Imperial College London

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Refining Gaze Prediction for Collaboration within a Virtual Environment

Author:
George Castle

Supervisors:
Thomas Heinis
Riccardo Bovo

Submitted in partial fulfillment of the requirements for the MSc degree in Advanced Computing MSc of Imperial College London

April 2021

Contents

Chapter 1

Introduction

1.1 Abstract

The widespread adoption of virtual reality (VR) within commercial and domestic applications creates new challenges in replicating human behaviour in physical reality (PR), in an attempt to provide a seamless experience when collaborating with other users in a virtual environment. VR collaboration has become common practice among myriad businesses as an alternative to travel and in-office meetings. The COVID-19 pandemic has drastically accelerated the need for this type of technology to allow for remote meetings without loss of communication. Yet the VR collaboration technology in use is far from replicating PR, often causing confusion among users and time wasted. The purpose of this work is to streamline the VR collaborative experience by increasing the accuracy of VR gaze prediction without the use of expensive eye-tracking technology or hand pointers, allowing collaborators to intuitively infer each other's gaze targets when working in a shared virtual environment. The data analysis results indicated a strong correlation between gaze positioning and head shift direction, suggesting a user's cone of vision may be reduced during gaze shifts of mid-range speed and amplitude, favouring the side of the cone analogous to the shift direction. Upon the processed data set, a gaze prediction model was trained to predict a user's gaze location in real-time whilst in a collaborative virtual environment (CVE). This prediction reduces the VR user's possible gaze location frustum, aiding a collaborator in estimating their associates target accurately, achieving mutual shared awareness.

1.2 Introduction

Gaze prediction is not widely adopted among VR collaboration software due to the lack of eye-tracking hardware in use and the inaccuracy of the predictions. However, the inclusion of accurate eye-tracking hardware in some modern, high end head-mounted display's (HMD's); such as the VIVE Pro Eye [1] and FOVE[6] [2], may create a discrepancy in shared awareness between VR users with eye-tracking technology and those without. The goal of this project was to increase the accuracy of gaze awareness, by creating a gaze prediction model for VR users without eye-

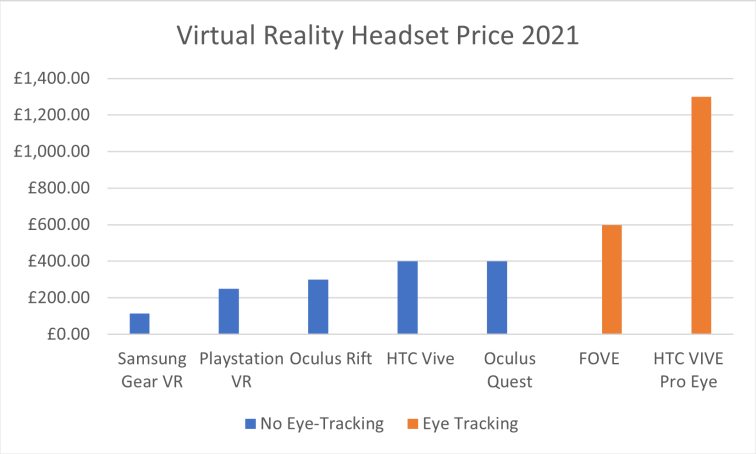


Figure 1.1: VR HMD price comparison.

tracking hardware installed. Enabling users who cannot afford high-end headsets with eye trackers to easily collaborate with other users, uninterrupted by a lack of mutual gaze awareness.

Many applications within VR require a shared awareness of a collaborators gaze, especially in a commercial setting. Applications such as data visualisation analysis rely heavily on a mutual understanding of where a collaborator is looking. However, due to the possible VR view frustum of each user covering an angle of 50° relative to their current head location, shown in Figure 2.1 [3], a joint understanding is difficult to attain. An accurate comprehension of a collaborators gaze target is currently only possible through the use of deictic pointing (indicating referents with pointing gestures during an utterance) or installed eye trackers. As mentioned previously, HMD's with eye tracking technology installed are more expensive than standard HMD's in widespread use, as shown in Figure 1.1. This price difference may prevent users purchasing an HMD with tracking, creating a disparity between the accuracy of each user's gaze tracking, until the headsets are more affordable. While the use of deictic pointing does overcome this issue, it is another action a user must remember to maintain whilst performing their collaborative task, causing possible distraction and breaking user immersion.

To create a gaze prediction model based on a user's head movements in VR a data set of relevant head and eye location data was required. Agtzidis et al [4] gathered a data set containing head location data from the inertial measurement unit (IMU) of an HMD and gaze location data from installed eye tracking technology, during naturalistic 360° video playback. This data set was used in this project to characterise different groups of head shifts based on peak velocity and amplitude, guided by current biological research. The position of the eyes during these different shift characterisations was analysed by calculating the distance between the gaze target location and head target location. With the intention of finding common patterns in eye movement, in relation to head movement, among the study participants. However, the average distance between gaze target and head target remained consistent

between each shift grouping. Head shifts were then split in to shifts to the right and shifts to the left, where it was found that the eyes are frequently to the right of the head target when turning right and left of the head target when turning left. This directional bias was predominant in mid range shift groups, allowing for a momentary reduction of the user's cone of vision during a gaze shift of this type. The method and results for this finding are detailed in Section 3.1.

From the processed data, a gaze prediction model was built using machine learning to predict eye gaze location based on real-time HMD head position and velocity received from the IMU of a virtual reality headset. The first models were built in MATLAB with the focus on minimising the potential cone of vision of a user during a gaze shift to below that of modern VR gaze research, increasing the accuracy of gaze prediction. Three models were created, one for each x, y and z coordinate. The models successfully reduced the possible cone of vision of a user throughout their VR usage. The models were then implemented in a VR setting using a two way Transmission Control Protocol (TCP) connection between MATLAB [5] and Unity [6]. This implementation was able to provide the accurate predictions generated by the models for the users gaze location in Unity. However, due to the complexity of each model and speed of the TCP connection the predictions could only be made every three seconds, creating a large amount of lag between user movement input and predicted gaze location output. The model could also only run on VR systems that were connected to a personal computer (PC) running MATLAB, not standalone devices such as the Oculus Quest [7]. To fix this, three new models were generated using TensorFlow [8] and converted to TensorFlow Lite [9], which is able to run locally on the Oculus Quest, without a PC connection. Unfortunately, the model with the highest accuracy was unable to be implemented, this is addressed in Chapter 4. However, the final implementation successfully reduced the high frequency cone of vision of a collaborator to below the current 40° high frequency cone of modern gaze research. These models were then implemented in a collaborative VR setting, built in Unity, ready for user testing.

Chapter 2

Background Research

To contextualise the project, existing neuroscience research on head contribution to gaze shifts was explored. Providing an insight in to how our eyes and head move in coordination with each other when unrestrained. Current computer science research on head and eye movement within a virtual environment was then reviewed to find how the constrictions to field-of-view (FOV) and the added weight of an HMD affect gaze shift movement.

2.1 Eye-Head Gaze Fundamentals

The eyes have a physical range of 50° with a fixed head position, however it has been found that they rarely rotate beyond 30° relative to the head [10] and are more frequently within 20° [10, 11, 12]. The eyes and head together provide a movement range of around 130°-140° in the horizontal axis and 110°-120° in the vertical axis [13] due to the limited anatomical movement range of the head and neck in the vertical axis [14]. Further work suggests that within the high frequency 40° cone of vision a greater reduction can be made based on high frequency eye positions. Sitzmann et al. (2016) [3] created visual saliency maps of users viewing a virtual environment and observed a mean gaze direction of 14°$\pm$12° relative to the head orientation Figure 2.1.

In this paper we suggest that the high frequency cone may be further reduced during a horizontal gaze shift involving the head where the target is more than 25° away. Gaze shifts are used to align an object of interest with the fovea (a pit in the retina that produces the clearest vision). Sidenmark and Gellersen (2020) [13] introduced the concept of 3 amplitudes of gaze zones: the first being shifts up to 25° from the current point of focus that can be comfortably achieved using the eyes alone. The second are shifts in the range of 25° to 50° where the eyes and head work together to achieve comfortable viewing, and the third zone is gaze shifts with amplitudes greater than 50° where eyes, head and torso must be used in conjunction for comfortable viewing Figure 2.2. Shifts within the first gaze zone will be challenging for a model to predict accurately as there is little head movement data to reduce the possible cone of vision, this zone should have the largest view frustum for possible

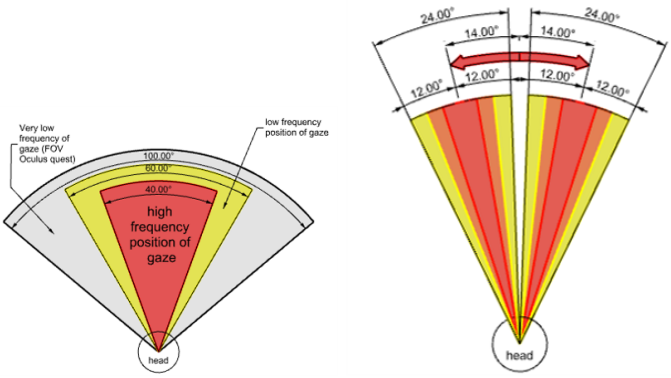


Figure 2.1: Cone of vision diagrams.

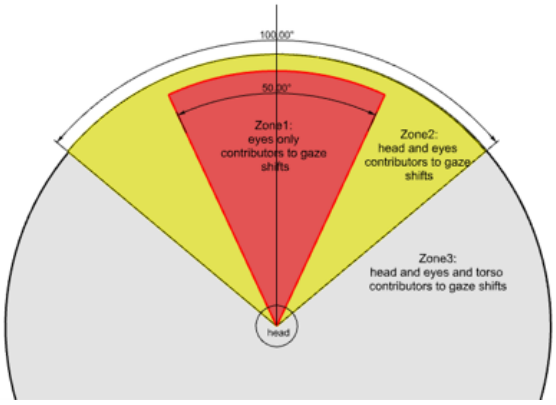


Figure 2.2: Gaze shift zones.

gaze targets of the three groups. However, the visualised frustum could be reduced to the high frequency gaze zone of 20° relative to the head, accurately encompassing the majority of gaze targets when the head is moving slowly [10, 11]. The project aims to reduce the possible cone of vision for shifts in zones two and three Figure 2.3 using gaze shift amplitude and velocity data for both the eyes and head.

The coordinated movements of the eyes and the head during fast gaze shifts are called saccadic eye and head movements [15]. These movements are usually conducted in two phases: during phase one the gaze is quickly shifted to the target using both the eyes and head. Phase two, the head continues to move until it reaches a comfortable viewing position, whilst the eyes move backwards at the same velocity as the head, keeping the gaze on the target using the vestibulo-ocular reflex (VOR) [16]. During this shift the eyes never go beyond the target, both eyes and head contribution to the shift are always positive allowing the following approach to be considered: When phase one is initiated the eyes will have a mean amplitude of $14\pm12°$ in the chosen direction [3], narrowing the possible cone of vision. However, such a cone may only be valid for a short duration as the body can quickly readjust to comfortable viewing, the head stops moving and the gaze once again only depends on eye movement which increases back to the 40° cone of vision Figure 2.4.

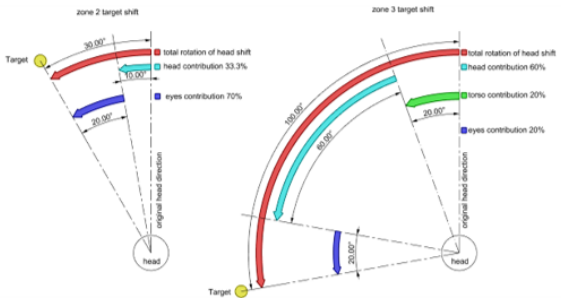


Figure 2.3: Shift contribution zone 2 and 3.

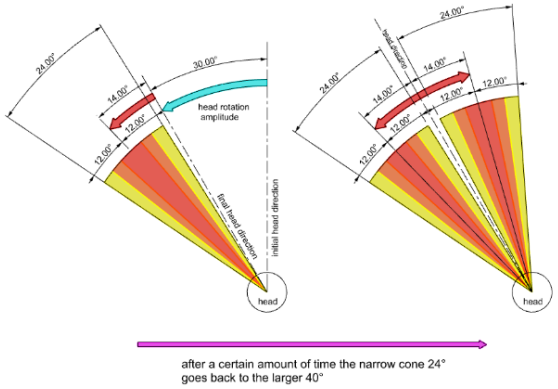


Figure 2.4: Simple model for gaze inference.

There is also variation between individuals in head contribution to gaze shifts within the eye-in-head range [17, 18, 19]. Head contribution is determined by numerous factors: energy expenditure, amplitude, initial eye-in-head position, gaze duration and position of the next target. This internal cost analysis leads to different outcomes among individuals[20, 18, 21, 19]. However, the data used for training the prediction model was gathered from 13 participants, 10 male, 3 female, which should reduce the variance between user's head contribution.

2.2 Perception Within Virtual Reality

Marmitt et al. (2002) [22] created an algorithm to evaluate predicted scanpaths (the path followed by the viewers eyes) in VR and showed that predictors built for classical viewing conditions perform significantly worse in a virtual environment. This suggests that there may be a difference to a viewers perception of a classical environment compared to a virtual environment, which may need to be considered when using characterisations of head and eye movement from neurological studies that were conducted in a traditional environment without the use of HMDs. However, the differences appear to be with how the evaluation algorithm interprets the environment as still images and stores no memory of them, not with how the user interprets the environment. The study also found that the VR user's 'fixations tend

to cluster about the central image region'[22, p. 9] [23, 24, 25], this is believed to be caused by the restricted viewing time. The data set used in this project [4] was also gathered in a restricted viewing time, yet the user's were unaware of the time limit and were encouraged to explore the environment, which should reduce the central fixation bias. There is also evidence to suggest that users perform more head movements when using an HMD in a virtual environment 'VR (Virtual Reality) users tend to rotate their head in some capacity for the majority of tasks, whereas our PR (Physical Reality) users tend to keep their heads still' [26, p. 5]. This must be taken in to consideration when assessing gaze behaviour in VR, as it indicates head movement is preferential when centering the gaze on a target and the VR cone of vision may be reduced from the classical cone of vision in Figure 2.1. Combining these findings indicates that the traditional cone of vision may be reduced for all types of gaze shifts when using an HMD in VR.

Despite these differences in perception between traditional and virtual environments, Sindenmark and Gellersen found that 'gaze shifts are generally performed in the same way in VR as in the real world.'[13, p. 21]. They observed that the amount of head movements observed was comparable to that of gaze shifts in the real world. This implies that current research of head contribution to gaze shifts in a traditional environment can be transferred to gaze shifts with HMDs in VR. As previously discussed, the study observes that larger gaze shifts are sometimes completed in stages of saccadic and VOR eye movements to keep the eyes in a comfortable position within the head whilst moving. The paper goes on to suggest that whilst this is similar to how large gaze shifts are conducted in the real world [27] the limited FOV of the HMD may affect how often the eyes wait for the head during these stages. If the eyes rotate further they may be outside the display's boundary without simulation of the virtual scene, and this might trigger the eyes to wait.

During the data evaluation described in Section 3.1, larger gaze shifts were be considered to be completed in stages to prevent a shift being incorrectly split and characterised using local minima of velocity or amplitude profiles. This would lead to shifts being evaluated upon only one of their component parts. An alternative was to characterise a shift based on head velocity returning to resting velocity of less than 3°/s 'The head was considered stationary when the velocity was less than 3°/s'[28, p. 2].

2.3 Eye-Head Gaze Data Set

As mentioned previously a large data-set of accurate eye and head positions, recorded whilst in a virtual environment, was required to perform the gaze analysis and for training the models to make gaze predictions based on head movements. The technical report '360-degree Video Gaze Behaviour' [4] includes a ground truth data-set detailing the head and eye x and y equirectangular coordinates during multiple naturalistic 360-degree videos, played to 13 subjects, using an HMD with eye-tracking technology (FOVE[6] virtual reality headset with integrated 120 Hz eye tracker)[2].

The paper introduces an algorithm for automatic classification of eye movements within the data set. The algorithm annotates the data in to component eye movements, fixation, smooth pursuit, saccades etc. The authors published the labelled data set with the original data set. However, despite the labels providing useful additional information, the unlabelled data-set was utilised for this project. This was because the model had to be trained only using data that would be available from a standard HMD, without eye-tracking software.

The data set [4] was gathered from 14 naturalistic videos, as well as one synthetically generated video that followed the target recommendations of Thaler et al.[29]. This created a comparison of guided gaze shifts in a controlled setting and unguided in a variety of natural surroundings, producing a diverse set of head and eye movement data. The videos were shown to each participant in a pseudo-random order to prevent fatigue bias of head movements in the same environment. This was excluding the synthetic video which was shown to each subject last, to prevent them thinking about the way they moved their eyes and head before it was necessary. There was also a re-calibration stage before each clip so that the subject is facing the same direction at the start of each playback session. The FOVE[6] HMD allowed for the eye tracking data to be gathered at 120 Hz, which is relatively high frequency when compared to previously published data sets such as Santini et al. [30, 31], who provided eye tracking data at 30 Hz (state of the art tracking at the time). The data was also gathered without any restraints placed on the user such as chin bars or chest straps, to prevent head and torso movement. Other studies [31] have opted for a chin bar to focus on eye movement relative to the head. However, these data sets are limited by the synthetic nature of the data capture and would be unsuitable for training a model to predict unrestrained, exploratory gaze shifts within a naturalistic environment.

The Author of the Technical Report [4], Ioannis Agtzidis, also wrote partner utility functions for processing ARFF data files [32] and for handling 360-degree data in MATLAB [33]. Making the data easy to utilise in a wide range of research topics. Functions such as the 'LoadARFF' function and the 'Rodrigues' rotation matrix function were used within this project for loading and manipulating the data set, prior to training the prediction model, as described in Section 2.4. Other functions were also used for validating the accuracy of conversion results, such as the 'EquirectToSpherical' and 'SphericalToCart' functions.

Another potential data set was gathered by Sitzmann et al.[3] Which focused on visual saliency within a virtual environment. Whilst most data sets from visual saliency analysis research are gathered at a lower frequency, limiting their versatility for accurate eye movement research, the eye tracking data was recorded at 120 Hz, in keeping with modern eye tracking technology. This data was also gathered over 122 participants, a much larger and potentially more reliable sample size. Unfortunately, the published data provides scanpaths in sequences of fixations, which would severely limit our interpretation of the eye and head movements, this is a common

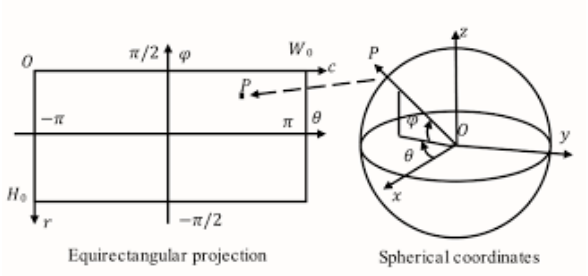


Figure 2.5: Equirectangular projection coordinates to spherical coordinates conversion.

occurrence among eye tracking data sets[3, 34, 35, 36]. Whereas, Agtzidis et al.[4] provided all head and eye movements recorded, which allowed the prediction model in this project to be trained with the same raw data that would be received from a modern HMD.

2.4 Data Processing

The data set provided by Agtzidis et al.[4] underwent certain manipulation prior to training the MATLAB regression learner model [5]. For example, any entry with a 'confidence' level lower than 1 was removed from the data set to prevent potentially erroneous data misleading the analysis. After which, the head and eye (x, y) equirectangular coordinates were converted to spherical coordinates (θ, ϕ) using the following formulas:

$$\theta = (\frac{x}{width}) \times 2\pi \tag{2.1}$$

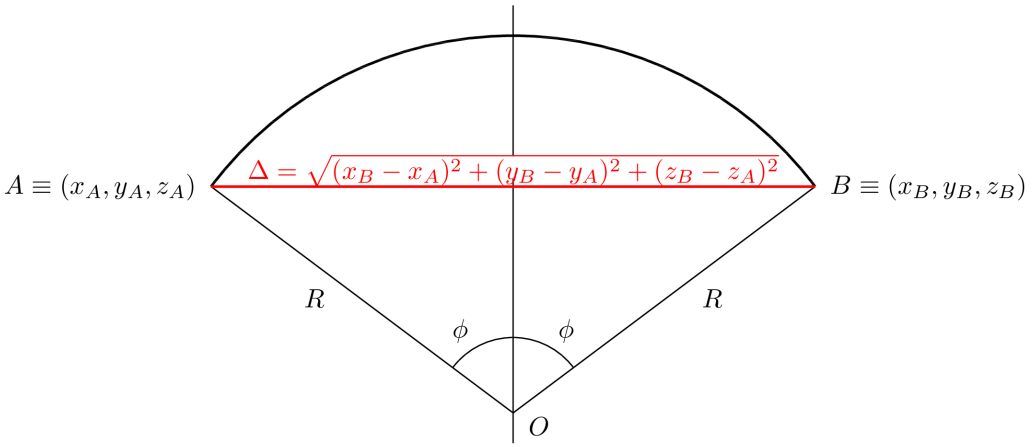
$$\phi = (\frac{height - y}{height}) \times \pi \tag{2.2}$$

With $width$ and $height$ being the pixel width and height of the corresponding equirectangular 360° video, which was provided in the metadata of each ARFF data file. The spherical coordinates were then converted in to world Cartesian (x, y, z) coordinates in 3 dimensions. This conversion was necessary as the coordinates of each ARFF data file were stored in 2D equirectangular projection (Latitude/Longitude format), shown in Figure 2.5, whereas the measurements provided by the IMU of the HMD and Unity both use world Cartesian (x, y, z) coordinates. A radius of 1 will be used for the conversion to world coordinates as the HMD coordinates are returned as a unit vector in the forward direction away from the headset, it is also possible to get this value whilst in a Unity environment using 'Transform.forward' [37] on the player object. The conversion from (r, θ, ϕ) to world Cartesian (x, y, z) coordinates can be done using the following formulas with a radius of $r = 1$ (unit sphere):

$$x = r \times \cos\theta \times \sin\phi \tag{2.3}$$

$$y = r \times \sin\theta \times \sin\phi \tag{2.4}$$

$$z = r \times \cos\phi \tag{2.5}$$



1. Δ, the $\overline{AB}$ distance, is known in terms of the coordinates of the two points.
2. By trigonometry, it is $\Delta/2 = R\sin\phi$ and $\phi = \arcsin(\Delta/(2R))$.
3. The length of the minimum arc on the sphere is $2R\phi$.

Figure 2.6: Angle between two points.

Once the data was converted to world Cartesian coordinates the velocity and amplitude of each entry could be calculated with respect to the previous entry using the distance in degrees between each (x, y, z) coordinate. The distance in degrees between two (x, y, z) coordinates can be calculated by finding the distance of the chord between the two points, formula 2.6, then using trigonometric identities and the radius (unit radius) to find the angle as shown in Figure 2.6.

$$\Delta = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \tag{2.6}$$

$$seconds = microseconds \times 1000000 \tag{2.7}$$

The velocity (°/s) of each head movement was then calculated by dividing the degree distance by the time difference between each entry and converting from microseconds to seconds, formula 2.7. The data was then split in to individual head shifts based on the velocity profile. A shift began when the head moved above the stationary velocity threshold $(> 3°/s)$ [28] and ended when the head went back to a stationary position $(< 3°/s)$. These shifts were then characterised, using their peak head velocity, in to groups. The groups were defined using the work of Saeb et al.[15] who measured how head rotation velocity variate depending on the target angle, Figure 2.7. Peak velocity of 120 °/s corresponds to a shift target angle of 90° and a low peak velocity of 25 °/s corresponds to a target at 50°. Overall head shift movement duration reported by Saeb et al. is approximately 1 second from the peak speed. However, despite the data from Saeb et al. being captured with a 'head-free' condition [15], the stimuli is of a synthetic nature with precise target locations, not of an exploratory nature. Consequently the shift classifications may be inaccurate for our data set [4] which almost entirely consists of natural, exploratory gaze shifts.

Unfortunately there is not a wealth of published research in unrestrained head shift

classifications during naturalistic 360° video playback. Fang et al. [38] investigated the coordination of the eyes and head during visual search and defined head shifts using classifications of peak speed and amplitude, similar to the work of Saeb et al. Despite this study using a synthetic stimuli, it did utilise exploratory, unrestrained head movements, not predefined shifts at a target angle. Fang et al also researched and gave a talk on 'Eye Position Distribution Depending on Head Orientation in Natural Scene Viewing' [39]. Unfortunately only an abstract is available, in which, they write 'The results showed a clear correlation between horizontal eye position and horizontal head orientation' [39, p. 1] and the 'results are consistent with our previous study in which a visual search task was performed' [39, p. 1], referring to the aforementioned study [38]. This suggests that the classifications of Saeb et al are similar to head shifts during natural, exploratory viewing, allowing for their results to be used when defining head shift groups in such a setting. Using Figure 2.7 [15, p. 8] four peak velocity groups were created to classify different head shits:

- group 1 $< 20°/s$
- $20°/s \leq$ group 2 $< 60°/s$
- $60°/s \leq$ group 3 $< 120°/s$
- $120°/s \leq$ group 4

The mean distance between eye and head gaze during a gaze shift was then calculated for each of these groups to determine if there was a discernable difference in gaze behaviour for one particular group, allowing us to refine the possible cone of vision for shifts of that classification. After which, the groups were further divided by amplitude and direction of shift. Amplitude groups were also defined using Figure 2.7 to correspond with their matching peak velocity profiles in to the following groups (with groups 1 and 6 being used to catch very small or large shifts that may mislead findings due to incorrect shift classification or erroneous data):

- group 1 $< 10°$
- $10° \leq$ group 2 $< 30°$
- $30° \leq$ group 3 $< 50°$
- $50° \leq$ group 4 $< 70°$
- $70° \leq$ group 5 $< 90°$
- $90° \leq$ group 6

After categorising each gaze shift by its velocity and amplitude profiles the data was then split by direction of gaze shift. To achieve this, each shift must first be redefined from world Cartesian coordinates to local Cartesian coordinates, relative to the first head position of the corresponding shift. The localisation of the shifts makes it easier to calculate in which direction the shift propagates, without going through each shift manually. This is due to the nature of world coordinate axes, if the shift starts 180°

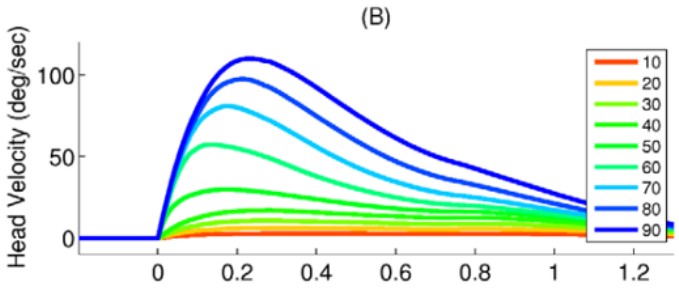


Figure 2.7: Shift amplitude against head velocity.[15, p. 8]

from the starting HMD pose, the positive and negative x axis is flipped, a shift to the right would appear to be a shift left when viewed from the initial HMD pose. The localisation process utilised three rotations, one around each axis, to align the head with the positive y axis ([0,1,0]). No translation was required as the HMD is always at the origin of the world axis in the data set coordinates.The Rodrigues' rotation formula [40] was applied to perform these rotations:

$$\mathbf{v}_{\text{rot}} = \mathbf{v}\cos\theta + (\mathbf{k}\times\mathbf{v})\sin\theta + \mathbf{k}\,(\mathbf{k}\cdot\mathbf{v})(1-\cos\theta) \tag{2.8}$$

$\mathbf{k}$ *is a unit vector describing an axis of rotation about which vector* $\mathbf{v}$ *rotates by angle* θ

The 'Rodrigues' rotation matrix function from the 360-degree data MATLAB utilities repository [33] written by Ioannis Agtzidis was used to implement the rotation matrix for each rotation, after finding the angle by which to rotate. Each rotation angle had to be found to align the head forward axis with the corresponding world axis plane. To do this the normal vector was first calculated between the head forward vector and the axis perpendicular to the rotation (world z axis), using the cross product of these vectors:

$$\mathbf{v}_{\text{ear}} = \mathbf{v}_{\text{head}} \times \mathbf{z} \tag{2.9}$$

The dot product of the head forward vector and desired aligning axis was then used with the determinant of the head forward vector, ear vector and aligning axis to find the rotation angle, using the following formulas:

$$\mathbf{dot} = \mathbf{v}_{\text{head}} \cdot \mathbf{z} \tag{2.10}$$

$$\mathbf{det} = x_{\text{head}}y_{\text{z}}z_{\text{ear}} + x_{\text{z}}y_{\text{ear}}z_{\text{head}} + x_{\text{ear}}y_{\text{head}}z_{\text{z}} - z_{\text{head}}y_{\text{z}}x_{\text{ear}} - z_{\text{z}}y_{\text{ear}}x_{\text{head}} - z_{\text{ear}}y_{\text{head}}x_{\text{z}} \tag{2.11}$$

$$\theta = \arctan(\mathbf{det}, \mathbf{dot}) \tag{2.12}$$

The rotation around each axis was then performed using the rotation matrix and the angle calculated. First, the head forward vector was aligned with the y, z plane by rotating around the z axis. After which, the vector was rotated around the x axis to align with the positive y axis. For the final rotation, to match the roll of the head, the provided 'angle_deg_head' variable, unique to each data entry, was used. This allowed for a complete recovery of the HMD pose for each entry of capture by rotating around the y axis (and aligned head vector).

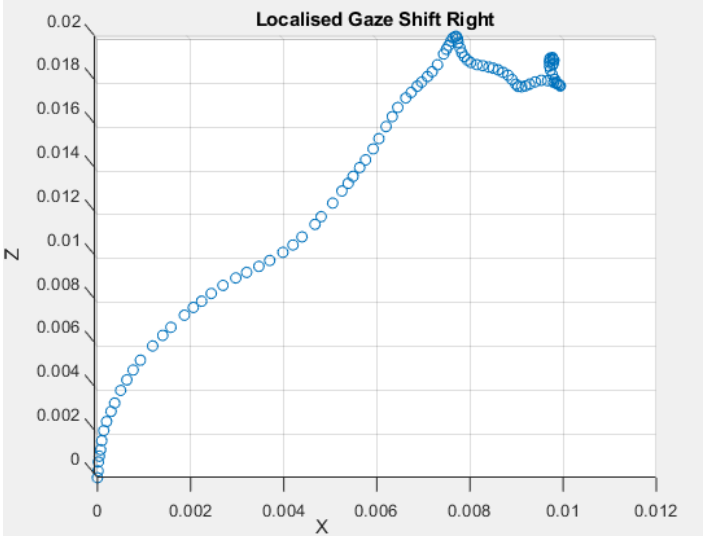


Figure 2.8: Localised gaze shift right looking down positive y axis

After the start of a shift was localised to the positive y axis, the following head shift entries were then made local to the first entry by multiplying with the same rotation matrix to align the axis. This allowed for visualisation of each shift from its origin, showing in which direction the shift propagated, relative to the users body orientation, an example shift is shown in Figure 2.8. Using this localised form, the shifts were then split in to shifts left and shifts right by checking if the shift proceeds in the negative or positive x direction respectively. After this split the final eye gaze position was compared to the final head position, by aligning the final head position to the positive y axis and, using the same rotation matrix, to localise the eye gaze world coordinate. The x value of localised eye vector was then observed to check if the eye gaze favours the direction of the gaze shift. For example, there was a higher frequency of eye gaze data positioned to the right of the head after a gaze shift to the right. This conclusion allows the cone of vision to be reduced when a user performs a gaze shift left or right. These results are recorded in Section 3.1.

2.5 Head and Eye Position as Input

Head pointing and eye pointing have been in development and use within human computer interaction (HCI) since the eighties, in an attempt to provide users with an alternative to mouse input [41, 42, 43, 44]. Gellersen and Sidenmark [41] Found that eye-only gaze allows for exploration around objects selected by the head supported gaze, affording a gaze target to be determined based on the head position even after the eyes have moved. In general it was shown that eye movement was faster and required less energy, yet was jittery and less controlled than head movement. However, this could be due to the low frequency eye tracking software utilised in these studies.

Due to the accuracy constraints of eye tracking it has also been found that VR and augmented reality (AR) users tend to prefer head input over eye input, [45, 46, 47] these studies propose to fix this issue by using eye movement for general gaze direction and head movement to refine the gaze positioning and confirmation. The focus of this project is on collaboration and awareness of a collaborators gaze within VR. However, input type could have an affect on the applicability of the final model. For example, if a user is using their head movement as input with a visible reticle centred between the eyes for selection, this could affect how much the eyes move and may produce unnatural head movements or gaze bias towards the reticle. This bias would not be present in the data set used for training, which is of a natural exploratory nature with no incentive for head over eye movement as input.

The model was implemented in Unity without a reticle, to prevent unnatural head movements. However, this limits the implementation to use within applications that do not use a reticle. Further work could be done to find the impact of a reticle on gaze behaviour in VR, but such applications can use the reticle to achieve shared awareness, without the need for a gaze prediction model. This is also why each user's predicted gaze location is not shown to them in the final implementation, only their collaborators; if each user could see their predicted gaze location this may produce an unnatural gaze direction bias towards the prediction, increasing the model's accuracy. Instead, each users predicted gaze is only visualised to their collaborators, not themselves.

2.6 Validation with Users

Further validation via a user study was necessary to understand how the models of gaze prediction affect mutual understanding of view. A collaborative task was created for two users to perform in a virtual environment with a visualisation of the gaze prediction model, via four view frustums, which the user could swap between at any time. Ideally, their eye and head movements would be tracked to find the precise error of the model. However, due to a lack of time and hardware the data gathering was unable to be performed. After the task was completed, each user was interviewed to determine if they found the model aided their collaborative understanding. The user study is described in Chapter 6.1

The collaborative task was based on the work of Wong and Gutwin (2014) [48] who carried out a study of deictic pointing in CVEs. In the study, the participants were asked to complete a set of referencing tasks where one participant was asked to indicate different referents to their partner, as well as three creative tasks. During these tasks the pairs were presented with scenarios where they would have to construct stories and make decisions using the CVE. For which, they created a cityscape visualisation [48] with the participants located on a balcony overlooking the city. Participants were asked to pretend to be roommates looking for an apartment to live in together, whilst taking in to consideration influencing factors such as 'type of neighbourhood, distance to work sites and traffic' [48, p5 (1381)].

This type of open-ended task without a defined solution is appropriate for analysing shared gaze as each participant will have a unique viewing experience, each finding different salient regions, with no predefined gaze targets. This allows for unbiased assessment of the accuracy of the generated model, whilst preventing any prediction or tailoring of the task to provide a result to fit the model. The freedom of the task should provide a unique result for each participant pair and create a realistic testing environment for the model. The only bias introduced could be created by an obvious choice, or salient area, within the CVE used. The created balcony view of the city was only from one side, preventing participants from having to turn more than 180°. While this type of task is suited to deictic pointing, which may be performed from a distance and in one primary direction, a small scale 360° environment may be better suited for testing short distance gaze prediction based on head movement.

Sitzmann et al.[3] studied visual exploratory behaviour in stereoscopic, static omni-directional panoramas recording head orientation and gaze direction to observe how existing saliency predictors may be applied to VR. The environments used were a combination of 14 indoor scenes and 8 outdoor scenes with no landmarks that may be recognised by the user, which would influence their gaze direction. Each environment was computer generated by an artist. They observed nearly 2,000 trajectories from 169 viewers, in both seated and standing position with a pseudo-randomised starting position. Users were then instructed to freely explore the scene and were provided with a pair of earmuffs to avoid auditory interference.

For a thorough evaluation of the gaze prediction model a combination of each testing environment was necessary. The model implementation relies on user's head movement to predict and is built for short-distance gaze collaboration, common in data visualisation analysis. Consequently, a localised 360° environment was developed for user testing similar to the environments of the 'Furniture World' [49, 50, 51]. Predicting at long distances increases the predicted potential cone of vision exponentially, due to the inaccuracy of the model, making deictic pointing a necessary requirement. If the model were more accurate, a smaller gaze indicator could be used, such as the beam used by Wong and Gutwin [48, p4 (1380)]. This type of indicator would be suited to larger environments. The inaccuracy of the gaze prediction model makes the use of indoor environments, such as were used by Sitzmann et al [3], appropriate for user validation. However, the viewing time of 30 seconds was increased to produce a more realistic testing scenario, allowing time for exploration and task completion. A set of collaborative tasks to perform with a partner could also be introduced within the localised CVE's, such as were used by Wong and Gutwin [48]. For example the same collaborative task of finding an apartment could be used, but instead of viewing a cityscape the participants could be inside the property surveying different rooms and assessing the pros and cons of the environment. There could also be a window view for testing long-distance gaze with similar objectives of type of neighbourhood and traffic. This would lead to a more realistic example of a collaborative task that may be conducted within VR. A sample

environment based on these suggestions was made as part of this project for user testing, the details of which are outlined in Section 5.3.

Whilst an apartment environment will provide a realistic example of a collaborative task, the familiarity of such an environment may make the users rely heavily on verbal referencing to achieve mutual awareness, instead of the gaze prediction model. However, there is not enough research on the impact of environmental familiarity on mutual awareness in VR to validate this hypothesis. The removal of verbal referencing from the test environment would inhibit the collaborative nature of the testing environment as shown by Eynard et al. [52], who found that a lack of verbal communication in a VR collaborative task impacts the performance, satisfaction and hedonism of the subject. To provide an insight in to the effect of spatial familiarity, a second section of the testing environment was created to display an apartment data visualisation. For example, graphs about traffic or theft in the local area were generated and displayed providing insights about the apartment to be discussed between collaborators. The introduction of unseen data in a novel environment encourages meaningful exploration and collaboration among the users, with only the gaze prediction model to gather a shared awareness of gaze target. An example data visualisation environment built for this project is detailed in Section 5.3.

The introduction of gaze indication alternatives such as 'pointing arms' or a 'laser pointer' [48] could also help to find user preference in comparison with the view frustum model. However, the user testing in this project was used to find a preference of model implementation, the use of pointers would have detracted from the evaluation of the model. The users did have controller visualisations which could be used to point, however it was found that this feature was used sparingly. Upon completion of the collaborative task the model's efficacy in supporting mutual awareness of visual focus was determined using questionnaires involving a rating system of each of the conditions with a system usability scale behavioural measure such as:

- The amount of shared focus
- Amount of verbal communication required
- Number of times pointers were used: higher mutual awareness should lead to reduced usage of hand pointers

The model was synthesised with multiple visual representations of each participants cone of vision, only visible to present collaborators. Participants could swap which visualisation was in use at any time; some of which adjusted based on the users head movements, showing where it predicts their gaze to be as the users moved. The cones were designed in a way that would minimise obstruction to the other users view.

Fraser et al.[49] and Piumsomboon et al.[53] visualised the cone of vision by explicitly outlining the view frustum using a wire frame model, affirming the view

of each collaborator. However, Duval and Nguyen[54] suggest that a problem that may arise when using wire frame view frustums; when there are many collaborators working in the same narrow space, the wire framed representations can clutter the environment leading to confusion. To prevent this, Piumsomboon et al.[53] represented the cones using coloured shading to lightly outline where each collaborator is looking, this approach was clear and did not obstruct the users view. This method was used for frustum visualisation in the gaze prediction model. However, most of the developed frustums only extend a short vicinity in front of the user, not through the entire environment, this prevents clutter in the scene and keeps the gaze prediction limited to the accuracy threshold of the forward value predicted by the model. A zoom option, similar to the work of Wong and Gutwin [48], to extend the cone for long distance viewing could be added in future work, but was not relevant for the model at this stage of user validation. A frustum toggle button could also be implemented that allows the user to see their collaborators view for a short duration, affirming their gaze target without having their view cluttered throughout the collaboration. However, this requirement depended on the results of the user testing, which suggested that the users preferred having a view visualisation present.

Chapter 3

Eye-in-Head Gaze Shift Analysis

After processing the data and characterising head shift groupings based upon modern gaze research, the processed data set was analysed to validate the suggested hypothesis: a users high frequency cone of potential gaze targets may be minimised during and after head shifts of a qualifying velocity and direction.

3.1 Data Analysis

The data set from Agtzidis et al.[4] was processed as detailed in Section 2.4. The head shifts were first characterised using the local minima of the head velocity profile. However, when using matlab's 'islocalmin' function [55] the shifts become highly fragmented as shown in Figure 3.1, creating shifts with a maximum amplitude of 20°. This led to each shift being split in to multiple component parts that could not be used to characterise the overall shift. To revise the characterisations, two methods were presented and considered: *threshold-based classification* and *Gaussian minima-based classification*.

In threshold-based classification each shift was characterised by finding when the head velocity rose above a threshold velocity of 3°/s as the start of the shift until the velocity went back below 3°/s which is considered to be the 'stationary' velocity [28]. This classification style improved the number of shifts characterised but the shifts became excessively large and encompassed multiple direction changes per shift whilst the head remained only slight above 3°/s.

Gaussian minima-based classification once again used the head velocity profile, but after a Gaussian smoothing filter was applied to remove some of the smaller peaks and reduce the amount of local minima present. The 'islocalmin' was then used to find the start and end of each shift at the minima, where the head transitions from deceleration to acceleration, as shown in Figure 3.2. The shifts characterised were concise and accurate, which led to the minima based method being used during the data analysis portion of the project. However, the Gaussian minima-based classification can only be used retrospectively, when all head movement data has been collected and a head velocity profile can be built. This prevented it's use in training

the gaze prediction model, as the model has to be provided velocity data and make predictions in real-time. For training the machine learning model threshold-based classification was used. This allowed for head shifts to be characterised in real-time by monitoring if the head velocity rose above the threshold value and storing the data entry at which it did until the shift concluded.

After the head shifts were defined they were grouped using their amplitude and peak velocity. Amplitude and peak velocity groups were as described in Section 2.4. The head-gaze target distance will henceforth be referred to as 'error'. The error was calculated for each group by finding the distance between the gaze target and head target in degrees using the method described in Figure 2.6. The calculation was performed to investigate whether during a head shift, of a distinct velocity or amplitude, the gaze target remains in close proximity to the head target. First, the error values throughout each shift were averaged, then the results were combined to find an error average for each group. Unfortunately, no group had an outlying error value and all results were consistent with the findings of Sitzmann et al.[3], which suggests that the eye gaze is frequently within 20° of the head. After which, the error value at the end of each shift (when the head is stationary) was taken and the values averaged for each group. The ultimate shift entry was examined to assess if eye-shift mechanics, such as VOR [16], increase the error value during a shift, by keeping the eyes still whilst the head moves. This would also determine if the eyes and head align on the gaze shift target at the end of specified shifts.

This analysis was performed to validate the hypothesis that certain shift groupings produce a mean error result lower than 10°, or a significant error discrepancy in one of the groups at the end of a shift. Once again, the hypothesis remained unvalidated, no group had an outlying error value and the results were consistent with the findings of Sitzmann et al.[3]. However, the results do exhibit an accuracy increase of 1° when using only the error values at the end of each shift, which suggests that biological eye mechanisms such as VOR may marginally affect the distance between head target and gaze target during a shift, as hypothesised. Below are the results of these calculations, which show there is no significant difference in eye/head gaze target distance depending on amplitude of shift or peak velocity of shift.

Group	Amplitude Mean Error(°) All Shift Values	Peak Velocity Mean Error(°) All Shift Values
1	11.1513	10.1539
2	13.4612	13.0736
3	13.6284	14.2640
4	13.6162	12.8045
5	12.7421	N/A
6	13.5690	N/A

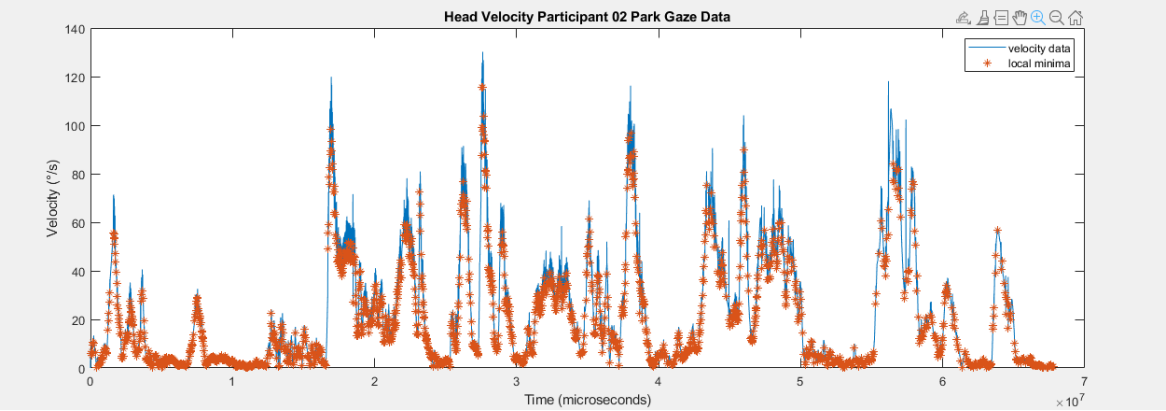


Figure 3.1: local minima of head velocity profile

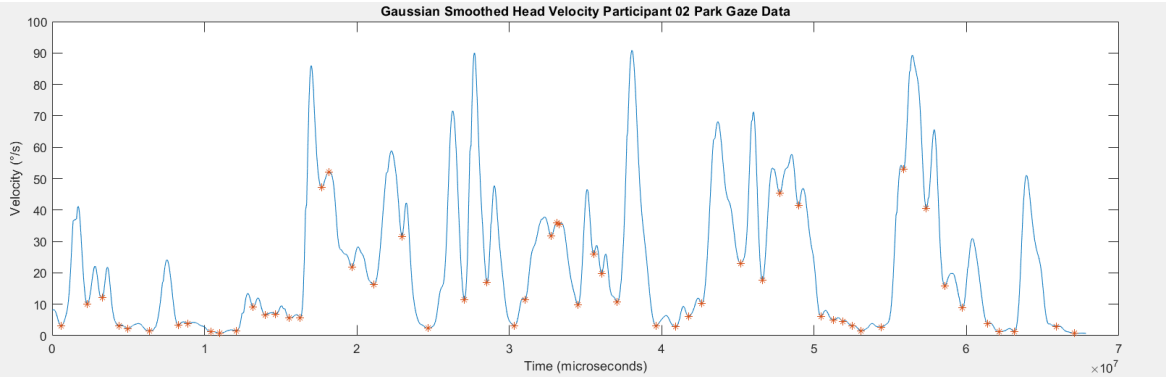


Figure 3.2: local minima of head velocity profile after smoothing

Group	Amplitude Mean Error(°) End of Shift Values	Peak Velocity Mean Error(°) End of Shift Values
1	11.0501	10.1096
2	12.7130	12.4865
3	12.7544	13.0506
4	12.2976	11.3869
5	10.5866	N/A
6	12.3729	N/A

The next hypothesis considered was whether the direction of a head shift impacts gaze location. This was investigated by splitting the data set in to two separate data sets of shifts right and shifts left, by localising the start of each shift to the positive world y axis and localising all other shift entries relative to the initial entry, as described in Section 2.4. If the intermediate and final x values of the head were positive the shift propagated to the right, and if negative the shift moved left. After splitting in to left and right shifts the rate at which the eye gaze ended to the right or left of the head gaze at the end of a shift was calculated. Looking to validate the hypothesis that there was a significant bias for the direction the head was moving. For example, if the head shift was moving from left to right the eyes would end on the right side of the head target. Such a result would allow the possible cone of

vision to be reduced after the user has completed a head shift.

The results indicate that a head shift right ended with the eye gaze to the right of the head for 57.8% of shifts and to the left for the other 42.2% of shifts, which validates the hypothesis. With the same being shown for head shifts to the left the eyes were to the left of the head gaze for 57.7% of shifts and to the right for the other 42.3% of shifts. These results show there is a slight bias for the eyes being to the same side of the head as the direction of the shift, this balances out when checking the results for all shifts: 50.0780% eyes left, 49.9220% eyes right. To find if the directional bias is more prevalent in certain types of shifts the right and left data sets were split in to amplitude and peak velocity groups, in an attempt to localise the type of shift that results in the eyes favouring each side of the head, below are the results:

Peak Velocity Group	Shifts Left		Shifts Right	
	Eyes Left (%)	Eyes Right (%)	Eyes Left (%)	Eyes Right (%)
1	52.2333	47.7667	48.2643	51.7357
2	61.4345	38.5655	38.4150	61.5850
3	65.5738	34.4262	34.0361	65.9639
4	57.1429	42.8571	37.4648	62.5352

Amplitude Group	Shifts Left		Shifts Right	
	Eyes Left (%)	Eyes Right (%)	Eyes Left (%)	Eyes Right (%)
1	55.1328	44.8672	45.1451	54.8549
2	63.2520	36.7480	37.3096	62.6904
3	65.3916	34.6084	32.6489	67.3511
4	55.7447	44.2553	40.7407	59.2593
5	49.4845	50.5155	46.7391	53.2609
6	52.8736	47.1264	42.1053	57.8947

As seen in the results above the gaze directional bias appears consistently among all groups. Groups 1 and 6 of the amplitude groups contain the very small and very large shifts which are not indicative of the standard saccadic head shift, removing these groups creates the following averages for left shifts: 58.4682% eyes left, 41.5318% eyes right and for right shifts: 39.3596% eyes left, 60.6404% eyes right. The directional bias appears more frequently within peak velocity and amplitude groups 2 & 3, this encompasses the mid-range distance and speed shifts. To ensure the results apply universally the stimuli were examined to find if they were particularly one-sided, with salient regions often being to the right or left, making the bias stimulus dependent as opposed to a biological reflex. This check involved watching the videos and checking the percentage of eyes right and left for each stimulus separately, which produced the following results:

Stimulus Video	Shifts Left		Shifts Right		All Shifts	
	Left(%)	Right(%)	Left(%)	Right(%)	Left(%)	Right(%)
Park	55.5	44.5	43.1	56.9	49.7	50.3
Festival	62.0	38.0	49.2	50.8	55.6	44.4
Drone	60.5	39.5	41.2	58.8	50.9	49.1
Turtle Rescue	47.8	52.2	39.7	60.3	43.5	56.5
Cycling	56.4	43.6	41.6	58.4	49.2	50.8
Forest	61.3	38.7	39.8	60.2	50.5	49.5
Football	61.1	38.9	43.3	56.7	52.0	48.0
Courtyard	57.3	42.7	41.4	58.6	49.3	50.7
Expo	52.0	48.0	38.5	61.5	45.3	54.7
Eiffel Tower	58.8	41.2	38.9	61.1	48.9	51.1
Chicago	64.6	35.4	43.5	56.5	54.3	45.7
Driving	65.3	34.8	45.1	54.9	55.3	44.7
Drone Low	61.8	38.2	39.6	60.4	50.8	49.2
Cats	56.8	43.2	38.5	61.5	47.6	52.4
Synthetic	44.7	55.3	52.1	47.9	48.1	51.9

Once again the eyes were predominantly on the right side of the head gaze for shifts right and left for shifts left throughout the entire stimuli catalogue. Interestingly, the 'Synthetic Video' [4] was one of the only stimuli which did not display the directional bias, the synthetic stimulus was generated to test specific types of gaze shifts at different amplitudes in a controlled unnatural manner. Whereas, all other stimuli in which the participant was allowed to explore freely, displayed a directional gaze favouring as shown in Figure 3.3. This suggests that when a participant is exploring a naturalistic 360-degree video in an unrestrained manner they will often favour the matching side of their cone of vision when completing gaze shifts.

When dividing the analysis in to shift groupings, many results show a slight increase in right side preference; a possible influence may be due to each participant having an eye preference. Monocular preference can be detected in about 90% of the population [56], with 70% having a right eye preference. Leeuwen et al.[57] found that their subjects with the strongest monocular preference always fixated nearby targets with their preferred eye and the preferred eye was consistently faster at performing accurate vergence shifts than the disfavoured eye. In a real-world scenario vergence eye movements are paired with changes in accommodation to focus the eyes depending on depth of the gaze target. However, in VR, the focal distance is constant and the eyes converge without changing accommodation to maintain a clear retinal image [58]. Consequently, gaze targets in VR are always nearby targets, which may incur a frequent use of the preferred eye to find a fixation on the target. Unfortunately, it is not stated which eye is dominant for each participant in the data set [4], preventing the evaluation of the impact of ocular dominance. Monocular preference in VR gaze shifts could be explored in future work and may help to explain the directional bias found in the analysis results.

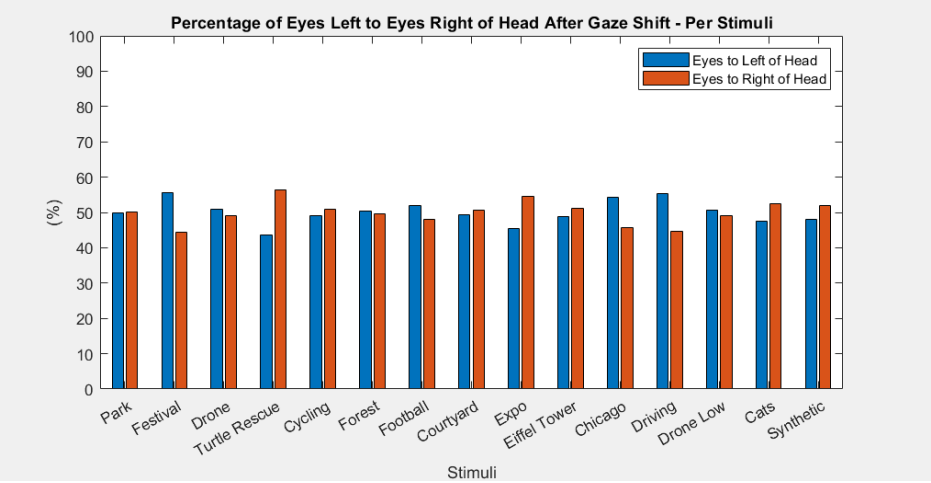


Figure 3.3: Local minima of head velocity profile.

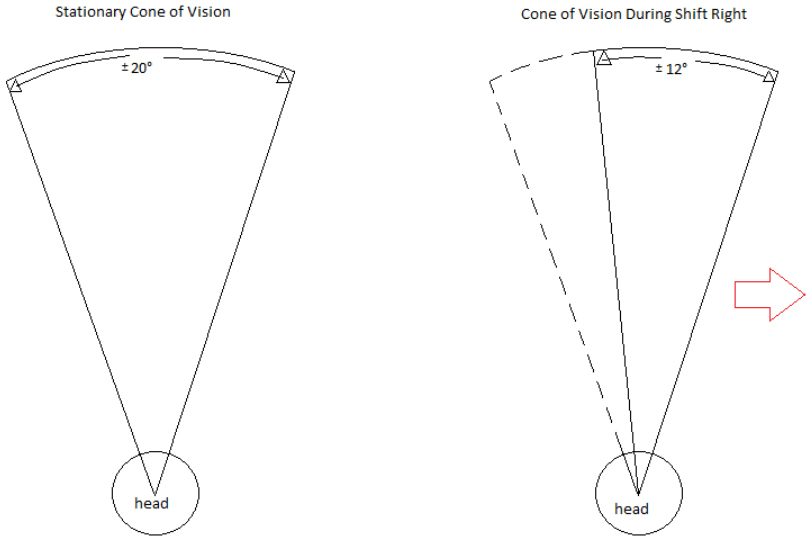


Figure 3.4: High Frequency View Frustum Shift Reduction.

The previous results were all obtained by checking the gaze position for the last entry of each shift, allowing the visible view frustum to be reduced at the end of a shift. However, when checking the gaze location compared to head location for the entire shift the results are consistent, and the bias displayed throughout the head movement:

	Entire Shift		End of Shift	
	Eyes Left (%)	Eyes Right (%)	Eyes Left (%)	Eyes Right (%)
Shifts Left	61.0309	38.9691	57.6995	42.3005
Shifts Right	39.1021	60.8979	42.2312	57.7688
All Shifts	50.1873	49.8127	50.0780	49.9220

The bias is shown to be slightly more prominent when checking the entire shift, allowing the high frequency view frustum to be reduced throughout a head shift, not

only at the end. This reduction can be implemented using the above averages to reduce the cone appropriately for each movement. The previous research by Sitzmann et al.[3] suggests a 40% reduction of the high frequency cone in the direction of the shift, shown in Figure 3.4.

The final part of the data analysis was for checking how much time the view frustum should remain minimised after a head shift. To achieve this a MATLAB script was written to find the average time that the participants gaze stays within the minimised 12° zone after the end of a head shift. The algorithm checks the next 200 entries after each shift ends and finds the time that that the gaze goes beyond the minimised 12° region. If the gaze stays within the region, the time 200 entries later is added to the accumulation. The total time was then averaged producing result of: 0.7609 seconds for the threshold-based classification shifts and 0.6356 seconds for the Gaussian minima-based classification, before the cone needs to revert back to its stationary size. The high frequency visualised view frustum should return to a size of 40° as shown in Figure 2.4.

Chapter 4

Model Training

This chapter delineates the steps taken to train a machine learning model upon the processed data set, building a gaze prediction model that may predict a collaborators gaze to a higher degree of accuracy than the modern high frequency 40° cone of vision proposed by current gaze research.

4.1 Model Selection

After the data was processed the gaze prediction model could be trained to predict the eye gaze of a user using only the head gaze data available from modern, widely used HMD's without the use of eye tracking technology installed, within an error value less than $\pm 20°$. This was achieved using MATLAB's 'Regression Learner App' [5]. The 'Regression Learner' was chosen for this data set as the model will predict a continuous response, not discrete. Consequently making regression more appropriate for training on the data than classification. The learner performed supervised learning on the processed data using 8 of the possible 21 features:

'Time', 'Head World X', 'Head World Y', 'Head World Z', 'Eye World X', 'Eye World Y', 'Eye World Z', 'Eye Local X', 'Eye Local Y', 'Eye Local Z', 'Head Roll Angle', 'Head Velocity', 'Head Amplitude', 'Eye Amplitude', 'Peak Velocity Group', 'Cumulative Shift Amplitude', 'Head Gaze Distance', 'Shift Average Head Gaze Distance', 'Local Shift Coord X', 'Local Shift Coord Y', and 'Local Shift Coord Z'

The 8 features in use for training were:'Time', 'Head World X', 'Head World Y', 'Head World Z', 'Head Roll Angle', 'Head Velocity', 'Head Amplitude' and 'Shift Amplitude'. These features were all calculated without using any eye tracking data, which will not be present when using a traditional HMD. These features may also be calculated at run-time not retrospectively, allowing for real-time predictions. Unfortunately, the Regression Learner does not allow for multiple response regression to predict all three variables simultaneously. Consequently, a separate model had to be trained individually for each positional eye x, y and z coordinate value.

The processed data set was split in to training and testing data sets by leaving out

25% of the participants, this resulted in the models being trained on data from participants 02 - 11 and tested on data from participants 12 - 14. The sets of recorded participants in training and test sets did not intersect. During the training process 'Hold-Out Validation' was used with 25% of the data withheld for model validation, this validation method was chosen over cross-validation due to the magnitude of the data set. Hold-out validation simply tests against the unseen withheld data, whereas in cross-validation, you make a fixed number of partitions of the data, run the analysis on each partition, and then average the overall error estimate. The increased workload of cross-validation would drastically increase model computation time. To find the best model type for the data, a small subset consisting of participant 02's data was used to train all possible models [59] offered by the 'Regression Learner' [5]. The models that returned the lowest root mean squared error (RMSE) were then used upon the larger training data set. RMSE is the standard deviation of the prediction errors, it suggests how concentrated the data is around the line of best fit, the lower the RMSE the better the predictive performance of the model. The following models produced the three lowest RMSE: Fine Tree, Bagged Trees and Exponential Gaussian Process Regression. The chosen models produced the following RMSE when trained upon the full training data set:

	Fine Tree		Bagged Trees		Exponential GPR	
	Train	Test	Train	Test	Train	Test
Eye	RMSE (°)	RMSE (°)	RMSE (°)	RMSE (°)	RMSE (°)	RMSE (°)
X	3.6135	9.5962	2.9830	7.2522	N/C	N/C
Y	4.1134	11.4628	3.5337	8.5209	N/C	N/C
Z	2.7663	11.0621	2.3261	8.2911	N/C	N/C

The results produced are $\pm$ from the predicted gaze location, creating a cube around the possible gaze location in world space. As displayed above, even when the results are doubled to account for each direction, the possible cone of vision is reduced to below 40° for the 'Bagged Trees' and 'Fine Tree' models. Unfortunately the 'Exponential GPR' model was unable to complete training due to insufficient processing power on the training device. However, the data set could be too large for Exponential Gaussian Progress Regression, leading to over-fitting [60, 61] and model inaccuracies.

It was then hypothesised that training the model on world coordinates may cause the model to learn and predict based on salient regions of each environment, instead of training only based on biological head movements. This may cause a distortion in the predictions to match the environments from the data set. To prevent an inclination to salient regions, new models were trained using the localised shift coordinates that all originate from the positive y axis as described in Section 2.4, instead of world coordinates. Below are the results of models trained on local coordinates:

	Fine Tree		Bagged Trees	
	Train	Test	Train	Test
Eye	RMSE (°)	RMSE (°)	RMSE (°)	RMSE (°)
X	4.2589	13.8512	4.7819	10.4647
Y	1.5747	4.4691	1.6883	3.7431
Z	3.6042	11.8060	3.9620	9.2950

Whilst the RMSE values are worse than the previous models the result is below the 20° error threshold, and the models should be more reliable in a novel environment. This was the desired outcome for the model, the next stage was to implement the models to perform predictions in real-time as the user is moving. The 'Bagged Trees' models displayed the best performance in both testing and training, leading to their implementation in Unity. The implementation is described in Section 5.1. Unfortunately, due to the complexity of the models and delay created by the TCP connection, predictions were only able to be made every three seconds. This created an unusable lag, impeding frustum movement and gaze prediction.

The models were also unable to be transferred into Unity as they were generated in MATLAB written in C++, however Unity uses C# preventing their adoption in to the Unity environment. An attempt was made to convert the models to C# using community made libraries [62], however the complexity of the models prevented any functioning conversion. The models could be used on a machine with greater processing power to run both Unity and MATLAB simultaneously, or potentially using a dedicated MATLAB server to generate the predictions, however building such a server was outside the remit of this project.

To fix this issue new models were required written in C#. The ML.Net 'Model Builder' [63, 64] was used to train all available model types on the data to find the model with the best RMSE(°) as was done for the MATLAB models. Interestingly for the x value 'FastTreeRegression' was found to have the lowest RMSE value, similar to the Regression Learner results. However, for the y and z values 'LightGbmRegression' produced the most accurate predictions. This may be due to the high variance of the x values, causing an increased training time for the x models. This led to only 133 models completing training during the training time, whereas for y 295 models were trained and for z 314 models were trained. Below are the RMSE values of the ML.Net models:

	RMSE(°)	Model Type
X	9.9654	FastTreeRegression
Y	10.8547	LightGbmRegression
Z	10.5986	LightGbmRegression

The ML.Net models displayed a marginal increase in RMSE, but with an error value below the 20° prediction target; predictive performance is high enough for implementation. Unfortunately, after attempting to port the models in to Unity, it was found that while Unity can target .net 4.x [65], there has been no tested libraries built to support ML.Net models and the only way to use the model was via a TCP

connection. This would once again introduce the same lag as previously mentioned for the MATLAB models. Unity does, however, have a plugin which supports the use of TensorFlow 'an end-to-end open source platform for machine learning' [8].

The next models were built using TensorFlow; these models could then be converted to TFLite [66], targeting Android [67] architecture. TFLite files can be compiled directly on the Oculus Quest HMD using community made Unity Libraries, without the need for a PC connection. The details of this are explained in Section 5.1. The first TensorFlow models generated were 'Decision Forests' [68]; an ensemble of trees similar to the 'BaggedTrees' model, which produced the smallest RMSE from the Regression Learner. This model produced the following test RMSE values:

	TensorFlow Decision Forests RMSE (°)	MATLAB Bagged Trees RMSE (°)
X	4.3205	7.2522
Y	4.8109	8.5209
Z	3.5989	8.2911

As shown the TensorFlow Decision Forests model predicted with a significant increase in predictive performance when compared to the generated MATLAB models. Unfortunately, whilst Decision Forests may produce the best prediction results of all generated models, it is currently not supported by the TFLite library, preventing its use on mobile or IoT devices. Consequently, it cannot be converted to .tflite File and stored on the Oculus Quest. The final model built was a basic regression multivariate deep neural network model (DNN) [69], that could be compiled in Unity and converted to a TFLite model, stored as a .tflite file and run locally on the Oculus Quest HMD. This was the model used for the Unity Implementation discussed in Section 5.1.

A DNN model was chosen due to the common comparison made between deep neural networks and boosted decision trees, both methods can model data that has nonlinear relationships between variables, this nonlinear modelling power allows for both algorithms to have superior predictive performance and predictive stability when compared to alternative algorithms such as logistic regression and support vector machines [70]. Boosted decision trees tend to show a slight predictive accuracy increase over DNN models [71], however DNN models are a close alternative and often require a lesser amount of training data to achieve similar performance [70]. The DNN model used the Keras Sequential architecture [72, 73], which groups a stack of linear and non-linear layers. The model contained three layers, a normalisation layer, two hidden non-linear dense (deeply connected neural network layer) layers and a linear single-output layer. The normalisation layer stabilises the model training, by normalising each feature range to 0-1. This reduces the scale of the outputs when multiplying features by the model weights, leading to faster convergence. The two non-linear dense layers use the rectified linear activation function (ReLU) non-linearity[74]. ReLU is defined as:

$$max(0, x) \tag{4.1}$$

It is the most widely adopted activation function for neural networks as it is 'easy to calculate, simple to achieve and has fast convergence speed' [75]. It can approximate any function $f(x)$ using the following properties:

$$ReLU(x - c) = 0, \text{ for } x \leq c \tag{4.2}$$

Therefore:

$$f(x) + ReLU(x - c) = f(x), \text{ for } x \leq c \tag{4.3}$$

Generalised forumla to approximate $f(x)$ for $0 \leq x \leq n$:

$$f(x) \approx b + \sum_{k=0}^{n} a_k \times ReLU(x - k) \tag{4.4}$$

$ReLU(x - c)$ added to any function $f(x)$ will not affect the outputs of $f(x)$ for $x \leq c$, which allows each additional term to be multiplied by a constant, accommodating the non-linearity of the input whilst leaving the previous term's output unaffected. The addition of many ReLU terms, each multiplied by its own constant, allows the function to get the shape of any curve needed for the multi-layer neural network. The final output layer is where the final prediction result is produced by taking the input from the layers before it and computing the result via it's neurons. The DNN was trained for 100 epochs (number of passes of entire training data set), with a 20% validation set split. The Final models' RMSEs were:

	TensorFlow DNN RMSE (°)	TensorFlow Decision Forests RMSE (°)
X	5.3414	4.3205
Y	7.5987	4.8109
Z	7.6369	3.5989

Whilst the model's predictive performance is lower than that of the 'Decision Forests' models, the RMSE is within the 20° target threshold, allowing for a reduction of the current biological fixed view frustum. The Implementation could be improved using the previously discussed Decision Forests models, by utilising updated TFLite libraries. The model was also trained using world coordinates due to the increased processing time taken to localise each shift entry in real-time, as well as the localised coordinates containing numerous unmapped entries, described in Section 7.1. This may incline the model to predict salient regions from the data set stimuli, which are not present in the testing environment. However, the effect of this potential salient bias cannot be evaluated without user testing with an installed eye tracker, which was not performed for this project. This is discussed further in Chapter 6.1. A comparison of each developed machine learning model is shown in Figure 4.1

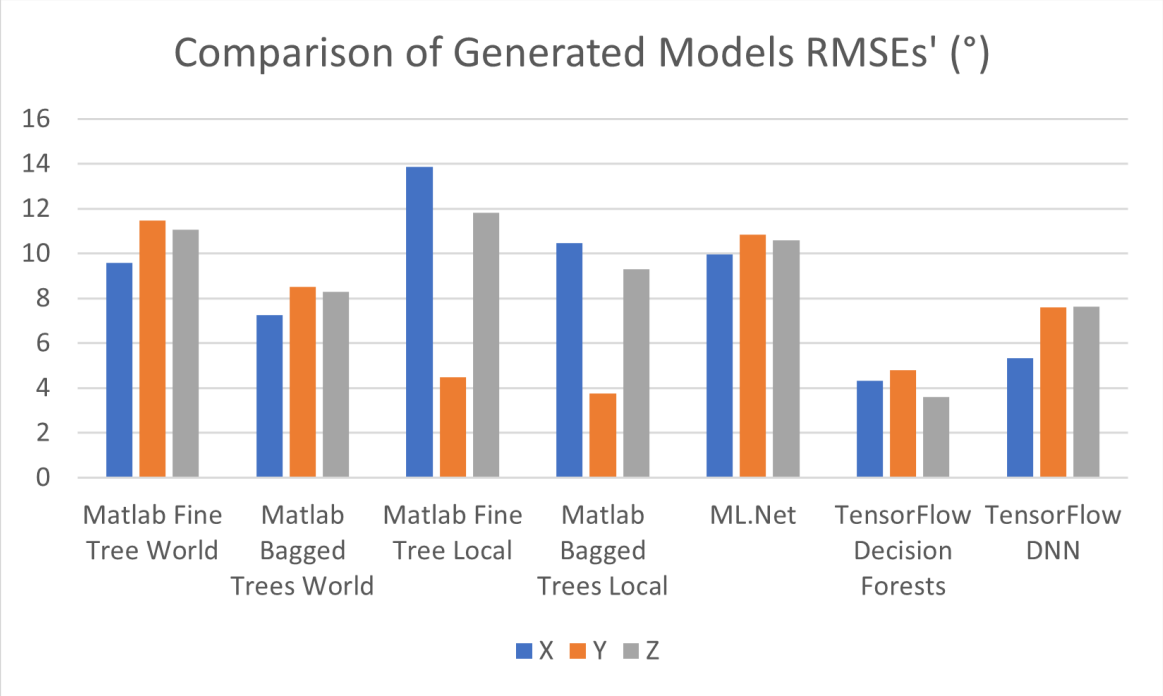


Figure 4.1: RMSE values of all generated models

Chapter 5

Unity Implementation

This chapter describes two approaches to implement the MATLAB and TensorFlow prediction models, built in chapter 4, within the Unity environment. It explains the development and design choices of two suggested testing environments and four approaches for visualising a collaborators view frustum.

5.1 Model Implementation

The model implementation began with using code translation to implement the MATLAB Bagged Trees models directly within the Unity environment. After trialing multiple C++ to C# converters [62, 76, 77, 78], it was found that the models were too complex for a direct conversion. Each converter flagged numerous compiler errors with no clear fix and building a MATLAB to Unity conversion library was outside the remit of this project. The models were instead implemented using a TCP connection between Unity and MATLAB. The server connection is initiated on MATLAB using a while true loop to keep the connection open and waiting for the Unity client to connect. Unity then connects to the MATLAB server using a TCP three-way handshake. Once each party has acknowledged the synchronisation of the sequence numbers the communication commences.

Sequence of events for Unity MATLAB Communication, after connection has been established:

Unity:

- Unity finds the elapsed time in seconds since the application was started.
- Unity accesses the head target vector and rotation vector from the camera rig eye anchor, using forward attribute and rotation attribute respectively.
- Each vector is converted in to a byte array of four byte blocks, and copied to the buffer.
- The buffer is written to the network stream object on the socket connection.

MATLAB:

- MATLAB receives the byte array, converting each byte to 'uint8' [79] 1-byte (8-bit) unsigned integers.
- each block of uint8 bytes is then converted to a single floating point value using a type cast.
- head amplitude and velocity are then calculated using the previously stored values, and velocity threshold checked for if a head shift has begun.
- If the movement is within a shift, the coordinates are localised to the first shift entry using the same method described in Section 2.4.
- The predictions are made using each Bagged Trees model, one for each coordinate.
- The predicted coordinates are placed in a string array and sent back to Unity.

Unity:

- Unity receives the byte array and reads each coordinate into a float array using a type cast, splitting the coordinates by recognising the spaces between the strings.
- The returned float values are then combined with the head position and set to the positional value of the visualised view frustum.

Whilst this implementation does function and provide accurate predictions, there was a significant delay caused by the server, shift localisation procedure and model prediction time. Consequently, MATLAB was only able to send a prediction back to Unity every 3.5 seconds. This created an unusable lag between the users head movements and the updated movement of the view frustum. When the frustum was updated the user had already shifted their gaze to a new position making the accuracy of the predictions irrelevant.

The use of predefined functions for data processing, such as the 'Rodrigues' function, created a disparity in the amount of server-side calculations that had to be performed in MATLAB. If the localisation and shift checking could have been performed client-side this may have reduced the delay time, making the prediction delay more manageable. Additionally if the MATLAB processing could have been outsourced to a dedicated server machine, this would have allowed for quicker localisation and model predictions, unfortunately the creation of such a server was outside the bounds of the project proposal and would have reduced frustum implementation and testing time. This model employment could provide accurate gaze prediction when using suitable hardware. However, using the resources available for this project the MATLAB models were unable to be efficiently implemented.

Whilst the TensorFlow DNN models were not as accurate as the Decision Forest

models, they were below the error threshold for all three values, and suitable for implementation in the testing environment. Unity Technologies recently developed the ML-Agents plugin [80], which allows for TensorFlow and Unity to work in conjunction. The plugin includes 'TensorFlowSharp', which allows previously trained Keras models [72] to be implemented within the Unity development environment, but not TFLite models. This enables the DNN Keras sequential models to be used in Unity when building and compiling, which could be potentially used on the now discontinued Oculus Rift[81]. However, standalone HMD devices such as the Oculus Quest (the deployment target of this project), are unable to compile a full Keras (.h5) model.

At its core the Oculus Quest is an Android Device, with limited resources to run and load files, which is why a conversion to a compact model with a smaller code footprint was required. As previously mentioned in Section 5.1, this was done using TensorFlow Lite [66] to convert the model in to an efficient, portable 'compressed flat buffer' [66, 82] format. Flat binary buffers allow data to be accessed directly without parsing or unpacking, whilst maintaining hierarchical data structure with a minimised code footprint. The format also improves memory efficiency, the only memory required to access the data is the buffer with no additional allocations. This enables 'TensorFlow Lite to execute efficiently on devices with limited compute and memory resources' [83], such as the Oculus Quest. The models can simply be converted using the 'TFLiteConverter' function from the 'lite' library.

The .tflite model files cannot be included within the Unity application Android Package File (APK), they must be separately stored in the Quest's persistent data directory, which can be accessed by an APK. To access the Quests persistent storage the device must have developer mode enabled and a third-party file access application must be used such as 'SideQuest' to sideload the required .tflite files. This process is described in the user guide in the Appendix. The final stage to implement the TensorFlow Lite models was installing an experimental TFLite Unity plugin 'based on the experimental TF Lite C API' [84], that facilitates model compilation in Unity through the use of a C# Interpreter wrapper.

The combined use of TFLite conversion and plugins, enabled a working implementation of the TensorFlow Lite models to be installed and used by the standalone Oculus Quest, without connection to a PC or TCP server. Owing to the increased efficiency of the TFLite models, the predictions can be made in real-time as the Unity head positional coordinates are simultaneously updated. Allowing for smooth, accurate frustum movement whilst providing gaze predictions within the 20° error threshold.

5.2 Multiplayer Frustum Visualisation

As stated in Section 2.6, each collaborators view frustum would be visualised following the recommendations of Piumsomboon et al. [53], who created a lightly shaded square-based pyramid frustum that extended away from each user, as shown

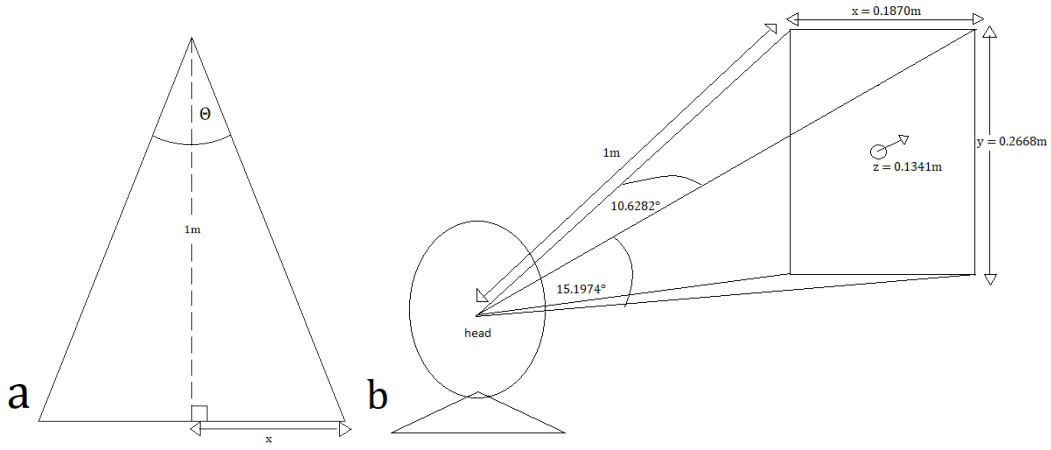


Figure 5.1: **a**: Frustum length calculation diagram, **b**: DNN RMSE frustum measurement value diagram

in Figure 5.2. This implementation provided an unobtrusive visualisation of a collaborators view, whilst also allowing for accurate frustum boundaries, guided by the RMSE of the implemented prediction model, as shown in Figure 5.1.

During the frustum implementation it was determined that Unity does not include a square-based pyramid primitive. A dedicated script was written 'MulPlayerFrustum' to create a pyramid shape emanating from each collaborators avatar to display their field of view. This was done by creating a simple mesh that starts by finding the distance from predicted point to the users head, the frustum base dimensions are then calculated using this value. The base length ratios were calculated using simple trigonometry, the setup for each frustum side is shown in Figure 5.1, each x value could then be calculated and doubled to find the full length of the frustum base. Below are the calculations for the x, y and z RMSE frustum length values:

X:

$$\begin{aligned} \tan(5.3414) &= \tfrac{x}{1} \\ 2 \times x &= 0.1870m \end{aligned} \tag{5.1}$$

Y:

$$\begin{aligned} \tan(7.5987) &= \tfrac{y}{1} \\ 2 \times y &= 0.2668m \end{aligned} \tag{5.2}$$

Z:

$$\begin{aligned} \tan(7.6369) &= \tfrac{z}{1} \\ z &= 0.1341m \end{aligned} \tag{5.3}$$

The values were calculated in metres due to one unit in Unity corresponding to one metre. The z error value was also calculated and added on the end of the frustum to ensure each error range is within the possible gaze location frustum. The z error value was also not doubled due to the other half already being within the frustum, starting at the prediction point and propagating backwards towards the users avatar. The values calculated were implemented as multipliers of the distance between the predicted gaze location and the users head, keeping the error accurate for the distance of the point predicted relative to the user.

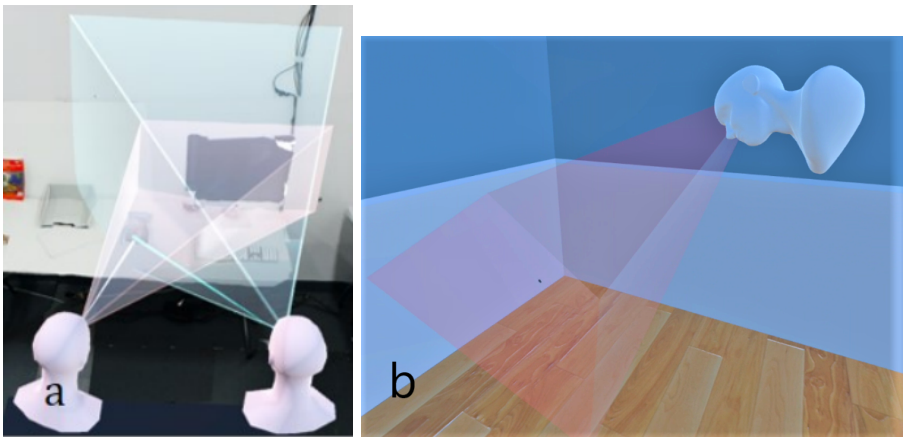


Figure 5.2: **a**: 'Virtual awareness cues, the FoV and Gaze cues, Changing views in CoVAR.' Piumsomboon et al. [53], **b**: Dynamic high frequency frustum implementation

The initial implementation displayed all predicted positions fed to Unity by the DNN models, with the reduced frustum. However, this made the frustum rapidly jump between predicted targets, creating erratic, disruptive frustum visualisation. The flickering created visible disturbance to the collaborators view, hindering their collaborative experience. To fix this, when the head is stationary (below 3°/s [28, p. 2]) the frustum reverts back to the default high frequency frustum size of 40°, fixed on the head forward position, that is guided by current gaze research [10, 11, 12]. This improved user comfort but reduced stationary prediction accuracy, as the cone remains at the original high frequency 40° when the user is still. This first 'Dynamic High Frequency Frustum' implementation can be seen in Figure 5.2.

To maintain high prediction accuracy, whilst minimising user view disruption, an alternative cone implementation was considered: A large 50° frustum that remained fixed on the users head forward vector, with a smaller frustum within, which represented the gaze prediction target. The large frustum is based on the high frequency range of the eyes with a fixed head position [10], shown if Figure 2.1. The smaller frustum is of the same dimensions as the predictions models' RMSEs, creating a reduced base surface area; minimising visual disruption when shifting between rapid target predictions. The larger cone, whilst remaining fixed in the forward position, dynamically adjusts to represent the 40% cone reduction during a shift right or left that was detected in Section 3.1. When the head turns at over 20°/s either right or left the frustum reduces on the opposing side, representing the 40% reduction of the high frequency 40° cone shown in Figure 3.4.

The initial implementation reduced the cone size whenever the head rose above a stationary velocity of 3°/s, this happened frequently causing constant frustum size adjustments. These quick size reductions created flickering and disruption to the collaborator. To prevent this the second implementation only reduces the cone when the users head moves above 20°/s. The threshold was determined by examining in which peak velocity groups the directional bias was most frequent; as previously

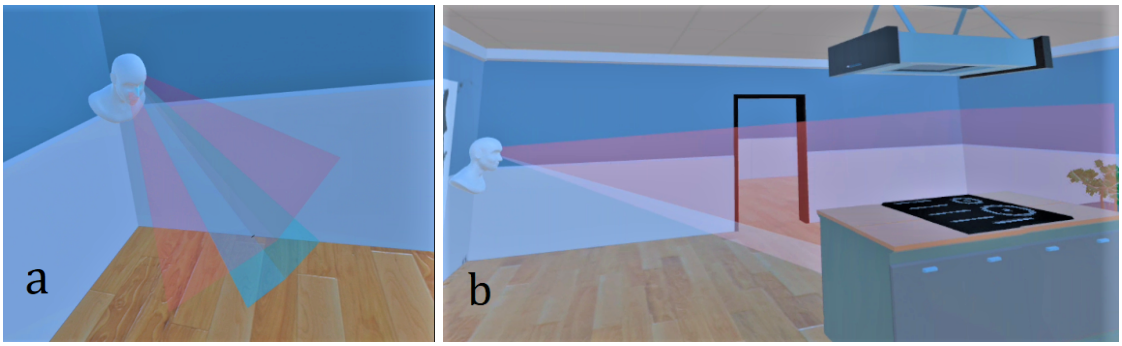


Figure 5.3: **a**: Dual high precision frustum implementation, **b**: Long distance view frustum

mentioned the bias was prevalent in groups 2 and 3 found in Section 3.1. Group 2 had a lower threshold of 20°/s, leading to this value being used for the frustum reduction threshold. The second 'Dual High Precision Frustum' implementation can be seen in Figure 5.3.

The Third 'Long Distance View Frustum', shown in Figure 5.3, was simply an expansion of the first; the frustum was extended in the negative z direction to encompass a greater area of the testing environment. This allowed for users to have their medium to long gaze targets predicted and covered by the view frustum. The model generated gaze prediction is frequently one unit away from the user, yet the training data set [4] was built using equirectangular naturalistic video clips that contained both short and long distance gaze targets. This suggests that the trained model should be able to predict all types of gaze targets, which is why the long distance cone was implemented as a testing option. However, due to the larger error range created by the TensorFlow prediction model, the range of possible gaze targets increases exponentially with distance, reducing accuracy and increasing environmental clutter. To minimise the disruption caused by an increased frustum size, the long distance cone is a fixed size based on the models RMSE values, it does not change dynamically based on a threshold value.

The final frustum implemented was used as a control for user testing; a large 100° frustum that covered the full 50° physical range of the eyes relative to the head. This frustum is fixed on the head forward position and doesn't change with predicted gaze. Current predictions are shown through the use of a small cross-hair for users to reference during testing; a more subtle depiction of a users predicted gaze. This implementation was used in the testing to find if users preferred the minimised cone implementations derived from the prediction model and data analysis, or whether they would rather have all possible gaze targets covered by the frustum and find referents using deictic referencing. The 'Default Large 100° Frustum' visualisation can be seen in Figure 5.4

The next stage in the development process was the addition of networking features; facilitating multiplayer collaboration whilst visualising each collaborator's view frus-

Figure 5.4: Default large 100° frustum

tum. This was made possible using the Photon Pun 2 software development kit (SDK) [85], which enables up to 20 concurrently connected users to connect to 'rooms' hosted on dedicated Photon servers. A network player object is assigned to each player who joins a room, which is where each frustum mesh script component is stored. the network player visualises the users head, Oculus touch controllers and chosen view frustum. Whilst the mesh of each users frustum is calculated and visualised on the collaborators device client-side, each users predicted gaze location is calculated on their own device using their HMD IMU data and local TFLite models. The predicted gaze location is then assigned to the world position of the 'Sphere' object. The network player script finds the position of the sphere object and uses this predicted position to build each collaborators view frustum over the Photon network. The prediction sequence is shown in Figure 5.5.

5.3 Testing Environment Development

The design of the proposed testing environment was based upon the previous testing environments of past research as stated in Section 2.6. One of the predominant design influences was the 'Furniture World' studies carried out using the MASSIVE environment [49, 50, 51]. The research studied the effect of technological limitations on collaborative interaction within a CVE. Each user was placed in a room and asked to collaboratively organise the furniture in the room, until a final layout was agreed upon by all participants. This type of design task encouraged users to 'discuss and discriminate features of the virtual world.' [51, p. 6 482], which required a mutual understanding of a collaborators gaze target. A design task, which is based upon individual preference creates a unique approach from each participant group, this type of task would work well for testing the prediction gaze model as each user will have unique environmental preferences and gaze target locations. However, an interactive task asking the user to rearrange objects within the environment may

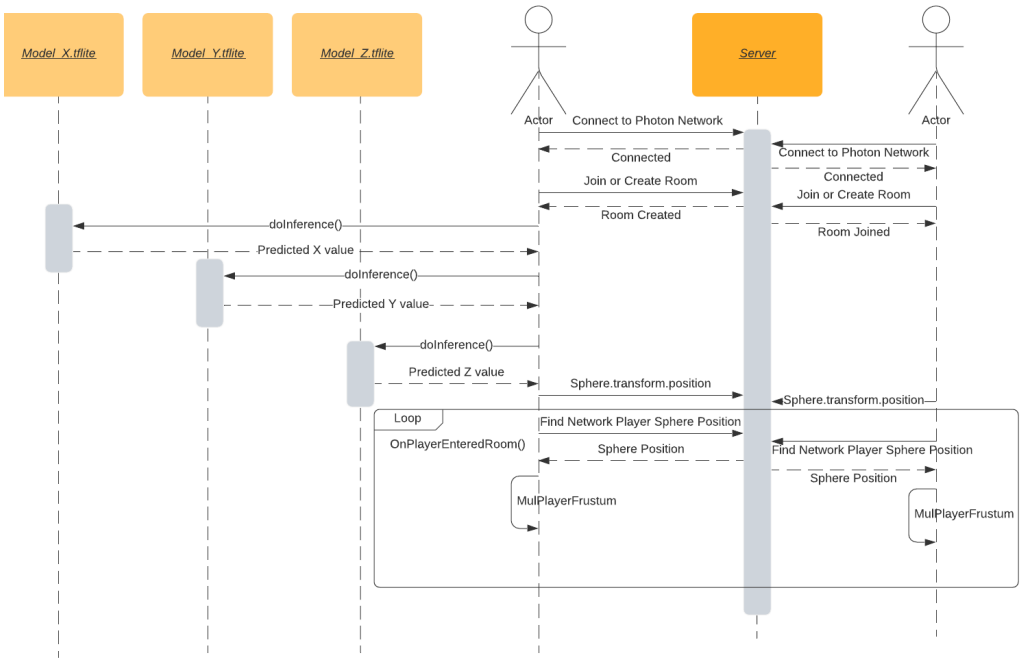


Figure 5.5: Frustum UML sequence diagram

influence their head movements whilst interacting. The model in this paper was trained upon data gathered from non-interactive naturalistic video clips [4], which may affect its accuracy within an interactive environment.

When building the CVE aspects of each study mentioned in Section 2.6 and the Furniture World studies were combined to create an example apartment environment shown in Figure 5.6. The study would involve performing a similar task to the work of Wong and Gutwin [48], where users are asked to find a suitable apartment to live in. However, instead of simply focusing on location and a birds eye view of the city, the users are placed within each apartment and asked to discuss the pros and cons of each interior. This approach prioritises short-distance gaze referents which the

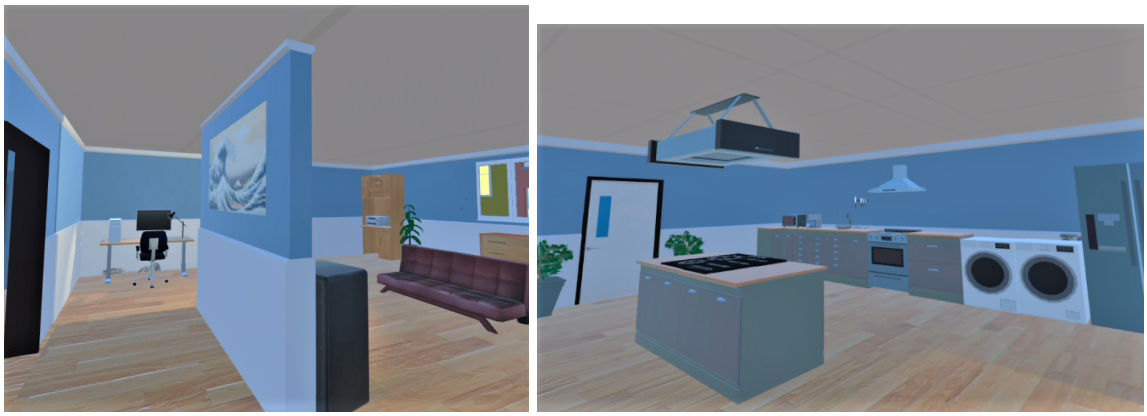


Figure 5.6: Developed example apartment environment

Figure 5.7: Small gaze referents built in to apartment environment

prediction model and frustum visualisations are intended to predict. The apartment environment was built to emulate the work of Sitzmann et al [3], who used internal environments with differing entropy of salient regions to find how saliency affects a viewers exploration of their environment. The results indicated 'that the viewer explores the scene faster in cases of low entropy, quickly discarding non-salient regions, and that her attention gets directed towards the few salient regions faster' [3, p. 4]. This suggests that if an environment has a high entropy of salient regions the user will spend longer examining the environment and be less drawn to individual salient regions. Consequently each apartment built for testing should contain many small, unique details for the participants to find, such as mould, art and property damage within the interior as shown in Figure 5.7. The inclusion of small-scale gaze referents should encourage the use of the smaller gaze frustum to localise a collaborators gaze target, as most of the larger referents will only require the larger 50° frustum to acquire a mutual understanding of referent. The inclusion of unusual details may also prevent the use of verbal referencing; if a user encounters an unexpected or unusual object they may struggle to describe it verbally and rely on the use of the gaze prediction cone to locate their intended gaze target.

The small details were all implemented to encourage the use of the view frustum over verbal communication. However, the familiarity of the apartment environment may prevent participants requiring a gaze prediction alternative to verbal referencing as explained in Section 2.6. This is why a second test environment of a synthetic nature was created, displaying statistical data about the apartment's local area, aiding the participants in deciding if they would like to move in to the neighbourhood in question, shown in Figure 5.8. The data was visualised in a similar fashion to the work of Fang et al. [38] who displayed their synthetic stimuli on a panoramic view, distributed between six screens, surrounding the user. Whilst environments of a synthetic nature were originally avoided due to the naturalistic nature of the training data stimuli, the data visualisation environment provides a more challenging envi-

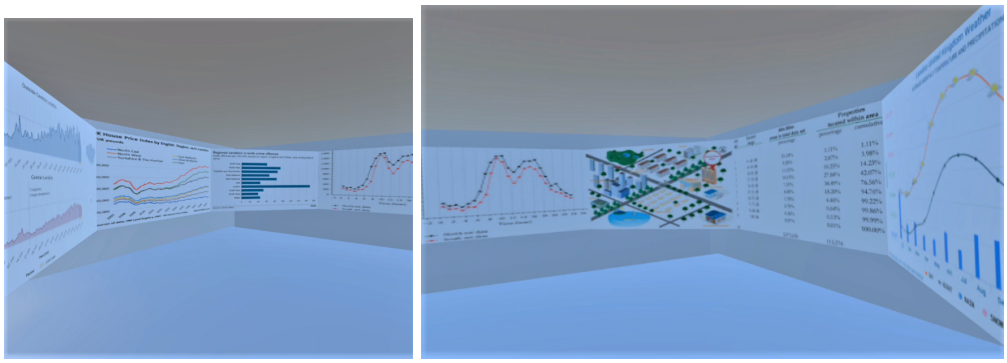


Figure 5.8: Apartment data visualisation environment

ronment for achieving a mutual understanding of gaze awareness, with less obvious verbal referents and more precise gaze locations. One of the original goals of the gaze prediction model was to help businesses perform collaborative remote meetings in VR. Data visualisation analysis is a common collaborative task conducted in commercial settings; a data visualisation testing environment creates an appropriate assessment of the models commercial applicability.

The testing environments were developed in Unity [6] using free community built assets from the Unity Asset Store [86]. When testing the model four developed apartments should be implemented, of varying size, interior and location. This creates a variety of choice for participants, but also prevents VR fatigue as each apartment has a both an interior and data room, more than four apartments may overload the participant with information and too long in VR may cause user discomfort and motion sickness.

Chapter 6

Evaluation

This chapter discusses the user validation performed to evaluate the predictive performance of the model and user design preferences for collaborator frustum visualisation. It also formulates the characteristics for a more comprehensive user study to be conducted in future work.

6.1 User Testing

Unfortunately, due to time and hardware limitations, a comprehensive user study was unable to be conducted for the model. However, a short user experiment was completed among 6 users who tried the model collaboratively, in pairs. The participants were tasked with simply deducing insights about the apartment and area based on the environment and data available. The instructions for the experiment are shown in the user guide at the end of the Appendix. Each user was asked to sign a form of consent, informing them what data would be gathered and stored from the experiment, as well as their right to withdraw at any time and any associated risks with the experiment, an unsigned copy of the consent form is also available in the Appendix. After the consent form was signed, participants were instructed to view the VR apartment environment and collaboratively decide if they would like to live in the location, 10 minutes were given for exploration, after which participants were instructed to enter the data room. In the data room participants were given another 10 minutes to deduce any insights about the surrounding area and decide if they would like to live in the area in question.

During the experiment participants were encouraged to try all four frustum implementations by pressing the 'A' button on the right touch controller of the Oculus to swap between each visualisation. The participants then filled out a short questionnaire which included the following questions:

- Which view visualisation did you prefer?
- Why did you prefer the chosen visualisation?
- How often did you and your collaborator have a mutual understanding of each other's gaze? (0 - never, 10 - always)

- How accurate do you think the predicted gaze view was? (0 - not at all, 10 - very)
- Did the visualisation clutter the scene or distract you?
- Which room did you find the view visualisation more helpful?
- Why did you find it more useful in that room?
- How often was verbal communication required to gain a mutual awareness? (0 - never, 10 - always)
- How often were the controllers used to point and gain a mutual understanding of gaze target? (0 - never, 10 - always)
- Did you experience any discomfort during the experiment?
- Would you choose to use the software again for a similar task? Why?
- Do you have any improvements which could be made to the view visualisation?

The results to the survey voting questions can be seen in Figure 6.1 and the complete results of the participant questionnaire can be found in the Appendix in their entirety. The user survey results displayed a significant predilection for the second dual high precision frustum implementation, with 4 of the six participants stating it as their preference. This was mostly due to the clarity of the blue frustum, created by the contrast between the red and blue cone colours and the accuracy of its predictions. Participants also stated that the cone 'seemed more stable', this will likely be caused by the high velocity threshold used for the dynamic frustum size adjustment. The dynamic high frequency frustum used a low velocity threshold, causing the cone to rapidly change size when the head moved, creating a flickering effect. This was discussed during the cone implementation in Section 5.1. The low velocity threshold was used to stay accurate to the gaze shift research that states the head is stationary when below 3°/s [28, p. 2], and that a gaze shift could be within a 40° cone of vision with no head movement. This is why the cone increases back to 40° when stationary and the model has no head movements to guide it. The testing does show that the dynamic high frequency frustum implementation, whilst staying true to the research, does so at the cost of usability and user comfort.

The users may also have preferred the smaller cone implementations due to a rendering issue present on the larger cones. Each cone was built using a mesh, which only encompasses the outside of the object, when the end of the cone clips through an object, the user is no longer able to see the base of the cone. This creates unwanted ambiguity in the predicted range of gaze target locations for collaborators and may break user immersion. The rendering issue was harder to recreate using the smaller frustums, as they were less likely to clip through scene objects. This design flaw could have been an influencing factor in users frustum preference, as most users stated they preferred the accuracy and clarity of the dual high precision frustum, with two users expressing that the larger frustums cluttered the view.

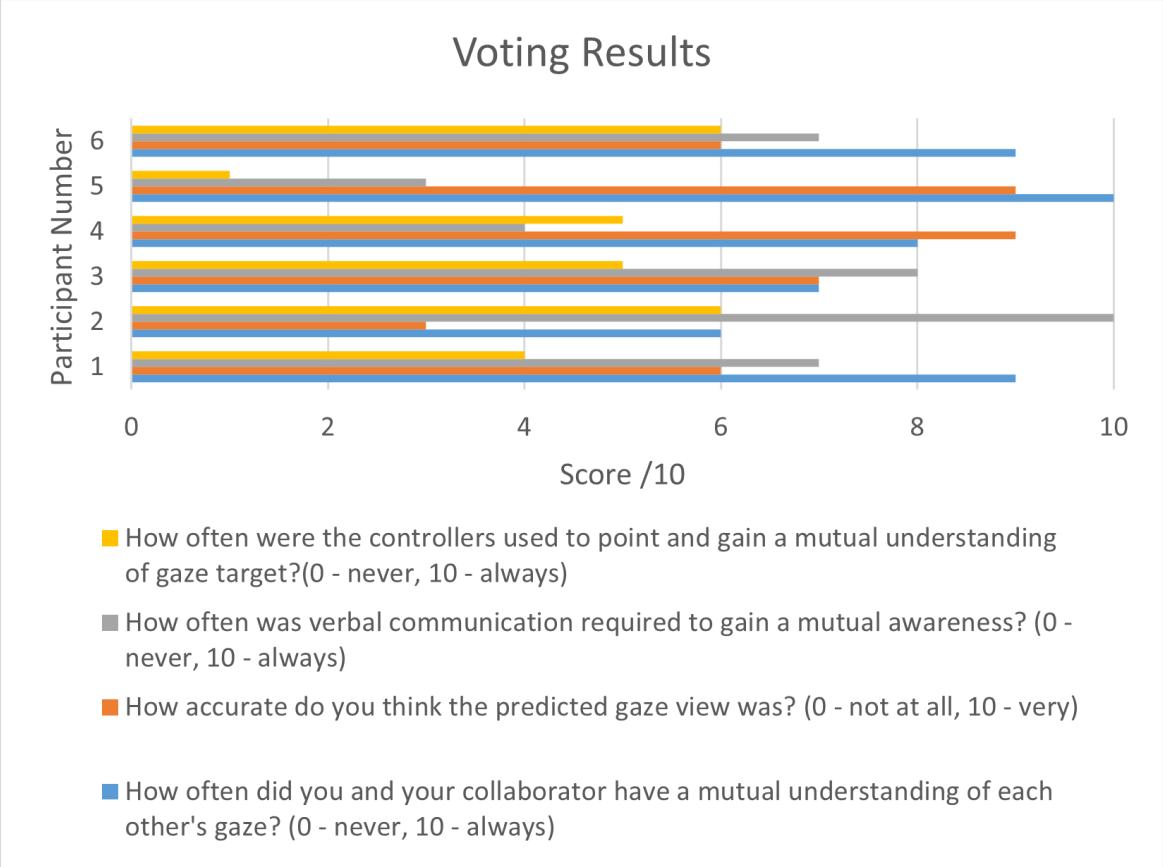


Figure 6.1: User survey voting question results

The survey results indicate a common sense of mutual awareness among collaborators; an average score of 8.2 out of 10 for a mutual understanding of collaborator's gaze coupled with the low average of 4.5 for the frequency controllers were needed as pointers, implies the software had a positive impact on the users shared awareness, as intended. However, the slightly lower average of 6.7 for the accuracy of the gaze prediction and a 6.5 for how often verbal referents were required, implies that the cone helped to find a general gaze location but lacked the precision for precise gaze targets. This outcome was expected from the lower accuracy models implemented, hopefully in future work the precise decision forests models can be applied for gaze inference, reducing the impact of this issue.

The addition of the data room environment provided useful insight in to the possible uses for the prediction software, with 4 of the six users stating it as their preferred environment for the visualisation software. This was commonly due to the lack of verbal referencing used to describe the smaller details observed on the graphs, which indicates that the model was accurate enough to assist users in achieving a mutual understanding of gaze target and could possibly be used within a similar commercial setting to assist the workflow of online VR collaborative meetings. However, the preference also stems from a lack of requiring shared awareness within the apart-

ment interior. This may be due to the familiarity of the environment and objects or possibly due to the lack of clustered small details within the environment. Upon analysis, whilst many small features were deployed in the apartment interior, they were not clustered or small enough to warrant the need for a precise gaze target location; the large frustum helped users find a general gaze direction and the final targeting was achieved using verbal referencing. This indicates that accurate gaze prediction may not be required in familiar environments with common verbal referencing, further environments with varying entropies of salient regions should be created to validate this hypothesis.

The user survey also indicated that most users would use the software again for a similar task, with only one user stating any discomfort during the experiment, this was due to the motion sickness that is common side effect of VR usage [87]. Most suggested user improvements were targeted at the frustum visualisation implementation, not model accuracy. The most frequent suggestions recommended the long distance frustum be removed, due to rendering issues, and the flickering fixed on the dynamic high frequency frustum. These are simple improvements that can be fixed within the Unity implementation. The user study analysis conveys the success of the model in aiding users to achieve a mutual shared awareness and promotes the use of dynamic frustum reduction based upon the gaze research analysis, which are the main contributions of this project, whilst also displaying a need for frustum alterations to be made before future user experiments are conducted.

6.2 Improved User Study

Whilst the user study executed provided useful insights in to the performance of the model, the evaluation would benefit from more comprehensive user validation; the limitations of the study are discussed in Section 7.2. A thorough model evaluation should be conducted with the following proposed properties:

- A minimum of 14 study participants - it has been shown that 5 users will uncover about 85% of issues during user testing [88]. However, the participants will be paired, requiring a minimum of 10 participants to run the test 5 times. The data used to train the model was also gathered from 13 participants, to create a comparable data set the testing should have a similar number.

- A minimum of 3 apartment environments - creating a wider variety of salient regions, thoroughly testing the accuracy of the model; giving the users increased choice, leading to comprehensive discrimination of each environment.

- Eye and head location tracking - enabling a comparison of live predictions against real-time gaze coordinates and an accurate prediction error value.

- Participant interviews - detailed discussion of the models to find possible hindrances caused by each frustum implementation and precise improvements, in addition to written questionnaires.
- Cash incentive - provide participants with a reward for their cooperation and accurate insights to encourage meaningful participation.

With more development time, money and access to eye-tracking hardware the proposed user study would have been conducted leading to a more exhaustive evaluation. However, the limited user study performed led to constructive insights. The results suggest that the model was successful in achieving shared awareness in a CVE, but design and accuracy could be improved.

Chapter 7

Conclusions

7.1 Model

The final project solution achieves the original desired outcome of predicting and visualising a collaborators gaze within an error range less than the high frequency view cone of 20° relative to the head. However, the final implementation would benefit from minor adjustments and improvements. The TensorFlow DNN model used for the Unity Implementation had an error range of:

- $-5.3414° \leq \text{x} \leq 5.3414°$
- $-7.5987° \leq \text{y} \leq 7.5987°$
- $-7.6369° \leq \text{z} \leq 7.6369°$

This creates an accuracy increase of 29.3172° for x values, 24.8026° for y values and 24.7262° for the z values, when comparing the model to the current gaze high frequency zone of a $\pm$20° error range. However, if the TensorFlow decision forests model could be converted to TFLite the model would have displayed an accuracy increase of 31.3590° for x values, 30.3782° for y values and 32.8022° for z values. The implementation of these models relies on TensorFlow's support of decision forest models within the TFLite converter, which should become available in future releases 'We expect to expand the set of supported operations in future TensorFlow Lite releases.'[89].

When implementing the model ready for user testing, it was also found that the model infrequently predicts a location outside of the physical 50° range of possible gaze target locations from a fixed head position. This error seems to occur at certain locations within the implemented testing environment, possibly due to floating point rounding errors or inaccuracies within the models. A hard-coded threshold fix was implemented to limit predictions to within the physical eye-in-head range, if a prediction is over 50° from the head forward vector the predicted frustum is simply set to the forward direction to show that the prediction was inaccurate. These impossible predictions highlight the need to improve model accuracy and reliability.

Training the model upon localised coordinates may help to achieve this, as mentioned in Section 4. This would prevent the model from learning salient regions from each video clip used for gathering the training data [4], possibly increasing the models performance in novel environments. Instead, the models implemented were trained upon world gaze and head coordinates, this was due to the MATLAB localised coordinate model's having higher RMSEs than the world coordinate models. The localised training data set also had large sections of unmapped coordinates that remained at the positive y axis (0,1,0) (the default localised coordinate vector); caused by the threshold velocity profile used to categorise head shifts, if entries went below 3°/s the head was considered stationary and not part of a shift (localised to (0,1,0)). If the Gaussian-based minima velocity profile could have been used at runtime to classify shifts, the localised shift data and world coordinate data would have been equally comprehensive. However, the Gaussian smoothing had to be applied retrospectively and the world coordinate data sets were used for model training.

To prevent salient regional bias from the model in future work, the model should be trained upon a larger data set gathered from a wider variety of environments, possibly a different environment stimuli set per user. Interactive environments would also help to diversify the gathered training data, creating different forms of head shifts for the model to train upon. The model generated in this work was limited by the stationary, naturalistic stimuli used to gather the training data.

Whilst there are small improvements to be made, the project was successful in finding a significant possible view frustum reduction in Section 3.1, when a user performs a head shift to the left or right and combining this gaze statistic with the prediction model. The implementation of this feature was favoured during user testing; the second dual high precision frustum implementation which only minimised the frustum when a user's shift was above 20°/s was preferred among test participants, helping to predict a collaborators gaze target before they had completed their gaze shift. This directional eye bias was found to be prominent in peak velocity groups 2 and 3, with a lower speed bound of 20°/s, making the second implementation more accurate to the head shift research whilst maintaining high user satisfaction.

7.2 User Validation Analysis

The user testing enabled an evaluation of the merits and failings of each frustum implementation and a clear preference for the second dual high precision frustum implementation was found, despite not covering the full 100° range of the eyes. However, the implemented experiment was limited in scope. An inadequate sample size of six participants was too small to gain a thorough evaluation of the models; the use of a collaborative task required the pairing of participants, leading to the task being performed only 3 times in total. Whilst it was suggested that 5 users are able to uncover 85% of software issues [88] and data was gathered from 6 users; it must also be considered that the training data [4] was gathered among 13 participants in a variety of environments. A minimum sample size of 14 participants should have

been used for model testing creating a comparable set of eye and head gaze data. The testing also lacks an accurate way to measure the efficacy of the models predictions. The only available metric is from the user surveys, where users are estimating accuracy based upon verbal confirmation of their collaborator. The implementation of eye tracking software would allow for a precise error value to be calculated for each models prediction; the data could then be combined with the training data set to develop a new model further increasing the variety of training data and model accuracy.

The use of a single environment also inhibits the model evaluation; the models predictive performance may prove to be environmentally dependent, with different types of head movements exhibited in novel environments. For example environments with a higher entropy of salient regions seem to produce smaller, yet more constant head movements and slower exploration [3]. The inclusion of stationary environments, where the user may only turn on the spot, would have created an analogous environment to the training data stimuli, appropriate for thorough evaluation of the model. A stationary environment was not developed due to most exploratory and collaborative VR applications involving some form of movement; testing only stationary environments would have limited the models commercial applicability.

7.3 Ethics

There were few legal, social ethical and professional considerations to acknowledge in this project. One ethical issue was identified in the ethics checklist: 'Section 2 - Does your project involve human participants?' [90]. Six human participants were involved in the user study described in Chapter 6.1. The participants were chosen among friends and colleagues of the author. Consent was gathered from each participant via a consent form that was signed and dated by both the participant and principal investigator, an unsigned copy of which can be found in the Appendix. The consent form informed participants:

- The purpose of the study.
- What the study will involve.
- That the participant may withdraw from the study at any time, without explanation and with no detrimental consequences.
- What data will be gathered and stored from the participant.
- How the data and results will be used and how the data will remain confidential.
- That the participant may decide that their data cannot be used.
- Who to contact with any concerns or questions and how they may contact them.

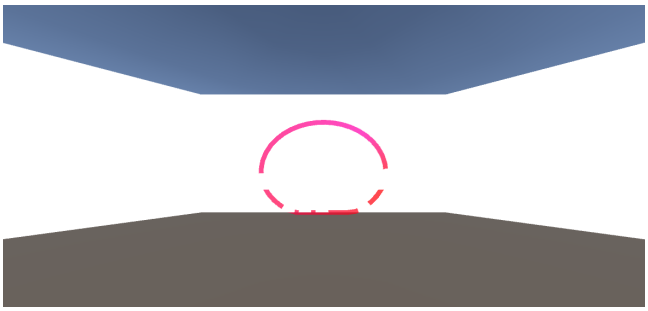


Figure 7.1: Riccardo Bovo's user field of view implementation

The study involved no particular risks associated with participation other than those associated with the use of Oculus Quest VR Headset such as 'Dizziness' or 'Eye Strain' [87]. After consent was gathered the user testing was performed without incident. The participants were also asked if they felt any discomfort or dizziness during the study to which only one user confirmed they felt slightly dizzy, however this was only stated upon completion of the experiment. Since the project had no funding and may be used for any intended application, it does not raise any other legal, social ethical and professional considerations.

7.4 Future Work

Going forward, future work will prioritise refining the model further to increase the accuracy of the predicted gaze location. This should first be attempted by gathering new training data using an eye-tracking HMD, the data set should be gathered from a wider range of stimuli: interactive environments, environments where the user can move and VR settings that intend to use the prediction model upon its release. The new data set could then be combined with the data gathered by Agtzidis et al. [4] creating a larger sample size, increasing the accuracy of mean values and providing a smaller margin of error. The expanded data set may also allow the model to be trained upon localised shift coordinates, with increased shift data to learn from. This could reduce any stimuli saliency bias found in the current model, improving its prediction accuracy in novel environments.

One of the main issues highlighted during the evaluation was the jittery, erratic prediction frustum movement. This was often caused by quick prediction values changing the frustums location drastically. Whilst this does match the jittery movement of the eyes [41], it caused user discomfort and broke user immersion. The erratic movement could be decreased by gradually shifting the cone towards a new predicted gaze location in small increments, creating a smooth movement in the right direction, however this will not accurately represent the error of the prediction and will not be a ground truth representation of the model. Another possible solution could implement a cross-hair that follows the predicted gaze location while the view frustum follows the direction of the users forward vector, similar to the fourth default large 100° frustum implementation. A cross-hair may be more appropriate

in a different view implementation such as one used in a study by Riccardo Bovo, which displayed the frustum as an oval projected on to the viewing surface, shown in Figure 7.1. This implementation prevents a cone frustum obstructing a collaborators view and clearly represents their possible gaze targets. A cross-hair could be added to the oval with its own smaller oval expressing the error values of the model. This type of implementation is better suited to the data analysis environment with clear flat surfaces for view projection, however this may be difficult to implement in a wider variety of settings such as the apartment environment.

If future implementations are successful in improving model accuracy the model could be extended to try and encompass long distance viewing. As mentioned previously, a zoom functionality could be introduced allowing the user to see further; extending their cone of view without increasing the error width of the view frustum. However, this would require a new set of training data built in environments that included the zoom feature, to understand how the eyes and head move whilst zoomed. This is also not possible when collaborating in real life and therefore not essential when recreating an accurate collaborative user experience. Alternatively, the user can simply use verbal referents or pointers for long distance gaze targets.

Finally, the role of monocular preference in VR could be explored by measuring each participant during the data gathering stage for indication of ocular dominance and in which eye the preference is evident. The influence of monocular preference in VR could then be determined to find if users are more likely to have their gaze favour the side of the head analogous to their eye preference as mentioned in Section 3.1. Whilst the right-side bias found in the data was marginal, the impact of ocular dominance in VR has not been covered in previous research and could be explored as an additional ambition during gaze data gathering for model improvements.

7.5 Conclusion

This paper aimed to provide a starting point for a comprehensive gaze prediction model in a fully immersive, collaborative VR setting. The model and implementation presented provides a basis of research for future VR gaze prediction models to be built upon. The novel implementation of a frustum that reduces based on head movements was well received in user testing and may provide inspiration to future collaborative user view implementations. Further improvements may be made to the model when the appropriate TensorFlow Lite libraries are released. The novel gaze research results that allowed this reduction of the potential cone of vision can provide support for future biological research in to the influence of directional head movement upon desired gaze targets.

Bibliography

[1] VIVE next-level immersion with precision eye tracking. https://www.vive.com/uk/product/vive-pro-eye/overview/. Accessed: 2021-07-02. pages 2

[2] KICKSTARTER fove: The world's first eye tracking virtual reality headset. https://www.kickstarter.com/projects/fove/fove-the-worlds-first-eye-tracking-virtual-reality. Accessed: 2021-05-20. pages 2, 8

[3] Pavel A Agrawala M Guitierrez D Masia B Wetzstein G Sitzmann V, Serrano A. How do people explore virtual environments? 2018. URL https://arxiv.org/pdf/1612.04335.pdf. pages 3, 5, 6, 9, 10, 16, 20, 25, 40, 49

[4] Dorr M Agtzidis I, Startsev M. 360-degree video gaze behaviour: A ground-truth data-set and a classification algorithm for eye movements. technical report. 2019. URL https://arxiv.org/pdf/1903.06474.pdf. pages 3, 8, 9, 10, 11, 19, 23, 37, 39, 48, 50

[5] MathWorks help center. https://uk.mathworks.com/help/stats/regression-learner-app.html. Accessed: 2021-05-25. pages 4, 10, 26, 27

[6] Unity the leading platform for creating interactive, real-time content. https://unity.com/. Accessed: 2021-06-20. pages 4, 41

[7] Oculus quest 2. https://www.oculus.com/quest-2/?locale=en_GB. Accessed: 2021-06-20. pages 4

[8] TensorFlow an end-to-end open source machine learning platform. https://www.tensorflow.org/, . Accessed: 2021-06-23. pages 4, 29

[9] Mín Abadi, A Agarwal, P Barham, E Brevdo, Z Chen, C Citro, G S. Corrado, A Davis, J Dean, M Devin, S Ghemawat, I Goodfellow, A Harp, G Irving, M Isard, Y Jia, R Jozefowicz, L Kaiser, M Kudlur, J Levenberg, D Mané, R Monga, S Moore, D Murray, C Olah, M Schuster, J Shlens, B Steiner, I Sutskever, K Talwar, P Tucker, V Vanhoucke, V Vasudevan, F Viégas, O Vinyals, P Warden, M Wattenberg, M Wicke, Y Yu, and X Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org. pages 4

[10] Tatler B Land M. Looking and acting: Vision and eye movements in natural behaviour. 2012. URL https://oxford.universitypressscholarship.com/view/10.1093/acprof:oso/9780198570943.001.0001/acprof-9780198570943. pages 5, 6, 36

[11] Freedman E.G. Coordination of the eyes and head during visual orienting. 2008. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2605952/. pages 5, 6, 36

[12] Van Opstal J Goossens J. Human eye-head coordination in two dimensions under different sensorimotor conditions. 1997. URL https://pubmed.ncbi.nlm.nih.gov/9187290/. pages 5, 36

[13] Gellersen H Sidenmark L. Eye, head, and torso coordination during gaze shifts in virtual reality. 2020. URL https://dl.acm.org/doi/pdf/10.1145/3361218. pages 5, 8

[14] Lee C Seo H. Head-free reading of horizontally and vertically arranged texts. 2000. URL https://www.sciencedirect.com/science/article/pii/S0042698902000639. pages 5

[15] Triesch J Saeb S, Weber C. Learning the optimal control of coordinated eye and head movements. 2011. URL https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1002253. pages 6, 11, 12, 13

[16] Roucoux A LLefvre P, Bottemanne I. Experimental study and modeling of vestibulo-ocular reflex modulation during large shifts of gaze in humans. 1992. URL https://link.springer.com/content/pdf/10.1007/BF00227846.pdf. pages 6, 20

[17] Bardins S Bartl K Böning G Schneider E König P Einhäuser W, Schumann F. Human eye-head co-ordination in natural exploration. 2007. URL https://doi.org/10.1080/09548980701671094. pages 7

[18] Stahl J.S Oommen B.S, Smith R.S. The influence of future gaze orientation upon eyehead coupling during saccades. 2004. URL https://doi.org/10.1007/s00221-003-1694-z. pages 7

[19] Stahl J.S Thumser Z.C. Eye-head coupling tendencies in stationary and moving subjects. 2009. URL https://doi.org/10.1007/s00221-009-1803-8. pages 7

[20] Fuller J.H. Head movement propensity. 1992. URL https://doi.org/10.1007/BF00230391. pages 7

[21] Stahl J.S. Amplitude of human head movements associated with horizontal saccades. 1999. URL :https://doi.org/10.1007/s002210050715. pages 7

[22] Duchowsk A. T Marmitt G. Modeling visual attention in vr: Measuring the accuracy of predicted scanpaths. 2002. URL https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.91.5486&rep=rep1&type=pdf. pages 7, 8

[23] Barnes G R. Vestibulo-ocular function during co-ordinated head and eye movements to acquire visual targets. 1979. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1281486/. pages 8

[24] Worden A Watson B Walker N, Hodges L F. Managing level of detail through peripheral degradation: effects on search performance with a head-mounted display. 1997. URL https://dl.acm.org/doi/10.1145/267135.267137. pages 8

[25] Duchowski A T. Serious gaze. 2017. URL https://par.nsf.gov/servlets/purl/10098312. pages 8

[26] Kulshreshth A Wisniewski P LaViola J.J Pfeil K, Taranta II E.M. A comparison of eye-head coordination between virtual and physical realities. 2018. URL http://eecs.ucf.edu/isuelab/publications/pubs/SAP2018.pdf. pages 8

[27] Land M.F. The coordination of rotations of the eyes, head and trunk in saccadic turns produced in natural situations. 2004. URL https://link.springer.com/article/10.1007/s00221-004-1951-9. pages 8

[28] Hatori Y Hiratani A Matsumiya K Kuriki I Shioiri S Nakashima R, Fang Y. Saliency-based gaze prediction based on head direction. 2015. URL https://www.sciencedirect.com/science/article/pii/S0042698915003338. pages 8, 11, 19, 36, 43

[29] Goodale M.A Gegenfurtner K.R Thaler L, Schütz A.C. What is the best fixation target? the effect of target shape on stability of fixational eye movements. 2013. URL https://doi.org/10.1016/j.visres.2012.10.012. pages 9

[30] Haines O Calway A Mayol-Cuevas W Damen D, Leelasawassuk T. You-do, i-learn: Discovering task relevant objects and their modes of interaction from multi-user egocentric video. 2014. URL http://www.bmva.org/bmvc/2014/papers/paper059/index.html. pages 9

[31] Kübler T Kasneci E Santini T, Fuhl W. Bayesian identification of fixations, saccades, and smooth pursuits. 2016. URL https://arxiv.org/pdf/1511.07732.pdf. pages 9

[32] Ioannis Agtzidis. matlab utils. https://gin.g-node.org/ioannis.agtzidis/matlab_utils, . Accessed: 2021-05-11. pages 9

[33] Ioannis Agtzidis. matlab 360 utils. https://gin.g-node.org/ioannis.agtzidis/matlab_360_utils, . Accessed: 2021-05-12. pages 9, 13

[34] Sidorchuk D Bolshakov A, Gracheva M. How many observers do you need to create a reliable saliency map in vr attention study? 2017. URL https://www.researchgate.net/publication/332468531_How_many_observers_do_you_need_to_create_a_reliable_saliency_map_in_VR_attention_study. pages 10

[35] Coutrot A Da Silva M.P-Le Callet P David E.J, Gutiérrez J. A dataset of head and eye movements for 360° videos. 2018. URL https://hal.archives-ouvertes.fr/hal-01994923/document. pages 10

[36] Le Callet P Rai Y, Gutiérrez J. A dataset of head and eye movements for 360 degree images. 2017. URL https://dl.acm.org/doi/10.1145/3083187.3083218. pages 10

[37] Unity Documentation transform.forward. https://docs.unity3d.com/ScriptReference/Transform-forward.html. Accessed: 2021-06-20. pages 10

[38] Matsumiya K Kuriki I Shioiri S Fang Y, Nakashima R. Eye-head coordination for visual cognitive processing. 2015. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4370616/pdf/pone.0121035.pdf. pages 12, 40

[39] Matsumiya K Kuriki I Shioiri S Tokunaga R Fang Y, Nakashima R. Eye position distribution depending on head orientation in natural scene viewing. 2015. URL https://journals.sagepub.com/doi/pdf/10.1068/if719. pages 12

[40] Rodrigues' rotation formula. https://en.wikipedia.org/wiki/Rodrigues%27_rotation_formula. Accessed: 2021-06-25. pages 13

[41] Gellersen H Sidenmark L. Eyehead: Synergetic eye and head movement for gaze pointing and selection. 2019. URL https://dl.acm.org/doi/10.1145/3332165.3347921. pages 14, 50

[42] Martin W.N Reichert K.C Frey L.A Hutchinson T.E, White K.P. Humancomputer interaction using eye-gaze input. 1989. URL https://ieeexplore.ieee.org/document/44068. pages 14

[43] Lin M Radwin R.G, Vanderheiden G.C. A method for evaluating head-controlled computer input devices using fitts' law. 1990. URL https://journals.sagepub.com/doi/abs/10.1177/001872089003200405. pages 14

[44] Mikaelian H.H Ware C. An evaluation of an eye tracker as a device for computer input2. 1986. URL https://dl.acm.org/doi/10.1145/30851.275627. pages 14

[45] Piumsomboon T Lee G. A Billinghurst M Kyto M, Ens B. Pinpointing: Precise head- and eye-based target selection for augmented reality. 2018. URL https://www.researchgate.net/publication/323970135_Pinpointing_Precise_Head-_and_Eye-Based_Target_Selection_for_Augmented_Reality. pages 15

[46] Majaranta P Spakov O, Isokoski P. Look and lean: Accurate head-assisted eye pointing. 2014. URL https://dl.acm.org/doi/10.1145/2578153.2578157. pages 15

[47] Pederson T Jalaliniya S, Mardanbegi D. Magic pointing for eyewear computers. 2015. URL https://dl.acm.org/doi/10.1145/2802083.2802094. pages 15

[48] Gutwin C Wong N. Support for deictic pointing in cves: Still fragmented after all these years? 2014. URL https://dl.acm.org/doi/abs/10.1145/2531602.2531691. pages 15, 16, 17, 18, 39

[49] Fraser M Benford S Hindmarsh J, Heath C. Supporting awareness and interaction through collaborative virtual interfaces. 1999. URL https://dl.acm.org/doi/pdf/10.1145/320719.322580. pages 16, 17, 38

[50] Heath C Benford S Greenhalgh C Hindmarsh J, Fraser M. Fragmented interaction: Establishing mutual orientation in virtual environments. 1998. URL https://dl.acm.org/doi/10.1145/289444.289496. pages 16, 38

[51] Heath C Benford S Greenhalgh C Hindmarsh J, Fraser M. Object-focused interaction in collaborative virtual environments. 2000. URL https://dl.acm.org/doi/10.1145/365058.365088. pages 16, 38

[52] Christmann O Richir S Eynard R, Pallot M. Impact of verbal communication on user experience in 3d immersive virtual environments. 2017. URL https://hal.archives-ouvertes.fr/hal-01186368/document. pages 17

[53] Lee Y Billinghurst M Piumsomboon T, Lee Y. Covar: A collaborative virtual and augmented reality system for remote collaboration. 2017. URL https://dl.acm.org/doi/10.1145/3132818.3132822. pages 17, 18, 34, 36

[54] Nguyen H.T.T Duval T. Improving awareness for 3d virtual collaboration by embedding the features of users' physical environments and by augmenting interaction tools with cognitive feedback cues. 2013. URL https://www.researchgate.net/publication/259866718_Improving_Awareness_for_3D_Virtual_Collaboration_by_Embedding_the_Features_of_Users. pages 18

[55] MathWorks help center. https://uk.mathworks.com/help/matlab/ref/islocalmin.html. Accessed: 2021-05-23. pages 19

[56] White L.E Purves D. Monocular preferences in binocular viewing. 1994. URL https://www.pnas.org/content/91/18/8339. pages 23

[57] van der Steen J de Faber J.H.N Collewijn H van Leeuwen A. F, Westen M. J. Gaze-shift dynamics in subjects with and without symptoms of convergence insufficiency: influence of monocular preference and the effect of training. 1999. URL https://www.sciencedirect.com/science/article/pii/S0042698999000668. pages 23

[58] Phillips J.R Turnbull P. R. K. Ocular efects of virtual reality headset wear in young adults. 2017. URL https://www.nature.com/articles/s41598-017-16320-6. pages 23

[59] MathWorks help center: Choose regression model options. https://uk.mathworks.com/help/stats/choose-regression-model-options.html#bvi2d8a-19. Accessed: 2021-06-20. pages 27

[60] Cawley G C. Mohammed R O. Over-fitting in model selection with gaussian process regression. 2013. URL https://ueaeprints.uea.ac.uk/id/eprint/67158/1/Chapter.pdf. pages 27

[61] MathWorks help center: Gaussian process regression models. https://uk.mathworks.com/help/stats/gaussian-process-regression-models.html. Accessed: 2021-06-25. pages 27

[62] Githhub cross compiling. https://github.com/Glympse/CrossCompiling. Accessed: 2021-06-20. pages 28, 32

[63] Microsoft ml.net. https://dotnet.microsoft.com/apps/machinelearning-ai/ml-dotnet. Accessed: 2021-06-20. pages 28

[64] Microsoft ml.net model builder. https://dotnet.microsoft.com/apps/machinelearning-ai/ml-dotnet/model-builder. Accessed: 2021-06-20. pages 28

[65] Microsoft download .net framework. https://dotnet.microsoft.com/download/dotnet-framework. Accessed: 2021-08-20. pages 28

[66] TensorFlow for mobile iot. https://www.tensorflow.org/lite, . Accessed: 2021-06-23. pages 29, 34

[67] Android introducing android 11. https://www.android.com/intl/en_uk/. Accessed: 2021-08-20. pages 29

[68] TensorFlow decision forests. https://www.tensorflow.org/decision_forests/tutorials/beginner_colab. Accessed: 2021-06-23. pages 29

[69] TensorFlow basic regression: Predict fuel efficiency. https://www.tensorflow.org/tutorials/keras/regression#a_dnn_regression. Accessed: 2021-06-23. pages 29

[70] Lai P Lin C Lee C Hung C, Chen W. Comparing deep neural network and other machine learning algorithms for stroke prediction in a large-scale population-based electronic medical claims database. 2017. URL https://ieeexplore.ieee.org/document/8037515. pages 29

[71] Gleyzer S. Machine learning developments in root. 2017. URL https://www.researchgate.net/publication/321237078_Machine_Learning_Developments_in_ROOT. pages 29

[72] TensorFlow tf.keras.model. https://www.tensorflow.org/api_docs/python/tf/keras/Model. Accessed: 2021-07-12. pages 29, 34

[73] TensorFlow tf.keras.sequential. https://www.tensorflow.org/api_docs/python/tf/keras/Sequential. Accessed: 2021-06-24. pages 29

[74] TensorFlow basic regression: Predict fuel efficiency. https://www.tensorflow.org/tutorials/keras/regression. Accessed: 2021-06-23. pages 29

[75] Shen W Lin G. Research on convolutional neural network based on improved relu piecewise activation function. 2018. URL https://www.sciencedirect.com/science/article/pii/S1877050918306197. pages 30

[76] Ispirer convert c/c++ to c. https://www.ispirer.com/application-conversion/c-c-to-cs-migration, . Accessed: 2021-05-28. pages 32

[77] Software Informer c++ to c converter (free edition) 3.5. https://c-to-c-converter-free-edition1.software.informer.com/, . Accessed: 2021-05-28. pages 32

[78] Tangible Software Solutions c++ to c converter. https://www.tangiblesoftwaresolutions.com/product_details/cplusplus_to_csharp_converter_details.html. Accessed: 2021-05-28. pages 32

[79] MathWorks help center: uint8. https://uk.mathworks.com/help/matlab/ref/uint8.html. Accessed: 2021-06-20. pages 33

[80] Unity unity machine learning agents. https://unity.com/products/machine-learning-agents. Accessed: 2021-07-11. pages 34

[81] Oculus oculus rift s is no longer available. https://www.oculus.com/rift-s/?locale=en_GB. Accessed: 2021-07-12. pages 34

[82] FlatBuffers flatbuffers. https://google.github.io/flatbuffers/. Accessed: 2021-07-09. pages 34

[83] TensorFlow tensorflow lite. https://www.tensorflow.org/lite/guide#2_convert_the_model. Accessed: 2021-07-09. pages 34

[84] GitHub tf lite experimental unity plugin. https://github.com/tensorflow/tensorflow/tree/master/tensorflow/lite/experimental/examples/unity/TensorFlowLitePlugin. Accessed: 2021-07-15. pages 34

[85] Photon pun. https://www.photonengine.com/en-US/PUN. Accessed: 2021-06-25. pages 38

[86] Unity asset store. https://assetstore.unity.com/. Accessed: 2021-06-10. pages 41

[87] Oculus legal documents. https://www.oculus.com/legal/health-and-safety-warnings/?locale=en_GB. Accessed: 2021-08-20. pages 45, 50

[88] Nielsen J. Why you only need to test with 5 users. 2000. URL https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/. pages 45, 48

[89] TensorFlow tensorflow lite and tensorflow operator compatibility. https://www.tensorflow.org/lite/guide/ops_compatibility. Accessed: 2021-08-10. pages 47

[90] Imperial College London legal, social, ethical and professional requirements. https://wiki.imperial.ac.uk/display/docteaching/Legal%2C+Social%2C+Ethical+and+Professional+Requirements. Accessed: 2021-08-20. pages 49

Appendices

Appendix A

User Study Results

Any mention of partner names has been replaced with *partner* to comply with the General Data Protection Act 2018.

1. Which view visualisation did you prefer?
    - Participant 1 - Red and Blue
    - Participant 2 - Short Red
    - Participant 3 - Red and Blue
    - Participant 4 - Red and Blue
    - Participant 5 - Red and Blue
    - Participant 6 - Short Red
2. Why did you prefer the chosen visualisation?
    - Participant 1 - 'It was the clearest of the three, easy to see where *partner* was looking and seemed the least glitchy'
    - Participant 2 - 'It was the smallest and didn't block my view as much'
    - Participant 3 - 'The blue was way easier to see'
    - Participant 4 - 'I preferred the look of the short red one but it kept flickering, whereas the red and blue seemed more stable but still helped to find where *partner* was looking'
    - Participant 5 - 'The moving blue and stable red cones made it easier to follow exactly what *partner* was looking at.'
    - Participant 6 - 'It was the smallest making it the most accurate to pinpoint what *partner* was talking about'
3. How often did you and your collaborator have a mutual understanding of each other's gaze? (0 - never, 10 - always)
    - Participant 1 - 9
    - Participant 2 - 6

- Participant 3 - 7
- Participant 4 - 8
- Participant 5 - 10
- Participant 6 - 9

4. How accurate do you think the predicted gaze view was? (0 - not at all, 10 - very)
    - Participant 1 - 6
    - Participant 2 - 3
    - Participant 3 - 7
    - Participant 4 - 9
    - Participant 5 - 9
    - Participant 6 - 6
5. Did the visualisation clutter the scene or distract you?
    - Participant 1 - 'No, the blue and red one helped to find where *partner* was looking. The long one did clutter the scene though which is why I didn't use it very much'
    - Participant 2 - 'Yes, especially the long one'
    - Participant 3 - 'Not really, it was useful to see which way *partner* was looking'
    - Participant 4 - 'Mostly not, I found it helpful to see what *partner* was looking at, occasionally in the data room it blocked the graphs.'
    - Participant 5 - 'Only the long view cluttered the scene, the others did not.'
    - Participant 6 - 'sometimes when it kept changing size yes, but not when it was stable'
6. Which room did you find the view visualisation more helpful?
    - Participant 1 - Data Room
    - Participant 2 - Apartment Interior
    - Participant 3 - Data Room
    - Participant 4 - Apartment Interior
    - Participant 5 - Data Room
    - Participant 6 - Data Room
7. Why did you find it more useful in that room?
    - Participant 1 - 'it was useful to find which graph *partner* was looking at and what part of it they was referencing. Didn't need it much in the apartment as I could see their avatar.'

- Participant 2 - 'In the data room the visualisation distracted me from the data'
- Participant 3 - 'It was harder to describe what I was referring to, the visualisation made it easy for *partner* to tell what I was looking at'
- Participant 4 - 'In the data room it blocked my view a bit, but was still useful to see what *partner* was talking about, but I relied on it more in the apartment to find *partner*.'
- Participant 5 - 'Describing what we were seeing was harder so I used the view cone more'
- Participant 6 - 'I used it more in the data room as it was easier than describing what I was looking at'

8. How often was verbal communication required to gain a mutual awareness? (0 - never, 10 - always)

   - Participant 1 - 7
   - Participant 2 - 10
   - Participant 3 - 8
   - Participant 4 - 4
   - Participant 5 - 3
   - Participant 6 - 7

9. How often were the controllers used to point and gain a mutual understanding of gaze target? (0 - never, 10 - always)

   - Participant 1 - 4
   - Participant 2 - 6
   - Participant 3 - 5
   - Participant 4 - 5
   - Participant 5 - 1
   - Participant 6 - 6

10. Did you experience any discomfort during the experiment?

    - Participant 1 - 'No'
    - Participant 2 - 'Yes I got slight vertigo/ dizziness'
    - Participant 3 - 'No'
    - Participant 4 - 'No not at all'
    - Participant 5 - 'No'
    - Participant 6 - 'No'

11. Would you choose to use the software again for a similar task? Why?

- Participant 1 - 'Yes, it seemed a good way to do a paired task'
- Participant 2 - 'No I would have found it easier to point to something and talk about it when referring to it.'
- Participant 3 - 'Yes I'd use the red and blue visualisation again because it helped when discussing the data, but not the other two'
- Participant 4 - 'Yes it seemed well suited to this task'
- Participant 5 - 'Yes, It made finding where *partner* was looking much easier, meaning the task was far easier.'
- Participant 6 - 'It could do with improvements but yes probably for a similar task'

12. Do you have any improvements which could be made to the view visualisation?

- Participant 1 - 'the long view seemed a bit buggy and was too large so couldn't tell what part of the area *partner* was looking at, the small view also flickered a lot, which stopped me using it'
- Participant 2 - 'I think the accuracy of the prediction could be improved and the cone should move smoother - it was very jittery.'
- Participant 3 - 'The long visualisation should be made clearer as you cant really tell what it's pointing at'
- Participant 4 - 'The flickering view needs to be fixed, and maybe make the colour of the blue a bit duller.'
- Participant 5 - 'Changing the colour from red to another more visible colour would be useful, in addition to removing the long frustum.'
- Participant 6 - 'Make the predictions more stable it jumps around too much, maybe make the cone a bit brighter as well'

Appendix B

Apartment Environment

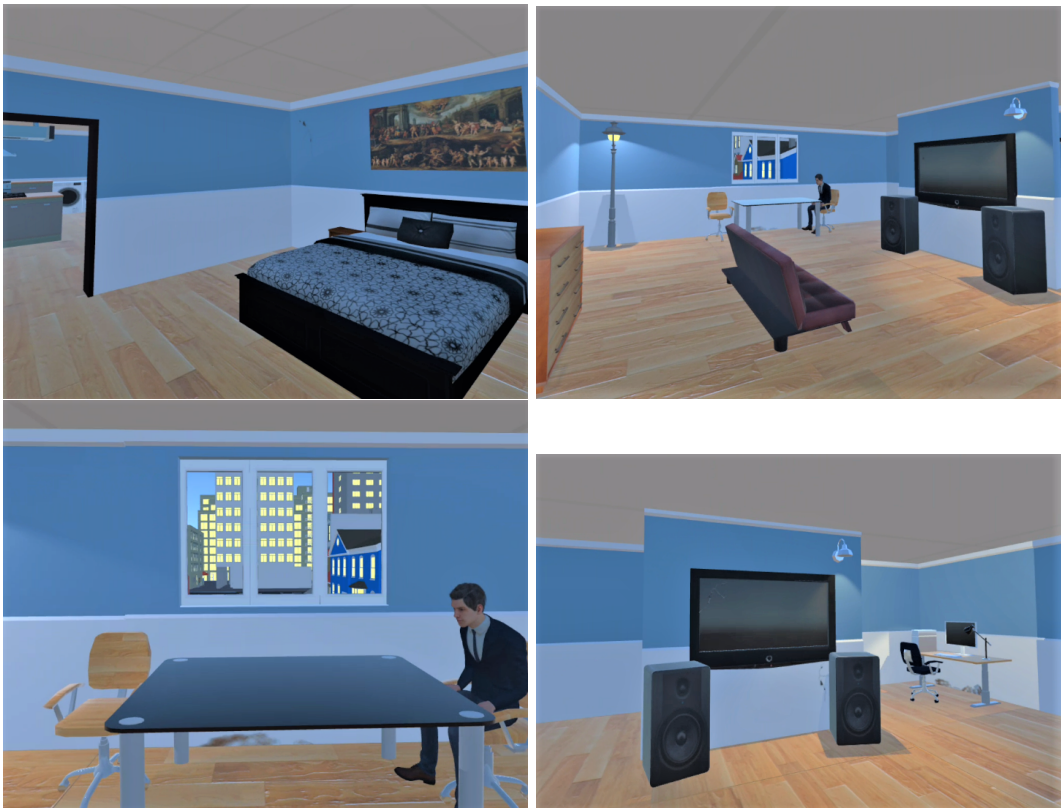


Figure B.1: Extra screenshots of the apartment environment

Appendix C

User Consent Form and User Guide

Imperial College London

Department of Computing

Information Sheet for Participants in Research Studies

You will be given a copy of this information sheet.

Title of Project: **Refining Gaze Prediction for Collaboration within a Virtual Environment**

Name, Address and Contact Details of Investigators:
Principal investigator:
George Castle
Imperial College London
South Kensington,
London SW7 2BU
United Kingdom
Email: george.castle20@imperial.ac.uk

We would like to invite you to participate in this research project directed by researchers at Imperial College London. You should only participate if you want to; choosing not to take part will not disadvantage you in any way. Before you decide whether you want to take part, it is important for you to read the following information carefully and discuss it with others if you wish. Ask us if there is anything that is not clear or if you would like more information.

Study Details: This research project aims to improve collaboration techniques in Virtual Reality (VR). During the study, you will be asked to complete a collaborative task in VR, using your own VR headset. You will collaborate with another participant who will see your VR avatar, and with whom you will be able to talk. The VR task will take around 20 minutes. You will then be asked to take part in a brief survey. It is expected that the entire study (including also setup) will take no longer than 45 mins. There are no particular risks associated with your participation other than those associated with the use of Oculus Quest VR Headset.

Data and Information: All data will be handled according to the General Data Protection Act 2018. The study will be anonymous and not require any personal information other than your name and signature on the consent form. The only data collected and stored will be the survey answers. The only individuals with access to this stored data will be the investigators of this study. Any information that is obtained in connection with this study and that can be identified with you will remain confidential

and will be disclosed only with your permission or as required by law. The information collected during the experiment will be kept separately from your personal identity. The information will be collected and stored in a password-protected computer. Any information published from this experiment will be done in aggregate or anonymous form. If any quotes from the survey answers are published, it will be in aggregated or anonymized form (i.e. not revealing your identity).

Withdrawing from the study: If you decide you do not want your answers to be stored, you may request that we delete the data from our dataset by contacting the principal investigator (e-mail george.castle20@imperial.ac.uk), within 3 days of completing the study. After this period, all data will be fully anonymized (i.e. anything that may identify the data as yours will be removed) and may be made available to other researchers – as this can be helpful for other researchers working on similar topics. You may withdraw at any time and for any reason during the study. You may access or withdraw your data at any time and for any reason within 3 days of completing the study when all the data is anonymized or securely destroyed.

Concerns or complaints: Should you have any concern or complaint, you can ask the investigator conducting the study in person or contact the principal investigator at any point (e-mail george.castle20@imperial.ac.uk). It is up to you to decide whether or not to take part. If you decide to take part, you will be given this information sheet to keep and be asked to sign a consent form. If you decide to take part, you are still free to withdraw at any time and without giving a reason.

Imperial College London

Informed Consent Form for Participants in Research Studies
(This form is to be completed independently by the participant after reading the Information Sheet and/or having listened to an explanation about the research.)

Title of Project: **Refining Gaze Prediction for Collaboration within a Virtual Environment**

Participant's Statement
I …….....
agree that I have:

- read the information sheet;
- had the opportunity to ask questions and discuss the study;
- received satisfactory answers to all my questions or have been advised of an individual to contact for answers to pertinent questions about the research and my rights as a participant and whom to contact in the event of a research-related injury.
- I understand that my survey answers will be stored and I am aware of and consent to the analysis of the storage.

I understand that I am free to withdraw from the study without penalty if I so wish. I understand that I consent to the processing of my personal information for the purposes of this study only. I understand that any such information will be treated as strictly confidential and handled in accordance with the provisions of the General Data Protection Act 2018.

Signed:

Date:

Investigator's Statement
I ……..
confirm that I have carefully explained the purpose of the study to the participant and outlined any reasonably foreseeable risks or benefits (where applicable).

Signed:

Date:

Imperial College London

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Gaze Prediction Model - User Guide

Author:
George Castle

Supervisor:
Thomas Heinis
Riccardo Bovo

Submitted in partial fulfillment of the requirements for the MSc degree in Advanced Computing MSc of Imperial College London

April 2021

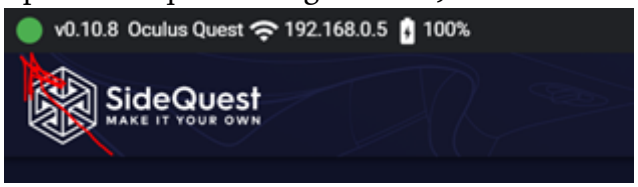
Contents

Chapter 1

Experiment User Guide

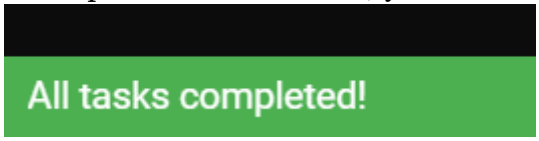
1.1 Installing the application

1. Download the experiment application at: `https://imperiallondon-my.sharepoint.com/:u:/g/personal/gwc20_ic_ac_uk/EZyCW8QBLplFvNxKKzUbdHYB2HIAbzCbIxaRI8WjuFSWrA?e=NLJkyd`
2. Download and install SideQuest at: `https://sidequestvr.com/setup-howto`
3. Ensure your device has developer mode enabled in case it does not follow this tutorial: `https://learn.adafruit.com/sideloading-on-oculus-quest/enable-developer-mode`
4. Start side Quest and Connect your Oculus to the computer (this should show up on side quest as a green dot):



5. On side quest select: INSTALL APK FROM FOLDER ON COMPUTER

6. Selected the downloaded Apartment_VR_Test_server.apk
7. If the process is successful, you'll be able to see:



8. Once installed access the application (**Library –>unknown sources –>FirstTry**).

1.2 Before the experiment

1. Ensure Oculus is fully charged.
2. Ensure Oculus controllers are fully charged.
3. Ensure your room is set up for a VR experience without obstacles.

Note:

- The experiment must be performed standing, you won't be able to take part if you are sitting down.
- The experiment will last a maximum of 35 minutes: (one 20 minute session in VR and one questionnaire), please make sure you haven't used your Oculus Quest for at least 20 minutes before the start of the experiment so that you are rested and ready to spend 20 minutes in a virtual environment.

1.3 Experiment Instructions

Setting: You and your collaborator have recently moved to London and are looking for an apartment to live in. This is your first viewing. Walk around the apartment and try to gain insights about the property and its former owner (who is sitting within the apartment during the viewing). Discuss these insights with your collaborator and decide if you would like to live in the apartment. After viewing for 10 minutes please enter the data visualisation room and look at the statistics displayed about the surrounding area. Discuss these statistics with your partner and decide if you would like to live in that area.
Instructions:

- Move backwards and forwards with the left analogue stick of your left touch controller.
- To turn simply turn your head.
- To change collaborator's view visualisation press the 'A' button on the right touch controller.
- To enter the apartment data room walk through the following door:

1.4 After the experiment

- You will be told when to move to the data room and how much observation time you have remaining.
- Upon completion of the experiment please fill out the following survey: `https://forms.office.com/r/1PESYv5xWy`