Deep Learning-based Simulator Sickness Estimation from 3D Motion

Junhong Zhao*
CMIC,
Victoria University of Wellington

Kien T. P. Tran*
HIT Lab,
University of Canterbury

Andrew Chalmers
CMIC,
Victoria University of Wellington

Weng Khuan Hoh
CMIC,
Victoria University of Wellington

Richard Yao
Reality Labs, Meta

Arindam Dey
Reality Labs, Meta

James Wilmott
Reality Labs, Meta

James Lin
Reality Labs, Meta

Mark Billinghurst
Empathic Computing Lab,
The University of Auckland

Robert W. Lindeman
HIT Lab,
University of Canterbury

Taehyun Rhee†
CMIC,
Victoria University of Wellington

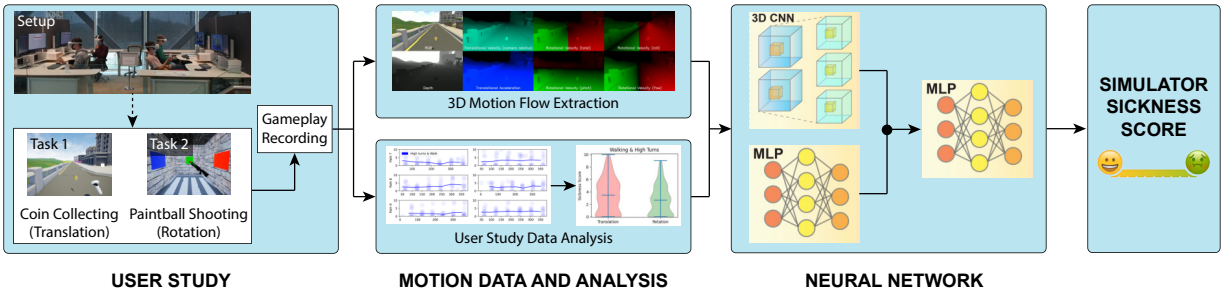

Figure 1: System diagram of our simulator sickness estimation solution, illustrating data capture from a user study, analysis of recorded user data, extraction of motion-flow data, training of the neural network, and the estimation of simulator sickness scores.

Abstract

This paper presents a novel solution for estimating simulator sickness in HMDs using machine learning and 3D motion data, informed by user-labeled simulator sickness data and user analysis. We conducted a novel VR user study, which decomposed motion data and used an instant dial-based sickness scoring mechanism. We were able to emulate typical VR usage and collect user simulator sickness scores. Our user analysis shows that translation and rotation differently impact user simulator sickness in HMDs. In addition, users' demographic information and self-assessed simulator sickness susceptibility data are collected and show some indication of potential simulator sickness. Guided by the findings from the user study, we developed a novel deep learning-based solution to better estimate simulator sickness with decomposed 3D motion features and user profile information. The model was trained and tested using the 3D motion dataset with user-labeled simulator sickness and profiles collected from the user study. The results show higher estimation accuracy when using the 3D motion data compared with methods based on optical flow extracted from the recorded video, as well as improved accuracy when decomposing the motion data and incorporating user profile information.

Index Terms: Virtual reality; machine learning; simulator sickness estimation; 3D motion; simulator sickness labelling

1 Introduction

Virtual reality (VR) users in head-mounted displays (HMDs) may feel simulator sickness during or after use. The after-effects can influence their driving, equipment operation, and other real-life activities. Simulator sickness using HMDs is theoretically caused by a conflict between the human visual system perceiving the information from the virtual environment and the vestibular system reacting to the movements in their physical environment. Researchers have studied the causes of simulator sickness, including content-based factors, individual characteristics, and hardware conditions [3,5,45,50]. From a hardware and software design perspective, there is a trade-off between higher immersion and lower simulator sickness in VR applications. By knowing the amount of simulator sickness in a VR application, and its causes, VR developers can optimize their solutions for an improved experience.

Properties of motion within the virtual environment are prominent factors in determining the degree of simulator sickness [14,22,42]. Previous studies investigated how movement characteristics in VR content, such as movement frequency and amplitude, axial rotation, and navigation speed, contribute to simulator sickness [1, 4, 22, 29, 36, 37, 48]. However, these prior works focus either solely on translational motion [48], or only rotational motion [1,2,21,22,37]. Many previous studies [14, 42] evaluate simulator sickness in a constrained setup (passive viewing without sufficient interaction or head movement), which is not representative of how VR users often interact in VR.

Recent research has started to utilize machine learning and user simulator sickness labels to predict simulator sickness from visual content [7, 19, 25, 32, 34, 43]. In these approaches, identifying the right features that cause simulator sickness is critical since machine learning prediction highly depends on the features to learn. Many prior approaches use optical flow [25, 34, 43] to obtain motion features from the video projected on HMDs when estimating simulator sickness. Optical flow provides the overall movement frequency and amplitude from the 2D projected plane [7, 34, 43], but it has limitations in accurately representing 3D motion features inherited from movements in the 3D environment. 3D motion flow has an explicit representation of the x, y, and z axes, while optical flow

*Authors contributed equally. e-mail: junhong.jennifer@gmail.com
†e-mail: taehyun.rhee@vuw.ac.nz

projects those motions onto a 2D image plane, struggling to capture forward-axis motion (changes in depth). The 3D motion flow is also well-suited for rotation and translation decomposition, while the optical flow is not.

In this paper, we present a novel deep neural network capable of interpreting 3D motion and learning to predict user-specific simulator sickness levels when moving through 3D virtual environments in HMDs. We specifically focus on decomposing the 3D motion features into translation and rotation components to improve the prediction of simulator sickness. In addition, our network is able to take into account user profile information, such as personal susceptibility to simulator sickness and demographic information, to improve simulator sickness estimation accuracy further.

To this end, we designed a gamelike VR task with enough interactions and decomposed motion guidance, which allowed us to assess the effects of different motion types on simulator sickness (Figure 1). We conducted user studies to identify correlations between simulator sickness and the decomposed 3D movement in translation and rotation, as well as build a user-labeled 3D motion dataset for machine learning to learn from. Our user study results showed that translation and rotation have different severities of simulator sickness in HMDs. We also conducted further analyses to find correlations between individual user profiles and simulator sickness levels. Based on these findings, we then developed a novel deep neural network architecture to better estimate simulator sickness with decomposed motion and user profile information. Our results show that the deep simulator sickness estimation model based on decomposed 3D motion features and user profiles outperforms methods based on optical flow and undecomposed motion.

Our contributions are summarized as follows:

- We propose a novel deep learning-based simulator sickness estimation solution, introducing decomposed 3D motion features, which proved to outperform traditional 2D motion features.
- We design and perform a user study with variations on decomposed movement, including translation and rotation, to explore the effects that different motion features have on simulator sickness in HMDs. The user study also provides a large 3D motion dataset paired with user-labeled simulator sickness scores for our network to train with.
- Our simulator sickness estimator is able to adapt personalized user profile information (e.g., individual simulator sickness susceptibility, demographic information) to improve simulator sickness estimation accuracy for a given user.

2 Related work

2.1 Movement and simulator sickness

User movement in VR often causes users to experience illusory self-motion, inducing simulator sickness. Hu et al. [14] found that virtual camera movement including translational acceleration and rotational velocity contributes to perceived simulator sickness. Nakamura et al. [39,40] suggest that the motion of the background induces vection while the motion of the foreground objects reduces the vection induced by the background because of the "inverted vection" they generate. Jeong [17] found that violent up and down camera motions, fast background transitions, and unintended camera movements are important factors contributing to simulator sickness.

Some studies have investigated the impact of the speed of the movement on simulator sickness, and mostly report that faster movement will induce more severe simulator sickness [4, 36]. So et al. [48] investigated what effects navigation speed have on the level of simulator sickness in the fore-and-aft axis during and after an exposure time in one virtual environment, and found that navigation speeds significantly impacted the onset times of simulator sickness. Some studies have investigated movement frequencies and amplitudes and their effects on simulator sickness [8, 9, 11, 12, 47] and tried to find the significant motion patterns and ranges that arouse simulator sickness. Some other works [1, 22, 29, 37] investigate the relative importance of different rotational axes on simulator sickness and the dual- and tri-axis combination effects. They demonstrated that the presence of rotational motions in the scene in any of the roll, pitch, and yaw axes consistently cause simulator sickness, and that dual-axis rotational movement will evoke greater feelings of nausea than single axis, but found no significant difference when comparing dual-axis and tri-axis rotations.

Although prior user studies have investigated either translation or rotation movement, no work has done a comparative study between these two different movement patterns. In our work, we try to decompose the rotation and translation in the user study to research the different effects they may have on simulator sickness.

2.2 Data Collection and Simulator Sickness Labelling

Some prior work used stereo 360° videos selected from YouTube as a raw stimulus [34, 43]. Padmanaban et al. [43] collected a set of 19 one-minute 360° videos from 96 participants. During simulator sickness labeling, the participant's head motion was constrained using a headrest to ensure that all users saw the same scene. So, strictly speaking, their data is not 360°, but regular video instead, although with a very wide field of view. In addition, passive video-watching made it quite different from normal VR usage. Jin et al. [19] focused on VR gameplay, which has more frequent viewpoint changes and interactions. They recorded the eye-screen video displayed in HMD. Head motion and some body movement (while sitting) were allowed during the experiments. Although their stimulus was a VR game, the simulator sickness estimation was still based on the 2D recorded video. Also, an off-the-shelf VR game lacks the ability to control movement styles.

Here, we designed a game-like VR environment and added various strategies of player control and movement guidance. Noteworthy, we decomposed different movement styles in our game-task design to study their difference in contributions to simulator sickness.

For simulator sickness labeling, Padmanaban et al. [43] asked users to fill out the Kennedy Simulator Sickness Questionnaire (SSQ) and the simulator sickness susceptibility questionnaire short-form (MSSQ-short) after watching the video as the measurement of simulator sickness. Porcino et al. [44] used a Virtual Reality Sickness Questionnaire (VRSQ) both before and after the user study to gauge the participant's simulator sickness. Participants played VR games for five minutes whilst rating when there was a simulator-sickness level change. The simulator-sickness level changes would be used to label the data, with the VRSQ used to validate any inconsistencies. Jin et al. [19] measured the simulator sickness in a "ranking-rating" (RR) measure that combined the relative comparisons (rankings) and ratings (based on an 11-point scale). The MSSQ was also filled out for each test as a measurement of individual features. The main limitation of these approaches is that they either break continuity or delay the reporting of momentary feelings during the game, which will impact the validity of the simulator sickness feedback. To fill this gap, recent work by Islam [15] started to use a Fast Motion Sickness Scale (FMS), a verbal rating scale (ranging from 0-20) method to capture user data during exposure quickly.

In our user study, we capture users' ratings based on a 0-10 points scale as simulator sickness labels. Unlike the verbal FMS, we follow the improved work by McHugh et al. [20, 38] to integrate a physical dial interface to periodically collect users' ratings of simulator sickness. This allows continuous measurement of the user's momentary simulator sickness feelings during VR immersion, while not breaking presence, and thus achieves more practical and precise user study data. We also take into account the users' subjective susceptibility and as well as their Visually Induced Motion Sickness Susceptibility

Questionnaire (VIMSSQ) [24] score, which has been proven to be a useful tool for measuring simulator sickness sensitivity [13, 23].

2.3 Learning Based Simulator Sickness Estimation

Recently, more and more machine learning-based methods have emerged due to the advancement of artificial intelligence. Some methods focused on visually-induced simulator sickness predictions [25–27, 34, 43] while others investigate physiological signals, including postural sway, gait motion, heart rate, breathing rate, galvanic skin response, and electroencephalogram (EEG) data [10, 16, 18, 30, 32, 35, 51], or the combination of visual content information and physiological signals [28, 31, 33]. For visual-based machine learning methods, gameplay video will always be analyzed first to extract the raw features of depth and optical flow, and then input them into the deployed machine learning method to regress simulator sickness level or classify simulator sickness arousal. Padmanaban et al. [43] trained a bagged decision tree model on hand-crafted features (quantifying speed, direction, and depth) from video content. Their model generally outperformed a naïve estimate, but was limited by the size of the dataset. Lee et al. [34] improved the neural network structure with a 3D CNN followed by several fully connected layers. And Du et al. [7] made further improvements by introducing an attention mechanism to realize the adaptive fusion of different inputs. Jin et al. [19] used long-short term memory recurrent neural networks (LSTM-RNN) models with many visual-based features and head movement. Instead of predicting a general simulator sickness score, some works [26, 31] have tried to advance the machine learning model to assess more specific symptoms (such as nausea, disorientation, and oculomotor), using either visual content [26] or a combination of visual content and physiological information [31].

In this work, we also focus on visual-induced simulator sickness estimation. But unlike previous work studying 2D motion or general appearance features, we focus on investigating more effective 3D motion features, studying different movement and their effect on predicting simulator sickness.

3 User Study

We had two main goals for the design of our user study. One goal was to encourage two types of motion, translation and rotation, to facilitate simulator sickness induced by these movement components. Another goal was to construct a large and varied user-labeled dataset for the machine learning model to train with. We designed a game-like environment that mimicked common choices game developers make in terms of player control, and allowed players to play using various strategies, so that the data set would cover a large range of play styles. Participants were presented with two alternating tasks that were triggered automatically during each trial:

Task 1 - Coin Collecting (translational focus).
Task 2 - Paintball Shooting (rotational focus).

Note that we will interchangeably use the terms Task 1/Coin Collecting and Task 2/Paintball Shooting. Each task was designed to focus on one motion type: translation or rotation. For translation, participants moved along a fixed path collecting coins laid out along the path with an emphasis on moving forward with occasional turns (Figure 2, top right). For rotation, the participant would be interrupted during the trial to perform a point-and-shoot task at a fixed location. While Task 1 is designed to focus on translation, we opted not to lock the user's head rotation to avoid undesirable vection and to also better mimic common VR experiences. Since Task 1 can include some rotation, we introduce independent variables within Task 1 to help analyze the differences caused by translation and rotation. Task 2 is then compared against Task 1 to gain further insight between these two motion types.

In total, our user study had 124 anonymized participants (70 males, 48 females, four others, and two chose not to say). The mean age was 24.2 years with a standard deviation of 6.62. There were 51 participants who had no previous VR experience, 68 used VR a few times per year, 4 used VR weekly, and 1 used VR daily.

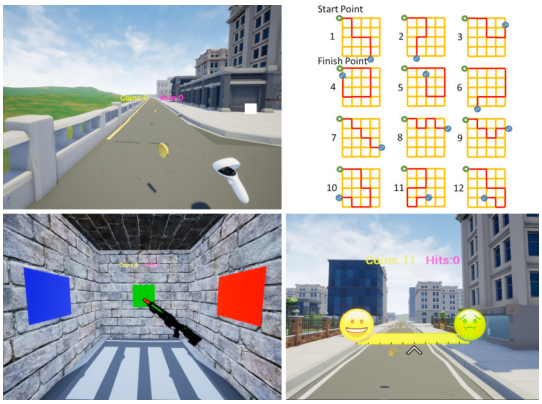

Figure 2: Task 1: Coin collecting (top left). Task 2: Paintball shooting (bottom left). The coins in Task 1 were placed in different patterns (top right). The participant used a Microsoft Surface dial to select their level of simulator sickness score (bottom right).

3.1 Conditions and Data collection

The study used a 2×2 design. The first independent variable was Movement Speed, consisting of two levels: Walking (1.7m/s) and Running (2.8m/s). The second independent variable was Number of Turns, again with two levels: Low (3 turns) and High (6 turns). We will refer to the combination of the variables as conditions WL, WH, RL and RH (where W, R, L and H refer to Walking, Running, Low number of turns, and High number of turns, respectively). To add more variability to the data, we made three different paths for each of the four variations of Movement-Speed × Number-of-Turns combinations. This resulted in a total of 12 possible paths. These conditions vary for Task 1 (Coin Collecting), while Task 2 (Paintball Shooting) does not have any independent variables. The dependent variables are described later, but include the user simulator sickness score, objective game performance, and other subjective self-reporting items.

We considered several threats to the validity of our study. Since we were studying simulator sickness, it was reasonable to assume the longer someone participated, the worse their symptoms may become. In order to limit this accumulation effect, we exposed each participant to only four paths (Latin Square), rather than all 12. Our reasoning is that by capturing data from enough participants (in our case, 124), we would be able to satisfactorily cover all conditions, while still collecting reliable data. We conducted an incomplete within-subjects design with a Latin square counterbalancing technique. For the first set of 12 paths, the first participant was assigned paths 1, 2, 3, and 4, and this same sequence was applied to the remaining three participants. For the second set of the same 12 paths, the first participant was assigned paths 2, 3, 4, and 5, while the last participant was assigned paths 10, 11, 12, and 1. This sequence was continued for the remaining participants. The counterbalancing and optional self-reported scores will slightly alter the degrees of freedom in our statistical analysis.

The participant sat in a swivel chair for the entire session and had 10 minutes to rest between sessions, undergoing a total of four sessions. We collected demographic information (asking for age, gender, VR experience, and normal or corrected-to-normal vision), simulator sickness susceptibility (with the VIMSSQ questionnaire), the simulator sickness scores during the trial, and their performance data (Collected Coins and Correct Color Hits).

3.2 Game Environment and Tasks

The game was developed with Unreal Engine 4.26, running at approximately 90 FPS. The map was 200m $\times$ 200m in size and used an urban-style landscape to minimize the impact on vection [46] (see supplementary materials). We employed a heads-up display (HUD) at 86cm from the participant [41], which included in-game progress. We used free sound for the audio content of the game and more digital assets from our in-house 3D artist.

We used the Meta Quest HMD for participants to experience VR with a Quest Link cable to connect with a desktop computer (Intel® Core™ i7-12700F, 32GB RAM, and RTX 3060Ti). The participants sat on swivel chairs with armrests for placing a Microsoft Surface Dial [20, 38] for reporting their simulator sickness.

3.2.1 Task 1: Coin Collecting

The participants needed to collect as many coins as they could (Figure 2, top left). The number of collected coins was updated on the HUD and saved to a file. The participant navigated around in "Tank mode," which allowed them to move forward while also being free to look around [41]. The participant used a Quest controller in their dominant hand to control movement and body rotation. We combined both movement and rotation control onto just one joystick, where any forward-back movement of the stick moved them at a constant speed along the fixed path at either 1.7m/s (W) or 2.8m/s (R). For rotation, we added a "dead zone," so body rotation would only be triggered if the stick was moved at least 60% of the way from center to the left or right. The rotation rate was fixed at 180 degrees-per-second. The paths were manually designed where paths had an appropriate length (based on the character's speed) and the number of turns in mind. We aimed to provide participants with as many straight segments as possible. Coin placement was done to encourage turning (L vs. H) and moving straight. Coins could be "collected" by moving through them. Each segment was 50m in length, with 10 coins per segment. Based on the moving speed, we had seven segments (70 coins) for conditions WL and WH, and 11 segments (110 coins) for RL and RH.

3.2.2 Task 2: Paintball Shooting

When the second task was triggered, movement using the joystick was disabled, but participants were still able to look around and point the paintball gun (the controller). A cabin with four brick walls suddenly appeared and surrounded the participant. On each of the front, left, and right walls, a colored rectangle with a randomized color (red, green, or blue) was displayed (Figure 2, bottom left). The task here was to pick the matching paintball color for the gun to shoot on the rectangles on each wall. The participant used the controller with the A/B (X/Y if left-handed) buttons to switch the color of the ball, as shown on an indicator on the barrel of the paintball gun. Then, the participant aimed the controller and fired at the rectangle using the trigger button.

If the color of the paintball and target matched, a correct hit was counted, the score was updated on the HUD, and the data was also recorded for later analysis. In the design of the environment, we chose to use simple textures to induce a sense of rotation while reducing the risk of simulator sickness that more complex textures may cause. We also used different sounds for the gunfire, indicating whether it was a correct hit or not, and provided haptic feedback through the controller when the participant pulled the trigger. When a target was hit, it turned to a default color of grey. After all three targets were properly hit, a new round with a random color placement was initialized. The participant continued this task for 1 minute before coming back to the Coin Collecting task. While in the Paintball shooting task, we encouraged the participant to keep their body still and only turn their head to the target on the left and right-hand sides. This task was identical for all the conditions.

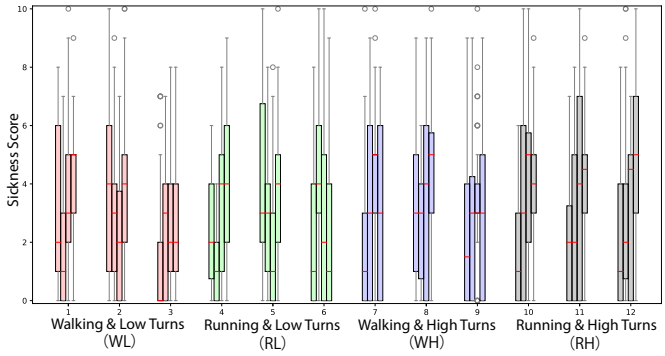

Figure 3: Boxplots of the user study results for each of the 12 paths during Task 1 (Coin Collecting), color-coded for each of the four conditions. Each path is further broken down into four sub-boxplots, representing each session the path was taken in.

3.2.3 Reporting User Simulator Sickness

A prompt reading "Please give your sickness a rating between 0-10" every minute for the participant to report their momentary simulator sickness level using a Microsoft Surface Dial (Figure 2, bottom right). The reporting interface had 11 levels (0 = comfortable, 10 = not comfortable), two emoji icons (happy and vomiting), and a ruler with marked ticks. At the bottom of the meter, there was a movable arrow for referencing/indicating a simulator sickness level. The interface initially started with a default value of 5. During a trial, the reporting interface was hidden and showed up with the last submitted score. The meter stayed visible for 7 seconds and faded out. The dial was located on the non-dominant hand's armrest. The participant rotated it to move the arrow along the meter for selecting a level and clicked (pushed) the dial to submit their score. Both the score and a time stamp were saved as subjective simulator sickness score data. The Dial provided haptic feedback for the participant's interactive actions. There was another option for reporting their momentary simulator sickness level. At any point, the participant could click (press) the Dial to call up the reporting interface. When answering and in the self-report mode, the participant also stopped moving and was released after completing the report.

3.3 Experimental Procedure and Flow

Experimental Procedure. We followed national COVID-19 guidelines when the experiment was conducted. The participant answered a demographic questionnaire and a VIMSSQ first. Afterward, the experimenter helped the participant put on their HMD, gave them the controller, and reminded the person of the dial before starting.

Trial Flow. In each trial, the participant had six minutes of VR gameplay. In order to provide a more realistic game scenario, we had participants start out by moving to a location, performing a task, then moving to another location, and performing another task. This is similar to the behaviour required in many games, where travel and other interactions are carried out. In our trial, participants started by doing Task 1 (Coin Collecting) for one minute, then reported their simulator sickness rating. After this, the participant was locked (could not perform translational movement using the joystick but was still able to look around) in the cabin to perform Task 2 (Paintball Shooting) for another minute. At the end of Task 2, the participant did another simulator sickness rating based on their recent experience before being released to continue with the next task (Coin Collecting). The participant could also invoke self-reporting at any point. The sequence of Task 1$\rightarrow$Simulator Sickness Reporting$\rightarrow$Task 2$\rightarrow$Simulator Sickness Reporting was repeated twice more to make a six-minute VR gameplay. At the end of the session, the game displayed a "Game Over" message and then

allowed users to take a 10-minute break. The amount of break time was empirically determined and flexible depending on the participant's request. After the break, the participant put the HMD back on and held the controller to get ready to start their next session. This sequence of Session→Break was repeated three more times, for a total of four sessions. Our surface dial kept participants engaged for 6-minute sessions, with a total exposure time of 24 minutes for participant safety. Note that previous work [7, 19] used 1-minute exposure times.

3.4 User Study Motion Analysis

From the user study, we were able to analyze the user-labeled simulator sickness scores with respect to the underlying 3D motion dataset. Additional analysis, including task performance, is provided in the supplementary. The simulator sickness scores for each path are shown in Figure 3, and each individual score is shown in the supplementary. The following results that include statistical analysis were tested for data normality using the D'Agostino-Pearson Test, which all indicated a positive result unless specified otherwise.

3.4.1 Translation and Rotation Comparison

We directly compared Task 1 (Coin Collecting) and Task 2 (Paintball Shooting) to understand the influence of translation and rotation motion data on simulator sickness. Note that the conditions were varied during Task 1 to introduce variability into the experiment. Therefore we will show results focusing on the conditions with respect to Task 1. However, we will also show the results of the conditions for both tasks as a point of comparison for Task 1.

The simulator sickness scores for each task, split into each condition, are visualized in Figure 4. While the conditions primarily affect Task 1, we still show the Task 2 results alongside it in case there is some carry-over influence between tasks. We can observe that both tasks cause a range of simulator sickness values, while Task 1 (translation) skews higher than Task 2 (rotation). A paired samples t-test between the two tasks under each condition indicated there was a statistically significant difference in WL, RL, WH, and no statistical significance in RH, the results are shown in Table 1. In general, translation-focused motion patterns tend to cause higher rates of simulator sickness than rotation-focused motion patterns, with variance in its significance among users.

Table 1: Comparing the two tasks under each condition (p-values in bold indicate statistical significance with Bonferroni correction).

		μ	σ	t	p	df
WL	Task 1	3.28	2.56	2.83	**0.0046**	244
	Task 2	2.41	2.23			
WH	Task 1	3.56	2.58	2.79	**0.0052**	250
	Task 2	2.69	2.31			
RL	Task 1	3.22	2.67	2.58	**0.0100**	246
	Task 2	2.42	2.16			
RH	Task 1	3.58	2.81	1.78	0.0750	244
	Task 2	2.97	2.53			

3.4.2 Evaluating Translation and Rotation Variation

We analyze the effect of varying translation and varying rotation separately. We analyze this under the four conditions (WL, WH, RL, RH) with respect to Task 1 (Coin Collecting), since the variables were strongly linked to this task. We ran a paired t-test between each pair and show the result in Table 2. We found no statistical significance in movement speed and number of turns.

3.4.3 User Profile

We looked at the self-reported profile, including the participants' VIMSSQ score and gender, to see if there was any correlation with simulator sickness using the Pearson correlation coefficient. Age and VR experience were also considered, but our sample size for each was not varied enough. For each user, we used their mean simulator sickness score in this analysis.

Table 2: Comparing pairs of conditions for Task 1 (p-values in bold indicate statistical significance with Bonferroni correction).

	μ	σ	t	p	df
WL	3.28	2.56	0.22	0.8275	244
RL	3.21	2.68			
WH	3.54	2.56	-0.12	0.9061	244
RH	3.58	2.81			
WL	3.28	2.56	-0.77	0.4419	244
WH	3.54	2.56			
RL	3.21	2.68	-1.04	0.2988	244
RH	3.58	2.81			

For simulator sickness and VIMSSQ, there was a very weak positive correlation in both Task 1 $[r(124) = 0.13, p = 0.153]$ and Task 2 $[r(124) = 0.19, p = 0.037]$, where Task 2 has statistical significance. We used a paired t-test between females and males regarding their simulator sickness scores; in Task 1, we found statistical significance between females ($\mu = 4.33; \sigma = 2.83$) and males ($\mu = 2.80; \sigma = 2.31$); $[t(382) = 5.79847, p < 0.001]$, and in Task 2, we also found statistical significance between females ($\mu = 3.30; \sigma = 2.55$) and males ($\mu = 2.17; \sigma = 2.03$); $[t(382) = 4.77523, p < 0.001]$.

3.4.4 Analysis Discussion

While coin collecting focused on the translational effects of simulator sickness, it still also contained rotations from both the HMD (active rotation) and controller (passive rotation) to include variability in the data and emulate typical VR experiences. Paintball shooting acts as a point of comparison with coin collecting, helping to identify the simulator sickness effects caused by translation, where we found statistical differences (Table 1). While collecting coins, participants tended to experience vection as their body was physically fixed in place versus having visual cues of translation in the virtual space. Regarding paintball shooting, their head rotation corresponded to what they saw. Thus, participants may have felt more comfortable during Paintball Shooting. The independent passive rotation variables (low, high) isolate their effects from translation. We found that there was no significant difference between moving speeds (Table 2), so the simulator sickness effects in coin collecting were most likely caused by the translation.

In this study, participants repeatedly performed Task 1 alternating with Task 2, always starting with Task 1. Building up sickness is a typical effect of VR, so it may be expected that earlier tasks can increase sickness levels in later tasks. However, we see that Task 1 always has higher average sickness scores than Task 2 (Table 1). Future work could explore passive rotation tasks to act as a baseline, similar to the paintball shooting task. Higher rotation speeds can also be investigated. Similarly, no statistical difference was found in the number of turns users took. Using the surface dial, we can keep participants engaged for four 6-minute sessions. While this is an improvement on prior work [7, 19], future work can consider adapting the game exposure time to be more comparable with typical VR gaming experiences.

We found that there are some slight differences between individual profiles and demographics. There is a weak positive relationship between the self-reported VIMSSQ scores and their sickness scores in both tasks. We also found that females reported higher sickness compared to males, which was more pronounced in Task 1. We did not have enough diversity in our samples for age or VR experience to make any conclusive results. We also asked about normal or corrected-to-normal vision, with 100% affirmation.

Based on the analysis, the machine learning model could benefit from separating the translational and rotational components as they each impact simulator sickness with varying severities. User profiles showed minor impact but with statistical significance.

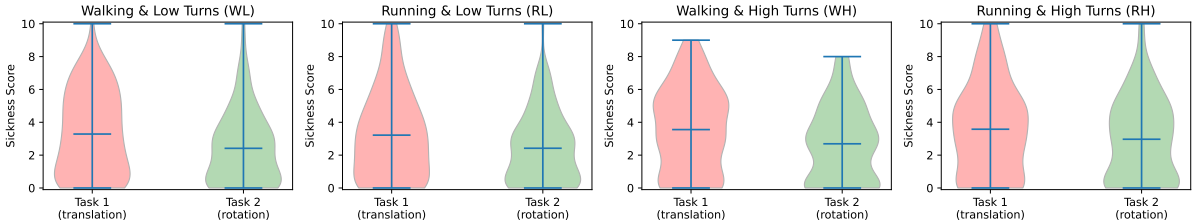

Figure 4: Violin plots visualizing the distribution of user simulator sickness scores for each condition, comparing both tasks - Coin Collecting and Paintball Shooting - which emphasize translation and rotation motion, respectively.

4 Simulator Sickness Estimation

We developed our simulator sickness estimator using a deep neural network based on the 3D motion dataset collected from our user study. Our aim was to explore 3D motion in VR that intrinsically existed within the 3D environment, and its relationship with simulator sickness. We train the network to learn the complex mapping between the 3D motion features and the user's simulator sickness level. The pipeline of our proposed method is shown in Figure 5. We extracted decomposed motion flow, including the translational velocity/acceleration and the rotational velocity, with the depth to provide distance information. We fed these pixel-wise motion and depth maps into the network as the input. Moreover, individual demographic details and VIMSSQ scores serve as supplementary inputs for characterizing user profiles. The projected outcome is a personalized simulator sickness score prediction.

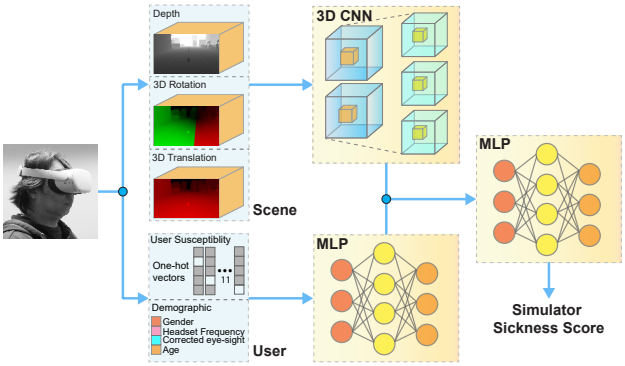

Figure 5: Neural network structure for simulator sickness estimation.

4.1 3D Motion Flow Extraction

We developed our 3D motion flow extraction functionality within a game engine (Unreal Engine) so that we have full access to the 3D scene and intrinsic motion data. We extracted different motion features from the simulation for the input of our network, including (1) translational velocity and acceleration with respect to the camera, and (2) rotational velocity (See Figure 6). In addition to the motion data, we extract the depth map of the scene. The data is extracted and saved into high dynamic range images. We refer to the images from (1) and (2) as motion maps.

Gameplay Recording: Extracting the motion data while the user is playing the game during the user study runs the risk of reducing the framerate. This is critical to avoid due to simulator sickness induced by a low framerate. As such, we opted to first record the user's playthrough (running at approximately 90 FPS) within the game engine, allowing us to replay their session offline for data extraction. Recording their playthrough has the added benefit for researchers to replay, inspect, and analyse individual playthroughs. To record the gameplay, the cartesian positions and rotations of the virtual character, dynamic objects, and the camera (x, y, z, yaw, $pitch$, and $roll$) were recorded for each participant during the user study.

Shader Processing: The recorded gameplay from the user study was replayed to generate pixel-level motion maps for (1) translational velocity and acceleration with respect to the camera, (2) rotational velocity, (3) depth. Before playback of the recorded gameplay, the default shader of all scene objects was changed to our motion-flow shader. The shader extracted pixel-level 3D motion features based on the camera's viewport, which were then saved as high dynamic range images. These images are the input to the network.

Motion flow - Translation: The translation velocity was calculated based on the position (translational) difference of an object relative to the camera from the previous frame (see supplementary for equation, where T is the unnormalized direction vector from the camera to a point on the object and t is the frame number). The translational velocity was decomposed into (x,y,z). The transitional acceleration was calculated as the first order difference of translational velocity.

Motion flow - Rotation: The rotational velocity was calculated based on the angular difference of an object relative to the camera from the previous frame, as shown in Equation **??**, where N is normalized direction vectors from the camera to a point on an object. We are able to track the position of each vertex from one frame to the next, enabling us to measure the velocity of the vertex. This is evaluated per pixel in a fragment shader. The rotational velocity (see supplementary for schematic and equations) is then decomposed into ($\gamma(yaw)$: rotation on the z-axis, $\beta(pitch)$: rotation on the y-axis, $\alpha(roll)$: rotation on the x-axis). Compared to a single aggregated rotational velocity, these decomposed rotational velocities provided an additional analytical dimension to isolate contributors to simulator sickness arising from the individual axial rotations.

Depth: The depth maps were calculated based on the shortest Euclidean distance of every visible rendered pixel of an object relative to the camera for every frame t.

4.2 Neural Network for Simulator Sickness Estimation

In this work, we used a 3D CNN network module to model the motion maps to obtain compact motion representations. Different from previous work (e.g., [7, 34]) that used optical flow, we used motion flow with decomposition to provide a more distinguishable motion interpretation of the scene. Since it is infeasible to conduct the simulator sickness estimation without considering a user's individual characteristics, we also treat the users' susceptibility as one condition of the simulator sickness estimation. The user's demographic information and past experience on digital devices (collected by VIMSSQ) were encoded in a one-hot vector and were projected into compact latent representations by a multi-layer perceptron (MLP) module. The depth, translation, rotation, and the users' susceptibility representations were then concatenated and fed into a fusion MLP module to regress the simulator sickness score, allowing us to obtain user-specific estimated simulator sickness scores. We didn't use RGB images as one input for simulator sickness estimation as the prior work had done [7, 34] because motion attributes are a more dominant factor for simulator sickness than general appearance information. Focusing on motion features can alleviate the data

demand for network training, which is critical when labelled data is challenging to collect.

Details of the network setups are listed in the supplementary materials. The 3D CNN module for the motion representation extraction had an input size of (171, 90, 160) (temporal length, height, width). It has four submodules stacked with each other, each including a 3D convolution layer, a batch normalization, and a ReLU non-linear activation function followed by a max-pooling layer. The max-pooling layers had a kernel size of (3, 3, 3) and stride of (2, 2, 2). The latent space had 32 channels for depth and 64 for rotation and translation, with feature maps of the size (6, 2, 2). The latent vector was flattened before feeding into the following fusion MLP module.

The MLP module for the user susceptibility representation extraction had an input dimension of 341 (275 for VIMSSQ and 66 for demography), and consisted of two linear fully-connected layers followed by a ReLU activation function. The fusion MLP module had an input dimension of $3,904$ ($1,280$ for depth, rotation, and translation latent space and 64 for user susceptibility). It was composed of three fully-connected layers, of which a ReLU projection followed the first two linear layers, and a sigmoid activation function followed the last linear layer as the output.

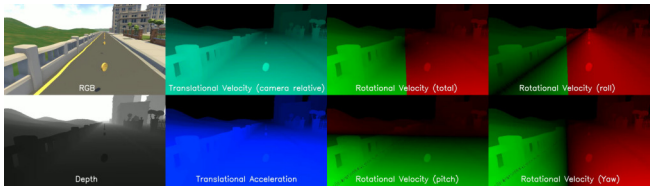

Figure 6: The extracted motion data, including depth, translational and rotational velocities (normalized for display). In the rotational velocity images, the GREEN and RED show positive and negative angular values, respectively. Here, the camera was moving forward.

4.3 Data Processing and Augmentation

From the user study, we collected a total of $2,976$ minutes (124 participants, 24 minutes per person) of recorded gameplay and simulator sickness labeling with sufficient data variations regarding motion patterns (see supplementary Table). We treated each 1-minute clip as one data sample for model training. The simulator sickness score at each minute was chosen as the ground truth label. If there were voluntary reports of simulator sickness scores (in addition to that collected at every minute) that were larger, the maximum score was chosen as the ground truth instead. The gameplay was downsampled to 3fps, yielding 180 frames per sample (3fps x 60 seconds). Before feeding the motion maps into the network, we applied max-min scaling to each scene feature to normalize the data into $[0,1]$. For the user's simulator sickness labels, we re-scaled them into $[0,1]$ by dividing the score by 10 and then centralizing them into $[-1,1]$.

Data augmentation was applied to enlarge the training dataset. We followed prior work [7, 34] to perturb the dataset with a frameshift. We assumed the simulator sickness level remained unchanged when there was just a subtle time shift in the visual content (e.g., 1 out of 60 seconds). In our implementation, we shift each video clip three times for 1 second each (augmented 4 times).

4.4 Loss functions and Training Process

The Mean Square Error (MSE) loss functions $\mathscr{L}$ were applied to optimize the network parameter by minimizing the distance between the predicted simulator sickness score and user-labeled ground truth score S_{gt}. To address model overfitting and help with feature selection, the $L2$ and $L1$ regularization are used as the penalty terms on all trainable weights (β). λ_{L1} was chosen 0.0003 and λ_{L2} was chosen 0.003 in our experiments. We used Adam Optimizer to train the model, with a learning rate of 0.0001 and batch size of 16. An early stopping strategy was applied to stop the training procedure when the loss on the validation dataset started to increase.

$$\mathscr{L} = \sum_{n=1}^{N} (S_{gt} - S_{pred})^2 + \lambda_{L2} \sum_{j=1}^{p} (\beta_{(}j))^2 + \lambda_{L1} \sum_{j=1}^{p} |\beta_{(}j)| \quad (1)$$

5 Evaluation

Dataset: We randomly chose the data from 30 participants as testing data, and another 94 participants' data as training data, bringing the total of 720 testing samples and 9,024 training samples after data augmentation (94 participants x 24 samples x 4 times augmentation).

Metrics: We do both regression and binary classification of simulator sickness levels in our experiments. We used two metrics to measure the model performance. For regression, we used MSE and output a range of $0-10$ as the simulator sickness score. In addition, since in some scenarios people just want to know if there was an occurrence of simulator sickness, we also show the accuracy of the binary classification as another metric. We applied a threshold to the ground truth and predicted simulator sickness scores (6 for ground truth and 4.5 for predictions in our implementation) to quantify them into binary sick/not-sick categories and calculate the classification accuracy. The threshold was chosen based on the performance of validation data, and then applied as a hyperparameter to other cases.

Runtime: All the tests were done on one GeForce RTX A6000 card with a runtime of 0.0167s (average across 100 samples).

5.1 Optical Flow vs. Motion Flow

Previous work [7, 34, 43] used optical flow extracted from 2D videos as motion features. For comparison purposes, we extracted optical flow from our dataset and compared it with our motion flow. We first simulated the RGB data from our replay recordings and used the state-of-the-art RAFT [49] method to extract optical flow, which works in consecutive frames. Following the RAFT implementation, we store the optical flow vectors in an RGB representation, where the displacement (large or small) scales the saturation of the RGB values. This RGB mapping is then fed into the neural network. Besides, we used the ground-truth depth to go along with it as part of the input to obtain the final estimation. It's noteworthy that RAFT has been reported to be more accurate for optical flow extraction than FlowNet [6] that the prior works had used. The ground-truth depth extracted from the engine is also more accurate than the estimated depth based on RGB images that previous work used.

We swapped out optical flow with motion flow, including the translation velocity and acceleration maps (divided by depth, together denoted as "Motion flow - Translation" in Table 3), and the rotation velocity maps (divided by depth, denoted as "Motion flow - Rotation"), to see if the motion flow features achieved more accurate predictions. Since our user study analysis indicated decomposing rotation from translation could be beneficial, we also experimented with and without rotation features.

Our results on testing dataset show that motion flow performed better than optical flow when predicting simulator sickness, improving the MSE from 8.50 to 6.50 and the binary classification from 77.82% to 80.65%. Adding rotation features further improves the estimation performance. Compared with optical flow extracted from rendered 2D images, motion flow extracted from the original 3D scene has a better capability of encoding motion cues in the experienced 3D space. Additionally, including more specific decomposed rotational motion features allows the neural network better to learn the relationship between motion and simulator sickness. Similar conclusions can be obtained on 5-fold cross-validation (See supplementary Table.).

Table 3: Optical flow and motion flow comparison.

Input feature	MSE↓	Binary classification (Acc (%)) ↑
Optical flow	8.50	77.82
Motion Flow - Translation	6.50	80.65
Motion flow - Translation & Rotation	**6.08**	**84.07**

5.2 Ablation Study

In order to see how the model performs on different motion patterns, we evaluated our model on data from Task 1 and Task 2 separately, allowing us to see how our model performs on translation-dominant and rotation-dominant motions.

Using 360 samples for each condition, the results are shown in Table 4. This shows that simulator sickness prediction for rotational motion data (Task 2) is more reliable than translational motion data (Task 1), with a lower MSE and higher classification accuracy for all input feature conditions. One possible reason is that the translation-focused motion data from Task 1 has a higher variation of simulator sickness levels than rotation-focused motion data from Task 2 (see Figure 4 and Table 1), increasing the prediction's uncertainty. While Task 1 is focused on translation, it still contains a few occasional rotations. This increases complexity compared with Task 2, thus more challenging for machine learning to interpret.

Compared with optical flow, motion flow can improve the simulator sickness prediction performance on both tasks. Adding rotation features also aids not only rotation-induced but also translation-induced sickness predictions This is possibly because the neural network can better discriminate different types of motions.

Table 4: Ablation evaluation for translation and rotation data.

Input Feature	Testing data	MSE ↓	Binary classification (accuracy (%)) ↑
Optical flow	Task 1 (translation)	9.57	74.19
	Task 2 (rotation)	7.47	81.45
Motion flow - Translation	Task 1 (translation)	7.96	70.50
	Task 2 (rotation)	5.03	91.90
Motion flow - Translation & Rotation	Task 1 (translation)	**6.35**	**75.81**
	Task 2 (rotation)	**5.81**	**92.30**

5.3 Impact of User Profiles

To investigate the impact of users' profiles on simulator sickness prediction, we experimented with switching ON and OFF the user susceptibility features on top of the motion flow features with rotation. We chose different lengths of the latent feature vector (32 & 64) to evaluate how compactness impacts the final prediction accuracy. As shown in Table 5, adding user-profile features can generally improve the performance of simulator sickness estimation, although not as significant as motion decomposition. It aligns with the analysis results 3.4.3 in which only minor correlations were found between user variables and reported simulator sickness. The performance is also partially limited by the number of participants we can have. Providing user profile information can improve rotation-induced simulator sickness prediction by a larger margin in both 32- and 64-dimensional cases. In terms of latent vector dimension, the model slightly prefers a more compact latent feature space, which may be due to the training dataset size limitation.

Table 5: Simulator sickness estimation with and without user susceptibility feature ("User") in 32 and 64 dimensions.

Input Feature	MSE ↓	MSE (Task 1)↓	MSE (Task 2)↓	Binary classification (Acc (%)) ↑
(A) Motion flow	6.08	**6.35**	5.81	84.07
(A) + User$_{fea,32}$	**5.67**	6.77	4.52	**86.49**
(A) +User$_{fea,64}$	5.77	7.04	**4.50**	85.08

5.4 Comparison with prior work

We compare our method with state-of-the-art deep learning-based simulator sickness estimation methods. For this, we chose the work by Lee et al. [34], and Du et al. [7](each denoted as [Lee et.al 19] and [Du et.al. 23]). They primarily use optical flow, in addition to disparity and saliency maps as the input, and use a 3DCNN [34] or an enhanced version using attention [7] to model their relationship with the simulator sickness. We re-implemented the two methods and apply the model to our dataset. From the results shown in Table 6 - our method with motion flow features including both rotation and translation, together with another branch of MLP for profile information aggregation, performed better in both MSE and binary classification accuracy metrics.

Table 6: Comparisons with other works.

Prior Works	Input feature	Network	MSE↓	Binary classification (Acc (%)) ↑
[Lee et.al 19] [34]	Optical flow Disparity map Saliency map	3DCNN	8.67	78.31
[Du et.al. 23] [7]	Optical flow Disparity map Saliency map	3DCNN + attention	7.72	79.94
Ours	3D Motion flow Depth map User Profiles	3DCNN + MLP fusion	**5.67**	**86.49**

6 Discussion

From our user study, we were able to collect a diverse 3D motion dataset paired with simulator sickness labels that our deep learning model was able to train on. In our analysis, we were able to observe that translational and rotational movement cause simulator sickness with different severities. Correspondingly, we were able to observe a noticeable increase in simulator sickness estimation accuracy when decomposing translation and rotation features into the deep neural network. User profiles showed a slight correlation with simulator sickness, and we observed a proportional increase in the model's accuracy when feeding the user profile as part of the input feature.

Our user study mainly considered users' movement through a scene. Although some dynamic objects were included, they were not the main focus of this work. Future research could study object-level movement and its impact on simulator sickness, including object size, movement, and placement. Texture, colour, and dynamic lighting could be another interesting research direction. While we introduced variability into our experiments with conditions on Task 1, future work could consider extending this with conditions on Task 2, introducing variability on rotation. Our current system provides a good baseline for such research.

7 Conclusions

We have presented a deep simulator sickness estimation method to estimate better the motion-induced simulator sickness level of a user wearing an HMD. We use 3D motion flow extracted from the original VR scene rather than optical flow extracted from 2D images. We conducted a user study to identify the impact of different movement styles on simulator sickness. Guided by findings from the user study, we trained our deep model. The results showed the benefits of using decomposed 3D motion for simulator sickness estimation. Our estimator can provide VR developers and players with a better tool to measure the likelihood of simulator sickness in VR applications.

Acknowledgments

This work was supported by the Entrepreneurial University Programme from the TEC.

REFERENCES

[1] F. Bonato, A. Bubka, and S. Palmisano. Combined pitch and roll and cybersickness in a virtual environment. *Aviation, space, and environmental medicine*, 80(11):941–945, 2009.

[2] P. Budhiraja, M. R. Miller, A. K. Modi, and D. Forsyth. Rotation blurring: use of artificial blurring to reduce cybersickness in virtual reality first person shooters. *arXiv preprint arXiv:1710.02599*, 2017.

[3] E. Chang, H. T. Kim, and B. Yoo. Virtual reality sickness: a review of causes and measurements. *International Journal of Human–Computer Interaction*, 36(17):1658–1682, 2020.

[4] J.-R. Chardonnet, M. A. Mirzaei, and F. Merienne. Visually induced motion sickness estimation and prediction in virtual reality using frequency components analysis of postural sway signal. 2015.

[5] S. Davis, K. Nesbitt, and E. Nalivaiko. A systematic review of cybersickness. In *Proceedings of the 2014 conference on interactive entertainment*, pp. 1–9, 2014.

[6] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 2758–2766, 2015.

[7] M. Du, H. Cui, Y. Wang, and H. Duh. Learning from deep stereoscopic attention for simulator sickness prediction. *IEEE Transactions on Visualization and Computer Graphics*, 2021.

[8] H. B.-L. Duh, D. E. Parker, and T. A. Furness. An independent visual background reduced simulator sickness in a driving simulator. *Presence: Teleoperators & Virtual Environments*, 13(5):578–588, 2004.

[9] H.-L. Duh, J. Lin, R. V. Kenyon, D. E. Parker, and T. A. Furness. Effects of field of view on balance in an immersive environment. In *Proceedings IEEE Virtual Reality 2001*, pp. 235–240. IEEE, 2001.

[10] T. Feigl, D. Roth, S. Gradl, M. Wirth, M. E. Latoschik, B. M. Eskofier, M. Philippsen, and C. Mutschler. Sick moves! motion parameters as indicators of simulator sickness. *IEEE Transactions on Visualization and Computer Graphics*, 25(11):3146–3157, 2019. doi: 10.1109/TVCG.2019.2932224

[11] J. F. Golding and M. A. Gresty. Biodynamic hypothesis for the frequency tuning of motion sickness. *Aerospace medicine and human performance*, 87(1):65–68, 2016.

[12] J. F. Golding, A. Mueller, and M. A. Gresty. A motion sickness maximum around the 0.2 hz frequency range of horizontal translational oscillation. *Aviation, space, and environmental medicine*, 72(3):188–192, 2001.

[13] J. F. Golding, A. Rafiq, and B. Keshavarz. Predicting individual susceptibility to visually induced motion sickness by questionnaire. *Frontiers in Virtual Reality*, 2:576871, 2021.

[14] P. Hu, Q. Sun, P. Didyk, L.-Y. Wei, and A. E. Kaufman. Reducing simulator sickness with perceptual camera control. *ACM Transactions on Graphics (TOG)*, 38(6):1–12, 2019.

[15] R. Islam, K. Desai, and J. Quarles. Towards forecasting the onset of cybersickness by fusing physiological, head-tracking and eye-tracking with multimodal deep fusion network. In *2022 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 121–130. IEEE, 2022.

[16] R. Islam, Y. Lee, M. Jaloli, I. Muhammad, D. Zhu, and J. Quarles. Automatic detection of cybersickness from physiological signal in a virtual roller coaster simulation. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pp. 649–650. IEEE, 2020.

[17] D. Jeong, S. Yoo, and J. Yun. Cybersickness analysis with eeg using deep learning algorithms. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 827–835. IEEE, 2019.

[18] D. K. Jeong, S. Yoo, and Y. Jang. Vr sickness measurement with eeg using dnn algorithm. In *Proceedings of the 24th ACM Symposium on Virtual Reality Software and Technology*, pp. 1–2, 2018.

[19] W. Jin, J. Fan, D. Gromala, and P. Pasquier. Automatic prediction of cybersickness for virtual reality games. In *2018 IEEE Games, Entertainment, Media Conference (GEM)*, pp. 1–9. IEEE, 2018.

[20] S. Jung, R. Li, R. McKee, M. C. Whitton, and R. W. Lindeman. Floor-vibration vr: Mitigating cybersickness using whole-body tactile stimuli in highly realistic vehicle driving experiences. *IEEE Transactions on Visualization and Computer Graphics*, 27:2669–2680, 5 2021. doi: 10.1109/TVCG.2021.3067773

[21] A. Kemeny, P. George, F. Mérienne, and F. Colombet. New vr navigation techniques to reduce cybersickness. *Electronic Imaging*, 2017(3):48–53, 2017.

[22] B. Keshavarz and H. Hecht. Axis rotation and visually induced motion sickness: the role of combined roll, pitch, and yaw motion. *Aviation, space, and environmental medicine*, 82(11):1023–1029, 2011.

[23] B. Keshavarz, B. Murovec, N. Mohanathas, and J. F. Golding. The visually induced motion sickness susceptibility questionnaire (vimssq): Estimating individual susceptibility to motion sickness-like symptoms when using visual devices. *Human Factors*, p. 00187208211008687, 2021.

[24] B. Keshavarz, R. Saryazdi, J. L. Campos, and J. F. Golding. Introducing the vimssq: Measuring susceptibility to visually induced motion sickness. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 63, pp. 2267–2271. SAGE Publications Sage CA: Los Angeles, CA, 2019.

[25] H. G. Kim, W. J. Baddar, H.-t. Lim, H. Jeong, and Y. M. Ro. Measurement of exceptional motion in vr video contents for vr sickness assessment using deep convolutional autoencoder. In *Proceedings of the 23rd ACM Symposium on Virtual Reality Software and Technology*, pp. 1–7, 2017.

[26] H. G. Kim, S. Lee, S. Kim, H.-t. Lim, and Y. M. Ro. Towards a better understanding of vr sickness: Physical symptom prediction for vr contents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, pp. 836–844, 2021.

[27] H. G. Kim, H.-T. Lim, S. Lee, and Y. M. Ro. Vrsa net: Vr sickness assessment considering exceptional motion for 360 vr video. *IEEE transactions on image processing*, 28(4):1646–1660, 2018.

[28] J. Kim, W. Kim, H. Oh, S. Lee, and S. Lee. A deep cybersickness predictor based on brain signal analysis for virtual reality contents. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10580–10589, 2019.

[29] J. Kim and T. Park. The onset threshold of cybersickness in constant and accelerating optical flow. *Applied Sciences*, 10(21):7808, 2020.

[30] R. K. Kundu, R. Islam, P. Calyam, and K. A. Hoque. Truvr: Trustworthy cybersickness detection using explainable machine learning. *arXiv preprint arXiv:2209.05257*, 2022.

[31] S. Lee, J. U. Kim, H. G. Kim, S. Kim, and Y. M. Ro. Saca net: Cybersickness assessment of individual viewers for vr content via graph-based symptom relation embedding. In *European Conference on Computer Vision*, pp. 170–186. Springer, 2020.

[32] S. Lee, S. Kim, H. G. Kim, M. S. Kim, S. Yun, B. Jeong, and Y. M. Ro. Physiological fusion net: Quantifying individual vr sickness with content stimulus and physiological response. In *2019 IEEE International Conference on Image Processing (ICIP)*, pp. 440–444. IEEE, 2019.

[33] S. Lee, S. Kim, H. G. Kim, and Y. M. Ro. Assessing individual vr sickness through deep feature fusion of vr video and physiological response. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(5):2895–2907, 2021.

[34] T. M. Lee, J.-C. Yoon, and I.-K. Lee. Motion sickness prediction in stereoscopic videos using 3d convolutional neural networks. *IEEE transactions on visualization and computer graphics*, 25(5):1919–1927, 2019.

[35] C.-Y. Liao, S.-K. Tai, R.-C. Chen, and H. Hendry. Using eeg and deep learning to predict motion sickness under wearing a virtual reality device. *IEEE Access*, 8:126784–126796, 2020.

[36] C.-L. Liu and S.-T. Uang. A study of sickness induced within a 3d virtual store and combated with fuzzy control in the elderly. In *2012 9th International Conference on Fuzzy Systems and Knowledge Discovery*, pp. 334–338. IEEE, 2012.

[37] W. Lo and R. H. So. Cybersickness in the presence of scene rotational movements along different axes. *Applied ergonomics*, 32(1):1–14, 2001.

[38] N. McHugh, S. Jung, S. Hoermann, and R. W. Lindeman. Investigating a physical dial as a measurement tool for cybersickness in virtual reality. Association for Computing Machinery, 11 2019. doi: 10.1145/3359996.3364259

[39] S. Nakamura. Depth separation between foreground and background

on visually induced perception of self-motion. *Perceptual and motor skills*, 102(3):871–877, 2006.

[40] S. Nakamura, S. Palmisano, and J. Kim. Relative visual oscillation can facilitate visually induced self-motion perception. *i-Perception*, 7(4):2041669516661903, 2016.

[41] Oculus. Oculus best practices, 2017.

[42] H. Oh and W. Son. Cybersickness and its severity arising from virtual reality content: A comprehensive study. *Sensors*, 22(4):1314, 2022.

[43] N. Padmanaban, T. Ruban, V. Sitzmann, A. M. Norcia, and G. Wetzstein. Towards a machine-learning approach for sickness prediction in 360 stereoscopic videos. *IEEE transactions on visualization and computer graphics*, 24(4):1594–1603, 2018.

[44] T. Porcino, E. O. Rodrigues, A. Silva, E. Clua, and D. Trevisan. Using the gameplay and user data to predict and identify causes of cybersickness manifestation in virtual reality games. In *2020 IEEE 8th International Conference on Serious Games and Applications for Health (SeGAH)*, pp. 1–8. IEEE, 2020.

[45] L. Rebenitsch and C. Owen. Estimating cybersickness from virtual reality applications. *Virtual Reality*, 25(1):165–174, 2021.

[46] B. E. Riecke, J. Schulte-Pelkum, H. H. Bülthoff, and M. von der Heyde. Cognitive factors can influence self-motion perception (vection) in virtual reality. *ACM Transactions on Applied Perception (TAP)*, 3:194–216, 7 2006. doi: 10.1145/1166087.1166091

[47] H. Shigemasu, T. Morita, N. Matsuzaki, T. Sato, M. Harasawa, and K. Aizawa. Effects of physical display size and amplitude of oscillation on visually induced motion sickness. In *Proceedings of the ACM symposium on Virtual reality software and technology*, pp. 372–375, 2006.

[48] R. H. So, W. Lo, and A. T. Ho. Effects of navigation speed on motion sickness caused by an immersive virtual environment. *Human factors*, 43(3):452–461, 2001.

[49] Z. Teed and J. Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European conference on computer vision*, pp. 402–419. Springer, 2020.

[50] N. Tian, P. Lopes, and R. Boulic. A review of cybersickness in head-mounted displays: raising attention to individual susceptibility. *Virtual Reality*, pp. 1–33, 2022.

[51] Y. Wang, J.-R. Chardonnet, and F. Merienne. Vr sickness prediction for navigation in immersive virtual environments using a deep long short term memory model. In *2019 IEEE conference on virtual reality and 3D user interfaces (VR)*, pp. 1874–1881. IEEE, 2019.