) = \begin{cases} 1, g_i(\delta) > 1 \\ g_i, g_i(\delta) \in [0,1] \\ 0, g_i(\delta) < 0 \end{cases} \tag{1}$$

If recorded timestamps were to not correspond with the test case design, the latter was recalculated to match given flight log using predefined set of path curves. Next, the data was discreetly integrated using trapezoidal approximation, after which the outcome was normalised by comparing it to corresponding maximal possible score (as shown in equation 2).

$$T_i = \frac{\int_{t=0}^{t_{end}} g\left(\delta(\tau)\right) d\tau}{\int_{t=0}^{t_{end}} g\left(0\right) d\tau} \tag{2}$$

A comparison between shape of the defined methods is shown in Figure 7. As seen in the image, function $g_2(\delta)$ is the most strict among the proposed methods, aggressively punishing errors beyond the acceptance zone, while highly promoting flight with error below the a_i boundary. It is worth noting, that this method is not sensitive to small deviations, leaving some acceptance for human error. Such a behaviour may simulate better a real instructor's assessment, albeit a strict one. On the opposite side is $g_1(\delta)$ function, rising score proportionally to lowering path error. Given its flow, it is capable to show any mistakes done by pilot or autopilot with high accuracy, however the difficulty of achieving higher scores may be discouraging to some trainees. The $g_3(\delta)$ method blends both of the mentioned functions, allowing for some drift from the designed path while at the same time expressing almost proportional score increase within range of $50-150\%$ of the a_i parameter.

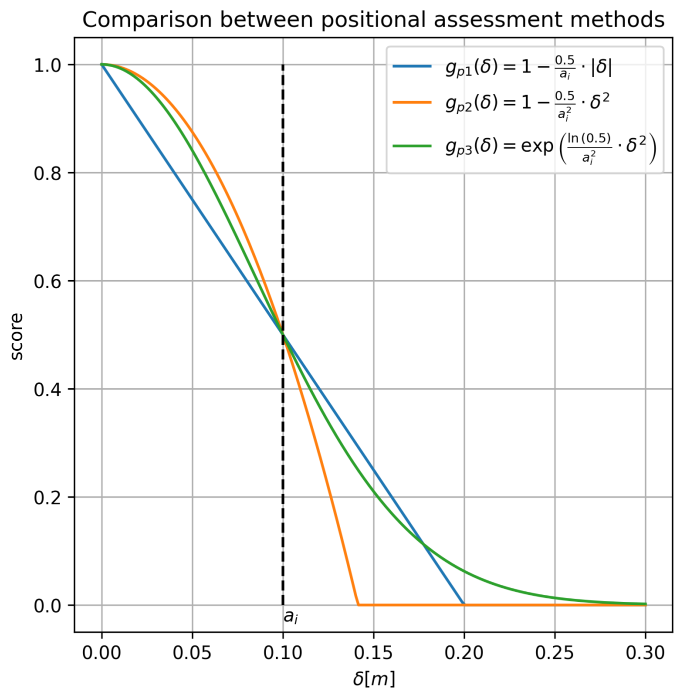


Figure 7 – Comparison between positional assessment methods' shapes

5.2 Velocity based parameters assessment

For velocity assessment, a different grading approach was taken. Instead of penalizing deviation from level design, pilot's progress is awarded by scores approaching $T_i = 1$ for a better performance. Calculations were performed using a set of equations similar to those defined in section 5.1with some variations:

1. absolute velocity value $g_{v1}(v) = \frac{0.5}{a_i} \cdot |v|$
2. square rooted velocity value $g_{v2}(v) = 0.5\sqrt{\frac{v}{a_i}}$
3. hyperbolic tangent assessment method $g_{v3}(v) = tanh\left(v \cdot \frac{\tanh^{-1}(0.5)}{a_i}\right)$

Due to a possibility of reaching out of range score values, condition expressed with equation 1 was also implemented into calculations.

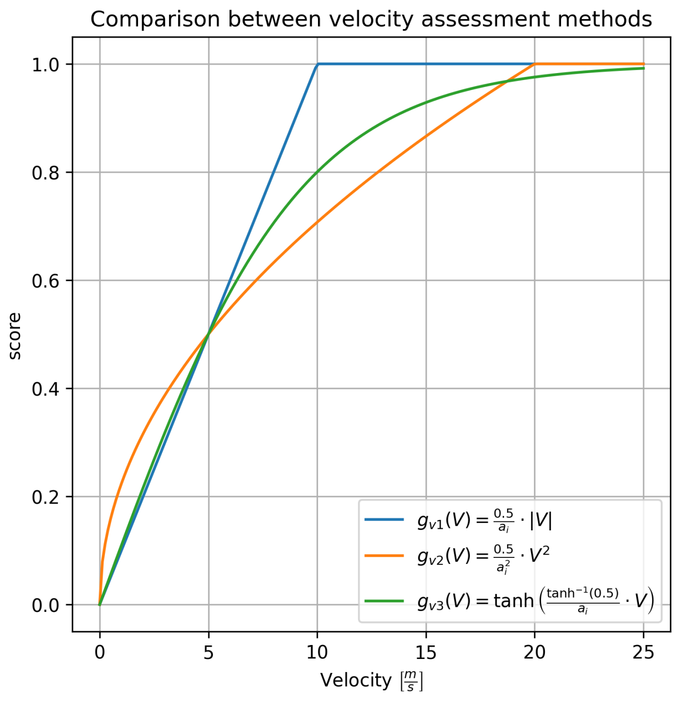


Figure 8 – Comparison between velocity assessment methods' shapes

A comparison between shape of the defined methods is shown in Figure 8. As seen in the image, both g_{v1} and g_{v2} methods allow to define a finite perfect mean velocity, while g_3 only approaches 1, giving a constant score improvement possibility. Such a behaviour effectively renders achieving a perfect score impossible, however usefulness of this approach is strongly dependant on whether a designed exercise requires a defined maximal mean velocity. Alternatively, a positional approach may be taken in order to simulate a requirement of a specified constant velocity during mission. This shows, that a level designer should be able to freely change the assessment method based on the training's requirements.

5.3 Simulated flight path assessment

Testing the positional methods was performed using a circular trajectory around a specified point in space. Such an exercise is often called "orbiting" as it's main goal is to revolve repeatedly around a well defined point of interest. In Cartesian coordinate system it's design path is expressed by a set of curves acquired by intersecting a sphere with a plane:

$$p(t) = \begin{cases} x = \cos(t) \cdot R \\ y = \frac{-2abx \pm \sqrt{\Delta}}{2+2b^2} \\ z = ax + by + c \end{cases} \tag{3}$$

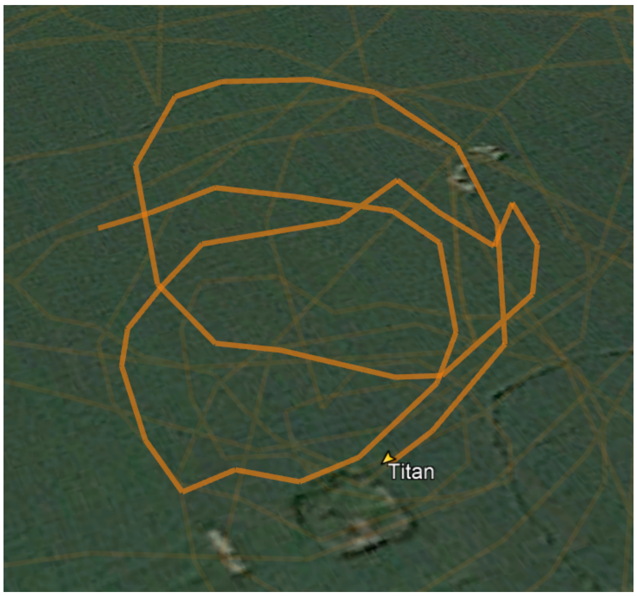
where a, b define path's pitch and roll, c - center point's elevation, R - orbit's radius and $\Delta = (2abx)^2 - 4 \cdot \left(1-b^2\right) \cdot \left(x^2\left(1+a^2\right) - R^2\right)$.

In order to introduce a realistic disturbance into the designed path a field test was performed and recorded using a private iFlight Titan XL5 quadcopter with an on-board GPS module. Data acquired during the session is visualised in figure 9 using Google Earth Pro. As shown in the images, the "orbit" is often oblated relative to a base circle, which may occur in presence of wind or due to pilot's inability to remain in constant distance to point of interest. Moreover, in the absence of altitude control module on board of the UAV, a heavy z-axis error is introduced often due to the pilot's slower reaction time to positional than rotational changes. Based on the findings, a disturbance profile may be introduced as a time-based three dimensional vector function:

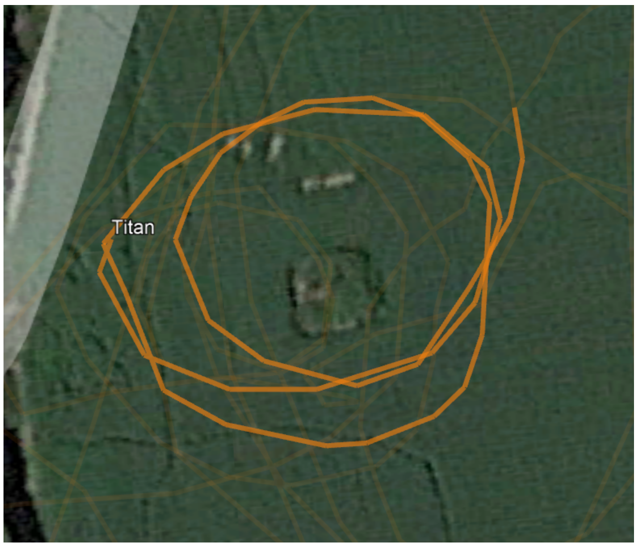
$$\vec{w}(t) = \left\langle \sin\left(\frac{1}{k} \cdot t\right) \cdot w_{sx}; 0; \sin\left(l \cdot t + \phi_0\right) \cdot w_{sz} \right\rangle \tag{4}$$

where k, l are disturbance period parameters, w_{sx}, w_{sz} are disturbance scale parameters and ϕ_0 is the z-axis disturbance starting phase parameter.

The data gathered during the field test was not used in the calculations due to sparseness of logged



(a) Perspective view

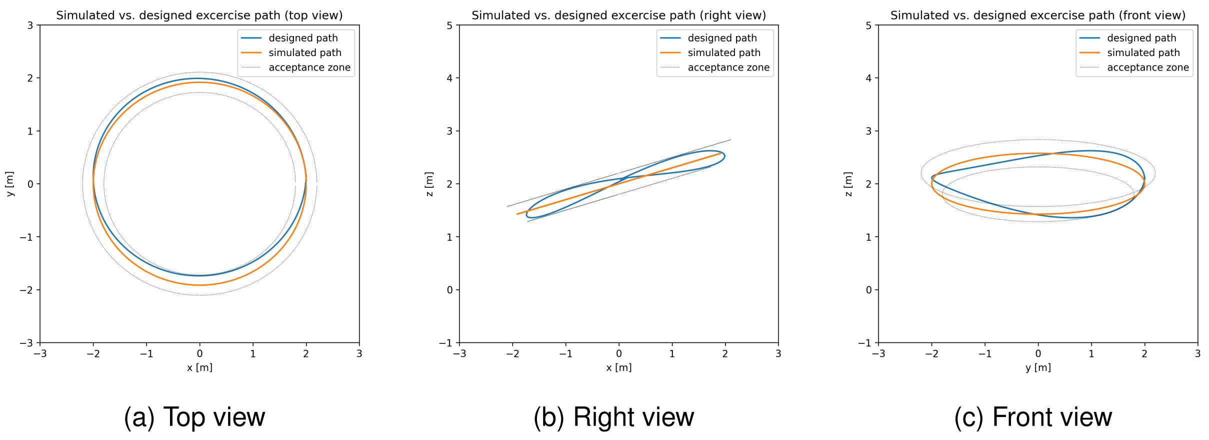


(b) Top view

Figure 9 – Field test flight path visualisation

points caused by low GPS refresh rate, limited access to IMU readings and lack of connection between the developed simulator and the betaflight-based multirotor used instead of px4 or ardupilot one.

The $\vec{w}(t)$ function has therefore been applied to the designed path, returning a disturbed trajectory, as shown in figure 10. Results from applying equations described in section 5.1before integration and normalisation are shown in figure 11. As predicted, the $g_{p2}(\delta)$ function (parabolic) yields the most dynamic change in flight score, while g_{p1} (linear) is the most lenient towards deviations and returns the lowest values while in acceptance zone. Scores yielded by functions are respectively $T_{p1} = 0.55267,\ T_{p2} = 0.56852,\ T_{p3} = 0.57482$.



(a) Top view (b) Right view (c) Front view

Figure 10 – A comparison between designed and disturbed flight path

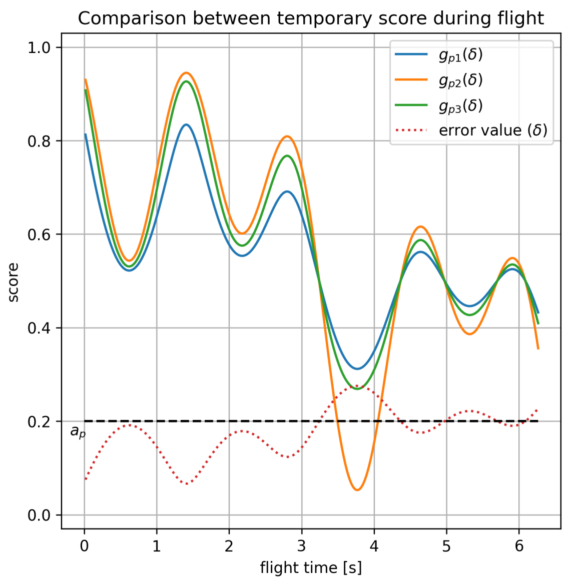


Figure 11 – Positional assessment equations' results before integration and normalisation

5.4 Artificially induced error

As forcefully induced errors, two additional scenarios were investigated:

- intensity differing oscillations,
- temporary high value error (amplitude of 1m).

As shown in figure 12, change in oscillations' frequency does affect overall score, however the grade difference is periodical, which confirms the need for an additional parameter evaluating control input adequateness. The latter scenario evaluates algorithm's response to an error in communication or on-board navigation of the drone. In this case non of the presented functions are immune to rapid high error value appearance (see figure 13), dropping the positional grade by $3-5\%$. This may indicate a high priority need for a robust navigation and communication systems, as well as well tuned filters, providing reliable data to the simulator.

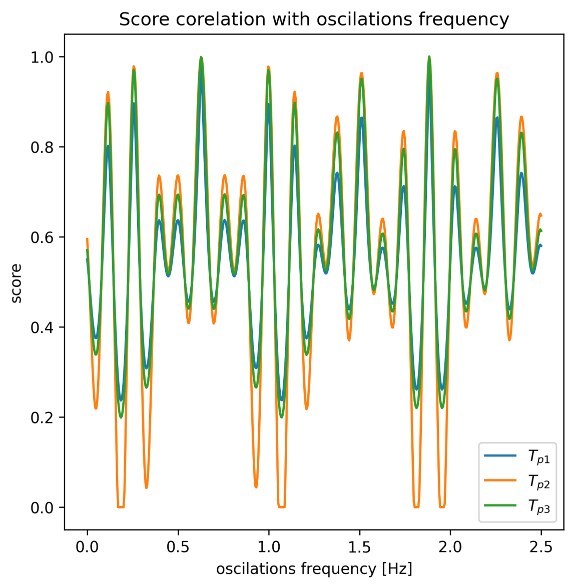


Figure 12 – Influence of oscillations' frequency on overall flight score

Overall, in terms of positional assessment, based on shown findings the squared error value method (g_{p2}), otherwise known as Integrated Squared Error [9], seems to provide best grading for short term training. However, for hard tasks meant to be learnt over greater time span the exponential function (g_{p3}) might be more suitable, as it provides a more lenient, yet still informative scores. The linear function (g_{p1}) might be useful in software testing, as it provides grading easy to quickly decode by a human designer.

Testing of the remaining evaluation parameters will be conducted in the near future.

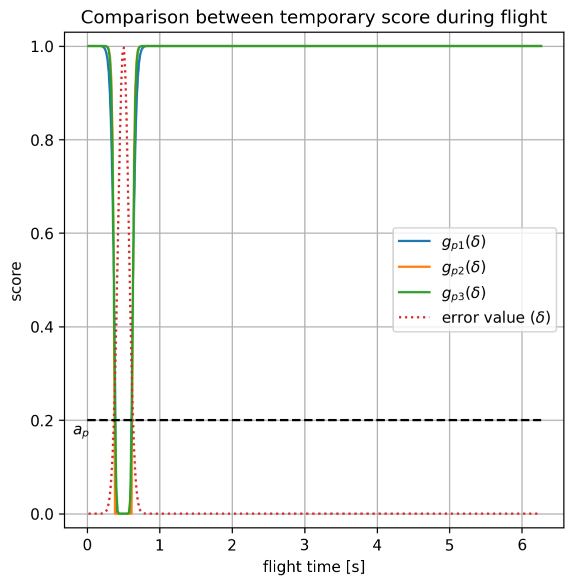


Figure 13 – Temporary grading of momentary high value error

6. Capabilities of simulated environment

6.1 Custom environment

Using Unreal Engine 4 made by Epic Games allows to create simulation environments with incredible quality rarely seen in the simulator market, which often offer a highly simplified design.

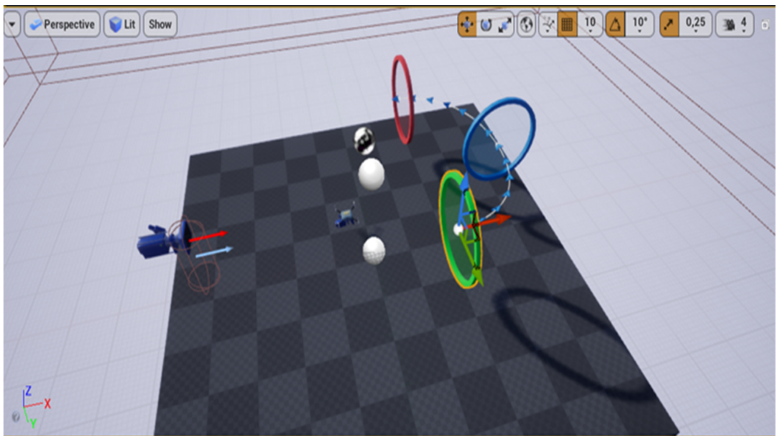


Figure 14 – A frame from the Unreal Engine 4 editor showing the process of designing a sample obstacle course

Providing photo-realistic graphics to trainees can impact learning outcomes and enhance the simulation flight experience. The engine itself offers great tools to communicate the architecture proposed in the paper and optimise the development process. Shifting the effort from coding the graphical-physical engine towards creating a functional product allows it to be polished.

6.2 Mission scenarios

Using C++ language and Blueprints (scripts inside Unreal Engine 4) it is possible to create freely defined mission scenarios as in an interactive application or computer game. It can have simple dimensions, from scripting moving obstacles to spawning objects, conditionally changing weather or mission objectives to plotted training missions designed to train and prepare a drone pilot for specific roles or tasks.

6.3 Multi-stage design

The architecture and design of the simulator allows for training in three aspects:

1. VR (Virtual Reality) with virtual drone within virtual environment
2. AR (Augmented Reality) with real drone within real environment enriched with virtual features
3. MR (Mixed Reality) combining the interaction of a real drone with virtual obstacles generated and placed in a real environment

With the use of Microsoft Hololens2 the developed system provides the ability not only to display simulation flight and simulate a drone inside a designed environment, but also create a mixed reality course. Using a real control apparatus and a physical UAV in a real environment we would be able to add elements of artificial obstacles, detect the interaction between the drone and the generated obstacles. This will enhance the training process by introducing elements of risk and testing the reaction of trainees without exposing to risk the equipment used.

Using the ability to transform objects from the user's Microsoft Hololens series goggles, it is possible in the future to expand the project with the ability to reorganise obstacles and the course by also interacting with these objects.

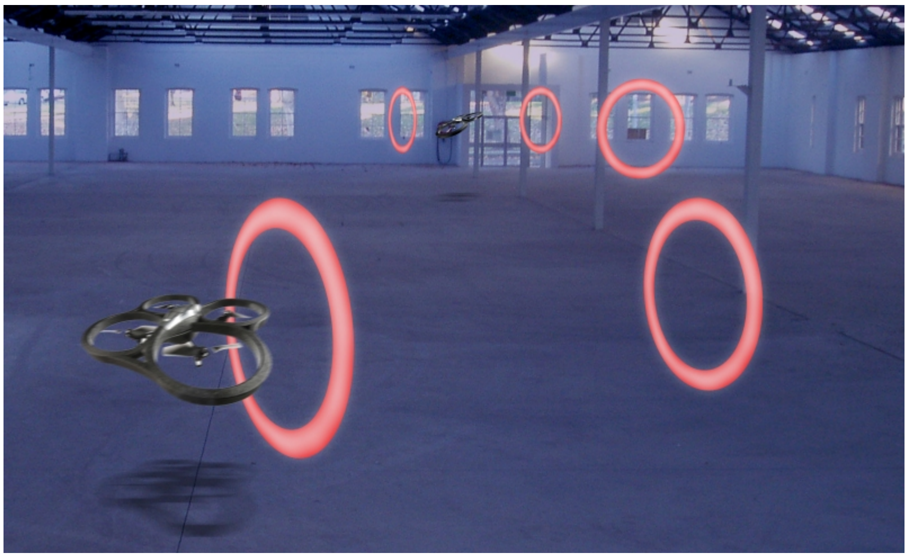


Figure 15 – Concept showing the possibility of using artificial obstacles in a real environment [10]

7. Conclusion

In face of recent legal, market and societal shift in view on Unmanned Aerial Vehicles it is necessary to provide a viable and modern training methods. Given the advantages of current home computing, it is possible to design a visually pleasing simulator that provides tools aiding both pilot in their flight exercises, as well as instructor in the assessment of trainee's flight abilities.

Main goal of the developed project was to create a UAV simulator, based on a combination of virtual, augmented and mixed realities proposed by Ph.D Antoni Kopyt, designed for drone pilot training. The course is supported by Flight Assessment Module providing a real-time evaluation of pilot's performance. Moreover the design supports both SITL and HITL approaches to software and hardware testing.

Unreal Engine 4 provides the flexibility to quickly create artificial, photo-realistic environments and enables shifting the programming effort from developing a physics and graphics engine to creating the core solution. The proposed architecture allows for free reconfiguration and expansion of the system which makes it flexible. The combination with a biofeedback solutions could provide invaluable data to further development of the system's ergonomics.

The combination of software already known on the market with emerging AR technologies such as modern goggles Microsoft Hololens2 series and original assessment methods allows the construction of a new type of simulator that will enrich the experience of drone pilot trainees and improve the training process.

References

[1] EASA, *"Easy Access Rules for Unmanned Aircraft Systems (Regulation (EU) 2019/947 and Regulation (EU) 2019/945)"*,
Available:`https://www.easa.europa.eu/document-library/easy-access-rules/easy-access-rules-unmanned-aircraft-systems-regulation-eu`,
Accessed: 20.10.2021,
2021, EASA site

[2] Brown, C., Hicks, J., Rinaudo, C. H., Burch, R., *"The Use of Augmented Reality and Virtual Reality in Ergonomic Applications for Education, Aviation, and Maintenance."*,
Available:`https://doi.org/10.1177/10648046211003469`,
Accessed: 20.10.2021,
2021, Ergonomics in Design

[3] Lorenz Meier, *"MAVLink Developer Guide"*,
Available:`https://github.com/mavlink/mavlink/`,
Accessed: 20.10.2021

[4] Matthias Kestenholz, *"PDFDocument"* package on GitHub,
Available: `https://github.com/matthiask/pdfdocument`
Accessed: 13.10.2021

[5] Hunter, J. D., *"Matplotlib: A 2D graphics environment"*
Computing in Science & Engineering, vol. 9, nr. 3, pg. 90-95,
IEEE COMPUTER SOC, 2007,
DOI: 10.1109/MCSE.2007.55

[6] Pawel W. Olszta, *"The freeglut Project"*, Available: `http://freeglut.sourceforge.net/` Accessed: 13.10.2021

[7] cplusplus.com, *"std::thread"* Available: `https://www.cplusplus.com/reference/thread/thread/` Accessed: 13.10.2021

[8] Denis Howe, *"Free On-Line Dictionary Of Computing"*,
Available: `http://foldoc.org/middleware`,
Accessed: 09.10.2021

[9] Paulina Anna Tomaszewska, *"A study on objective evaluation method for steering quality taking into consideration ride comfort"*,
2019, Warsaw University of Technology

[10] Len Calderone, *"The Fastest Upcoming Sport: Drone Racing"*,
Available:`https://www.roboticstomorrow.com/article/2017/03/the-fastest-upcoming-sport-drone-racing/9665`,
Accessed: 14.10.2021,
2017, RoboticsTomorrow site