ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΜΗΧΑΝΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΤΟΜΕΑΣ ΒΙΟΜΗΧΑΝΙΚΗΣ ΔΙΟΙΚΗΣΗΣ &
ΕΠΙΧΕΙΡΗΣΙΑΚΗΣ ΕΡΕΥΝΑΣ

Development and evaluation of a wearable motion tracking system for sensorimotor tasks in VR environments

---

Ανάπτυξη και αξιολόγηση ενός ενδυόμενου συστήματος ανίχνευσης κίνησης για κιναισθητικά καθήκοντα σε περιβάλλοντα εικονικής πραγματικότητας

Διπλωματική Εργασία

Μουρελάτος Ανδρέας

Επιβλέπων Καθηγητής:
Ναθαναήλ Δ.

Αθήνα

Φεβρουάριος 2018

Ευχαριστίες

Η παρούσα Διπλωματική Εργασία σηματοδοτεί, εκτός από πέρας των προπτυχιακών μου σπουδών, την ολοκλήρωση του μεγαλύτερου, πιθανότατα του ποιοτικότερου εγχειρήματός μου ως φοιτητής. Ωστόσο, το εγχείρημα αυτό δε θα μπορούσε να αποδώσει καρπούς χωρίς τη συμβολή ενός αριθμού ανθρώπων, οι οποίοι συνέβαλλαν, ο καθένας με τον δικό του ξεχωριστό τρόπο, στην πραγμάτωση αυτής της εργασίας. Αυτούς τους ανθρώπους θέλω να ευχαριστήσω ειλικρινά σε αυτό το σημείο.

Πρωτίστως, θέλω να ευχαριστήσω τον επιβλέποντα καθηγητή κ. Δημήτρη Ναθαναήλ. Από την πρώτη συζήτηση σχετικά με τη διπλωματική εργασία αυτή, μέχρι και την τελευταία ημέρα της συγγραφής της, επέδειξε καθημερινή πρόθεση να συνδράμει αφιερώνοντας χρόνο, ενέργεια και πραγματικό ενδιαφέρον σε κάθε νέα υπόθεση ή ζήτημα που πρόκυπτε. Παρότι το γνωστικό αντικείμενο της εργασίας δεν συμπίπτει εντελώς με το επιστημονικό του πεδίο, παρείχε ανελλιπή υποστήριξη με κάθε δυνατό τρόπο σε κάθε κομμάτι της, αλλά και αστείρευτη υπομονή απέναντι στον καταιγισμό ερωτήσεων στον οποίο τον υπέβαλλα.

Στη συνέχεια, θέλω να εκφράσω την ευγνωμοσύνη μου στο υπόλοιπο προσωπικό της μονάδας Εργονομίας. Στον Κώστα Γκίκα για την συνεισφορά του σε κάθε ζήτημα σχετικό με το σχεδιασμό και την κατασκευή του συστήματος και του περιβάλλοντος, και την αδιάλειπτη διάθεση του να συμμετέχει σε ανταλλαγή ιδεών και απόψεων σχετικά με κάθε μέρος της εργασίας. Στο Λοΐζο Ψαράκη για τις πολυάριθμες φορές που αφιέρωσε το χρόνο και το χώρο εργασίας του για την υποστήριξη αυτής της εργασίας, προσφέροντας γνώση, υπομονή και εποικοδομητική κριτική σε κάθε της βήμα.

Κυρίως όμως θέλω να ευχαριστήσω όλο το προσωπικό της μονάδας Εργονομίας για την ικανότητα τους να δημιουργούν ένα κλίμα εργασίας πιο φιλικό και ευχάριστο απ’ όσο θα μπορούσα ποτέ να υποθέσω. Η στάση και η υποστήριξη τους κατάφερε να καταστήσει τη διαδικασία της συγγραφής αυτής της διπλωματικής μία αυθεντικά ευχάριστη εμπειρία, την οποία θα ανακαλώ με χαρά στο μέλλον.

Κλείνοντας, ευχαριστώ θερμά όλους όσους συμμετείχαν στα πειράματα, καθώς και τους συμφοιτητές και φίλους που με υποστήριξαν ποικιλοτρόπως καθ’ ολη τη διάρκεια των σπουδών μου και αυτής της διπλωματικής. Ιδιαίτερη μνεία αξίζει σαφώς στο Μιχάλη Καρακικέ, του οποίου η δουλεία και προτροπή αποτέλεσε έμπνευση και εφαλτήριο για την παρούσα εργασία, και στον Πέτρο Κοντραζή για την σημαντικότατη συνεισφορά του στη σχεδίαση του Εικονικού Περιβάλλοντος.

Περίληψη

Η πρόοδος της τεχνολογίας στους τομείς των υπολογιστών, αισθητήρων και συσκευών απεικόνισης καθιστά τα περιβάλλοντα Εικονικής Πραγματικότητας διαρκώς ρεαλιστικότερα και οικονομικότερα. Οι άνθρωποι μπορούν πλέον με ευκολία να εμβυθιστούν σε Εικονικά Περιβάλλοντα μέσω οπτικών μέσων.

Ωστόσο, για να καταστεί εφικτή η πλήρης εκμετάλλευση των πλεονεκτημάτων που προσφέρουν αυτές οι τεχνολογίες, είναι σημαντικό να παρέχουν τη δυνατότητα ελέγχου ενός «εικονικού σώματος» εντός του περιβάλλοντος, και ιδιαίτερα των άνω άκρων. Η ενσωμάτωση της κίνησης των χεριών σε τέτοιου είδους περιβάλλοντα όχι μόνο αυξάνει την αίσθηση εμβύθισης του χρήστη στο περιβάλλον, άλλα ανοίγει νέες διόδους για την εξερεύνηση και διάδραση με το περιβάλλον. Επιπρόσθετα, διευρύνει τις δυνατότητες προσφοράς εκπαίδευσης και αρωγής ανθρώπων σε ποικίλα καθήκοντα.

Χρησιμοποιώντας τρείς Αδρανειακούς Αισθητήρες Κίνησης (Inertial Measurement Units, IMUs) και έναν μικροεπεξεργαστή Arduino Nano, αναπτύξαμε ένα πλήρες σύστημα ανίχνευσης κίνησης του άνω άκρου, από τον ώμο έως και την παλάμη. Το σύστημα αυτό χρησιμοποιήθηκε σε συνδυασμό με μία κάσκα Oculus Rift DK2 και ένα περιβάλλον σχεδιασμένο με την Unity Game Engine για τη δημιουργία ενός καθήκοντος «σκοποβολής» σε Εικονικό Περιβάλλον. Το παραπάνω σύστημα επέτρεπε τον πλήρη έλεγχο ενός εικονικού χεριού εντός του περιβάλλοντος μέσω της κίνησης του χεριού του χρήστη στον πραγματικό κόσμο.

Στην παρούσα εργασία παρουσιάζεται μια σειρά πειραμάτων τα οποία σχεδιάστηκαν με σκοπό να διερευνηθεί εάν η ορατότητα των κινήσεων του χεριού εντός ενός Εικονικού Περιβάλλοντος αυξάνει την απόδοση του χρήστη σε κιναισθητικά καθήκοντα εντός του περιβάλλοντος αυτού. Επεκτείνοντας τα παραπάνω, διερευνούμε επιπλέον το βαθμό στον οποίο το άτομο μπορεί να αφομοιώσει μια εικονική αναπαράσταση του άκρου του στην αντιληπτή εικόνα του σώματός του, και στη συνέχεια να ενσωματώσει την αναπαράσταση αυτή στο ανάλογο κινησιακό σχήμα , καθώς και την ταχύτητα και ευκολία με την οποία συμβαίνει η αφομοίωση αυτή.

Τα αποτελέσματα των πειραμάτων παρέχουν στοιχεία σχετικά με την επίδραση της ορατότητας του χεριού στην επίδοση και την καμπύλη εκμάθησης στο καθήκον. Δεν παρατηρήθηκε σημαντική συσχέτιση ανάμεσα στην ορατότητα και την επίδοση. Η επικρατούσα υπόθεση ήταν ότι αυτό οφείλεται στη φύση του πειραματικού καθήκοντος. Για τον λόγο αυτό το καθήκον διαφοροποιήθηκε, επιτρέποντας τη διερεύνηση των αλλαγών στη στρατηγική και τα αποτελέσματα των χρηστών.

Executive Summary

Advances in computing, sensor and display technology mean that immersive virtual and augmented reality environments are becoming increasingly more realistic and affordable. Humans can now easily experience virtual environments (VEs) through visual means.

However, in order to reap the full range of benefits these technologies can offer, it is of essence to provide the ability to control an avatar body within a VE and especially the upper limbs. The integration of arm movements in such environments both increases the immersion of the user, and opens new pathways for exploring and interacting with the VE. Moreover, it allows for increased capabilities in training and assisting humans in various tasks.

Using three MPU9250 IMUs and an Arduino Nano microprocessor, we developed a wearable system for unobtrusively tracking the movement of the arm from the shoulder up to and including the palm. The system was used in conjunction with an Oculus Rift DK2 and an environment designed in the Unity Game Engine, to create a VE “shooting target practice” task. The above setting allowed a virtual arm to be completely controlled by the movements of the user’s arm in the real world with negligible lag.

In this thesis a series of experiments will be presented, designed to determine whether the visibility of one’s limb movements in real time in a VE, improves the effectiveness in the execution of sensorimotor tasks within that environment. Extending from the above, we also explore the degree to which a human can assimilate a virtual representation of their arm with their body image, and subsequently incorporate this representation into their body schemas, as well as the speed and ease with which that assimilation occurs.

The results of the experiments provide evidence on the effect of avatar arm visibility on the users’ performance and learning curve in the VE task. Visibility did not appear to have a significant effect on performance. Our assumption is that this is related to the nature of the experimental task. For this reason the experimental task was changed, allowing us to explore the diversification of the subjects’ strategy and results.

Contents

A. Table of Figures

B. Table of Tables

C. Table of Charts

1 Introduction

Advances in computing and display technology have rendered Virtual Reality a rapidly evolving field of research, with a constantly growing number of applications and an increasingly commonplace presence in everyday life. VR environments ant technologies are constantly becoming ever more inexpensive and readily accessible to humans, and seeing use for purposes exceeding entertainment.

The interest of the scientific community in Virtual Environments (VEs) has been both extensive and manifold, concerning issues ranging from the possible applications of such technologies in various aspects of life to the psychological effect that virtual stimuli can have on users.

The present thesis revolves around two main objectives

The first of these was the re-design and adaptation of an innovative technology originally developed for measuring the angles of the wrist and forearm by the NTUA Lab of Cognitive Ergonomics, for use inside VR environments. This included the construction and programming of a functional prototype, as well as the development of a VE in which the prototype could be tested.

Secondly, after the completion of the above, a series of experiments were designed and conducted, with the aim of both evaluating the prototype and exploring certain issues regarding limb visibility and performance inside VEs. Namely, it was hypothesised that the ability to see, as well as control, the entirety one's upper limbs inside a VE would have a positive effect on the performance and experience of the humans operating the system. This hypothesis challenges the practice currently employed by various existing VR systems, which tend to display only the hands and fingers (e.g. Oculus Rift, HTC Vive, and Manus VR).

2 Theoretical background: VR and Avatar Bodies

In this section, the theoretical basis for this thesis is presented. The concepts of immersion and presence are explained as used in the context of VEs. Moreover, the term “avatar body” and its meaning are introduced. Research related to the representation and control of body parts within immersive VEs is also presented.

2.1 Immersion & Presence

Immersion is a crucial factor when judging the quality of a Virtual Environment (VE). As described by Slater & Wilbur (1997), immersion is a distinct concept from presence. Presence describes the user’s psychological sense of being *inside* the virtual world and is a central goal of Virtual Reality (Steuer, 1992). Immersion, by contrast, refers to the ability of the utilised technology to create “an inclusive, extensive, surrounding and vivid illusion of reality” (Slater & Wilbur, 1997)—essentially a world providing the user with a sense of presence. It is worth noting at this point that while an increase in immersion is often observed to have an analogous effect on the performance of users in tasks within the VE, no such conclusion can be drawn regarding an increased sense of presence. This is attributed to the fact that while increased immersion results in an increase of the quality of the virtual world, allowing for better performance in tasks within it, increased presence regards the similarity of the actions performed in the real and virtual worlds and as such has no effect on the performance in these tasks. (Slater et.al, 1996)

2.2 Avatar Body

One of the main components of immersion is self-representation within the VE—what we call a *virtual* or *avatar body.* An avatar body is both a part of the environment perceived by the user, and a representation of that user’s physical body within the VE. For the sense of self-representation to be complete, a match is required between the user’s proprioceptive feedback about the movements of their physical body and those performed by the avatar body—essentially, the ability to control the avatar body by one’s own movements. To achieve this, of course, real-time tracking of the head and body movements of the person operating in the VE is required, as well the ability to translate these movements into the corresponding ones for the avatar body with minimal lag.

An avatar body need not be similar in appearance and anatomy to the user’s physical body. As has been demonstrated by research and asserted by users of VR applications, humans can experience an illusion of ownership of bodies largely different from their own, and adapt their existing body schemas to facilitate these bodies. This observation merits research, as it not only provides insight into the function of human cognition, but also allows humans to incorporate novel training and interaction schemes within VEs.

2.3 Related research - immersive VR with First Person integrated body parts

Avatar body representation and control in VEs allows for many types of research in various fields such as game development, H-R collaboration, training, medical rehabilitation, ergonomics, etc. Examples of such research include:

Lange et.al (2011), used the PrimeSense depth sensor technology (also utilised in the Microsoft Kinect) to develop and evaluate a game-based rehabilitation tool for the balance training of adults after neurological injury. Wittmann et. al (2015) developed a VR therapy game that continuously estimates the patient's arm reachable three-dimensional (3D) workspace based on Inertial Measurement Units (IMUs). Luo et. al (2011), created an interactive VR system for both arm and hand rehabilitation utilising both optical linear encoders (OLEs) and IMUs. Osumi et. al (2017) developed a quantitative method to measure movement representations of a phantom upper limb, and investigated whether short-term neurorehabilitation with a VR system would restore voluntary movement representations and alleviate phantom limb pain (PLP), using a combination of the Microsoft Kinect and Leap Motion. Merians et. al (2009) developed a complex system capable of exercising the arm and hand together or in isolation, providing for both unilateral and bilateral hand and arm activities in three-dimensional space. The system incorporated CyberGlove instrumented gloves for hand tracking and a CyberGrasp exoskeleton for haptic effects in a number of VR simulations. Moreover, MRI imaging was used to observe the engaged areas of the brain, in order to test the feasibility of using VE-based sensory manipulations to recruit select sensorimotor networks.

All of the above research yielded encouraging results regarding the rehabilitation of patients, as well as gaining positive feedback from both patients and clinicians, thus demonstrating the applicability of combining motion tracking technologies and VR environments for rehabilitation purposes, and especially for offering a solution for the at-home rehabilitation of patients, in an enjoyable environment.

Heidicker et. Al (2017) studied the effect of avatar appearance and motion control on communication and interaction in social virtual reality scenarios within immersive VEs. To that end, three different types of avatar in different VEs were compared. The results demonstrated that motion control of avatar bodies plays an important role in the sense of presence within the VE, with full body avatars with full motion control exhibiting the best results regarding co-presence and behavioural interdependence. It is worth noting however that avatars consisting of head and hands with motion control showed better results than complete avatar bodies with pre-defined animations. A question left open for future research by the paper was how many and which body parts have to be visible to reproduce or even surpass that degree of co-presence.

Specifically regarding the field of Ergonomics/Human Factors, by studying the relationship between an avatar body and the person operating it, it is possible to draw valuable conclusions regarding issues such as:

- The correlation between avatar body control and task effectiveness in a VE
- The user's ability to assimilate a virtual representation of their body with their real-world body image

- The ability to subsequently incorporate this representation into their body schemas.

For example, Kilteni et.al (2013) observed that subjects' behavioral and movement patterns within a VE can change depending on the visual aspects of an avatar body within said VE, by simulating a drumming task with avatar bodies of different skin tones and clothing. Moreover, it was observed that a stronger body ownership illusion corresponded with a greater behavioral change of the subjects. Slater et.al (2010) studied the illusion of ownership of an avatar body different than the subjects' physical one, demonstrating that a virtual female body that appears to substitute the male subjects' own bodies was sufficient to generate a body transfer illusion.

In a different study, Kilteni et. al (2012) studied the ability of subjects to incorporate an avatar body exhibiting asymmetries in comparison to their physical one into their body schemas. This was achieved by creating elongating one the users' virtual arms to up to 4 times their normal length, and using questionnaire scores and defensive withdrawal movements in response to a threat to measure the degree of ownership of that arm experienced by the users. Results showed that users experienced a sense of ownership towards the elongated limb and were able to adapt their responses to this unnatural body image. That illusion did decline, however, with the length of the virtual arm, especially when the virtual arm exceeded three times the length of the physical one.

Won et al. presented congruent results by examining the concept of "homuncular flexibility"—the idea that humans can learn to control bodies different from their own by changing the relationship between tracked and rendered motion. To that end, the researchers conducted two different experiments. In one, the movements of the upper and lower limbs of the users real and virtual were remapped, making the physical arms control the virtual legs and vice versa, or attributing far increased range-of-motion to the virtual legs than the arms. In the other, a third arm was added to the avatar body, controlled by the rotation of the users' physical arms. The results of both experiments demonstrated that subjects were able to adapt their body schemas to the virtual bodies' capabilities, quickly learning how to utilise their more flexible limbs in the first experiment and their "third arm" in the second one to achieve better performance in tasks when compared to "normal" body representations.

3 Review of Motion Tracking Technology

3.1 Importance of arm tracking

The upper limbs play a paramount role in the experience of a VR user's avatar body, as they provide the main tool for interacting with the environment. In light of this, various technologies have been developed to incorporate the arms' movement into the VE, utilising different approaches, each with its own benefits and limitations.

3.2 Existing systems

In this section, some of the most popular VR technologies are presented, along with their working principles and approach regarding position and movement tracking.

3.2.1 Microsoft Kinect

Figure 1: Microsoft Kinect

The Microsoft Kinect, originally released in November 2010, is a motion sensing device that allows the user to interact with a computer or gaming console (namely the Xbox 360/One) without need of a controller, using gestures and spoken commands. While the Kinect itself is not, strictly speaking, a VR technology, since it does not display a VE to the user, it has been used extensively in conjunction with other systems such as the Oculus Rift (presented below) due to its motion tracking capabilities.

The Kinect achieves motion tracking by combining the data acquired from two sensors:

i) An RGB camera
ii) An IR emitter and an IR sensor

The working principle is as follows: A speckle pattern of infrared laser light is first projected onto the scene by the IR emitter. The IR sensor acquires the reflected pattern and analyses with structured light algorithms in order to compute the depth map of the scene. In other words, a depth value is assigned to each pixel of the image. The Kinect combines this with the information acquired by the RGB camera, in order to produce information about the red, green and blue colours and the distance from the sensor for each pixel of the image.

The acquired map is then segmented in order to recognize human silhouettes. Body parts are inferred using a randomized decision forest, learned from over one million training examples (Berliner et al., 2010; Shotton, 2011) and matched to one of 15 body models. From the knowledge of location and attitude of body parts, the position of the articular joints between them are computed (Valentini, 2012).

While the Kinect's motion tracking capabilities are impressive, its design makes it subject to certain limitations. Namely, the objects (or users) tracked need to be within Line-of-Sight of the camera and IR sensor at all times. This means that users need to stand in front of the sensors when operating Kinect-based systems, and be directly "visible" by them (not standing behind objects) otherwise the sensing system suffers from occlusions. Moreover, the speckle pattern projected by the IR emitter is corrupted by natural light, rendering the Kinect incapable of being used outdoors. It is also worth noting that, due to a lack in popularity, the Kinect was recently discontinued by Microsoft

3.2.2 Oculus Rift

Figure 2: Oculus Rift

The Oculus Rift is perhaps the most well-known VR technology currently available to consumers. Initially funded via a Kickstarter campaign started in 2012, Oculus released two Development Kits (DKs) in 2013 and 2014, and finally released the first consumer version on March 28, 2016. The Rift hardware includes a Head-Mounted Display (HMD), an IR sensor consisting of a specially filtered camera in the form of a desk lamp, as well as two handheld controllers known as Oculus Touch.

The Oculus Rift approach to motion tracking is based on the Constellation system. The Rift HMD as well as the Oculus Touch controllers are fitted with a series of precisely positioned infrared LEDs under or above the surface, set to blink in a specific pattern. The pattern is recorded by the infrared sensor which is usually placed on the user's desk. By knowing the configuration of the LEDs on the objects and the pattern at which they blink, the system can determine the position of each LED as a point in space. By combining these points to create a 3D wire frame, it is able to pinpoint the precise position and orientation of the tracked device (HMD or controller) with sub-millimeter accuracy and near-zero latency. This data is combined with information obtained through gyroscopes and accelerometers embedded in the devices, as well as prediction algorithms, to provide continuous position and orientation tracking. Due to this architecture, previous versions of the Rift suffered from occlusions when the user was facing away from the sensor, as the system was unable to track the LEDs. This was tackled in the consumer version, however, with the inclusion of tracking LEDs in the back of the headset.

It should be noted that two separate IR sensors are required to track the HMD and Touch controllers at the same time, since a single sensor could be easily confused and occluded by one or more of the Touch controllers and hence block tracking of the other controller, the headset, or both.

Moreover, the Oculus IR sensors are still prone to occlusion issues, since any object (walls, furniture, or even the user's hands) interposed between the sensor and the tracked devices "hides" the LEDs from the sensor, resulting in a loss of tracking data. Furthermore, since Constellation is limited by the resolution of the cameras, at a certain distance away from the camera the devices take up too few pixels and the system can no longer identify the individual points corresponding to each LED.

Figure 3: Oculus IR Sensor

3.2.3 HTC Vive

Figure 4: HTC Vive

The HTC Vive is a virtual reality headset developed by HTC and Valve Corporation.The first Consumer version of the device was released on April 5th, 2016.

Its main advantage over previous existing systems is the ability to provide room-scale VR with motion tracking, allowing the user to move in space freely.

This is implemented by what Vive names “Lighthouse” technology. The technology’s working principle is similar to that of Constellation, with a few important differences. The Vive headset and controllers are accompanied by a pair of “base stations” or “beacons", which provide the means for positional tracking of both the HMD and controllers. These need to be placed within the room where the Vive will be used at an elevated height. The stations emit pulses of non-visible light from stationary Infrared LED arrays within them, along with two laser beams, which sweep the room at periodic intervals (60 fps). The HMD and controllers feature a large number of photosensors (37 on the headset, 24 on each controller) which detect the light emitted from the base stations, both the LED pulses and the beams. By measuring the time elapsed between detecting and LED pulse and a laser beam, the position of each photosensor in space can be determined. By combining data from enough of these sensors, as well as an accelerometer-gyroscope combination within the headset, it is possible for the Vive’s software to pinpoint both the position and orientation of the headset and controllers in 3D space.

By positioning the base stations higher than the level at which the user operates the HMD and controller, Vive manages to omit occlusion issues, since the beacons are able to “look down” at the entirety of the room without being obstructed by interposing objects. Moreover, even if one of the beacons is unable to detect the equipment, correct placement of the other one should ensure accurate tracking (see Fig. 5) In addition to the above, since the tracking technology is not dependent on any form of camera, rather utilising simple clocks and light detectors, the resolution issues

appearing in the Oculus hardware are not encountered by Vive. The obvious downside to this is the requirement for proper placement of the base stations within the space where the system is used.

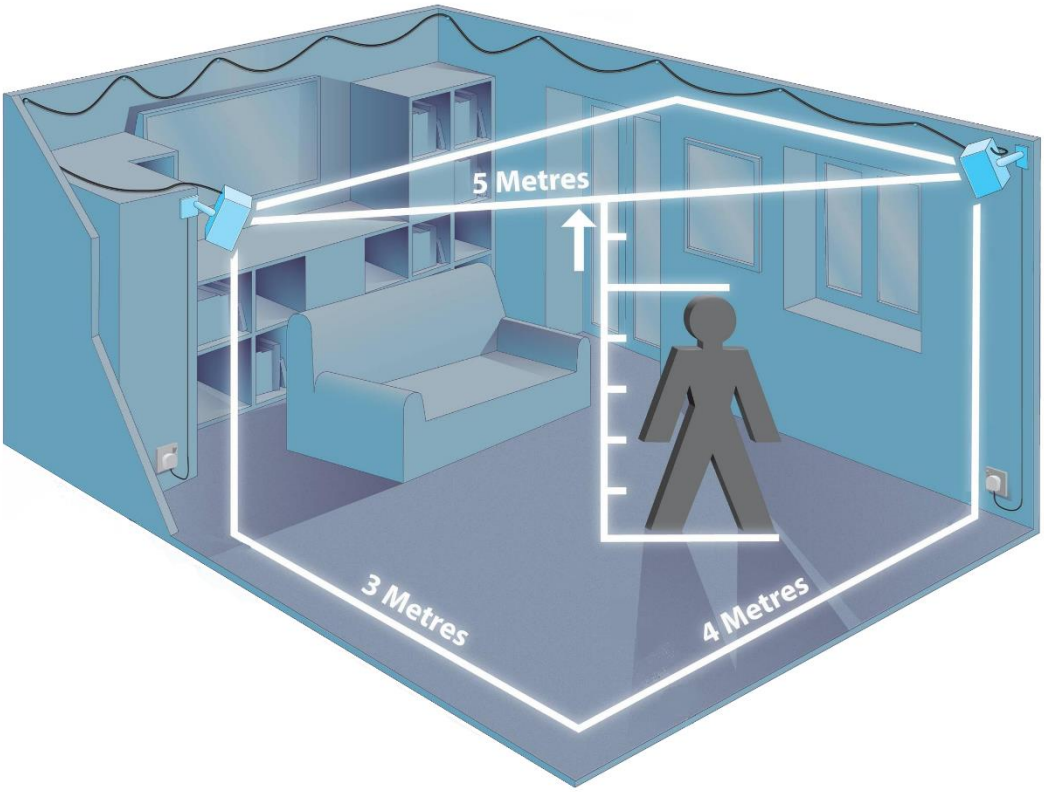


Figure 5: Correct placement of the Vive Base Stations

3.2.4 Manus VR

Figure 6: Manus VR

The Manus VR is one of the latest additions to VR motion tracking, and focuses on incorporating the movements of the fingers and palm into a VE. The gloves contain two flexible analog sensors to track the movement of each finger, as well as one 9DOF Inertial Measurement Unit (IMU) which tracks the orientation of the thumb, and another for the orientation of the hand's dorsal surface. Details as to how the data from these sensors is processed remain unreleased, as does the final product. A video was recently released demonstrating a combination of Manus VR with the Vive HTC controller, to provide full arm and hand control to the user.

4 Development of a Motion Tracking System

4.1 Hardware/Wearable

As part of this thesis, a motion tracking system was developed, capable of tracking the movement of the human arm beginning from the shoulder up to and including the palm.

The wearable system was based on the one previously constructed by Michael Karakikes (2017), with a number of changes made to accommodate the needs of the current research. It consists of three MPU 9250 IMUs connected to an Arduino Nano with 24AWG wires (the connections are explained in section 5.1.2, “Wiring/Connections”) and mounted on a fingerless glove and elbow patch. The main characteristics which favored the selection of IMU technology are:

- Inexpensiveness
- Small size, non-intrusiveness
- No obstruction of the user’s movement
- Sufficiency of accuracy and precision, for arm motion tracking and visualisation(< 3°)
- Availability of prototyping platforms
- Portability, ability for wireless communication
- Independent function from specific VR equipment (HMDs, software, etc.)

Figure 7: Proprietary Motion Tracker

Although various systems implementing arm tracking in VR already exist, with the most popular and recently developed ones being described above, the system developed by our lab retains certain benefits not found in existing systems. The main advantage it provides is the fact that it can function independently from any type of position tracking technology or display. The data from the IMUs goes directly from the Arduino to the PC, without needing to be within Line-Of-Sight of any externally mounted camera or position tracker, rendering it entirely portable. This characteristic would make it usable not only within various different VR environments, but in augmented and real world applications as well, requiring only access to a computer in order to send and receive data, a process which can be achieved wirelessly.

This feature, stemming from the fact that the system was not originally designed to be used in VR, is not found in any of the aforementioned motion tracking systems. Even the Vive, despite its impressive accuracy, depends on the light-emitting base stations to provide “room-scale VR”. The system presented here, by contrast, can be used in different rooms or even outdoors, as long as wireless support is provided. In this manner, it paves the way for what we have dubbed “world-scale” VR and, perhaps more importantly, AR applications. It is worth noting here that the Manus VR, although its functional details are unclear, seems to utilise technologies which also function without need of external position tracking. This could mean that, by combining it with our proprietary motion tracking system, we would be able to provide complete tracking of the arm and hand on world-scale applications.

Moreover, since IMUs are very inexpensive to obtain, the system has a very low construction cost; excluding the equipment required to perform the soldering connections, the cost required to build the existing prototype was 15€ (4€ for the Arduino, 3x3€ for the sensors, 2€ for the glove and elbow patch). It should be noted that this does not come at the expense of accuracy, since IMUs are capable of delivering adequately accurate measurements of a human arm’s motions, as is made evident by their use in commercial VR in systems such as the Oculus Rift, HTC Vive and Manus VR, as well as related research such as the rehabilitation-oriented systems presented in section 3.3, “Related Research” (Merians et. al 2009, Wittmann et. al. 2015, etc)

This system was used to control the movement of a virtual human arm, in a virtual environment created with the Unity Game Engine, and displayed to the user through the Oculus DK2 Head-Mounted Display (HMD) explained in depth in section ??.??, “Environment Design”

4.1.1 Components

4.1.1.1Arduino Nano

For this project, a version of the *Arduino Nano v3.0* (Fig. 8) board was used as a host processor device, to capture IMU measurements and process them, in order to calculate the orientation and movements of the user’s arm. The board was powered through a Micro-USB connection with a PC. The same connection was used to exchange data with the computer via the *serial monitor* included in the Arduino software, which allows simple textual data to be sent to and from the board. The Arduino receives data from the three MPU 9250 IMUs utilising I2C communication through the A4 (SDA) and A5 (SCL) pins. I2C communication is supported by the

appropriate libraries. The Arduino Integrated Development Environment (IDE) was used for writing the code used in the project and uploading it to the processor.

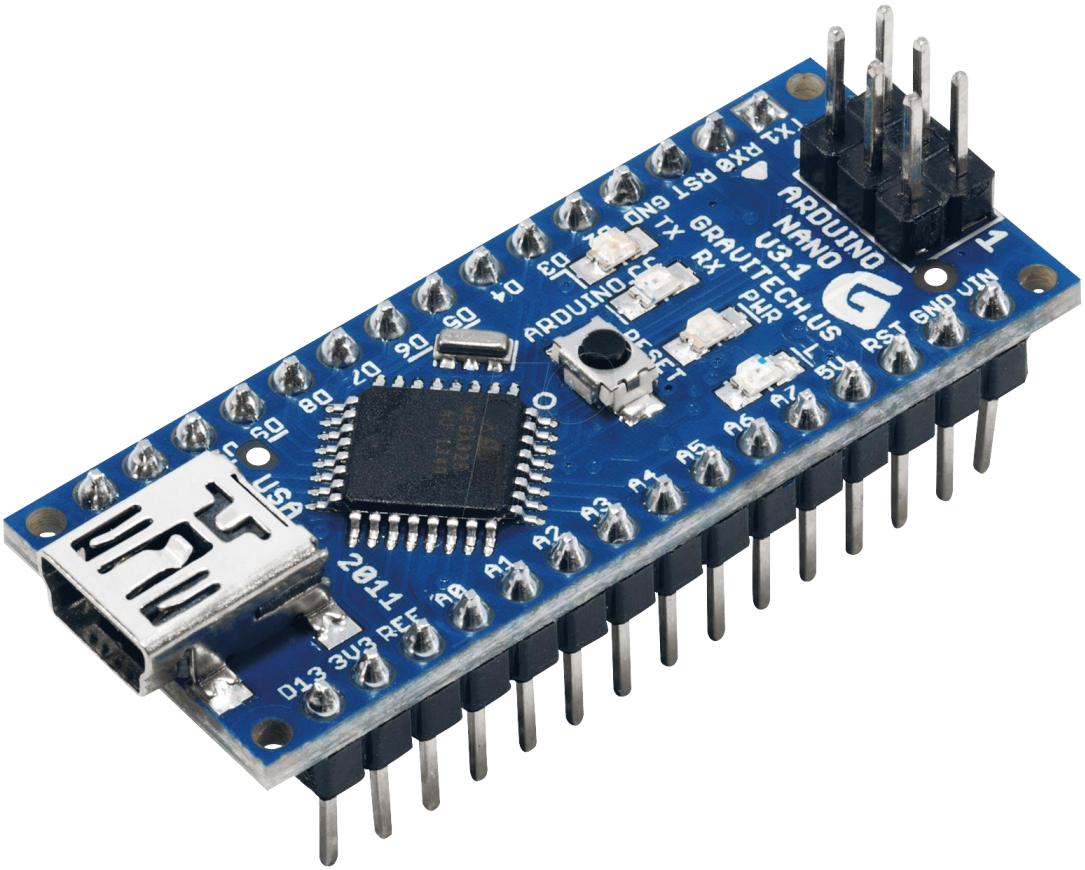


Figure 8: Arduino Nano V3.0

4.1.1.2MPU 9250

The MPU 9250, produced by InvenSense, is a 9-axis Motion Processing Unit™ (MPU), meaning that it combines an accelerometer, gyroscope and magnetometer for position, orientation and acceleration tracking. It additionally contains an embedded Digital Motion Processing (DMP) unit, which can acquire the data from these sensors, process those utilising data fusion algorithms, and return position and orientation information.

A breakout board of the chip was used to better facilitate prototyping and connection to the Arduino. The board (seen in Fig 9), also denotes the IMU's X and Y axes. The Z axis can be inferred from these by the right-hand rule.

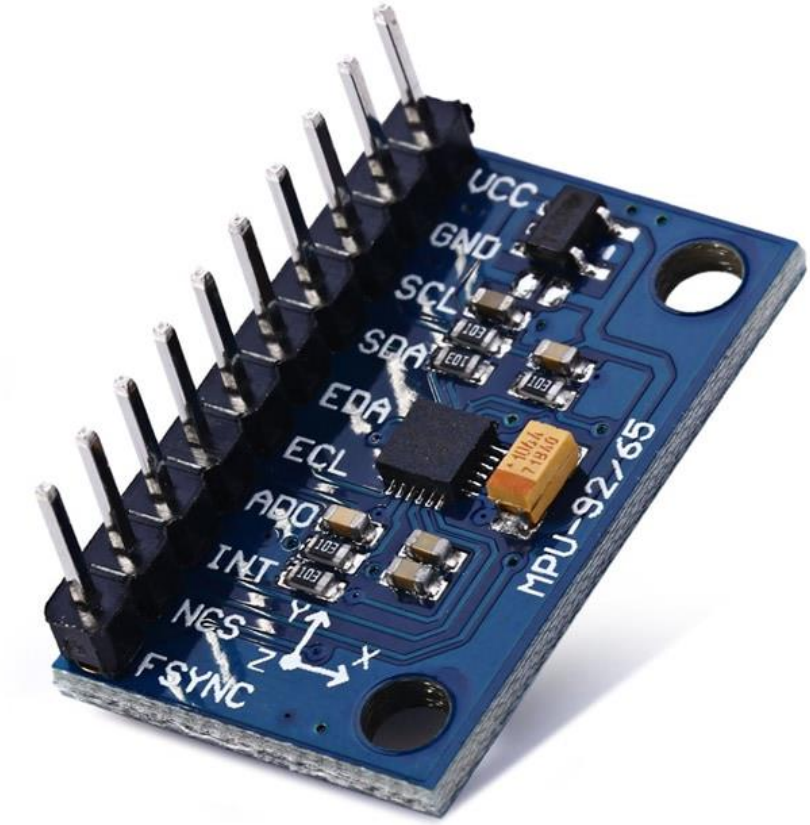


Figure 9: MPU 9250

4.1.1.3Oculus Rift DK2

The HMD used in conjunction with the motion tracking system was an Oculus Rift DK2 (Fig. 10). The DK2, released on July 24, 2014, is the predecessor of the Rift's first consumer iteration. The DK2 connects to the computer via an HDMI cable and two USB 3.0 ports (one for the HMD and one for the IR position tracker). It was calibrated using the Oculus Configuration Utility. The HMD was used to project the developed VE to the user, as well as track the position of the user's head within the environment.

Figure 10: Oculus Rift DK2

4.1.2 Wiring/ Connections

Initially, a prototype of the system was built on a Solder-less breadboard, for the purpose of testing the connections and functionality of the system. The Wiring code for the Arduino was tested and developed further using this prototype to suit the needs of the project. The prototype was also kept in place during later phases of development, and used as a tool for verifying any changes to the code or wiring prior to implementation.

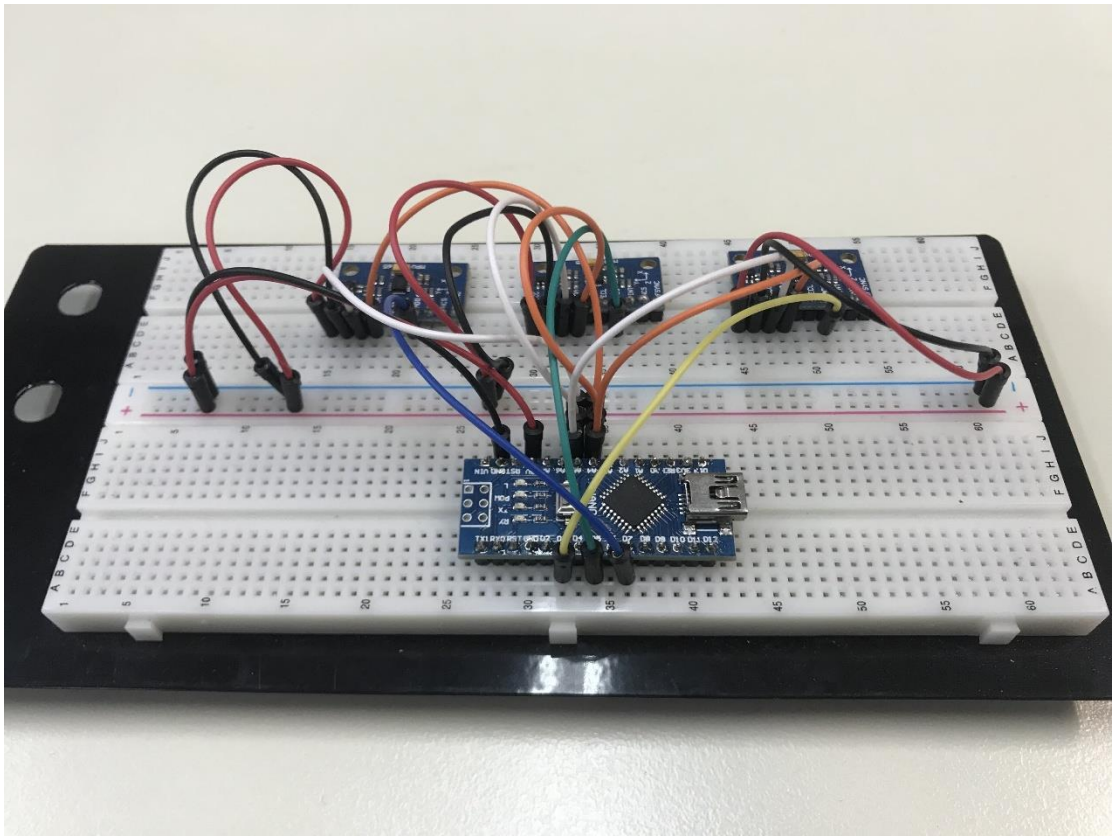


Figure 11: Solderless circuit

After ensuring the system's functionality with the current configuration, the circuit was soldered permanently. Stranded core wires were used, owing to their flexibility which would allow the mobility of the subject wearing the system. Small solderable breadboards were soldered to female pins, within which each sensor and the Arduino were fitted, allowing for the removal of any malfunctioning component from the circuit without the need to de-solder and re-solder it .Header length was clipped, to reduce size.

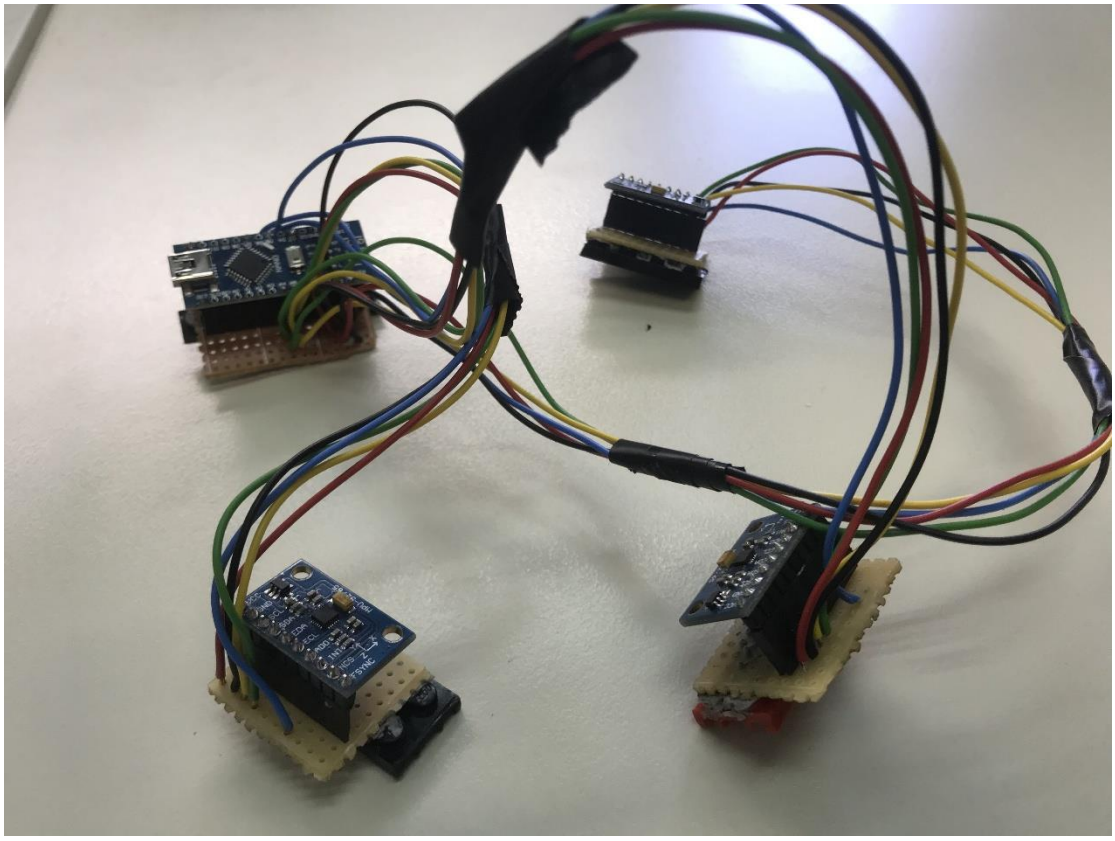


Figure 12: Soldered Circuit

The connections required for each of the IMUs are presented in the following Table:

Table 1: Connections between MPU 9250 & Arduino

MPU 9250	Arduino Nano
VCC	5V
GND	GND
SCL	A4
SDA	A5
AD0	D3/5/7

In the above connections, the A4 and A5 pins on the Arduino are used to implement I2C communication between the board and the IMUs. The same pins can be used for communicating with all three IMUs at once, with A4 being the “Clock” (SCL) and A5 the “Data” (SDA) pin. Pins D3, D5 and D7 are used for selecting the I2C address of each of the IMUs (see section 5.2, Code Development), and are each connected to the AD0 pin on one of the IMUs.

4.1.3 Placement on the arm

The position of the sensors on the body was determined by both empirical observation of upper limb joint and skin movement, and previous research (Chen X. ,

2013; Buchholz & Wellman, 1997; Leonard, et. al 2005; Smeragliuolo et al. 2016; Oberländer, 2015, (Karakikes, 2017). Namely:

One sensor (IMU 1) was placed of the dorsal surface of the palm, approximately over the third metacarpal bone

One sensor (IMU 2) was placed on the dorsal surface of the forearm, over the wrist joint

One sensor (IMU 3) was placed on the dorsal surface of the upper arm, over the elbow joint

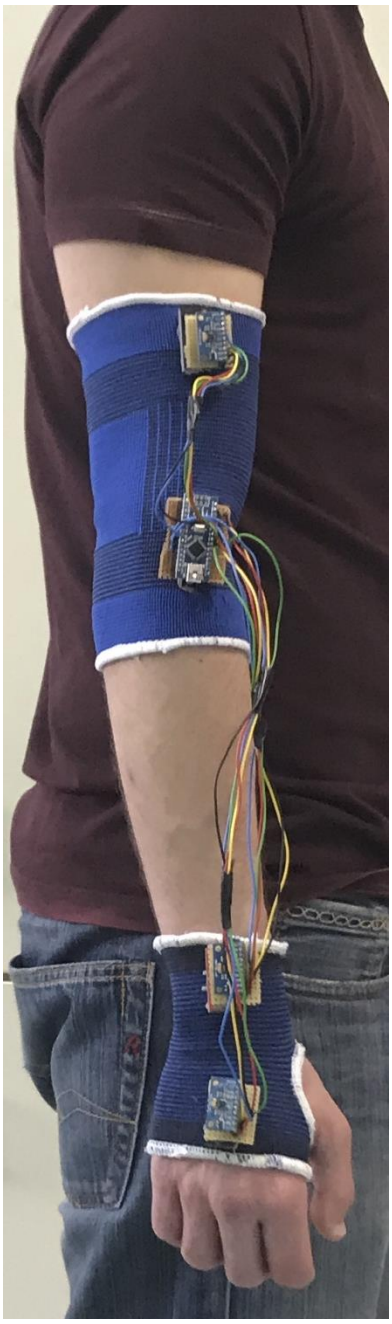


Figure 13: Placement

Each of the sensors captures the motion of one “segment” of the arm—IMU 1 captures the movement of the palm, IMU 2 that of the forearm and IMU 3 that of the upper arm

4.1.4 Placement justification

The placement of the sensors was based on the aforementioned research as well as on the experience of our lab’s personnel due to previous related projects. The work of Michael Karakikes (2017) had determined the correct placement of sensors for the measurement of wrist and forearm angles, and therefore palm and forearm orientation.

As indicated, to measure the movements of the palm, it is adequate for a sensor to be placed over the metacarpal bones of the hand. This placement ensures that the motions originating at the wrist (flexion/extension and radial/ulnar deviation) are accurately read by the IMU, but not influenced by the movement of the rest of the arm (e.g. the fingers or forearm).

For the accurate measurement of the forearm’s pronation or supination, the second sensor needs to be closer to the wrist than the elbow joint, since the effect of the motion is far more pronounced on the lower part of the forearm. The same sensor can be used to effectively measure the elbow’s flexion and extension, since the forearm cannot perform such movements independently from the elbow.

As regards the movements of the shoulder and upper arm, it was theorised that by mounting the third IMU close to the elbow joint, the motions originating at the shoulder joint would be clearly pronounced, as was the case with the pronation/supination of the forearm. This hypothesis was experimentally tested, and the measurements were indeed accurate, so the placement was maintained.

4.1.5 Mounting on the Arm

After reviewing a number of solutions, including using Velcro to mount the sensors and processor to the glove and elbow patch (as previously done by Karakikes, 2017), it was decided to use Lego™ bricks to mount the sensors. This solution offered the ability to quickly mount/dismount the sensors on the glove & elbow patch, while simultaneously providing increased stability over the Velcro patches, as well as the ability to very accurately calibrate the sensors, as explained below. Moreover, since the glove & patch are elastic, it was possible to shift the position of the sensors to accommodate the differences in shape and size of each user’s arms, while still maintaining their alignment.

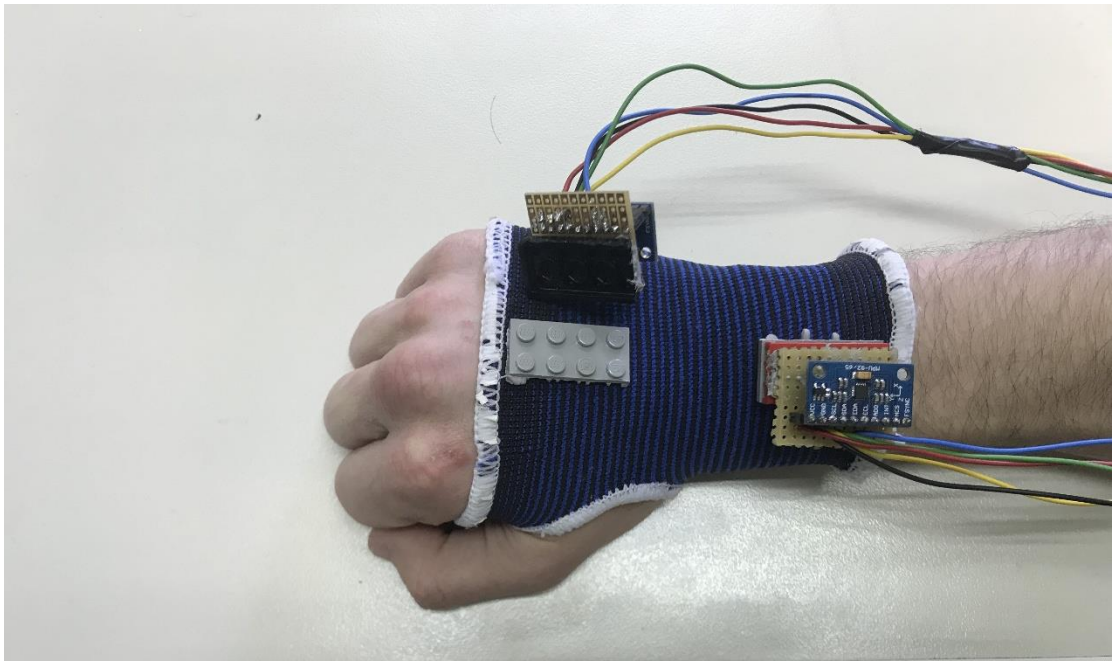


Figure 14: Mounting

4.1.6 Setup

Every time the system is initialised, the following setup process is executed

First, the sensors are attached to a large Lego™ brick with their Y axis facing forward and their Z axis facing up, as shown in the image, and allowed to rest for approx. 10 seconds. The sensors' X-Y needs to be parallel to each other during that time, and their Z axis parallel to the ground (Fig. 15). This is required in order to calibrate the sensors, since their magnetometer data is not obtained by the code, rendering their reference system random (around the gravity axis) each time they are initialized. Moreover, their coordinate systems need to be "matched" to those of the virtual arm in the VE controlled by the motion tracker, which starts at an extended (facing forward) position. This process is repeated every time the system is initialised, as well as when measurement error ("drift") has been accumulated.

Following that, the sensors are attached to the glove and elbow patch worn by the user. The user is asked to extend their arm in front of them, and the sensors are aligned with the user's extended arm position, in order to provide correct measurements. This is done to ensure the position of the virtual arm controlled by the sensors' motion matches that of the user's arm when that motion starts.

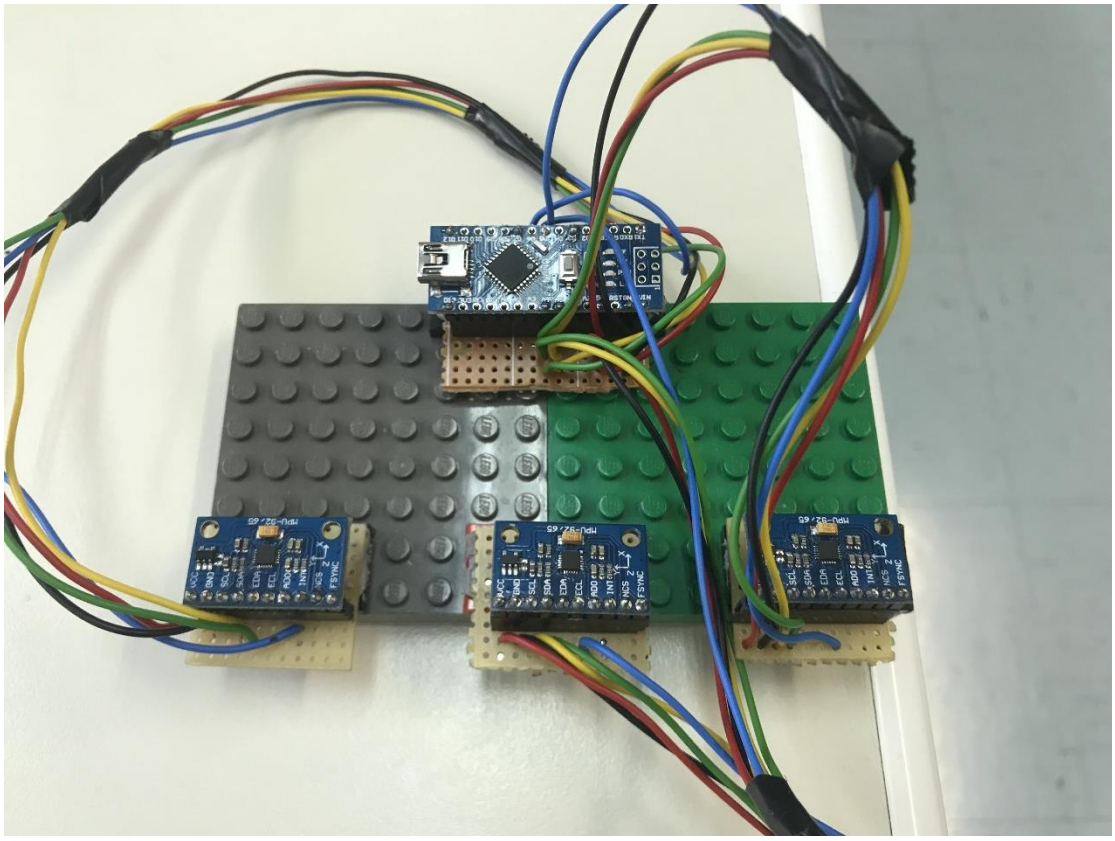


Figure 15: Calibration Setup

4.2 Code Development (Software)

The code developed for this thesis was required to cover two main objectives

a) Acquisition of real-time motion data from the three IMUs mounted on the user's arm
b) Translation of that data into movement of a virtual avatar arm within the developed VE, which was used to conduct experiments

To fulfill objective (a), the Arduino IDE was used to develop and test code, utilising I2C communication, as well as a number of libraries, especially the I2C devlib by Jeff Rowberg (jrowberg) which provides various functions for reading the data from the IMUs and displaying them in different formats (quaternions, Euler angles, yaw-pitch-roll, etc.) as well as the MPU6050_Wrapper library by GitHub user eadf which allows for periodically changing the I2C addresses used by the sensors.

Regarding objective (b), an environment was initially designed using the Processing IDE, utilising the OculusRiftP5 library by Sunao Hashimoto (kougaku) wherein the functionality of the virtual arm could be tested.

However, with the aid of Petros Kontrazis, a different environment was ultimately constructed using the Unity Game Engine (henceforth referred to simply as "Unity"). Unity offers various advantages over Processing, such as easier head-tracking, generally increased—and simpler to implement— VR support, increased

communication capabilities with the Arduino, as well as a more realistic and easily customisable environment.

Owing to the fact that the system was initially designed to function with Processing and later changed to Unity, a number of different solutions were tested and implemented regarding the acquisition of data from the sensors, and various scripts were changed within the Unity Scene to fine-tune the environment's functions and obtain the necessary experimental data.

In this section, the functionality of the main program is explained, together with the data exchange methods between Arduino and Unity. This program obtains the motion data from the three IMUs, transfers that data to Unity, and translates it into movement of the virtual arm. The code itself is presented in appendix A.

The changes made to scripts within Unity are described in section 6, "Environment Design", but not specifically presented in the form of C# code (as that would result in a needlessly large appendix). The program written for the Processing IDE is also not presented here, since it was not used in the final project.

4.2.1 Arduino IDE

The code for the Arduino Microprocessor has one main function: Obtaining the data from each IMU in the form of Quaternions, and printing these Quaternions to the Serial Port. This is achieved by reading raw data from the IMU's Digital Motion Processing (DMP) unit, and translating that data to readable quaternions using the *dmpGetQuaternion* function, provided with *I2Cdevlib.* For distinguishing between the three IMUs, a modified version of the *MPU6050_Wrapper* library is used, which "rotates" between reading each IMU after a set amount of time. This functionality was required because the MPU9250 can only obtain one of two I2C addresses: 0x68 or 0x69, depending on the logic level on pin AD0. Therefore, in order to simultaneously use more than two IMUs, these addresses need to be "rotated" and the motion data needs to be read from each of the IMUs in order.

The used library achieves that by providing "high" voltage for one AD0 pin and "low" voltage for all other AD0 pins (or vice versa). This forces one of the MPUs acquire a different address than the others, making it easy to read data from. Subsequently, the addresses change again, and the next MPU is made available for reading. This way, the MPUs are read one at a time, but with a very short time interval, more than adequate for the purpose of measuring the motions of the human arm.

The final developed program reads data from one IMU at a time, and then prints that data to the serial port in a single line, separated by commas and preceded by the number 1, 2 or 3 to distinguish between IMUs.

4.2.2 Unity

On the other side of the loop, the script *motionControl.cs* running in Unity reads the data sent to the Serial Port by the Arduino and translates it to movement of the corresponding game object. This is achieved by reading one line of data at a time from the open Serial Port, splitting that line into separate strings whenever a comma

is encountered, and parsing these scripts as integer (for the IMU identifier 1,2 or 3) or floating point variables (for the actual Quaternion data). These variables are then stored in 3 Quaternion objects (pre-existing in Unity) changing their order and orientation as required for the coordinate systems of the IMUs and the Unity environment to match (the correspondence appears in Table 2). Finally, the 3 joints of the virtual arm within Unity are each rotated according to the corresponding Quaternion, creating an accurate representation of the user's arm movements.

Table 2: Correspondence between MPU9250 & Unity coordinate systems

Unity	MPU9250
X	-X
Y	-Z
Z	-Y

Notes

1) The I2Cdevlib as well as the MPU6050_Wrapper libraries, used with the Arduino IDE, are not developed for the MPU 9250 but rather its predecessor, the MPU 6050.The reason for their use is two-fold. Firstly, since the 9250 is a fairly new unit, the existing libraries for using it with an Arduino are few and do not cover the needs of the current project. This necessity, coupled with the fact that the libraries for the 6050 could be used with ease to obtain accurate data from the DMP, resulting in simpler code and lessening the computational load on the Arduino, led to the decision of using the aforementioned libraries.

2) Due to the above decision, and owing to the fact that the MPU 6050 did not include a magnetometer, the magnetometer data is not read by the utilised libraries. This means that the sensors have no reference regarding their orientation in the XY plane at the beginning of the experiment, necessitating that they be allowed to rest before being used, in order to determine the world reference frame (as described in section 5.1.6, "Setup"). This was not an important hindrance in our case, as the time that the sensors were resting was used for the purpose of explaining the experimental process and setup to the subjects (see section 7, Experimental Design).

3) The used code allows for determining offsets for each of the sensors, in order to expedite the calibration process. However, these offsets are determined by the sensors' internal architecture, and need to be determined by dedicated algorithms, or otherwise estimated by iteratively testing various values. For that reason, as well as the fact that time for calibration was rather plentiful in our case (as described above), the offsets were left to the value of 0.0 in the code.

4) To minimise the lag between the movements of the real and virtual arm, the *motionControl* script was assigned to a separate processing thread from the rest of the processes. This was required because if allowed to run in a serial, rather than parallel, manner, the IMU measurements were displayed with a cumulative delay in the Unity environment, resulting in increased lag and making the virtual arm unresponsive.

5) The scripts as well as the virtual arm developed in the Unity environment are completely standalone. This means that they can be “removed” from the existing VE and placed into any other Unity-based environment, allowing for expedient and efficient use of the motion tracking system outside the developed scene.

An overview of the designed system can be seen in the figure below. The VR environment’s design and functionality are explained in detail in section 6, “Environment Design”

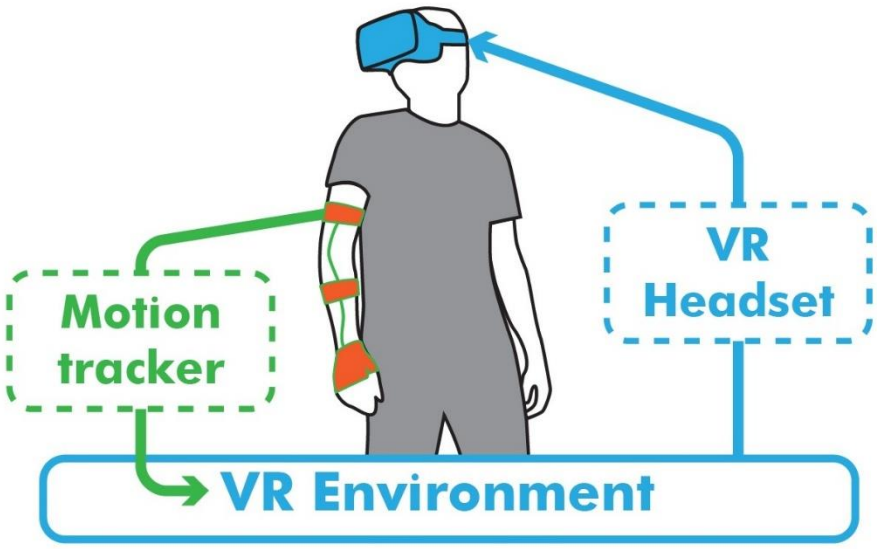


Figure 16: Working Principle of Motion Tracking System

5 Environment Design

The experiment conducted as part of this thesis was originally intended to be a VR "catch" task. The reason for this choice is that catching a flying object required the user to implement hand-eye coordination in a short amount of time, and reflexively move their arm in space. It was theorised that this would allow us to effectively study the level of incorporation of an avatar body which can be achieved in such environments. However, after designing such a task in the Processing environment, it was realised that the environment suffered from a lack in realism, due to the absence of haptic feedback when catching the flying object; moving their arm to catch something and then clicking a button to achieve the actual "grabbing" of it seemed confusing to users.

In light of this, it was decided to change the task from catching flying objects to shooting randomly appearing targets. The reasoning behind this was that trying to quickly aim and shoot at a target appearing at a random point in space would impose hand-eye coordination requirements on the user similar to those of the catching task. Moreover, clicking a button to simulate releasing a projectile from a gun seemed more lifelike, to researchers and users alike, than grabbing an object.

As mentioned above, the above task was also initially designed using the Processing language, and then re-adapted using the Unity Game Engine. Only the second game environment—the one used in the final experiment—is presented here.

5.1 Original Scene

The environment's background and most of the objects were originally found in the free Unity Asset VR Samples, and namely from that Asset's "Target Arena 360" scene.

That scene consists of a 360^{o} environment, with the player positioned centrally with their viewpoint coincident with the Main Camera (first person perspective). The player is unable to change their position within the environment, but they are capable of rotating around their axis and moving their head in order to look "behind" them in the environment, utilising the head tracking capability of the Oculus HMD. The scene's original functionality was as follows:

5.1.1 Intro

First, the player is presented with an intro screen, providing instructions on how the game is played (see Fig 17). Within that screen, the player can see a rigid, extended arm holding a gun, and a red rectangular reticle positioned where the gun is aiming. By moving their head around, the player can aim the reticle wherever they are looking. The arm moves to follow that movement. Positioned centrally on the screen underneath the instructions, is a grey box with the phrase "Ok, I got it!" displayed within. By holding their gaze over that box and holding down the left mouse button, the box "fills up" with a fuchsia colour, and the player can start the game.

Figure 17: Game Intro

5.1.2 Main Game

During the main “phase” of the game, targets spawn around the player in random locations within the environment (see Fig 18). The player has 1 minute to shoot as many targets as possible. By using the direction of their gaze to move the reticle over the targets and clicking the left mouse button, the player causes a laser beam to fire from the gun held in the displayed arm. When the beam hits the targets they shatter, and the player’s score increases by 1 point. The score is displayed on the screen during this phase, together with a blue curve representing the time remaining. When that time runs out, the game is over. If not hit, the targets remain on the screen for 2 seconds before disappearing. There is a script running behind the scene determining the desired number of targets which should be on-screen at any time, and reducing or increasing the probability of a new target spawning depending on how many already exist.

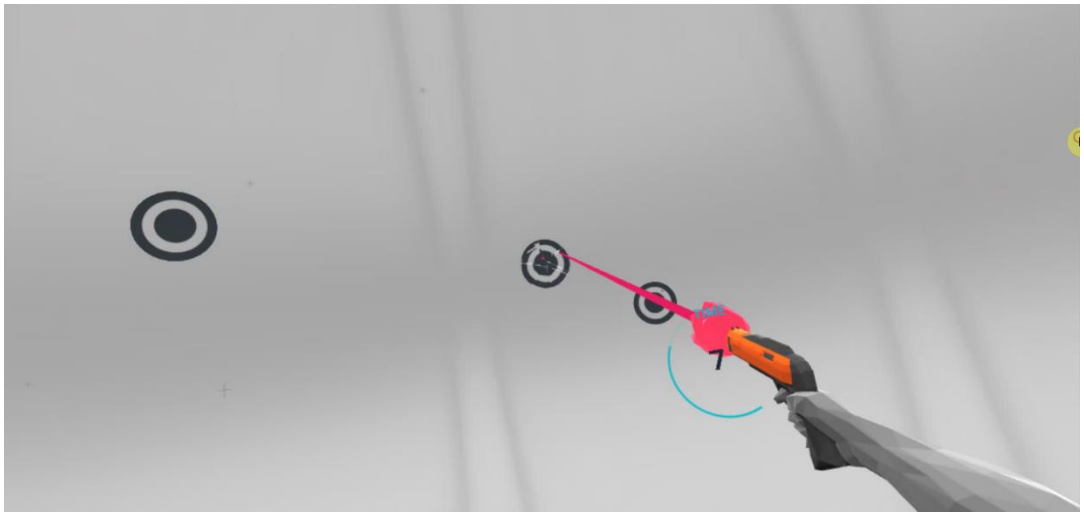


Figure 18: Main Game

5.1.3 Outro

When the time runs out, any existing targets disappear and the player is presented with an outro screen, displaying their score as well as the current high score (Fig 19). The rectangular reticle is replaced with a circular one, and the player is prompted to hold down the left mouse button to play again. If they choose to do so, the reticle "fills" up and the player is redirected to the intro screen, from where they can start the game again.

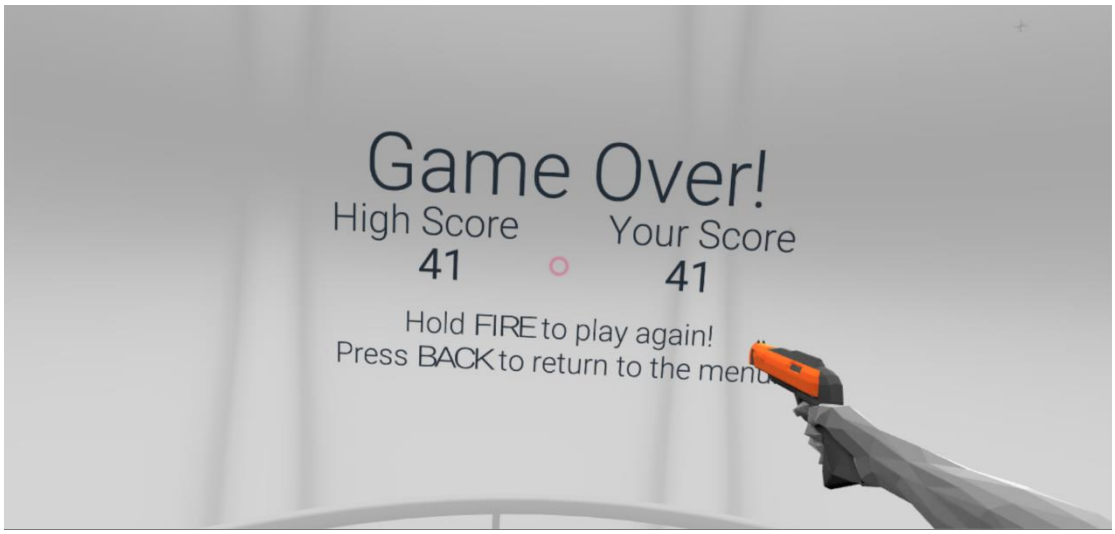


Figure 19: Game Outro

5.2 Adaptations to the needs of the project

The scene was changed appropriately to accommodate the needs of the experiment. The most important changes made to the environment were:

The original rigid arm existing in the environment was replaced with a jointed arm created for the purposes of the experiment, controlled by the signal from the three IMUs on the user's arm. This arm was created using simple capsule and sphere objects found in the Unity Game Engine, since our focus was directed more towards it being capable of accurately capturing the movements of the user than to it being aesthetically pleasing or realistic. Each capsule represents a segment of the arm (palm, forearm, and upper arm), and each sphere one of the arm's joints (wrist, elbow and shoulder). The signal from each IMU controls the position and orientation of the arm's corresponding joint—the IMU mounted on the upper arm controls the shoulder, the one mounted on the forearm controls the elbow, and the one on the palm controls the wrist. The capsules and spheres are connected via parent-child object relations starting from the shoulder and moving down to the palm. This way the movement of the virtual arm corresponds to the movement of the user's arm in the real world—for example, if the user rotates their forearm about the elbow, the wrist and palm move as well, but if the user simple flexes or extends their wrist while keeping the rest of the arm immobile, the same movement will occur on the virtual arm.

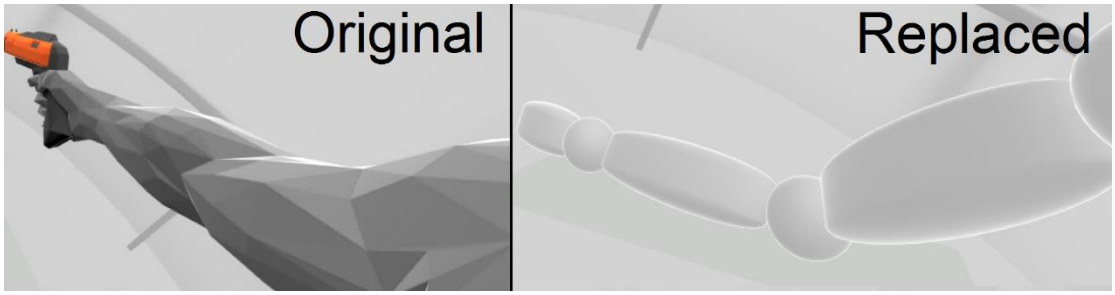


Figure 20: Replacement of the existing arm

The aiming system was redesigned so that the player uses their arm to aim, instead of their gaze as per the original design. This was done by repositioning the origin of the scene's Raycaster (the component that projects a "ray" to the aiming position) from the Main Camera to the end-effector of the virtual arm (a small sphere object positioned at the end of the "palm" capsule). This way, the scene's aiming reticle is positioned on the point in the background where the palm is pointing at, rather than wherever the player is looking. This change was made in order to force the players to utilise the movement of their arm while aiming. This was required in order to obtain useful data from the conducted experiment (described in section 6, "Experimental Design"). Namely since the hypotheses tested in the experiment focus on the motion of the arm within the VE, it was of paramount importance that the users aim with their arms. If they were able to aim using the direction of their gaze instead, the arm's movements would be rendered irrelevant to the task, and therefore any data obtained from the measurements of these movements would carry no significance whatsoever.

The origin of the laser beam was also placed at the arm's end-effector, in order to visually demonstrate this change in the aiming mechanism to the player.

The aiming reticle appearing in the scene was removed in order to shift the users' focus more towards the movement of their arm, rather than a moving point on the screen. The circular reticle described above (see "Outro") was maintained during the intro phase, to aid the users in understanding how the aiming system works and in starting the game. The circular reticle was chosen over the rectangular one simply due to its larger size—it was easier for the users to spot during their first contact with the environment.

The spawn mechanism of the targets was changed so that only one target spawns at a time and remains on-screen for 10 seconds, rather than 2, before de-spawning (if not shot down). The purpose of this was to compel the users to search around the environment in order to find the target, as it was theorised that would increase the observed hand-eye coordination activity, as well as the movement of the users' arm, during the experiment. The targets' lifespan was increased in order to provide enough time for the users to locate and shoot them. The original thought was to make the lifespan "infinite" (60 seconds), but this plan was abandoned after observing certain users getting "stuck" on targets they couldn't reach, during the pre-experiment runs.

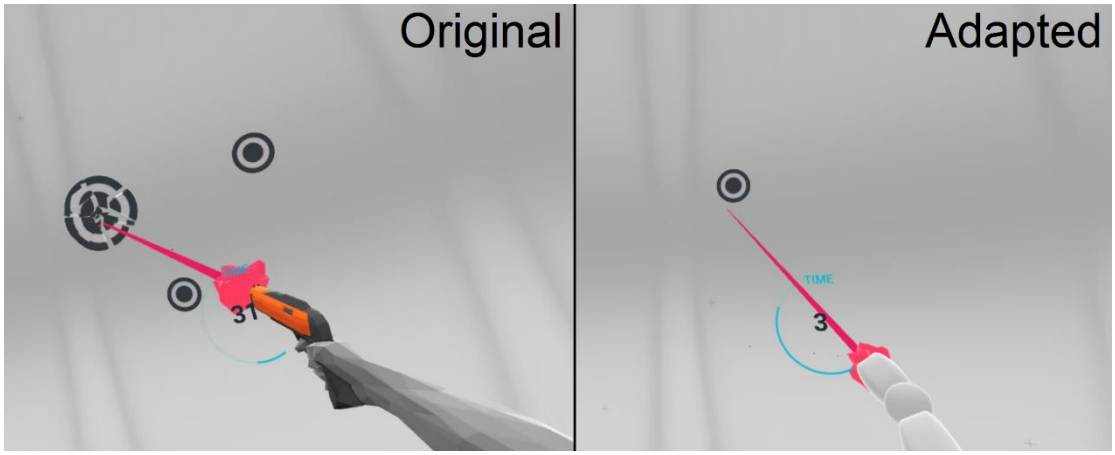


Figure 21: Original & Adapted Scene

A script was added to implement the control of the jointed arm, receiving the data from the Arduino via serial communication and translating them to movement of the arm in the environment, as explained above.

Several changes were made to the existing scripts in order to obtain the desired data while running the experiment (see “Experimental Design”).

6 Experimental Design

6.1 Subject Distribution

The experiment was initially designed to test whether a correlation exists between the visibility of the virtual arm and the test subjects' performance in the task. It was decided that the 22 subjects would be divided into two groups, one containing 12 and the other 10 participants, henceforth referred to as Group A & Group B, respectively. Group A consisted of 6 male and 6 female participants, while group B consisted of 5 male and 5 female participants. Participants were aged 20 to 29. The subjects' experience in First Person Shooter games varied from none to very high.

6.2 Runs and conditions

The experiment would consist of two "runs" for each subject. Each run was defined as a 1-minute period during which the subjects use a laser beam fired from the end of the virtual arm to shoot down targets spawning in a 360^{o} radius around them. The task was completed from a standing position, ensuring that each subject had sufficient space to rotate around their axis, in order to reach targets appearing behind them, without being obstructed by the control and display devices. Between runs, the subjects were asked to fill out a questionnaire regarding their experience during the experiment

Group A would first complete the experiment with full visibility of the arm, fill in the questionnaire after this first run, and subsequently run the experiment again, this time having visibility of only the wrist and palm. Group B would complete the runs in the opposing order, initially viewing the arm from the wrist down, and, after completing the (modified) questionnaire, re-run the experiment with a visible arm.

This was decided based on the assumption that the subjects would accumulate experience on the task during their first run, and consequently exhibit increased performance in their second run. To compensate for that learning effect, the above division was implemented. At the same time, this allowed for testing an increased number of subjects in each of the test conditions, since participants in both groups would eventually complete the experiment both with and without visibility of their arms. The completion of the questionnaire was also placed between the two runs to avoid the possibility of the subjects being biased by experiencing both test conditions and favouring the one in which they performed better.

However, during test runs before the start of the experiments, it quickly became apparent that the beam, rather than the arm, was the primary tool used for aiming by most subjects. In light of that information, a third run was added to the experiment, in which the subjects no longer had visibility of the beam. The visibility of the arm was determined accordingly to the groups' second run, therefore Group A had visibility from the wrist down, whereas Group B had visibility of the full arm.

The test conditions for each group and run are shown in the table below. Runs with identical testing conditions are signified by common (blue or green) shading of the

cells. Note that, in the conditions where the beam was invisible, only ten subjects experienced each condition, as opposed to twenty for the previous two (visible beam).

The addition of a fourth run, where the arm's visibility would again be reversed with the beam remaining invisible, was rejected under the assumption that subjects would have accumulated fatigue due to the repeated runs at that point, which might influence their results.

Table 3: Test Conditions per Group & Run

	Run 1	***Run 2***	***Run 3***
Group A	Arm Visible Beam Visible	Arm Invisible Beam Visible	Arm Invisible Beam Invisible
Group B	Arm Invisible Beam Visible	Arm Visible Beam Visible	Arm Visible Beam Invisible

The aim of this third run was to determine whether the usefulness of the arm's visibility was dependent on the nature of the task. It was theorised that, by removing the beam, the task of shooting targets was made not only harder, but different from the original, and more akin to real-world shooting, therefore forcing the subjects to adopt a different strategy for completing it. That means, of course, that any comparison of data between the first two runs and this third one is deemed invalid, as they originate from dissimilar tasks.

6.3 Experimental Process for each subject

Each subject completed the experiment as follows: Firstly, the equipment used for the experiment and its usage was briefly described to them (HMD, Motion tracker). This time was also used to calibrate the Motion Tracker. Following that, the HMD was mounted and fitted on their head, allowing them to see the Unity environment. They were prompted to look and move around, in order to ensure the HMD's cables did not obstruct their movement. While doing that, the experimental task was described to them, omitting the changes made in subsequent runs. The Motion Tracker was then mounted on their arms, and they were given a few (~10) seconds to understand the movement of the virtual arm within the environment, using the circular reticle described in section 6, "Environment Design". It was explained to the subjects that this reticle would disappear while running the actual experiment. Finally, the mouse was handed to the subjects, and they were allowed to begin the experiment.

After finishing their first run, the HMD was removed and the subjects were asked to complete the questionnaire corresponding to their group, with the examiner available nearby to answer any questions. The motion tracker was not removed during this process, except in the cases when recalibration was required, since it did not impede

the subjects' movements. After completing the questionnaire, the HMD was mounted on the subjects' head again, the changes made for the following run (rendering the arm visible or invisible) were explained (and observed by the subjects), and the subjects were allowed to run the experiment again. After this run was completed too, the HMD was removed for a short while to allow the subjects' eyes to relax and, when deemed necessary, the Motion Tracker was recalibrated. Finally, the control and display mechanisms were mounted again, the changes made (removal of the beam) were explained, and the experiment was run again. Finally, after the completion of this final run, the equipment was removed from the subjects.

6.4 Data acquired from subject

6.4.1 Objective Data

From the twenty subjects, the following data was received for each of the runs

- Number of shots Fired
- Number of targets hit
- Shots Fired off-target between subsequent hits
- Time (in seconds) each target remained on screen before disappearing (whether hit or naturally de-spawning)
- Position (XYZ) of the arm's end-effector during the experiment.

This data was used to conduct various types of analyses, described in section 8, "Results"

6.4.2 Questionnaire, subjective experience data

The questionnaire filled out by the users was used for the purpose of extracting information regarding the users' subjective experience of the experimental task, such as user satisfaction, adaptation to the environment, usability of the virtual arm, etc. The questionnaire was loosely based on the presence questionnaire by Witmer & Singer (1998), and adapted to suit the needs of the current experiment. The answers were given on a Linkert scale of 1 to 7. Each of the A and B groups was presented with a different version of questions 4 and 5 ("visible" for Group A, "invisible" for Group B) in order to gauge the effect of the arm's visibility on the users' subjective experience. The questionnaire is presented below, translated into English

1) How much delay did you experience between your movements and their representation in the Virtual Environment?

1-None 7-Very large

2) How much did the motion tracker and HMD interfere with your performance in the designated task?

1-Not at all 7-Very much

3) How capable in moving within the environment and interacting with it did you feel at the end of the experiment?

1-Not at all 7-Very much

4) How much harder or easier was the task made by the fact that your arm was invisible/visible?

1-Much harder 7-Much easier

5) How much was the naturalness of your interactions with the environment decreased or increased by the fact that your arm was invisible/visible?

1-Severely decreased 7-Severely increased

6) How experienced are you in playing such games (First Person Shooters)?

1-Not experienced at all 7-Very experienced

To supplement these results, the users were asked to freely comment on their experience after the culmination of the tests. Some such observations are presented below

“The beam was more helpful for aiming than the arm”

“When the beam was removed, I was forced to align my arm with the target to hit”

“I mainly used my wrist to aim, I felt surer that way, like I had better control”

“During the first run, I was aiming normally. During the third run, I used my arm as a controller of the virtual arm”

It is clear that these comments echo the hypotheses and observations made during the pre-experiment runs in the VE; subjects tend to aim using the beam as a guiding mechanism, and, when that is removed, re-adapt their strategy in order to hit the targets. Most users also commented that the experience was overall enjoyable, and many stressed the importance of accurate measurement by noting that they got confused when the representation of their movements was delayed or inaccurate (due to drift).

7 Results

The data acquired from the subjects was used to draw a number of different conclusions regarding

- the importance of the arm's visibility in performance
- the assimilation of the virtual arm by the users, and the speed with which that assimilation occurred
- the speed with which the users adapted to the virtual environment and the task they were asked to complete
- the differences in strategy and results after the task was differentiated (removal of the beam)

7.1 Scores per Run

Firstly, as regards the scores (targets hit) of the groups in each of the runs, the following data was acquired:

Table 4: Scores per Group & Run

Group 1	Run 1	Run 2	Run 3	**Group 2**	Run 1	Run 2	Run 3
F1	10	9	7	***F1***	9	12	1
F2	10	15	10	***F2***	6	9	0
F3	8	12	2	***F3***	6	11	2
F4	10	11	14	***F4***	5	9	5
F5	7	14	2	***F5***	14	13	8
F6	9	15	3	***M1***	16	15	13
M1	6	8	1	***M2***	10	16	7
M2	8	15	11	***M3***	14	15	6
M3	9	10	8	***M4***	7	11	6
M4	10	18	2	***M5***	5	11	6
M5	8	15	3	***Average***	9.2	12.2	5.4
M6	12	13	1				
Average	8.92	12.92	5.3				

The scores of Groups A and B in each of the runs were compared using a two-sample unequal variance t-test and found to not be statistically different. Rather, the average scores of each group are impressively similar throughout in the first, second and third runs. We are thus led to believe that the arm's visibility has no significant effect on the performance of the subjects, as theorised. It is also obvious that familiarisation with the task improves performance, as seen by the fact that the scores for Run 2 are higher than those in Run 1. Removal of the beam, on the other hand, has an adverse effect on performance, as it forces the subjects to re-adapt their strategy, and increases the difficulty of the task. Still, the two groups exhibit very similar results on this run as well, indicating that the arm's visibility is not of paramount importance even after modifying the task.

7.2 Target On-screen Time charts

After these preliminary results were demonstrated, the following plots were produced. These show the time (in seconds) each target remained on screen before disappearing or being shot down on the Y axis, plotted during the total run-time of the experiment, displayed on the X-axis. In these graphs, each blue point corresponds to a target. If that target was shot down, it lies underneath the 10-second gridline. The targets above that line were not hit by the players, instead de-spawning naturally. The trend lines were obtaining by using linear regression on the data. Linear regression was chosen as a simple evaluation tool for observing trends appearing in the data, due to the lack of any sort of information as to what kind of formula would best represent the learning effect. The results are presented for each of the groups separately, as well as for both groups on the same axes, for the first two runs of the experiment. It is repeated at this point that Runs 1 & 2 were analysed separately from Run 3, as they concern differentiated experimental tasks.

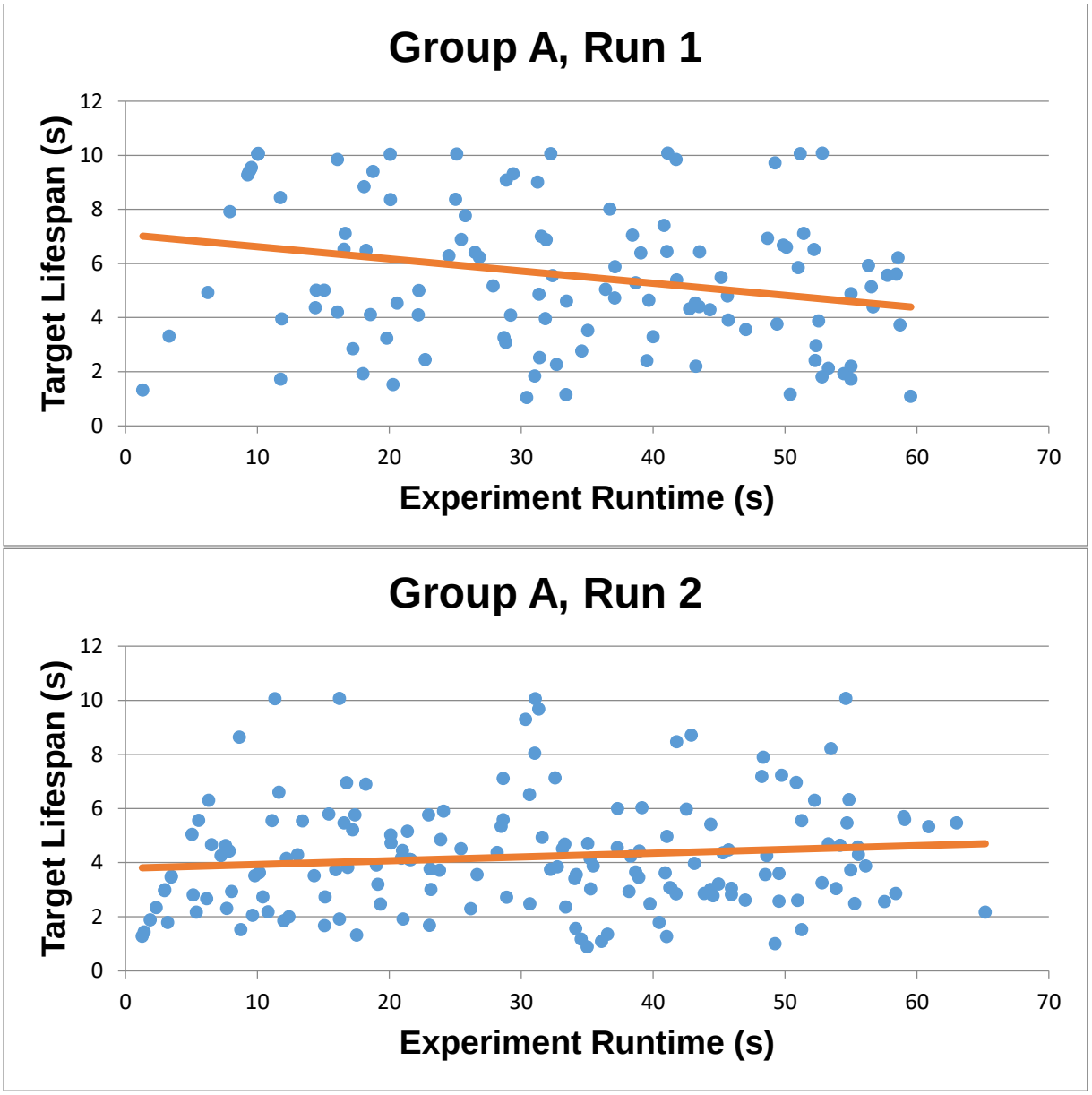


Chart 1: Target Time on Screen, Group A, Runs 1 & 2

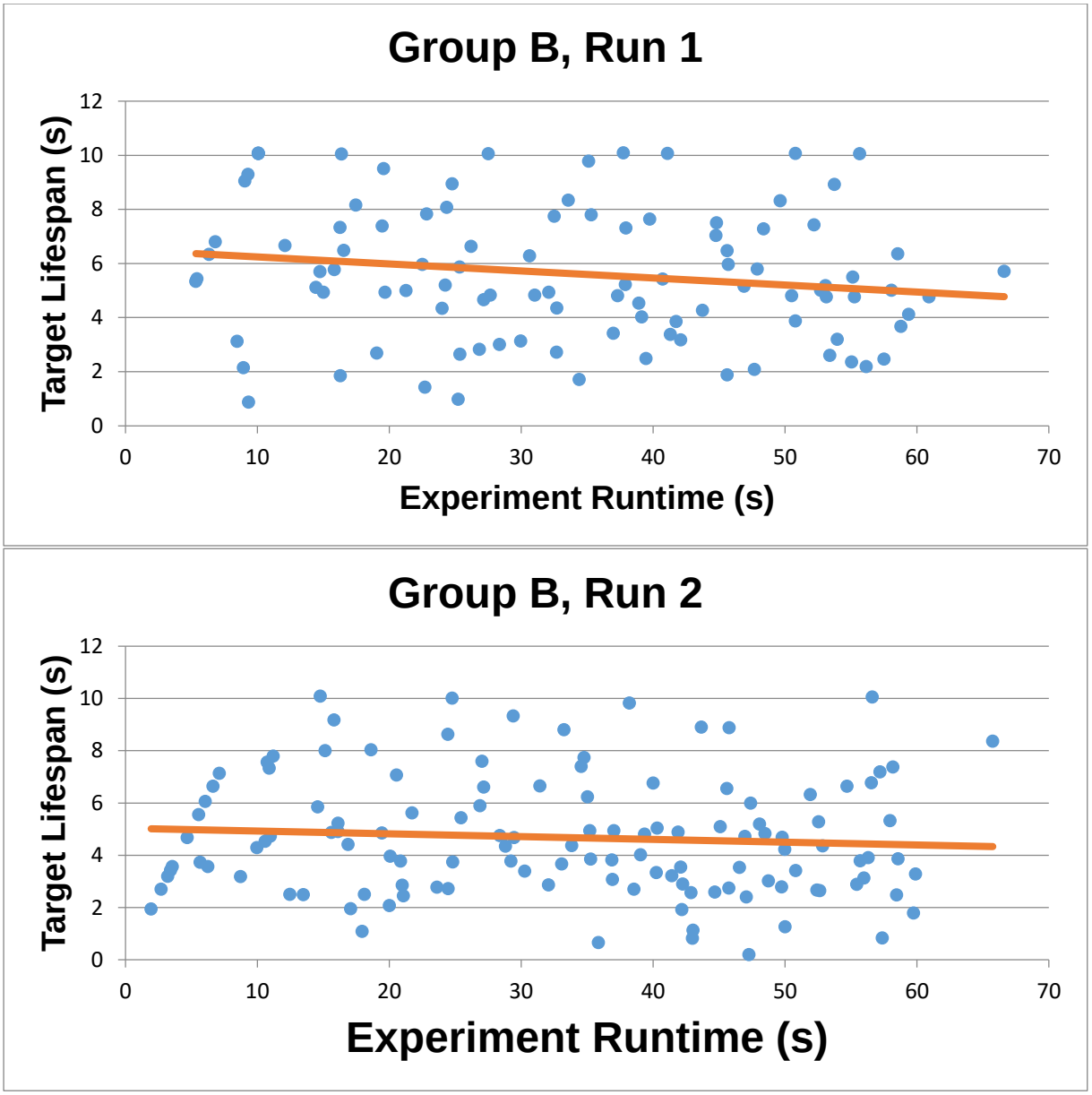


Chart 2: Target Time on Screen, Group B, Runs 1 & 2

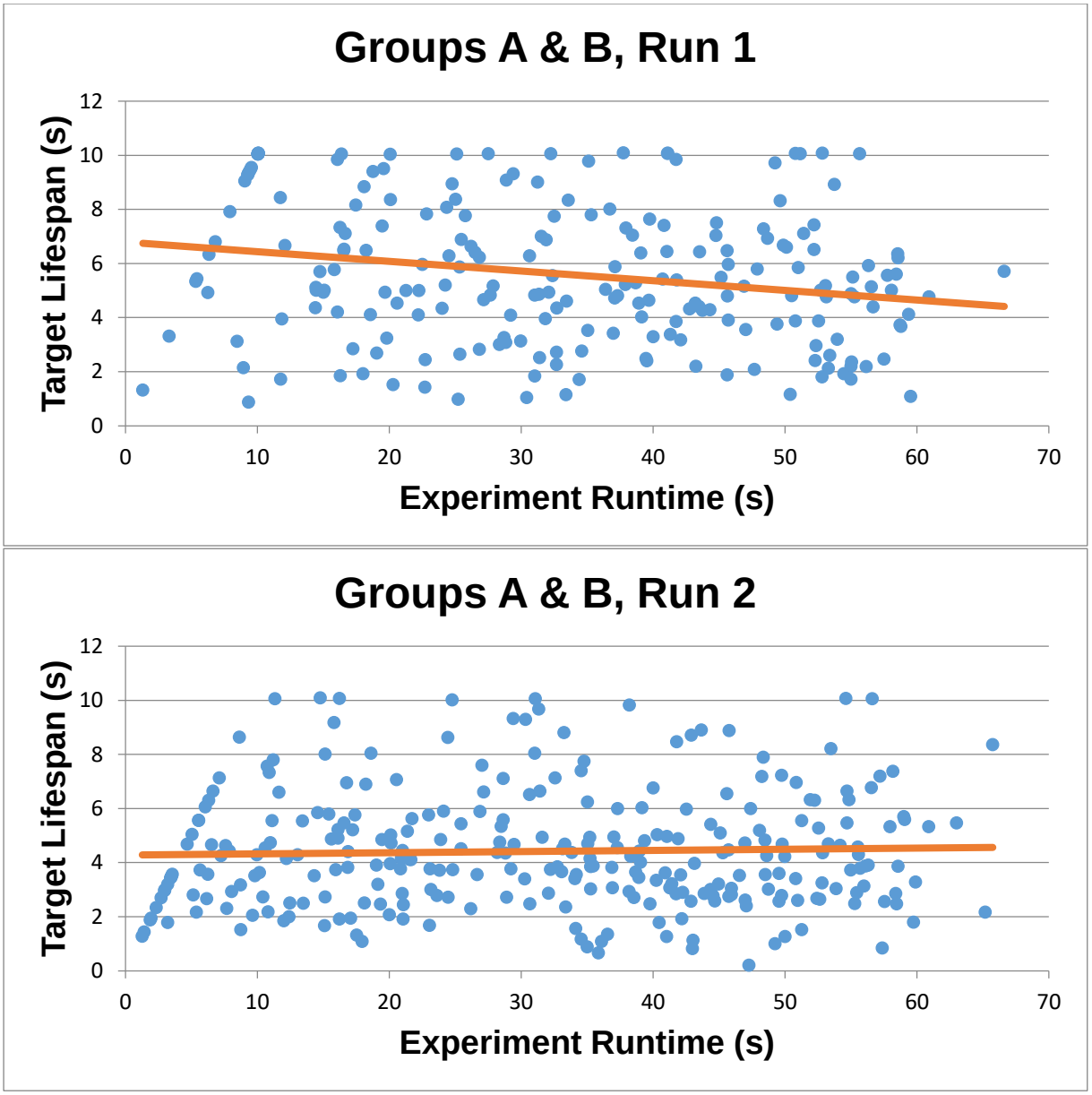


Chart 3: Target Time on Screen, Groups A & B, Runs 1 & 2

The presented results are intended to demonstrate the learning effect occurring during the experiment. While the data is scattered, the linear regression trend lines indicate the following:

During the first run, targets stay on-screen for decreasing amounts of time as the experiments progresses, meaning that they are shot down faster and naturally de-spawn less frequently towards the end of the experiment, compared to the start.

During the second run, that effect is no longer observed; targets remain on-screen for an amount of time that does not increase or decrease during the run. That amount is approximately equal to the amount of time that targets stay on-screen at the end of the first run (~4s). A t-test was conducted, comparing of the mean values of the second run for each group with the estimated value reached at the end of the first run

(considered as a set value) by the corresponding group, demonstrating that these values can indeed be considered statistically identical.

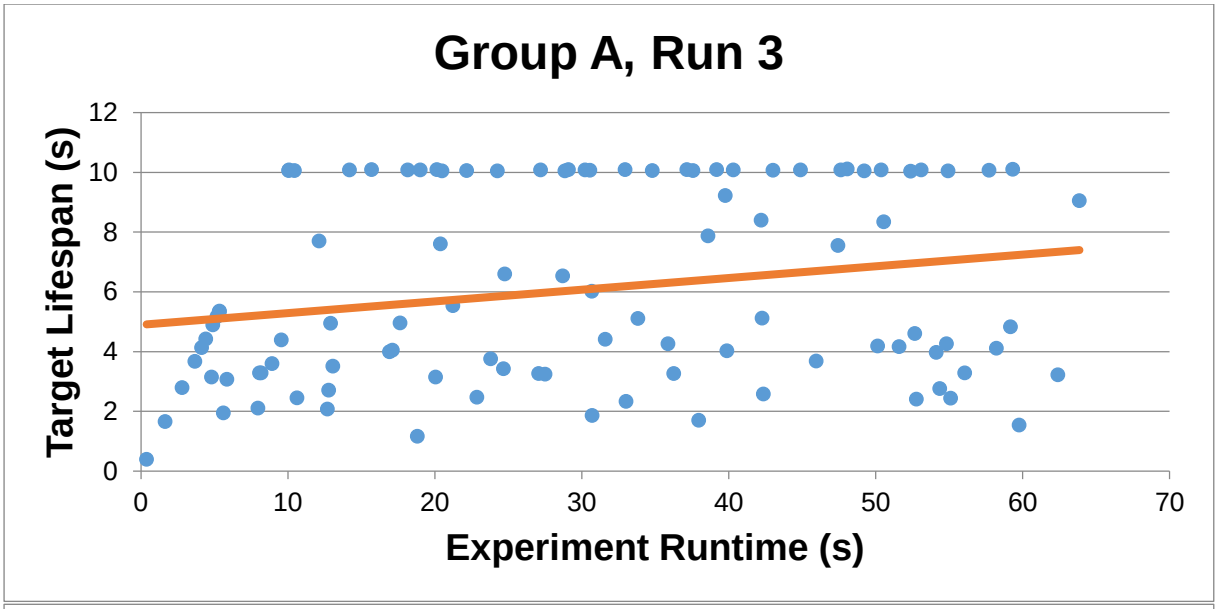
These observations were further strengthened by evaluating the p-value of the regression curves. For all cases, the first run showed a significant decreasing trend (p-value <0.05), while the second did not (p-value > 0.05). All p-values are presented in the following table

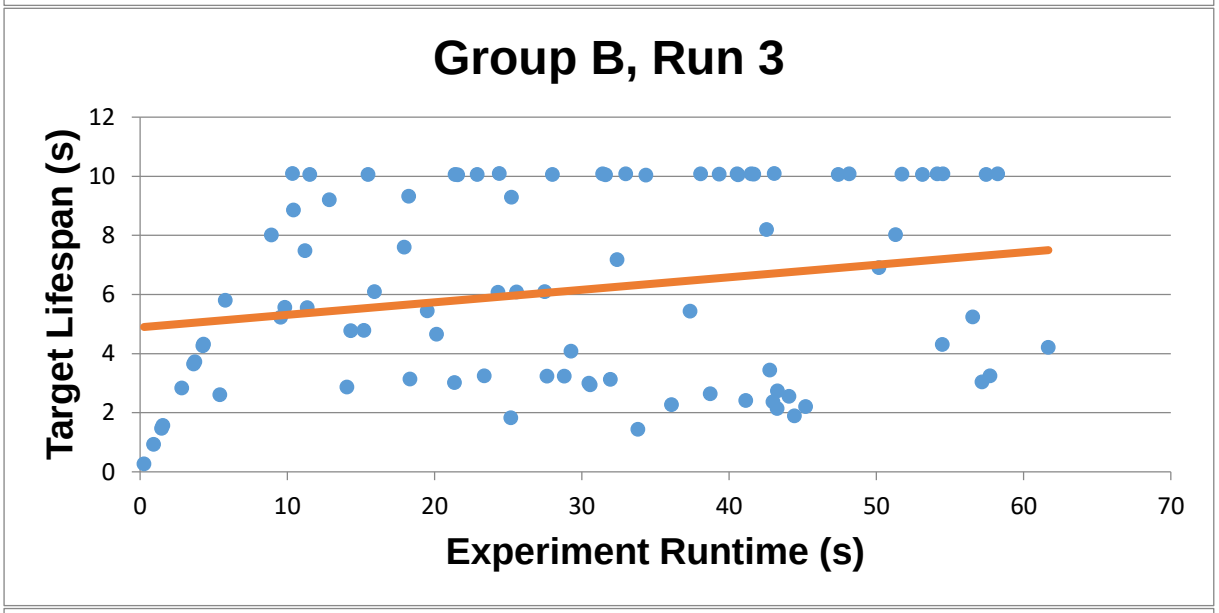
Table 5: P-values of Linear Regression trend-lines, Runs 1 & 2

	Run 1	**Run 2**
Group A	0.0041	0.1409
Group B	0.0307	0.3733
Groups A & B	0.0004	0.6743

It is theorised that these findings illustrate the learning effect experienced by subjects: during the first run, subjects progressively familiarise with the experimental environment, leading to an increase in performance, which however reaches a plateau during the second run. This speaks as to the speed with which the users incorporate the virtual arm's movement into their body schemas; each of the runs lasts for 1 minute, yet that time is enough for the subjects to learn how to use the virtual arm effectively to achieve their goals. This assimilation occurs regardless of whether the arm is visible or invisible, as can be seen by evaluating the "plateau" values reached by each group at the end of the first run, which, as has been stated, are statistically identical to the mean value achieved during the second run. These values were again compared using t-tests, assuming both equal and unequal variances and found to not be significantly different in all cases.

The same analysis was conducted for the third run, producing the plots displayed below:





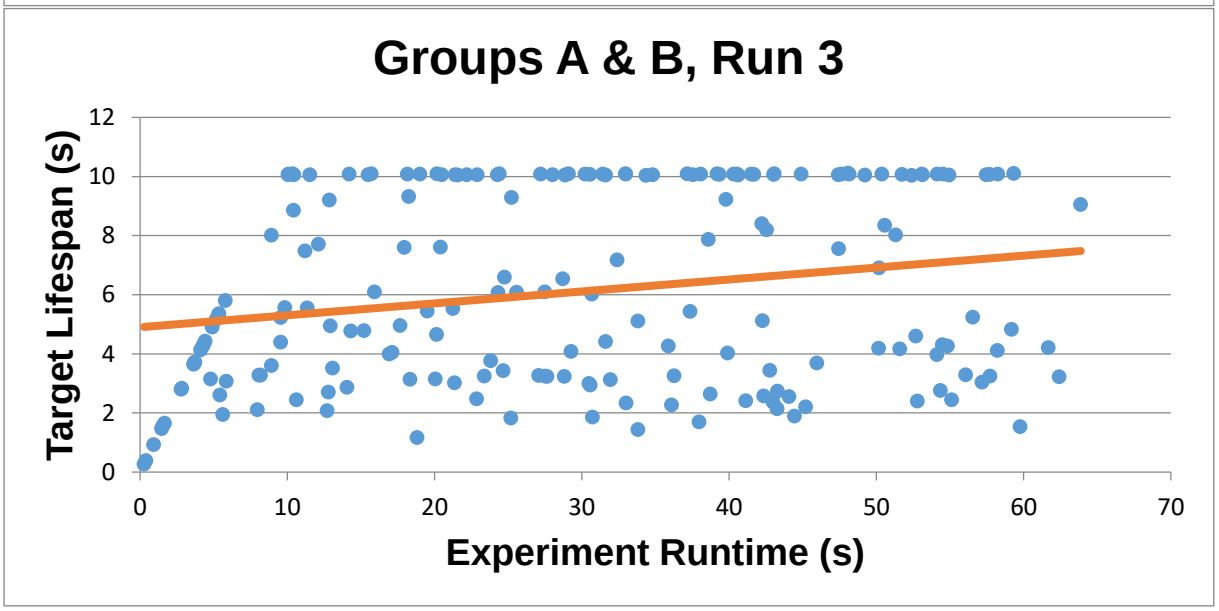


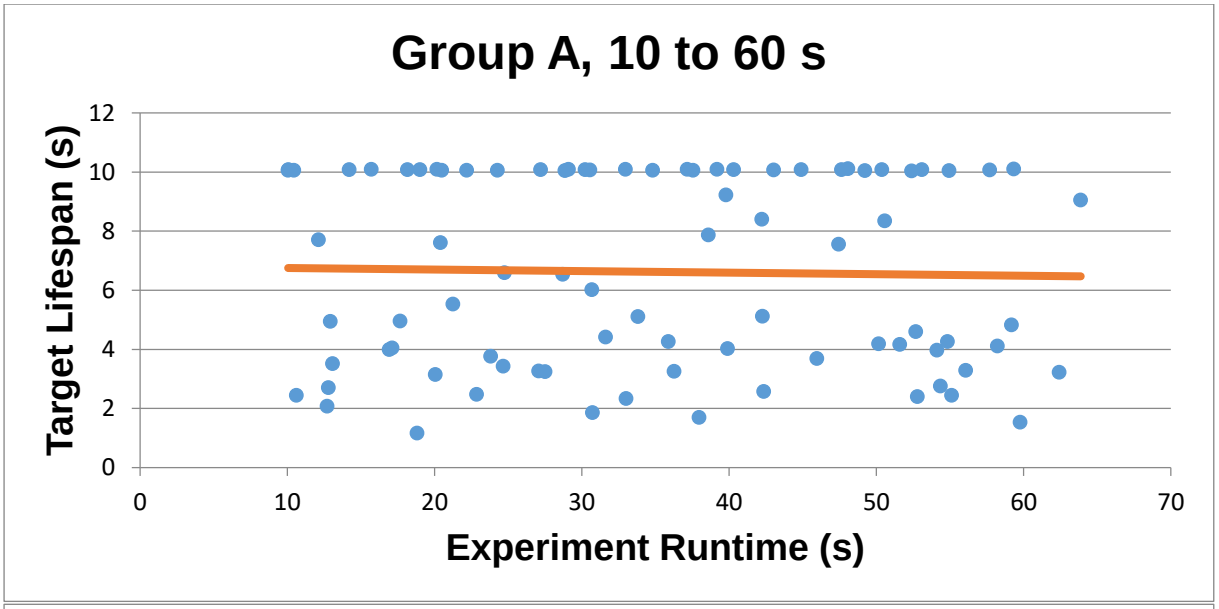
Chart 4: Target Time on Screen, Groups A & B, Run 3

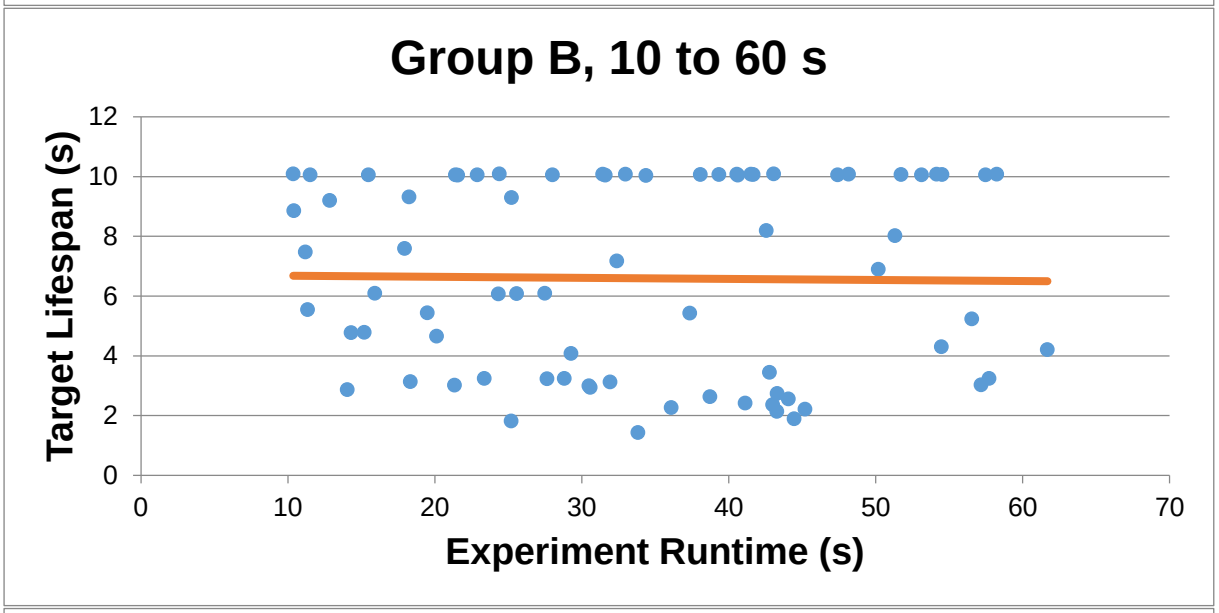
In these graphs, an increasing trend can be observed in the time that targets remain on-screen during the experiment. This result is due to a statistical artefact, originating from the combination of two factors:

Firstly, it can be easily observed that the top-left corners of the graph is devoid of data. This is entirely reasonable given the nature of the axes: since the X axis represents the total time elapsed since the start of the experiment, and the Y axis the time for which each target remained on the screen, it is impossible for data to exist in that corner—it would mean that a target had been on-screen since before the start of the experiment. Nevertheless, the concentration of data in that area attributes a clearly visible increasing trend to the chart. M

Moreover, it is evident that in the third run, a large number of targets during the third run were never shot down, instead remaining on-screen and de-spawning naturally after 10 seconds (easily observable especially in the third graph by the large number of points over the 10-second gridline), much more so than in the previous two. This can simply be attributed to the increased difficulty of the task during this run. However, this means that the graphs contain greatly reduced useful data when compared to the ones obtained from the first and second runs, since missed targets simply form a “horizontal line” of data points at the 10-second mark, providing no real imformation regarding the evolution of a trend.

The combination of the increasing trend in the first 10 seconds elapsed with the lack of data “within” the chart, create the “illusion” of an increase in the targets’ lifespan. In light of that, the plots were recreated, omitting the data regarding the first ten seconds. The new plots are presented below:





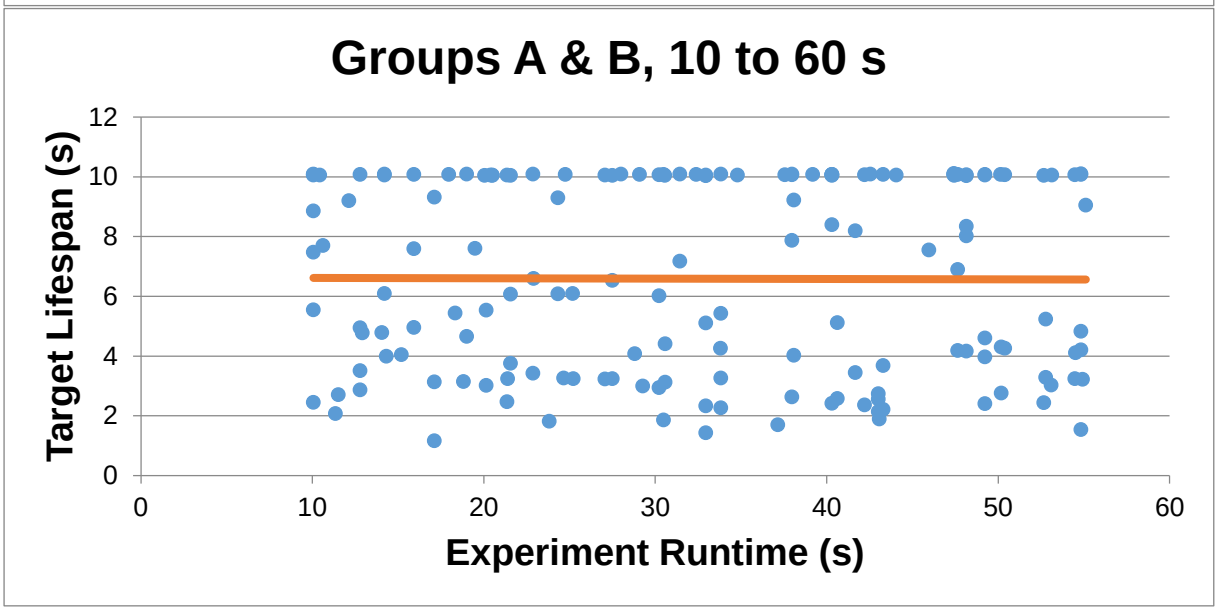


Chart 5: Target Time on Screen, Groups A & B, Run 3, 10 to 60 seconds

It can easily be seen that by removing the data regarding the first 10 seconds, the increasing trend previously appearing on the data has disappeared, a result confirmed by the p-values of the new trendlines (Table 6). However, no decreasing trend is observed either. This could be attributed to a number of reasons. Our prevalent hypotheses are that either

a) The modified task was too difficult for the participants to be able to adapt/ develop their strategy for effectively completing it in 1 minute, or
b) The participants had already achieved the maximum level of familiarisation with the virtual arm and its control mechanism, and therefore exhibited the same "plateau" effect observed in the second run

It is worth noting that these two hypotheses are somewhat contradictory. Further experimentation on the modified task is necessary to determine which, if any, of the above hypotheses is true.

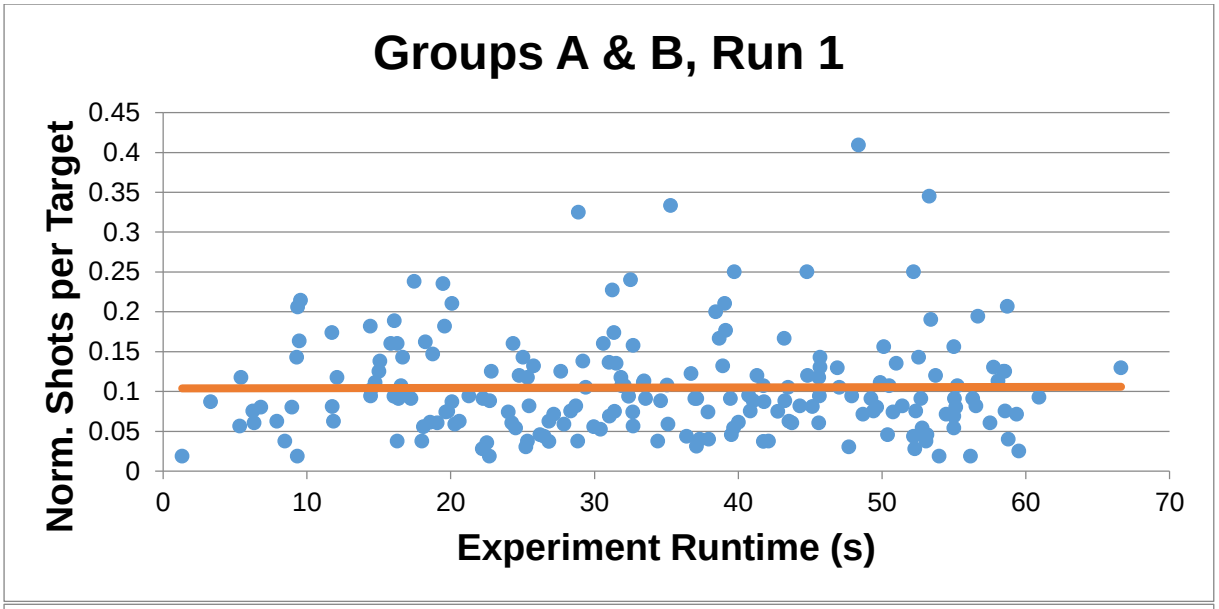
Table 6: P-values of Linear Regression trend-lines, Run 3

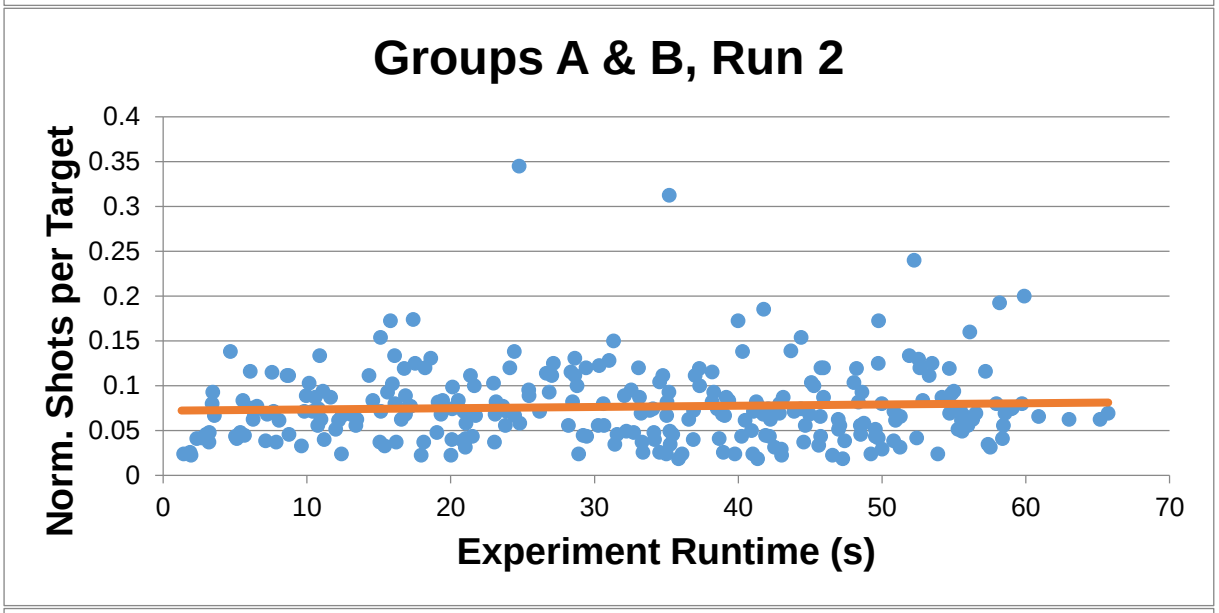
	Run 3
Group A	0.8195
Group B	0.8948
Groups A & B	0.6554

7.3 Shots per Target charts

The next step was confirming the above results using the data regarding the shots fired by each player. It was already known which of the shots fired by the players were on-target, making it easy to determine how many shots were required to shoot down each of the targets. However, as no restriction was imposed on the number of shots the players could fire, the total number of shots fired in each run by each player varied wildly, ranging from 16 to 93. To offset that effect, the following procedure was used: the shots fired to hit of the targets were "normalised" by dividing them with the total number of shots fired during the run. Missed targets were ignored during this procedure.Therefore, if a subject were to shoot 3 times at a target but miss it, and afterwards shoot 4 times before hitting the next target, the algorithm would divide all of these 7 shots with the total number of shots fired during the subject's run. Few such cases were observed however, especially in the first two runs, and therefore the results can be considered unaffected by them.

It was theorised that the resulting plots would mirror the tendency shown in the target screen-time plots; as the subjects became more familiar with the task, they would need fewer shots to hit each of the targets. That was not the case however, as demonstrated below:





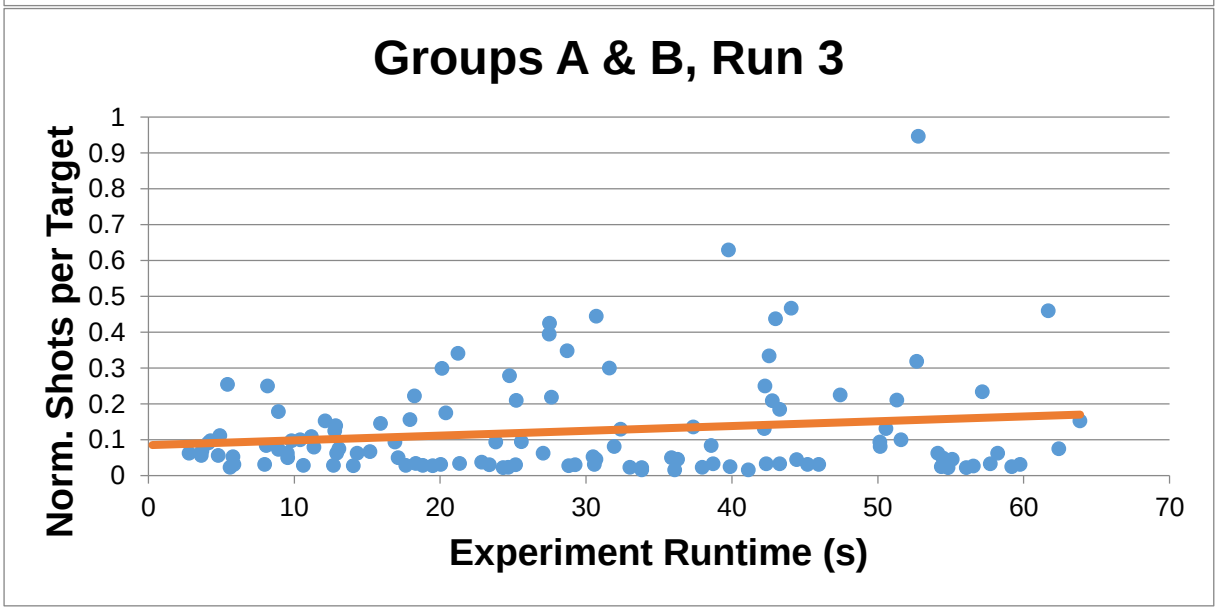


Chart 6: Normalised Shots between Hits, Groups A & B, Runs 1, 2 & 3

As is made obvious by the above graphs, no significant trend is observed during any of the runs. The increasing trend seemingly observed in the third run was deemed insignificant due to a high p-value of the regression line (see Table 7). Similar results were observed in the graphs representing each group and run separetely, (not presented here to maintain conciseness.)

Table 7: P-values of Linear Regression trend-lines, Shots per Target charts

	Run 1	**Run 2**	**Run 3**
Groups A & B	0.9129	0.3743	0.0854

Our prevalent hypothesis for the interpretation of these results stems from the fact that the targets appear randomly in 3D space during the experiment. We theorise that the location of the target, rather than the skill of the test subject, is the defining factor determining the amount of shots required to hit it. This conclusion was drawn based on the observation that nearby targets were much easier for subjects to hit than faraway ones, especially in the third run were the beam was removed. Further testing is deemed necessary to support this hypothesis.

7.4 End Effector Position Data

As mentioned before, the Position of the virtual arm's end-effector was also tracked during the experiment. This was done by placing a small spherical object at the end of the capsule representing the palm, and acquiring its transformation vector for each frame of the experiment's duration. This data was used to obtain the length of the curves traced by the virtual arm during the experiment. To achieve this, the data was imported into Matlab, wherein it were parsed as XYZ position vectors. Each of these vectors was then subtracted from the following one, thus determining the vector representing the length traced by the arm between the two positions in space. The norm of this last vector was then computed and added to the overall length of the curve. By doing this for all data points, the full length of the curve for each run was obtained.

It was initially theorised that the results would show an increase in curve length during the third run of the experiment. This hypothesis was based on the observation that test subjects would more often than not change the way they moved during the third run. Namely, during the first two runs when the beam was visible, many subjects would retain their upper arm and forearm in an approximately fixed position, instead using the movement of their wrist to aim, and recalibrating their shots after seeing the point in space that the beam would reach. By contrast, during the third run, subjects were unable to use that mechanism, as the beam was rendered invisible. Therefore, they would attempt to aim at targets by extending their physical arm in an attempt to align the virtual arm with the target in the VE. This movement seemed to have a much larger range-of-motion than the one seen in previous runs, leading us to assume that the corresponding curves would exhibit longer length. The results did not confirm this hypothesis, however. As can be seen in Table 8, the third runs have the shortest average curve length of all, as well as the shortest of the three runs for

each subject. Even in the rare cases where it exhibits longer length than the first or second runs, it still remains shorter than the other one.

Table 8: Curve Lengths per Group & Run

Group 1	Run 1	Run 2	Run 3	**Group 2**	Run 1	Run 2	Run 3
F1	67.46	81.69	67.71	***F1***	79.80	88.90	74.42
F2	101.92	107.17	83.63	***F2***	59.17	65.31	81.99
F3	75.25	90.32	52.45	***F3***	110.54	90.70	67.81
F4	82.62	84.42	69.67	***F4***	77.37	63.70	52.03
F5	80.78	72.47	64.91	***F5***	60.59	57.51	37.28
F6	65.89	91.59	57.15	***M1***	70.70	68.80	40.25
M1	55.30	76.87	52.23	***M2***	72.04	73.32	45.82
M2	89.74	64.59	68.99	***M3***	67.76	80.93	40.36
M3	81.51	75.95	67.90	***M4***	70.67	68.59	62.70
M4	89.54	81.14	55.71	***M5***	58.45	82.40	75.43
M5	89.84	78.37	61.49	***Average***	72.71	74.02	57.815
M6	84.99	84.80	54.64				
Average	80.40	82.45	63.04				

There are two proposed explanations for this:

i) Because the length measured corresponds to the end of the virtual arm's palm, it is heavily affected by the movements of the wrist. As a result, the many small-range movements at the wrist during the first runs accumulate to provide a larger length when compared to much fewer but larger-range motions of the whiole arm observed during the third one

ii) The largest contribution to the overall curve length is not actually provided by aiming at one target, but rather by the large movements of the arm when moving from one target to the next. As a result, the second run, which usually exhibits the highest number of targets hit, contains more of these movements than any of the other two, especially the third one

(where the scores are usually lowest), thus resulting in a larger overall curve length.

The seconds explanation seems more probable to our research team at this point. However, both of these hypotheses will be tested by future experiments.

7.5 Questionnaire data

The results from the questionnaire generally illustrated a positive user experience during the experiment. The answers to each question are displayed in column chart form, followed by explanatory comments.

Question 1: How much delay did you experience between your movements and their representation in the Virtual Environment?

1-None *7-Very large*

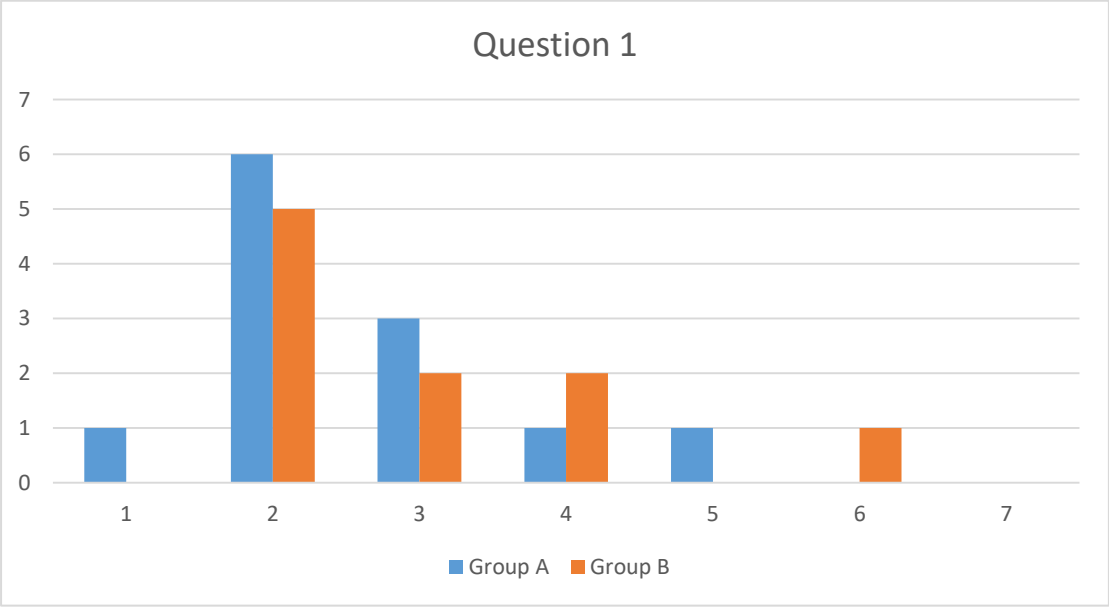


Chart 7: Questionnaire Answers, Question 1

12 out of 22 subjects felt that they experienced a very small delay or no delay at all in the translation of their movements into the VE, while only 2 commented that the delay was above average and still not “very large”.

Question 2: How much did the motion tracker and HMD interfere with your performance in the designated task?

1-Not at all *7-Very much*

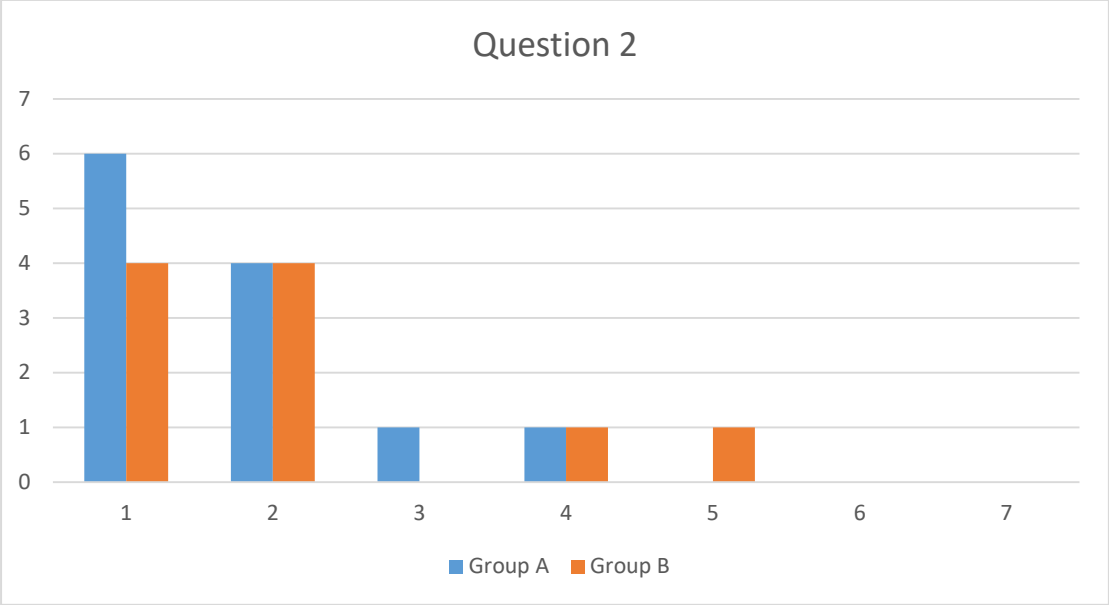


Chart 8: Questionnaire Answers, Question 2

18 out of 22 subjects were not at all or very slightly obstructed by the motion tracker and HMD during their movement (the users' comments illustrate that this obstruction was mainly attributed to the HMD's cable, especially for taller users).

Question 3: How capable in moving within the environment and interacting with it did you feel at the end of the experiment?

1-Not at all *7-Very much*

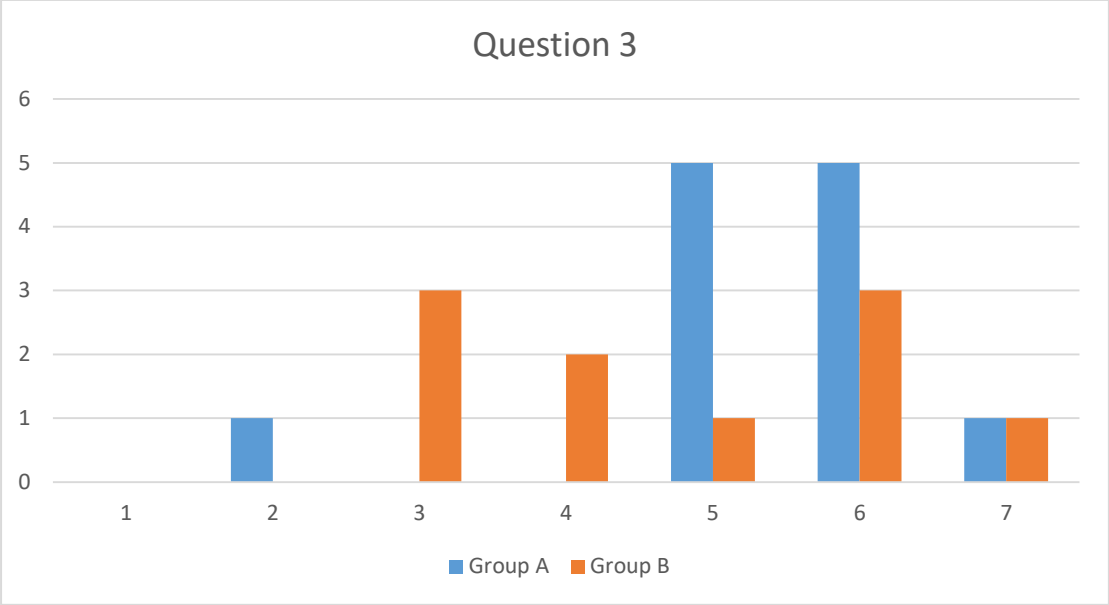


Chart 9: Questionnaire Answers, Question 3

16 out of 22 subjects felt they became skilled in moving and interacting with the VE by the end of the experiment. However, the subjects belonging to the first group reported a higher level of adaptation than those in the second group

This disparity was further illustrated by questions 4 and 5, presented below

Question 4: How much harder or easier was the task made by the fact that your arm was invisible/visible?

1-Much harder *7-Much easier*

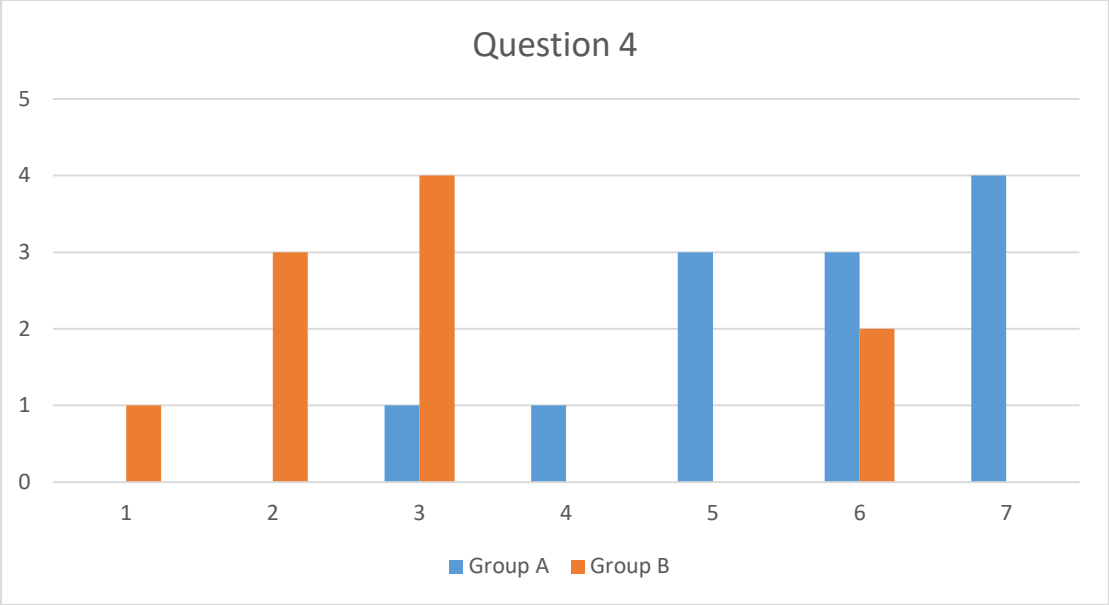


Chart 10: Questionnaire Answers, Question 4

Question 5: How much was the naturalness of your interactions with the environment decreased or increased by the fact that your arm was invisible/visible?

1-Severely decreased *7-Severely increased*

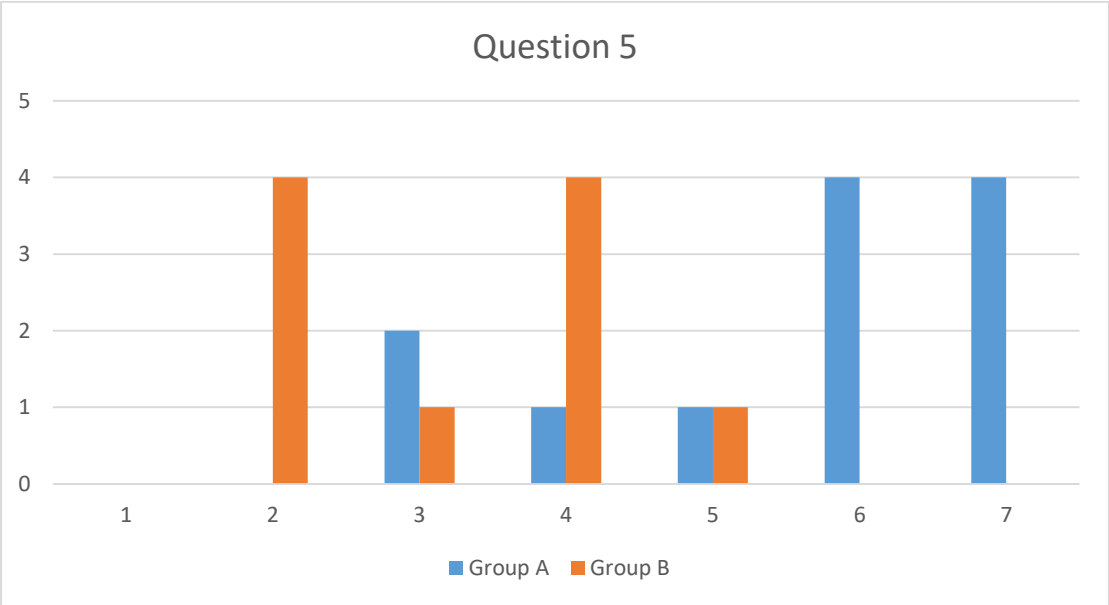


Chart 11: Questionnaire Answers, Question 5

The results make evident the fact that subjects belonging to Group A found that the visibility of their arms increased both their performance and the naturalness of their interactions, whereas those in Group B report the opposite effect occuring owing to the fact that their arms were invisible. This leads to the conclusion that, while the visibility of the arm seems to have had no significant effect on the users' objective measures of performance, it does seriously affect satisfaction—namely,users prefer being able to see their whole arm in the VE, rather than just the wrist and palm. This finding echoes the results of previous research, such as Heidicker et. Al (2017), whose findings illustrate that full body avatars with full motion control exhibit better results in social VR environments, and Mohler et.al (2008), who demonstrated that full-body avatars improve the users' ability to judge distance in immersive VEs

Question 6: How experienced are you in playing such games (First Person Shooters)?

1-Not experienced at all *7-Very experienced*

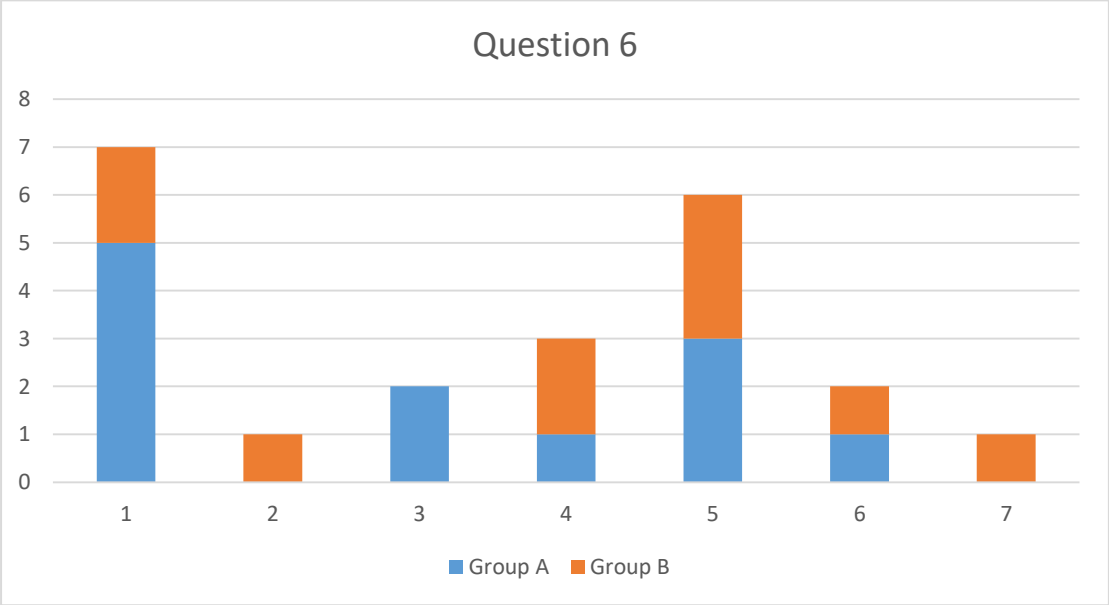


Chart 12: Questionnaire Answers, Question 6

The answers to Question 6 are presented in order to provide a complete picture of the data, although this was a demographic question and does not particularly affect the above data

8 Conclusions

The results of the experiment described above echo the findings of pre-existing research in the field. Namely, it is clearly demonstrated that the ability to control one's arms plays a paramount role in increasing task performance in VEs, regardless of whether the limbs are entirely visible (same as results from Heidicker et.al 2017) . Visibility of an avatar body is still of essence, however, as it substantially increases the subjective sense of presence in the environment experienced by the users. This could be compared to the conclusion drawn by Slater et. al (1996) in "*Immersion, presence, and performance in virtual environments: An experiment with tri-dimensional chess."* The researchers wrote:

"We argue that although increased immersion may well improve performance in certain tasks due to the higher quality and quantity of information available, there is no particular reason to expect presence to improve performance. "

In the particular case of the target practice tasks, since the level of immersion remained equal throughout tests, the results were very similar, despite the variance in the experienced presence.

Our emergent hypothesis that the importance of visibility is dependent on the nature of the task was not confirmed by the results of the experiments—subjects exhibited similar results between groups in the third run as well as the first two. No definite conclusion can be drawn as regards the general validity of the hypothesis, however, since it remains undetermined whether the visibility of the arm plays a paramount role in a different kind of task

Moreover, both the objective and subjective results provided by the experiment and questionnaires, as well as the corresponding analyses, clearly demonstrate the fact that IMU technology is a viable tool for human motion tracking in VR and AR environments. While this has already been indicated by its use in rehabilitation research, (such as Merians et. al 2009,Wittmann et. al. 2015, etc, also mentioned above), this thesis sheds light on the applicability of this technology in various other fields, such as game development, training, remote machine operation, industrial applications, etc.

9 Future work

Our intention is to continue our work in this field in two main directions

Firstly, the improvement of our proprietary motion tracking system is a crucial step in furthering research. By incorporating magnetometer measurements, determining the exact offsets of each sensory unit and fine-tuning the code for both the Arduino and Unity, we hope to achieve vastly reduced drift errors and calibration times, as well as eliminate or at least minimise the need to recalibrate the sensors as often as is required now. Following that, our intent is to expand our design to include the movements of the second arm,as well as the fingers, progressing toward complete body tracking in a VE. Our hope is that due to its ease of use, reduced demands in processing power, and minimal obstructions to the user's movements, the developed system can replace or enhance existing tools used for avatar body control in VR environments.Moreover, by incorporating other technologies, such as haptic feedback devices, into our design schemes, we will be able to vastly expand our ability to simulate different tasks in the existing (or another) VE.

Secondly, by utilising the VE developed during this thesis, combined with our lab's expertise in the field of cognitive Ergonomics, we propose to create a number of simulations of different tasks requiring the use of one's arm and hand motions in VR, and continue the experimentation process started by the presented shooting task. In this manner, we hope to be able to determine the answer to questions raised by this thesis, such as the importance of limb visibility in VR environments and the dependence of said importance on the nature of the task. In addition to that, however, we intend to create and evaluate simulations of various different tasks, for purposes exceeding Ergonomics/Human Factors research. For example, simulations of industrial situations incorporating Human-Robot Interaction can be designed, building on the work of Matsas & Vosniakos (2015), or of tasks requiring tele-operation of machinery. It is our belief that such applications can be used both as a testbed for experimentally evaluating theoretical advances in various fields of research, and as a tool for training humans for the corresponding real-world tasks.

10 Bibliography

Buchholz, B., & Wellman, H. (1997). Practical Operation of a Biaxial Goniometer at the Wrist Joint. Human Factors , 39, 119-129.

Chen, X. (2013). Human motion analysis with wearable inertial sensors. Ph.D. dissertation, University of Tennessee.

Heidicker, P., Langbehn, E., & Steinicke, F. (2017, March). Influence of avatar appearance on presence in social VR. In 3D User Interfaces (3DUI), 2017 IEEE Symposium on (pp. 233-234). IEEE.

Karakikes M. (2017). Development and Evaluation of a Wearable Motion Tracking System, to Support Hand-Tool Design, Diploma Thesis, National Technical University of Athens

Kilteni, K., Bergstrom, I., & Slater, M. (2013). Drumming in immersive virtual reality: the body shapes the way we play. IEEE transactions on visualization and computer graphics, 19(4), 597-605.

Kilteni, K., Normand, J. M., Sanchez-Vives, M. V., & Slater, M. (2012). Extending body space in immersive virtual reality: a very long arm illusion. PloS one, 7(7), e40867.

Lange, B., Suma, E. A., Newman, B., Phan, T., Chang, C. Y., Rizzo, A., & Bolas, M. (2011, July). Leveraging unencumbered full body control of animated virtual characters for game-based rehabilitation. In International Conference on Virtual and Mixed Reality (pp. 243-252). Springer, Berlin, Heidelberg.

Leonard, L., Sirkett, D., Mullineux, G., Giddins, G. E., & Miles, A. W. (2005). Development of an in-vivo method of wrist joint motion analysis . Clinical Biomechanics , 20, 166-171.

Luo, Z., Lim, C. K., Chen, I. M., & Yeo, S. H. (2011). A virtual reality system for arm and hand rehabilitation. Frontiers of Mechanical Engineering, 6(1), 23-32.

Matsas, E. & Vosniakos, GC. (2015). Design of a virtual reality training system for human–robot collaboration in manufacturing tasks. Int J Interact Des Manuf (2017) 11: 139.

Merians, A. S., Tunik, E., & Adamovich, S. V. (2009). Virtual reality to maximize function for hand and arm rehabilitation: exploration of neural mechanisms. Studies in health technology and informatics, 145, 109.

Mohler, B. J., Bülthoff, H. H., Thompson, W. B., & Creem-Regehr, S. H. (2008, August). A full-body avatar improves egocentric distance judgments in an immersive virtual environment. In Proceedings of the 5th symposium on Applied perception in graphics and visualization (p. 194). ACM.

Oberlander, K. (2015). Inertial Measurement Unit (IMU) Technology. Inverse Kinematics: Joint Considerations and the Maths for Deriving Anatomical Angles.

Osumi, M., Ichinose, A., Sumitani, M., Wake, N., Sano, Y., Yozu, A., ... & Morioka, S. (2017). Restoring movement representation and alleviating phantom limb pain

through short-term neurorehabilitation with a virtual reality system. European journal of pain, 21(1), 140-147.

Riva, G., Bacchetta, M., Baruffi, M., & Molinari, E. (2002). Virtual-reality-based multidimensional therapy for the treatment of body image disturbances in binge eating disorders: a preliminary controlled study. IEEE Transactions on Information Technology in Biomedicine, 6(3), 224-234.

Slater, M., & Wilbur, S. (1997). A framework for immersive virtual environments (FIVE): Speculations on the role of presence in virtual environments. Presence: Teleoperators and virtual environments, 6(6), 603-616.

Slater, M., Linakis, V., Usoh, M., Kooper, R., & Street, G. (1996, July). Immersion, presence, and performance in virtual environments: An experiment with tri-dimensional chess. In ACM virtual reality software and technology (VRST) (Vol. 163, p. 72). New York, NY: ACM Press.

Slater, M., Spanlang, B., Sanchez-Vives, M. V., & Blanke, O. (2010). First person experience of body transfer in virtual reality. PloS one, 5(5), e10564.

Smeragliuolo, A. H., Hill, N. J., Disla, L., & Putrino, D. (2016). Validation of the Leap Motion Controller using markered motion capture technology . Journal of Biomechanics , 49, 1742-1750.

Steuer, J. (1992). Defining virtual reality: Dimensions determining telepresence. Journal of communication, 42(4), 73-93.

Witmer, B. G., & Singer, M. J. (1998). Measuring presence in virtual environments: A presence questionnaire. Presence, 7(3), 225-240.

Wittmann, F., Lambercy, O., Gonzenbach, R. R., van Raai, M. A., Höver, R., Held, J., ... & Gassert, R. (2015, August). Assessment-driven arm therapy at home using an IMU-based virtual reality system. In Rehabilitation Robotics (ICORR), 2015 IEEE International Conference on (pp. 707-712). IEEE.

Won, A. S., Bailenson, J., Lee, J., & Lanier, J. (2015). Homuncular flexibility in virtual reality. Journal of Computer-Mediated Communication, 20(3), 241-259.

Appendix A: Code

A.1 Arduino Sketch

The following Sketch, running in the Arduino Microprocessor, obtains the data from the MPU9250 and translates it into the form of quaternions. Afterwards, these quaternions are passed through Unity through the serial port.

```
//This sketch obtains the data from the DMP of 3 MPU6050 or 9250
units, translates it into Quaternions,
//and writes them to the serial port to be read by Unity.

//The number of printouts here as well as in the MPU6050_Wrapper and
DeathTimer libraries has been minimised
//because they were causing error and warning messages in Unity, and
slowing the process.

// I2Cdev and MPU6050 must be installed as libraries, or else the
.cpp/.h files
// for both classes must be in the include path of your project

#include "I2Cdev.h"
#include "MPU6050_Wrapper2.h"
#include "TogglePin.h"
#include "DeathTimer2.h"
#include "MatrixMath.h"

// Arduino Wire library is required if I2Cdev I2CDEV_ARDUINO_WIRE
implementation
// is used in I2Cdev.h
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
#include "Wire.h"
#endif

// define the output as a Quaternion, to be read by Unity
#define OUTPUT_READABLE_QUATERNION

// if using 3 MPUs, create and array
const bool useThirdMpu = true;
MPU6050_Array mpus(useThirdMpu ? 3 : 1);

//define the pins where the MPUs are connected
#define AD0_PIN_0 3  // Connect this pin to the AD0 pin on MPU#1
#define AD0_PIN_1 5  // Connect this pin to the AD0 pin on MPU#2
#define AD0_PIN_2 7  // Connect this pin to the AD0 pin on MPU#3

#define LED_PIN 13 // (Arduino is 13, Teensy is 11, Teensy++ is 6)

#define OUTPUT_SERIAL Serial

uint8_t fifoBuffer[64]; // FIFO storage buffer

// orientation/motion vars
Quaternion q;        // [w, x, y, z]         quaternion container
```

```
TogglePin activityLed(LED_PIN, 100);
DeathTimer deathTimer(5000L);

// ================================================================
// ===                      INITIAL SETUP                       ===
// ================================================================

void setup() {
  // join I2C bus (I2Cdev library doesn't do this automatically)
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
  Wire.begin();
  Wire.setClock(400000); // 400kHz I2C clock. Comment this line if
having compilation difficulties
#elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
  Fastwire::setup(400, true);
#endif

  // initialize serial communication
  // (9600 rate chosen because Unity can "catch up" to it, but it's
  // really up to you depending on your project)
  Serial.begin(9600);

  while (!Serial)
    ; // wait for Leonardo enumeration, others continue immediately

  // initialize device
  mpus.add(AD0_PIN_0);
  if (useThirdMpu){
  mpus.add(AD0_PIN_1);
  mpus.add(AD0_PIN_2);
  }

  mpus.initialize();

  // configure LED for output
  pinMode(LED_PIN, OUTPUT);


  // load and configure the DMP
  mpus.dmpInitialize();

  // supply your own offsets here, scaled for min sensitivity
  MPU6050_Wrapper* currentMPU = mpus.select(0);  // offsets for IMU#1
 currentMPU->_mpu.setXGyroOffset(0.0);
  currentMPU->_mpu.setYGyroOffset(0.0);
  currentMPU->_mpu.setZGyroOffset(-50.0);
  currentMPU->_mpu.setXAccelOffset(0.0);
  currentMPU->_mpu.setYAccelOffset(0.0);
  currentMPU->_mpu.setZAccelOffset(0.0);
  if (useThirdMpu) {
    currentMPU = mpus.select(1);    // offsets for IMU#2
currentMPU->_mpu.setXGyroOffset(0.0);
  currentMPU->_mpu.setYGyroOffset(0.0);
  currentMPU->_mpu.setZGyroOffset(0.0);
  currentMPU->_mpu.setXAccelOffset(0.0);
  currentMPU->_mpu.setYAccelOffset(0.0);
  currentMPU->_mpu.setZAccelOffset(0.0);

    currentMPU = mpus.select(2);    // offsets for IMU#3
currentMPU->_mpu.setXGyroOffset(0.0);
```

```
  currentMPU->_mpu.setYGyroOffset(0.0);
  currentMPU->_mpu.setZGyroOffset(0.0);
  currentMPU->_mpu.setXAccelOffset(0.0);
  currentMPU->_mpu.setYAccelOffset(0.0);
  currentMPU->_mpu.setZAccelOffset(0.0);
  }
  mpus.programDmp(0);
  if (useThirdMpu){
    mpus.programDmp(1);
    mpus.programDmp(2);
  }
}

// ============================================================
// === handleMPUevent function ===
// ============================================================

void handleMPUevent(uint8_t mpu) {

  MPU6050_Wrapper* currentMPU = mpus.select(mpu);
  // reset interrupt flag and get INT_STATUS byte
  currentMPU->getIntStatus();

  // check for overflow (this should never happen unless our code is
too inefficient)
  if ((currentMPU->_mpuIntStatus &
_BV(MPU6050_INTERRUPT_FIFO_OFLOW_BIT))
      || currentMPU->_fifoCount >= 1024) {
    // reset so we can continue cleanly
    currentMPU->resetFIFO();
    //Serial.println(F("FIFO overflow!"));
    return;
  }

  // otherwise, check for DMP data ready interrupt (this should
happen frequently)
  if (currentMPU->_mpuIntStatus & _BV(MPU6050_INTERRUPT_DMP_INT_BIT))
{

    // read and dump a packet if the queue contains more than one
    while (currentMPU->_fifoCount >= 2 * currentMPU->_packetSize) {
      // read and dump one sample
     // Serial.print("DUMP"); // this trace will be removed soon
      currentMPU->getFIFOBytes(fifoBuffer);
    }

    // read a packet from FIFO
    currentMPU->getFIFOBytes(fifoBuffer);

#ifdef OUTPUT_READABLE_QUATERNION

    // obtain the Quaternion values from the DMP
    currentMPU->_mpu.dmpGetQuaternion(&q, fifoBuffer);

    //Print the number 1, 2 or 3 to distinguish between IMUs
    if (mpu==0){
      OUTPUT_SERIAL.print('1');
    }else if(mpu==1){
      OUTPUT_SERIAL.print('2');
    }else{
      OUTPUT_SERIAL.print('3');
```

```
    }
    //simply print the Quaternion Values to the Serial Port,
seperated by commas
    OUTPUT_SERIAL.print(",");
    OUTPUT_SERIAL.print(q.w);
    OUTPUT_SERIAL.print(",");
    OUTPUT_SERIAL.print(q.x);
    OUTPUT_SERIAL.print(",");
    OUTPUT_SERIAL.print(q.y);
    OUTPUT_SERIAL.print(",");
    OUTPUT_SERIAL.println(q.z);

#endif


  }
}

// ================================================================
// ===                      MAIN PROGRAM LOOP                     ===
// ================================================================

void loop() {

  static uint8_t mpu = 0;
  static MPU6050_Wrapper* currentMPU = NULL;
  if (useThirdMpu) {
    for (int i=0;i<3;i++) {
      mpu=(mpu+1)%3;
      currentMPU = mpus.select(mpu);
      if (currentMPU->isDue()) {
        handleMPUevent(mpu);
      }
    }
  } else {
    mpu=0;
    //choose the MPU that is due to be read
    currentMPU = mpus.select(mpu);
    if (currentMPU->isDue()) {
      //read from MPU
      handleMPUevent(mpu);
    }
  }

  int incomingByte = 0;   // Incoming from serial

  // If incoming data is available in the serial
  if (Serial.available() > 0) {
    incomingByte = Serial.read();   // read the incoming byte
  }

  activityLed.update();
  deathTimer.update();
}
```

A.2 Unity Script

The following script, running in the Unity environment, obtains the quaternion data transmitted to the serial port by the Arduino, and translates it into the movement of the virtual arm.

```
using UnityEngine;
using System.Collections;
using System.IO.Ports;
using System.Threading;

public class Oculus_motionControl : MonoBehaviour {

	public GameObject joint1,joint2,joint3;
	public float ardW, ardX, ardY, ardZ;
	public int mpu;
	public string dataFromArduino;
	private bool shouldExit = false;
	string[] sInput = new string[5] {"0", "0", "0", "0", "0" };
	public Vector3 axis=Vector3.zero;
	public float angle;
	public GameObject tracker;
	public Vector3 curr;
	SerialPort mySerialPort = new SerialPort ("COM1", 9600);

	Quaternion quat1= new Quaternion(1,0,0,0);
	Quaternion quat2= new Quaternion(1,0,0,0);
	Quaternion quat3= new Quaternion(1,0,0,0);



	void Start ()
	{
		curr = tracker.transform.position;
		Thread myThread = new Thread (new ThreadStart
(ThreadWorker));
		myThread.Start();
		mySerialPort.Open ();

	}

	void Update ()
	{
		if (mySerialPort.IsOpen == true) {
			if (sInput.Length == 5) {
				mpu = int.Parse (sInput [0]);
				ardW = float.Parse (sInput [1]);

				ardX = float.Parse (sInput [2]);
				ardY = float.Parse (sInput [3]);
				ardZ = float.Parse (sInput [4]);
				if (mpu == 1) {
					quat1.w = ardW;
					quat1.x = -ardX;
					quat1.y = -ardZ;
					quat1.z = -ardY;
				} else if (mpu == 2) {
```

```
                        quat2.w = ardW;
                        quat2.x = -ardX;
                        quat2.y = -ardZ;
                        quat2.z = -ardY;
                  } else if (mpu == 3) {
                        quat3.w = ardW;
                        quat3.x = -ardX;
                        quat3.y = -ardZ;
                        quat3.z = -ardY;

                  }

                  joint1.transform.rotation = quat1;
                  joint2.transform.rotation = quat2;
                  joint3.transform.rotation = quat3;

                  curr = tracker.transform.position;
            }
      } else {
            Debug.Log ("Connect the serial Port");
      }

}
void ThreadWorker ()
{
      while (shouldExit == false) {
            try{
            dataFromArduino = mySerialPort.ReadLine ();
            sInput = dataFromArduino.Split (',');
            }catch(System.Exception){
            }
      }
}

void OnApplicationQuit()
{
      mySerialPort.Close ();
      shouldExit = true;
}
}
```