Research Flight Simulator
Review of Components and Development of the Motion Cueing System

Eric Alexandre Félix da Silva Friman Loewenthal
ericloewenthal@tecno.ulisboa.pt

Instituto Superior Técnico, Universidade de Lisboa, Portugal
October 2018

Abstract
A flight simulator is an important tool in human factors research and other fields, allowing for experiments to be carried out even if too costly or dangerous for a real flight. This approach requires a convincing simulation, if its results are to be trusted. The Research Flight Simulator (SVI) at Instituto Superior Técnico was found to be lacking – its motion cueing system did not provide reasonable cues, according to pilot opinions and was hard to operate, relying on old versions of MATLAB; the flight deck layout and external visuals were also criticized. The objective of this thesis is to review and eliminate these limitations.

To validate motion cueing, the classical washout algorithm used to process aircraft data into motion base commands was reviewed using a synthetic testbench and was found to be incorrect. Corrections, including a new tuning procedure, were made and validated iteratively, using synthetic testing and feedback from test flights with airline pilots. A discrete-time version of the algorithm was also developed in native C++ code. Possible upgrades to the flight deck and to the external visuals were laid out for future implementation.

Synthetic testing showed that the revised motion cueing system behaves in a reasonable manner. Test pilots found the cues provided to be coherent but noted that some cues were too weak. A baseline was established for future developments to be compared against.

Keywords: Flight Simulation; Classical Washout Algorithm; Motion Cueing; Motion Drive Algorithm; Research Flight Simulator

1. Introduction
Flight simulation has long been an indispensable tool in the aerospace field. The safe environment of a flight simulator facilitates pilot training, allowing pilots to acclimate themselves to the aircraft before taking the controls of the real thing and to understand the limits of what the aircraft can do. Flight simulators are often used in air accident investigations, for instance, to determine whether other pilots would be able to perform better in the same situation [1] or to determine if an aircraft would have been capable of performing a certain flight in both ideal and real conditions [2]. Research also makes heavy use of flight simulators, to acquire insights into fields such as training procedures, understanding the limitations of the pilot/aircraft interface and ergonomics.

The Research Flight Simulator (known as SVI – Simulador de Voo de Investigação) at Instituto Superior Técnico fits in the latter category. In development since 2007 [3], the SVI provides motion cueing capabilities in addition to the basic functions of any flight simulator.

The SVI has been the focus of five previous Master’s theses, which aimed to develop its capabilities in various ways [3] [4] [5] [6] [7]. It has been used for projects such as examining the correlation of flight events with physiologic data from airline pilots in simulated flight [8]; as well as being used to familiarize students with aircraft instruments as part of their coursework.

The desire is to have a flight simulator which can be used in an increasing number of projects that take advantage of its motion cueing capabilities, which set it apart from most simulators and allow for a wider range of scientific activities.

This paper is laid out in five Sections, including this introduction: In Section 2, the SVI is presented and analyzed; areas of improvement are identified for study. The SVI’s Motion Cueing System is discussed in Section 3. In Section 4, improvements to other simulator subsystems are presented and proposed. Finally, Section 5 contains the conclusions of the project.
2. Overview of the SVI

The Research Flight Simulator at Instituto Superior Técnico, known as Simulador de Voo de Investigação (SVI), is, as its name implies, a flight simulator used for research purposes. It provides the pilots with an environment, based on a Fokker 27 nose section mounted on a motion platform capable of movement in six degrees of freedom. This environment is somewhat generic and is meant to represent a wide range of aircraft that can be simulated. The simulation is handled by general-purpose computer hardware and the X-Plane flight simulator software.

2.1 Architecture of the SVI

The SVI can be roughly divided into two main units: the Flight Simulation System and the Motion Cueing System, corresponding respectively to the top and bottom halves of the diagram in Figure 1.

The Flight Simulation System is responsible for running the simulation, handling pilot input and generating external visuals – most of the work is executed by the Simulation Computer, Master (SIMC-M), with the external visuals being offloaded to the Simulation Computer, External Visuals (SIMC-EV). X-Plane 9 is used as the flight simulation software and is configured to simulate a Boeing 777-200. Both SIMC machines are Pentium 4-class workstations. The SVI’s flight controls are provided by a Logitech joystick, as well as a touchscreen for all other controls. A Primary Flight Display and a Navigation Display are provided to the pilot on an additional monitor.

The motion cueing system is built around a Moog MB-E-6DOF/12/1000kg Model 170-131 [9] Motion Base (MB), capable of movement in six Degrees Of Freedom (DOFs): three axes of translation and three rotations. It is controlled via Ethernet by the System Control Computer.

2.2 Analysis and Evaluation of the SVI

The SVI has several limitations that hinder its usefulness. X-Plane 9, the flight simulation software in use, is an old, unsupported version. The latest major version, X-Plane 11, was released in March 2017 [10]. Its linear acceleration data output is incorrect and sends phantom accelerations to the motion cueing system when forces are exerted on a stationary aircraft on the ground. Pilots with widebody experience have noted that the flight model for the simulated aircraft does not seem to behave correctly at the edges of the flight envelope. In particular, stall characteristics at high angles of attack were perceived to be lacking in fidelity. Other aspects lacking fidelity include the flaps settings and autothrottle behavior.

In addition to this, the computers used in the SVI are extremely outdated in both software and hardware. The simulation runs at sub-optimal framerate in the 20 frames per second (FPS) range, 30 FPS being the minimum acceptable value and 60 FPS the ideal target. Furthermore, the System Control Computer struggles to run the Simulink model that implements the Motion Cueing Interface System, occasionally dropping output packets long enough for the motion base to detect a communications failure.

![Figure 1: Architecture of the SVI](image-url)
throttle quadrant [4]. The lack of right seat controls and instruments limits the SVI in multi-crew scenarios, where much can be learned. The lack of a physical throttle quadrant, though mitigated with a virtual control on the center touchscreen, was cited by every pilot who volunteered to try out the simulator as a significant hindrance. A more minor recurring complaint is that the sidestick feels too light, due to a weak spring. The instruments on the touchscreen were noted to be cramped, due to limited display area, and hard to use – in part due to inaccuracies in the optical touch technology used, which could not be calibrated for, despite multiple attempts; and in part due to a heavy parallax effect.

Sound reproduction was found to be adequate. However, the persistent pre-recorded air traffic control radio chatter, enabled by default in X-Plane 9, was found to be a nuisance due to its near-constant presence and complete irrelevance to the flight being simulated. Communication between pilots and the simulator operator is hindered by the lack of headsets for simulated radio communications.

Users of the simulator with commercial flying experience have noted that the motion cueing system’s response is not coherent with the simulation. A lack of low-frequency, long-term cues was also cited as a negative aspect of the system.

Previous work done on the motion cueing system has already highlighted other flaws, not directly related to performance. The reliance on MATLAB and Simulink was previously identified as detrimental to the operational workflow of the simulator, along with needless use of UDP broadcasts for communication which could be done using UDP unicast [3]. Jerky movements on startup are also described [4]. Inconsistencies in the motion base’s documentation raised questions about its conversion of degrees-of-freedom commands into actuator extensions, in the version of the software currently installed in the Motion Base Computer. It appears that linear approximations are used for this conversion – although not strictly accurate, the results were deemed good enough to not require immediate fixes. In the future, the conversion may be done on the System Control Computer and the motion base commanded with actuator extensions.

2.3 Areas of intervention

Motion cueing was deemed the highest-priority target, as it is the distinguishing feature of the SVI and most changes can be easily implemented in software, minimizing the uncertainty inherent in acquiring hardware. The motion cueing system must be validated, as it had not yet been subjected to documented, formal testing, neither in isolation nor in conjunction with X-Plane. The problems identified during this process shall be corrected.

In addition, changes to the flight simulation system were analyzed, in particular the flight simulation software and workstations, exterior view and cockpit layout.

3. Motion Cueing

The objective of motion cueing is to approximate the feeling of movement inherent to piloting an aircraft. By moving the simulator cabin in six degrees of freedom, it is possible to suggest many of these stimuli in the comparatively limited envelope of a motion base. The use of motion cueing in flight simulators is important in a wide range of situations, both to aid and hinder the pilot. While cues such as that of the landing gear contacting the runway help by providing feedback that is otherwise unavailable, an overload of cues during tense situations such as an engine failure imposes additional noise that the pilot must be able to filter out to safely resolve the problem. Full Flight Simulators must have a motion cueing system to be certified as such [11].

In the research missions that the SVI is meant for, a capable motion cueing solution is a definite advantage. Among other possibilities, examining human factors with and without motion cueing can allow for determining when to train pilots using static simulators, which are naturally cheaper, or more elaborate simulators with motion cueing.

Previous users of the SVI have noted that the existing motion cueing system did not perform to their expectations, so it will be examined in this chapter.

3.1 Overview of the SVI Motion Cueing System

The SVI’s Motion Cueing System is made up of a Moog MB-E-6DOF/12/1000kg motion base [9], which is controlled by the System Control Computer, which receives the data outputs from the Flight Simulation System.
The motion base uses a Stewart platform, as is typical of motion bases, which leads to movement in one degree of freedom restricting the other degrees of freedom. In other words, the six DOFs “share” the platform and it is impossible to maximize more than one simultaneously – it is necessary to use low-amplitude commands to ensure that all DOFs can be replicated by the platform.

The System Control Computer executes the software that receives flight simulation data and provides commands for the motion base – initially, a bank of filters and integrators, implemented in MATLAB/Simulink, was used. For ease of documentation, this preexisting piece of software was retroactively named “MDA2007”.

MDA2007 receives four quantities, measured in the aircraft frame of reference, from the flight simulator software: linear acceleration vector, angular acceleration vector, angular velocity vector and Euler angles. Angular accelerations are used to offset the linear acceleration vector from the aircraft center of mass to the pilots’ position. Euler angles are used to subtract gravity from the linear accelerations to obtain specific forces. The actual filtering and integration use only the specific forces and angle rates, for a total of six degrees of freedom. The interface outputs the desired position in each of the six degrees of freedom (three axes of translation and three axes of rotation), which the Motion Base Computer (MBC) internally converts to actuator extensions.

3.2 Human Perception of Motion

In order to understand the limitations of MDA2007, it is important to review the theory of operation of an abstract classical washout filter, formalized by Reid and Nahon in 1985 [12]. MDA2007 implements such a system, based on Moog’s MDA software documentation [13].

Besides perceiving motion visually, humans also sense rotation and acceleration, using the vestibular system and, in the case of acceleration, using touch and muscle feedback to feel the force exerted by the vehicle [14].

The quantity sensed in the case of rotation is angular velocity, which is detected by the semicircular canal system, a part of the vestibular system. These can be modelled as three orthogonal non-linear sensors, one for each axis of rotation, each as shown in Figure 3. It acts as a band-pass filter with a non-linear threshold, beneath which there is no detection [12]. The other component of the vestibular system, the otolith, senses accelerations – more specifically, it senses specific forces $f$, defined as the difference between the acceleration vector and the gravity vector [12].

The otolith has also been characterized as three independent non-linear sensors, but with low-pass characteristics, as shown in Figure 4.

Linear accelerations are also sensed in the form of pressure exerted on the body by the aircraft – specifically, by the seat – and by muscles responding to inertial forces exerted on body parts – most significantly the head, as it is a large, unrestrained mass. These senses also present a non-linear low-pass behavior, but cut off around 13 rad/s, higher than the otolith [14].

![Figure 3: Model of the Semicircular Canal system [12]](Image)

![Figure 4: Model of the Otolith system [12]](Image)

3.3 Classical Washout theory of operation

By process of integration, acceleration and angular velocity can be related to position and angle, as used to control the motion base. Naively integrating these would result in replicating the flight – needless to say, the motion base is incapable of doing so and is limited by its actuator extensions. The Classical Washout algorithm addresses this by employing high-pass filters to “wash out” motion base movement [12]. The system comprised of filter and integrator has a response that quickly peaks and then slowly decays back to zero. As a consequence of the Final Value Theorem, the step response of this system tends towards zero if it has a filter of order one higher than that of the integration. This solution replicates high-pass sensations and can be used for both acceleration and angular velocity, but does nothing for constant accelerations, which are a relevant component of flight, as seen previously. These are simulated by Tilt Coordination, the process of rotating the simulator cabin, thereby reorienting the gravity vector perceived by the pilot to provide them with a low-pass acceleration cue in the x- or y-axes. To keep this motion limited to what the motion base can accomplish, scaling and limiting of the inputs is required. A tradeoff between a larger scale factor and a larger limit exists when fitting aircraft motion into a real motion base.

The Classical Washout algorithm for motion cueing is composed of three channels operating mainly on two vector inputs: angular velocity and linear acceleration. It outputs two other vector quantities, namely the Motion Base’s position in translation and rotation. The three channels are the rotational high-pass channel, which high-pass filters and integrates angular velocities to output an angular position; the tilt coordination channel, which applies low-pass filtering to accelerations to obtain an angular position, which is used to simulate low-pass-type accelerations; and the translational high-pass
channel, which high-pass filters and integrates accelerations to output a position.

Figure 5 illustrates the high-level signal flow of this algorithm, including the pre-processing block, which is responsible for offsetting accelerations to the pilot’s frame of reference and subtracting gravity from these to obtain specific forces; as well as applying gains [13]. The acceleration at a point P not coincident with the Center of Mass is given by [15]:

\[
\ddot{a}_P = \ddot{a}_c + \frac{d\ddot{\omega}}{dt} \times \vec{p} + \ddot{\omega} \times (\ddot{\omega} \times \vec{p})
\]  

(1)

Where \(\ddot{\omega}\) is the angular velocity vector and \(\ddot{a}\) is the linear acceleration vector. It should be noted that this calculation is the only one requiring the angular acceleration input \(\frac{d\ddot{\omega}}{dt}\).

The classical washout algorithm is defined as operating on a specific force vector \(\vec{f}\) [12], defined as:

\[
\vec{f} = \ddot{a} - \ddot{g}
\]  

(2)

Where \(\ddot{a}\) is the linear acceleration input and \(\ddot{g}\) the gravitational acceleration felt by the simulated aircraft.

The first step is to convert the angular velocity input, expressed in body axes, to inertial axes in the form of Euler angle rates. This step is needed, as commands to the Motion Base are issued in its Earth-fixed frame of reference. Note that the angle used to determine the rotation applied is the Motion Base’s commanded output. Once expressed in the inertial frame of reference, the inputs are clamped down to their limits, as defined by the algorithm’s tuning parameters. These limited inputs are then subjected to high-pass filtering, followed by a single integrator, resulting in an angle output.

The Tilt coordination channel converts an acceleration input into an angle output. The objective is to allow for the Motion Base to simulate low-pass acceleration by rotating, using gravity to simulate the acceleration in question. Figure 7 describes the algorithm used for Tilt coordination.

\[
\vec{f} = \vec{a} - \vec{g}
\]  

Figure 6: Overview of the Classical Washout algorithm [12] [13]

Like the Rotational High-Pass channel, the first step is to convert the input’s frame of reference from body axes, as provided by the flight simulator software, to the static frame of reference in which commands are issued to the Motion Base. Afterwards, the inputs are clamped down to their limits for this channel, which are tuning parameters independent of the other channels’ limits. This step is followed by a set of gains that linearly map a full-scale input to a full-scale output. After scaling, the signal is subject to low-pass filtering, to prevent high-frequency components of the acceleration signal from being output as high-frequency angular

Figure 7: Overview of the Tilt coordination channel
movements of the simulator cabin. The filter outputs are then rate-limited, in order to keep the Tilt coordination movement below the threshold of human perception, to avoid it being perceived as a rotation. Finally, the result of the Tilt coordination channel is summed with the previously-calculated angular position output, to give the final output that can be sent to the Motion Base.

The Translational High-Pass channel is solely responsible for the translational position of the Motion Base. Its operation is analogous to the Rotational High-Pass channel but operating on linear acceleration inputs and outputting a position via double integration. Figure 8 shows the two main differences to the Rotational High-Pass channel: A double integrator is used instead of a single integrator, since \( \sigma = \frac{d^2x}{dt^2} \), and the input is rotated from body to inertial axes.

These constraints are used to calculate the filter cutoff frequencies and the tilt coordination gain. If the high-pass cutoff frequencies are too high, steps 1 and 2 must be repeated with different values. The cutoff frequencies can be obtained with the following equations:

\[
\omega_{cX} = \sqrt{\frac{f_{\text{lim}X} \cdot k_x}{x_{\text{max}}}}
\]

\[
\omega_{c\phi} = \sqrt{\frac{\Omega_{\text{lim}\phi} \cdot k_\phi}{\phi_{\text{max}}}}
\]

With gains \( k \), translation cutoff frequency \( \omega_{cX} \) and rotation cutoff frequency \( \omega_{c\phi} \). \( x_{\text{max}} \) and \( \phi_{\text{max}} \) are the limits for the output DOF in question, as defined in the motion base partition. \( \Omega_{\text{lim}\phi} \) and \( f_{\text{lim}X} \) are the input DOF limits.

Finally, the filter parameters must be calculated to match the desired cutoff frequencies. The translational high-pass channel uses a third-order filter, because of the use of a double integrator. Similarly, the rotational high-pass channel uses a second-order filter due to the channel’s single integrator. For the tilt coordination channel, no such criterion exists, so a second-order filter was chosen for the sake of simplicity.

### 3.5 Evaluation and Validation Procedure

As no previously-validated solution exists, it was necessary to validate MDA2007 and any future developments. Therefore, a validation procedure was defined. Any changes to the algorithms will be tested according to this procedure. The process consists of three stages: Synthetic testing, to ascertain whether the algorithm’s outputs correspond to the desired outputs; Sanity check, to verify that operation with real inputs will be safe and that it corresponds to what is expected from the synthetic tests; General test flight, where experienced pilots report on their perceptions of the motion cueing system.

For the synthetic testing, a test bench with simplified inputs was created. These inputs, though unrealistic, are simpler to analyze and easier to interpret. Maneuvers were chosen to cover all degrees of freedom.

The sanity check consists of recorded test flights, obtained from X-Plane 11. It is meant to ensure that the motion cueing algorithm behaves correctly with flight simulation software inputs, and that operation will be safe.

The third stage of the process consists of test flights with experienced pilots. These should include basic maneuvers, such as ground operations, turns with various bank angles and crosswind landings. Given the subjective nature of the evaluation, test pilots are provided with a questionnaire, on which answers are given by tracing a line on an analog scale with reference markings [7].
3.6 Analysis and Evaluation of MDA2007

MDA2007 was subjected to the synthetic testbench. The results showed that every test resulted in unwanted behavior. Some examples include a constant Z-axis output that greatly exceeds the motion base’s capabilities, incorrect tilt coordination with high-pass behavior and significant overshoots when angle outputs decay back to zero.

These problems result from incorrectly configured signal limits; an unknown, seemingly arbitrary filter tuning; high-pass filters being used for tilt coordination instead of the correct low-pass filters; and an incorrect subtraction of gravity.

MDA2007 also suffers from being locked to MATLAB, limiting where and how it can be deployed. It also relies on values hardcoded into the Simulink model. Furthermore, the block diagrams that make up the Simulink model for MDA2007 are obfuscated and very difficult to understand. Finally, the GUI for MDA2007 relies on an obsolete library, which provides the GUI’s gauges, in addition to the standard GUIDE development environment. This library has been completely unsupported since at least MATLAB R2016a.

Given these flaws, it was decided that MDA2007 needed to be reimplemented, as it was unusable for serious work. This contrasts with the original assumption that MDA2007 needed only minor work.

3.7 SVI Motion Cueing Development

During development to resolve the outlined issues, the interface was renamed to the “Motion Cueing Interface System” (MCIS). This change is meant to reflect the wide range of fixes implemented and correct the lack of a coherent naming scheme for this simulator component.

Development was split into four stages: MCIS v1a attempts to fix MDA2007’s issues by applying the tuning procedure and correcting the most visible problems, such as replacing the tilt coordination filters with low-pass filters. MCIS v1b reimplements the algorithm from MCIS v1a in a new Simulink model, allowing for better understanding of what it does. MCIS v2a corrects issues that were identified by analysis of the more readable block diagrams, namely the conversions between frames of reference and the angles used for these. Finally, MCIS v2d implements a discrete-time equivalent of MCIS v2a, first in Simulink, to more easily validate changes, and later in C++, for ease of porting to other systems.

At each stage, the validation procedure was applied, as appropriate. Test flights were run for both MCIS v1 and MCIS v2d, the results of which can be used in the future as a baseline against which to compare new developments.

Figure 9 and Figure 10 show the response of MDA2007 and MCIS v1, respectively, to one of the synthetic tests. Although there are ten tests in total, the first test alone shows that MDA2007 has erratic behavior that does not correspond to what is desired. MCIS v1 corrects this and its behavior matches what was desired: Washout behavior on the position output and a low-pass tilt coordination, with maxima in line with those considered in the tuning procedure. MDA2007 does not wash out, does not have correct tilt coordination and does not even keep its outputs within the motion base’s envelope.

MCIS v1b was determined to be functionally identical to MCIS v1a, as desired. The changes in MCIS v2a were found to result in mostly small differences in the outputs. MCIS v2d was found to be very close to MCIS v2a, despite its discrete-time nature, with worst-case full-scale errors of the order of 10^{-4}. Its C++ implementation was found to be extremely close to the Simulink implementation.
3.8 Does subtraction of gravity make sense?

Reid and Nahon define the classical washout algorithm as operating on specific forces, that is, accelerations minus gravity. This approach makes intuitive sense, as the gravity component of the perceived acceleration is introduced by the rotation itself, leaving the acceleration channels of the algorithm to respond to accelerations caused by movement.

This approach is not without problems, as illustrated in Figure 11 with a coordinated turn. By definition, the resulting acceleration, as perceived by the occupants or an onboard accelerometer, is aligned with the body z-axis of the aircraft. This vector, \( \mathbf{a}_z \), is the sum of the gravity vector \( \mathbf{g} \) and the centrifugal acceleration vector \( \mathbf{a}_R \):

\[
\mathbf{a}_z = \mathbf{g} + \mathbf{a}_R
\]

Combining equations (2) and (5), it is evident that the classical washout algorithm will respond only to \( \mathbf{a}_R \):

\[
\ddot{f} = \dot{\mathbf{g}} + \mathbf{a}_R - \mathbf{g} = \mathbf{a}_R
\]

![Figure 11: Accelerations in a coordinated turn, adapted from [16]](image)

The effect of this behavior is that a constant bank angle of the aircraft, in a coordinated turn, will result in a constant bank angle of the simulator cabin in the same direction. This effect is described by Reid and Nahon [12], but is not examined to any extent. No comment is made on whether it is acceptable, desirable, harmful or irrelevant.

The alternative solution, not subtracting gravity and operating on accelerations instead of specific forces, introduces other errors. Given a pitch movement, e.g. from zero to positive ten degrees, the x-acceleration component will tend towards a negative value, as the x-axis of the aircraft is aligning itself with \(-\mathbf{g}\). This negative acceleration leads to tilt coordination tilting the cabin’s nose down, as tilt coordination is defined so as to respond to movement of the aircraft. Gravity, unlike these accelerations, acts in the same direction on the aircraft and its occupants, breaking tilt coordination.

Although subtracting gravity is, at first glance, the most appealing of the two options, as its error is in the same direction as the movement and it is the method described by Reid and Nahon, as well as other work based on theirs, the notion of a coordinated turn with an acceleration vector not aligned with the Z axis is dubious from an engineer’s point of view.

To better examine this issue, a volunteer airline pilot was recruited to run eight test flights, with and without gravity subtraction. Test flights 1 through 5 were executed with no wind or turbulence. Test flights 6 through 8 were done during a single simulated flight, without landing, and focused on more complex maneuvers. After each test, the pilot was asked to fill out the standard questionnaire.

Curiously, pitch movement was given, on average, a higher score (16.33, \( \sigma = 4.06 \)) in the no gravity subtraction tests than in the standard tests with gravity subtraction (14.00, \( \sigma = 2.08 \) – although this could easily be attributed to the variance inherent to the answers. Most other scores are better for gravity subtraction.

In test flight 4, which was considered to have the worst false cues, the pilot noted that the false cues observed were high frequency noise when combining roll and yaw movements, describing it as “stick shaker-like”. These do not seem to be relevant to the current tests. In a post-testing interview, the pilot expressed a preference for the version of MCIS with gravity subtraction. In particular, they stated that they had not perceived the constant roll of the cabin as a false cue.

Given this, it was decided that the SVI would adopt the use of specific forces, as described by Reid and Nahon. Further studies in this field may reveal a hitherto unknown solution that is more appropriate than either of those tested.

4. Flight Simulation System

Besides the the Motion Cueing system, other components of the SVI were analyzed and improved upon. In this chapter, these non-Motion Cueing changes are presented. Furthermore, a number of improvements to the Flight Simulation System, both in terms of hardware and software are proposed for future implementation.

4.1 SVI network IP addresses

The SVI operates on an isolated, private network. This network uses standard IEEE 802.3 Ethernet [17] hardware, with Internet Protocol version 4 (IPv4) [18] as the network layer.

It was found that this network had devices configured on two different subnets, without a router, and worked via brute-force broadcasting of packets. Furthermore, neither subnet used a private IP address range.

To fix these issues, all devices were assigned IP addresses in the 192.168.20.0/24 subnet, which is reserved for private use [19].
4.2 Flight Simulation Software

The SVI currently uses X-Plane 9 for all simulation, including graphics, sound and aircraft dynamics. X-Plane 9 suffers from problems such as incorrect data output on the ground and dubious aircraft flight models.

X-Plane 11 is the current major version of X-Plane, which has seen significant development since X-Plane 9. According to its developers, X-Plane 11 strives to realistically simulate its default aircraft, in particular the Boeing 737-800. Several third-party aircraft are available for purchase, if desired. X-Plane 11 has also been shown to correct the acceleration data output on the ground, no longer reporting phantom accelerations when the aircraft has the brakes set.

Upgrading to X-Plane 11 requires adapting MCIS to a new data format, as X-Plane 11 has removed the angular acceleration data output. A new output, Motion Platform Stats, would seem to provide exactly the data needed for motion cueing, at the pilot’s position. Unfortunately, its output is dominated by ridiculously large longitudinal accelerations.

X-Plane 11 has substantially higher system requirements than X-Plane 9 [20] [21]. The hardware currently in use is barely capable of handling X-Plane 9, with the single projector – typical frame rates hover around 20 FPS – so upgrading to X-Plane 11 necessitates new simulator workstations. Benchmarks suggest that a high-end gaming video card is required.

4.3 Improved cockpit layout

The existing flight deck layout is not ergonomic and does not come close to any aircraft or category of aircraft. The concentration of functions in the touchscreen significantly hinders flight operations, mostly due to the lack of a set of physical throttle levers. Pilot interviews suggested that throttle levers are a fundamental requirement, with speed brakes and flap controls being desirable functions to have available in a physical format. An additional hindrance is the number of instruments and controls presented in the sole touchscreen, which significantly limits their size so much as to make them nearly unusable. Furthermore, the lack of controls and instruments dedicated to the right seat precludes usage of the simulator with an active copilot and seriously limiting the role of pilot-monitoring.

These limitations can be addressed by replacing the existing monitors with four new touchscreens, laid out as per Figure 12: adding a throttle quadrant; and adding pedals and joystick for the right seat.

4.4 Exterior view

The exterior view suffers from a narrow field of view, low resolution, and poor attachment to the simulator cabin.

A cylindrical projection setup, with three projectors and a cylindrical screen, would allow for a near-180° field of view. Its main drawback is the need for a rigid structure to hold the screen (and possibly the projectors) in place relative to the cabin, even as the platform moves.

Edge blending was found to provide good results in eliminating seams between projectors, even on low-end projectors of different characteristics.

![Figure 12: Proposed layout of instruments and controls](image)

5. Conclusions

The original motion cueing interface, MDA2007, suffered from serious flaws both from an operational and a development standpoint. These were not fully understood at the beginning of the project and had to be painstakingly identified. To address these issues, MCIS was developed as a replacement and a procedure for its validation was laid out and executed. MCIS versions 1a, 1b, 2a and 2d represent a growing understanding of MDA2007 and of what changes were necessary. From a developer’s point of view, the improvements are undeniable as the algorithm in MCIS is clearly laid out and is also implemented in C++, which allows for increased flexibility. From a user’s standpoint, early testing with volunteers is encouraging and will serve as a baseline for further development of both the MCIS implementation of the classical washout algorithm and its tuning parameters. Synthetic testing suggests that the changes made from MDA2007 to MCIS resulted in better motion reproduction.

The question of whether to subtract gravity from the acceleration input was analyzed. The decision was made to subtract gravity, as it appears to be the approach commonly taken in the field of motion cueing. Furthermore, it was considered to be the lesser evil in a tradeoff between erroneous outputs in the same direction as aircraft movement, as is the case when gravity is subtracted, and erroneous outputs in the direction opposite to that of the aircraft’s movement, as happens when gravity is not subtracted. Experimental results with a single volunteer reinforced this decision, but further study may be warranted.
6. References


