PassengXR: A Low Cost Platform for Any-Car, Multi-User, Motion-Based Passenger XR Experiences

Mark McGill, Graham Wilson, Daniel Medeiros, Stephen Brewster
first.last@glasgow.ac.uk
School of Computing Science, University of Glasgow
Glasgow, UK

Figure 1: *PassengXR* lets practitioners create multi-passenger XR experiences based on the motion of any vehicle, supporting immersive gaming (bottom left), productivity (bottom middle) and collaboration (bottom right).

ABSTRACT

We present *PassengXR*, an open-source toolkit for creating passenger eXtended Reality (XR) experiences in Unity. XR allows travellers to move beyond the physical limitations of in-vehicle displays, rendering immersive virtual content based on - or ignoring - vehicle motion. There are considerable technical challenges to using headsets in moving environments: maintaining the forward bearing of IMU-based headsets; conflicts between optical and inertial tracking of inside-out headsets; obtaining vehicle telemetry; and the high cost of design given the necessity of testing in-car. As a consequence, existing vehicular XR research typically relies on controlled, simple routes to compensate. *PassengXR* is a cost-effective open-source in-car passenger XR solution. We provide a reference set of COTS hardware that enables the broadcasting of vehicle telemetry to multiple headsets. Our software toolkit then provides support to correct vehicle-headset alignment, and then create a variety of passenger XR experiences, including: vehicle-locked content; motion- and location-based content; and co-located multi-passenger applications. *PassengXR* also supports the recording and playback of vehicle telemetry, assisting offline design without resorting to costly in-car testing. Through an evaluation-by-demonstration, we show how our platform can assist practitioners in producing novel, multi-user passenger XR experiences.

CCS CONCEPTS

• **Human-centered computing** → **User interface toolkits**; **Mixed / augmented reality**.

KEYWORDS

Mixed Reality, Extended Reality, In-Car, Vehicle, Toolkit, Passenger

ACM Reference Format:
Mark McGill, Graham Wilson, Daniel Medeiros, Stephen Brewster. 2022. *PassengXR*: A Low Cost Platform for Any-Car, Multi-User, Motion-Based Passenger XR Experiences . In *The 35th Annual ACM Symposium on User Interface Software and Technology (UIST '22), October 29-November 2, 2022, Bend, OR, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3526113.3545657

1 INTRODUCTION

Conducting research into the design of in-vehicle eXtended Reality (XR) interfaces can have high costs (e.g., driving simulators, buying/hiring real cars, expensive sensors) and can be practically challenging: maintaining stable and accurate tracking, developing the necessary functionality, having limited access to real driving

UIST '22, October 29-November 2, 2022, Bend, OR, USA

ACM ISBN 978-1-4503-9320-1/22/10…$15.00
https://doi.org/10.1145/3526113.3545657

data, etc. With the rise of autonomous cars, research is increasingly looking to make use of, or counteract, the real motion and location of the vehicle for the design of in-car interfaces and experiences for passengers who can now use travel time for leisure, entertainment or productivity [13, 15, 17, 18, 52].

In an attempt to make it easier for designers to create or test in-car XR interfaces, research has started developing toolkits or platforms that provide templates, and open-source codebases for detecting and using data about the movement and actions of a vehicle, and a user within it. In these situations, it is crucial to separate the sensing of the Head-Mounted Display (HMD) and vehicle motion, so that car movement does not interfere with the user's agency over their experience, or the immersive effect of head-tracking [33, 35]. Most platforms use an Inertial Measurement Unit (IMU) to track vehicle orientation, combined with sensors attached to the On-Board Diagnostic (OBD-II) port of the vehicle to poll velocity [13–15, 17, 52]. This telemetry can then be used, for example, to match the movement of a virtual vehicle (also called the 'reference frame') to the motion of a real car, while the user's HMD is tracked separately to view the virtual content. Outside of research, Holoride [18] adapts virtual content based on the movement, GPS location and intended travel route of the car, but has limited compatibility with car models, and is only open to commercial partners.

Many of the existing in-vehicle systems are limited by their use of PC VR headsets connected to a laptop, restricting the experience to one user, and introducing high infrastructure costs. Crucially, many systems do not report, or make explicit efforts to mitigate, the yaw drift inherent in the IMU-based sensors in vehicular setups [34], or the platforms are only used in controlled settings involving limited turning, which is very different to real driving scenarios. These platforms are also typically limited to being real-time experiences that need to be used in a moving car. It is also important that research platforms provide tools to play back vehicle telemetry in the lab to support researchers who may not have ready access to a vehicle, or to give the opportunity to test an XR designs in different driving environments [47]. As commercial XR moves towards more accessible and lower cost standalone HMDs, it is important to minimise the hardware required for the sensing, computation and display of in-car experiences, to avoid the need for expensive and power-hungry laptop/PC VR setups. These expensive setups are not cost-effective nor scalable for multi-user setups in a vehicle.

In this paper we present *PassengXR*, a cost-effective and open-source in-car motion platform for passenger XR experiences that addresses the limitations of current research platforms. Built in Unity, and using ESP32 Arduino-compatible IoT modules to capture, broadcast and receive telemetry wirelessly at low latency, it supports the sensing, recording and playback of all car movement (IMU orientation, OBD-II velocity, GNSS global position) for multiple co-located standalone XR headsets. Our platform also supports a variety of approaches for maintaining headset alignment within the vehicle reference frame, enabling both motion-based and vehicle-based XR content. All sensor data can be recorded for analysis of participant testing, and using the same Unity scene configuration in a lab-based scenario, *PassengXR* can also play this recorded data back within Unity in real time, to recreate the same movements, locations, events etc as the real car journey. This means that designers who are unable to access a car can still design an interface and test how real car movements manifest in the virtual scene for local VR users. We also provide ready-made datasets for designers who are unable to record their own routes.

1.1 Contributions

PassengXR provides several key contributions over the existing research and commercial platforms:

- A means to create **passenger XR experiences in any vehicle (last ~15 years), at low cost** (~$480) vs. PC VR platforms.
- **Runs on standalone inside-out XR headsets**, with vehicle telemetry wirelessly broadcast to all passenger XR headsets.
- In-built support for **maintaining headset alignment within the reference frame of the car**, through both manual and automatic re-alignment.
- Supports **multiple concurrent users in shared and individual experiences**.
- **Flexible and extensible** software for configuring how individual data sources are used within an experience.
- **Recording and playback of real car data** for faster, cheaper and more ecologically valid development of experiences in lab settings.

2 RELATED WORK

Virtual Reality (VR) and Augmented Reality (AR) have been used in a range of automotive research. Most work is aimed at the driver of the vehicle (*e.g.,* [12, 13, 15, 21, 41, 49]) or, as autonomous vehicles become more prevalent, pedestrians who may need to gain information about the car (*e.g.,* [30, 39, 41]. Comparatively few papers have investigated the design of passenger experiences [41]. As our motion platform is intended for the design of passenger experiences in real cars, this section focuses on research that has utilised either real cars (including vehicle telemetry) in the design of an XR-based simulator, or has conducted in-car research using XR as a means to explore use cases or applications for passengers.

This is important, as a key challenge in creating stable in-car XR experiences is that the frequent turning of a vehicle (and therefore its passenger) leads to drift in IMU-based car and headset sensors [17, 34]. Also, headset sensors are not only detecting rotation of the head, but also rotation of the car. Therefore, the successful design of in-car platforms (and associated experiences/applications) is dependent on an ability to separately detect, and compensate the drift of, both vehicle and headset rotation. We first cover in-car XR simulators and research into in-car XR applications/use cases. As our primary contribution is the creation of an open-source toolkit for conducting in-car XR research, we then separately discuss existing and similar open-source XR research toolkits/platforms.

2.1 Vehicular XR Experiences

2.1.1 Simulation and Gamified XR. A handful of papers have instrumented real vehicles with movement sensors as a way to build XR experiences that leverage the physical motion of a car. *CarVR* [17] and *MAXIM* [52] both captured a car's rotation and velocity through an IMU and the vehicle's OBD-II port, respectively, and *MAXIM* also captured the physical environment around the car using a hood-mounted 360° camera. *COMS-VR* [23] took a different

	TECHNOLOGY				FEATURES			UTILITY			
	Standalone or PC	**VR & AR**	**GPS/GNSS**	**Environment Sensors**	**Multi-User Experiences**	**Headset Pose Correction**	**Recording & Playback**	**Low-Cost**	**Open-Source**	**Extensible & Configurable**	**Route Agnostic**
CarVR [17]	Standalone							✓			
COMS [23]	PC										
MAXIM [52]	PC	✓		✓			✓				
McGill [34]	Standalone		~			~		✓			✓
Daimler [16]	PC		✓	~		**6DoF Tracked**					
Volvo [12]	PC	✓									~
VR-OOM [15]	PC								✓	~	~
XR-OOM [13]	PC	✓		✓					✓	~	~
Holoride [18]	Standalone		✓	???	~	???	✓			✓	✓
PassengXR	Standalone	~	✓	~	✓	**3DoF ✓ 6DoF ~**	✓	✓	✓	✓	✓

Table 1: Table comparing *PassengXR* to other research and commercial platforms. Green tick indicates a present feature; amber "~" indicates a partially supported feature; "???" indicates that the feature is unknown but likely to be supported.

approach, reading vehicle telemetry directly from the Engine Control Unit of an electric vehicle. Some approaches rely on PC-based systems [23, 52] while one used a standalone phone-based VR headset [17], and they have been used for both passenger entertainment [17, 23] and producing simulators for drivers [52]. As well as being a real-time, live simulator, MAXIM can also annotate the 360° video with the IMU and OBD-II data in a way (which is unclear) that lets it be played back in static simulator setups.

However, these systems have several limitations. Only MAXIM took action to mitigate headset drift/misalignment (by using a VR controller as reference point) and no paper reports the level of drift that occurred. They also all relied on short, predefined driving routes with few turns. *COMS-VR* [23] only supports linear movement and is specific to the single-seat Toyota COMS electric vehicle. CarVR [17] can be used in any vehicle with an OBD-II port (manufactured after 2006), and so is a good example of a low-cost and practical approach to instrumenting a vehicle for in-car XR experiences, but it is limited in the number of sensors (e.g., there is no GPS or positional sensing for location-based experiences), it does not include playback and it only supports a single user. *MAXIM* [52] is a single-user PC-based system for drivers that lacks GPS/positional sensing.

2.1.2 Motion Sickness. McGill *et al.* [34] developed a platform on a GearVR headset, which captured vehicle rotation via the gyroscope (30Hz, latency of 40ms) of a Nexus 5 smartphone fixed to the dashboard and the vehicle velocity from OBD-II over Bluetooth (via OBDLink LX, ~14Hz, latency of 100ms). Passengers viewed a 360° video sphere while being driven round a set urban route, and the authors took continuous measurements of motion sickness under different viewing/rendering conditions that incorporated, or excluded, vehicle motion. Drift was measured at 20° per minute, and so the user's forward direction required manual re-alignment (via a button press) at any time the user perceived a misalignment. There were large individual differences in which conveyances of motion led to motion sickness or a preference of rendering approach. Therefore, in-car XR tools and systems need to be flexible in how they convey (or ignore) vehicle motion, or use motion to alter parts of a scene.

RoadVR [4] and RideVR [5] used a similar technical setup to CarVR [17] and McGill *et al.* [34], but with an additional GPS sensor for more accurate positioning in RoadVR. They dynamically distorted the visual scene when the car turned corners to more accurately produce optical flow and thereby significantly reduced the experience of motion sickness. QueasyRider [29] examined how interface interactions might affect motion sickness while participants wore an Oculus Quest headset in a back passenger seat of a car driven along 4km of highway. Car movement was detected via the Quest's own controller and the virtual content was locked to the car rotation. The authors do not report any measurement or experience of drift (which is surprising based on our own test, reported in Section 3), nor system latency.

2.1.3 Productivity. As well as providing more opportunities for entertainment [17, 18, 48], XR also offers passengers significant benefits for productivity, as workspaces are not restricted to the physical dimensions of desks or monitors. Research has begun investigating the effective, comfortable and socially acceptable design of virtual workspaces in XR [32] including in cars [28] and planes [37], letting passengers interact with multiple windows from a seated position while limiting head movements. In the future, autonomous cars may have internal surfaces instrumented with touchscreens[1], and so the passenger would be able to turn the entire cabin into an interactive 3D workspace. However, for these scenarios to be effective, the in-car motion platform needs to be able to accurately separate the movement of vehicle and user, and make efforts to maintain the user's alignment to the vehicle reference frame.

2.1.4 Commercial Applications of In-Car XR. Car manufacturers have also produced prototype XR-based motion platforms [12, 16] that typically fuse vendor-specific telemetry (e.g., GPS, wheel tick, IMU) directly from vehicles with high-end PC-based systems, including the professional-grade Varjo XR-1 headset [12]. Daimler [16] created a virtual environment of a digital Mercedes car interior as it moves through different driving environments (small town, cave system, forest) running off two computers. Passengers wore a 3DoF Oculus Rift headset complemented by a PST Base external 6DoF positional tracking system. No mention is made of any measurement or magnitude of drift over time. More recently, Volvo presented a proof-of-concept PC-based AR platform for designing hazard warnings for drivers in a Volvo XC90 [12] using a Varjo XR-1 headset and ART ARTTRACK5/c positional tracking.

2.1.5 Summary. The research in this section has shown how vehicles can be instrumented to display XR content to a headset in a way that either mimics, incorporates or ignores the physical motion

[1]https://www.mercedes-benz.com/en/innovation/autonomous/research-vehicle-f-015-luxury-in-motion/

of the car in a way that suits the intended application and/or minimises motion sickness. We illustrate how *PassengXR* expands and improves upon these existing vehicular XR approaches, addressing key limitations, in Table 1.

2.2 Research Toolkits and Platforms

With *PassengXR*, we are presenting an open-source motion platform toolkit for creating in-car XR experiences. There are a small number of similar endeavours in the research and commercial spheres, but they are either focused on different use cases, require expensive hardware, have fewer features, or have restrictions in terms of who can access/use them.

VR-OOM [15] and XR-OOM [13, 14] are open-source software platforms to help researchers design in-vehicle VR and AR interfaces for drivers, respectively. Both are PC-based systems that use an IMU to detect vehicle rotation and OBD-II for velocity, while XR-OOM also uses LIDAR and a ZED2 camera to track the vehicles position within a known external environment. VR-OOM utilises an Oculus Rift for display, while XR-OOM uses the enterprise-focused Varjo XR-1 headset. The authors discuss the issue of drift, but do not provide a measurement of it during their studies. While not directly comparable to our system, LoopAR [36] is an open-source Unity simulator toolkit for designing autonomous car takeover interaction techniques in static simulators. It includes pre-built Unity components for creating roads, vehicles and "critical traffic events", with obstacles to avoid.

Outside of research, Holoride [18] creates interactive passenger VR experiences for autonomous vehicles that adapt their content based on the movement, GPS location and intended travel route of the car. They also provide their "Elastic SDK" for developers to create experiences based on car telemetry, which includes real pre-recorded telemetry that can be played back within the companies developer tools to let designers test apps without access to a moving car. Two users can apparently engage in experiences in the same car[2], though not in shared experiences.

2.2.1 Current Limitations. We illustrate the limitations of these systems - and how *PassengXR* improves upon them - in Table 1. In summary, these systems have typically one or more of the following issues: PC-based (costly, cumbersome, limited scalability); single-user; lack compensation for headset pose drift and realignment; based on short or predefined routes (i.e., they have not been demonstrated as fully route-agnostic for use on any road); have restricted usage (licensing, specific vehicles); or lack support for lab-based playback of real vehicle telemetry.

Against this backdrop, we have developed *PassengXR* to provide a low-cost, accessible, open-source and flexible toolkit for standalone VR headsets to help researchers and designers develop single or multi-user in-car passenger experiences with recording and playback of car data, and offer several approaches for how to maintain headset-vehicle alignment during arbitrary driving routes.

[2]https://www.youtube.com/watch?v=q2KN_ZpQqNg

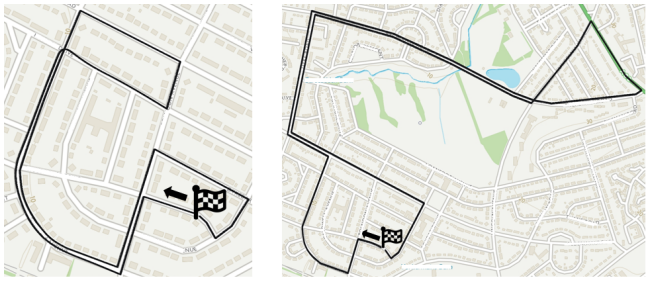

Figure 2: Routes for measuring headset drift: 2.6km with frequent turns (left), 6.5km with less frequent turns (right).

3 EVALUATING CONSUMER HEADSETS FOR IN-VEHICLE USE

XR headsets predominantly rely on fusing visual (optical) and inertial (IMU) data to arrive at an estimate of their position and orientation. With a move to inside-out 6DoF tracking methods, modern standalone VR headsets face additional tracking challenges in car environments that contain a mixture of static (e.g., dashboard) and moving components (e.g., outside world) as well as glass and temporally variable lighting conditions from cloud cover or shadows. Consumer XR headsets track the stable environment of the car interior (with varying success) and try to rectify that against the apparently moving environment as indicated by the IMU during angular and linear accelerations of the vehicle. This results in unpredictable and erratic tracking when the car turns, exhibited in our experience as significant yaw jitter where the user perceives the turn motion, but with the headset view frequently snapping back to looking straight ahead based on the ground-truth of the visual tracking. As this fusion algorithm is a black box, we are unable to do any significant corrections to the tracking.

Instead, our focus is on supporting 3DoF tracking in-car, and in particular identifying headsets that a) can be forced into 3DoF tracking by practitioners and b) whose 3DoF tracking implementation operates reliably in-car[3]. Given a vehicle mounted IMU, it should be straightforward to zero the headset tracking to the IMU, such that the IMU tracks the orientation of the vehicle, and the headset tracks the combined orientation of vehicle and head movement - the basis for vehicle-locked content. However, IMU-based sensors are subject to gyroscopic yaw (y-axis) drift over time, particularly in a vehicle that is frequently turning. The greater the extent of this drift, the more (and more frequently) a headset will have to be re-aligned/zeroed to the car, either manually or automatically. Therefore, we conducted a test to measure the extent to which current commercial standalone VR headsets are prone to drift during real driving, and are suited to deployment in-car.

3.1 Methodology

We conducted testing to determine what 6DoF standalone headsets supported a 3DoF fallback; to what extent 3DoF headsets behaved as expected (detecting car and head orientation in combination); and to what extent the headset IMU drifted over time - how far the

[3]While external 6DoF positional trackers are available, they typically require a PC, greatly increasing the cost of the system.

	Short Route (2.6km)						**Long Route** (6.5km)	
	1 Lap (~7 mins)		**2 Laps** (~14 mins)		**3 Laps** (~21 mins)		**1 Lap** (~15 mins)	
	Car	***Headset***	***Car***	***Headset***	***Car***	***Headset***	***Car***	***Headset***
Meta Quest 2	5.07 (5.23)	54.47 (87.67)	5.71 (5.50)	74.40 (73.44)	7.87 (7.95)	110.34 (222.99)	0.55	196.83
Pico Neo 3 Pro	2.21 (2.37)	4.43 (4.54)	3.47 (3.47)	9.46 (8.64)	3.08 (3.16)	8.54 (8.27)	0.59	7.29
Pico G2 4K	3.73 (3.75)	44.91 (46.52)	4.73 (4.99)	67.47 (67.64)	5.47 (5.53)	70.33 (70.30)	0.09	83.37
HTC Vive Focus 3	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Table 2: Total drift - change in estimated orientation from a set reference - in degrees, for the car and headset IMU at the end of each lap of two testing routes (before a 1-minute rest period). Values in brackets show the drift at the end of the 1-minute rest. HTC Vive Focus 3 lost tracking throughout both routes, and so no reliable measure of drift was possible.

intended 'forward' direction in a VR experience would move around the user. This is the first such systematic approach to measuring drift across headsets on public roads and routes representative of everyday driving. To do this, we set up the motion platform components as described below in a 2019 Citroën C3 and recorded the orientation of the vehicle IMU and headset IMU over the course of two different driving scenarios (shown in Figure 2):

- Three laps (separated by a one-minute break) of a 2.6km (~7 minutes) urban route made up of shorter roads and frequent turns, to approximate a potential experimental setup involving three conditions (similar to McGill *et al.* [34]). This route involved 6 left and 6 right turns.
- One longer urban route (6.5km, ~15 minutes) involving longer roads and less frequent turns, to understand if this led to reduced drift. This route involved 7 left turns and 7 right turns.

The headsets tested were the Meta Quest 2, Pico Neo 3 Pro, Pico G2 4k Enterprise and HTC Vive Focus 3 - all current cutting edge inside-out standalone VR headsets. To avoid any potential issues arising from the inside-out tracking interfering with the estimated orientation, each headset was to be set to a 3DoF mode, by turning off positional tracking (the Pico G2 4k is already 3DoF). However, it was not possible to do this on the HTC Vive Focus 3. Each headset was anchored to the passenger seat using the headrest and velcro straps to keep it facing directly forward. The recording of all data was started and the car remained stationary for one minute to set clear baseline forward directions for both the vehicle IMU and headset, which was along the yaw (y axis) Euler angle in degrees. The car started and stopped in the same position and angle, including during the breaks between the short repeated laps.

The total headset drift - in degrees - was calculated by comparing the original baseline forward orientation to the orientation at the end of each lap: arrived at by averaging the values over the first ~5 seconds after stopping (and the final ~5 seconds at the end of each short lap break) to allow for the signal to settle. The results can be seen in Table 2. The drift could have been calculated by comparing the headset to the car IMU, however, the latter may also be susceptible to drift, muddying the results. Therefore, both values were measured, to see the level of car IMU drift, and to compare how black-box headset tracking might lead to different levels of drift than a dedicated external sensor.

3.2 Results

The Pico Neo 3 Pro had by far the lowest total drift: 4.5° after a single 7-minute lap and up to 8.5° after all 3 laps (~22 minutes). Following the 15-minute route, the Pico had drifted by 7.3°. These values are considerably lower than the ~20° per minute observed by McGill *et al.* [34] (GearVR headset) and so, for short journeys or research studies, there may be no need for manual or automatic re-alignment using the Pico Neo 3. However, longer journeys will likely still lead to detrimental levels of misalignment. In comparison, the Meta Quest 2, currently the most popular standalone VR headset, performed erratically with significant drift, leaving it essentially unusable. Our interpretation of the logs is that, when in 3DoF mode, the Quest 2 tries to perform some form of drift correction, perhaps using magnetometer data[4], which leads to the headset performing slow and continuous yaw corrections even when the vehicle is not moving. In Table 2, this can be seen in the differences between the non-bracketed drift values (from the beginning of the 1-minute post-lap rest) and the bracketed drift values (at the end of each 1-minute rest).

The Pico G2 4K, a 3DoF headset, had similar total levels of drift to the Quest 2, but did not show the same continued drift (or attempted correction) when stationary (bracketed and un-bracketed values are similar). From the logs, it was apparent that the HTC Vive Focus 3 was not able to maintain tracking in the car: the orientation effectively alternated between points ±10° either side of the initial heading throughout the drive. During earlier subjective testing an experimenter wore each headset during a drive to view how it responded to movement, and they noted that the Focus 3 would frequently lose tracking entirely and attempt to reestablish the tracking space. The vehicle IMU (SparkFun BNO080) regularly maintained a low level of drift (~2-8°) even after 15-21 minutes of driving, however there were variations between testing runs, possibly due to unavoidable differences in traffic behaviour. It performed particularly well over the long route (<1°).

A note of caution in interpreting these results is that the headsets were anchored in a way that they could not move or rotate (to maintain reliable orientation readings). However, during real use, a passenger's head will regularly turn as they view or interact with content. These additional turns (some conflicting, some complementary) will likely lead to different drift values, but our data provides a baseline level and an indication of each headsets propensity to become misaligned. Based on our results, we chose to use the Pico Neo 3 Pro as the headset for *PassengXR*, and we also recommend this headset, or the more widely available Pico Neo 3 Link (though this remains untested) to other in-car XR researchers. The following section explains the *PassengXR* motion platform, the main hardware components and the main technical challenges we faced.

[4]http://msl.cs.illinois.edu/~lavalle/papers/LavYerKatAnt14.pdf

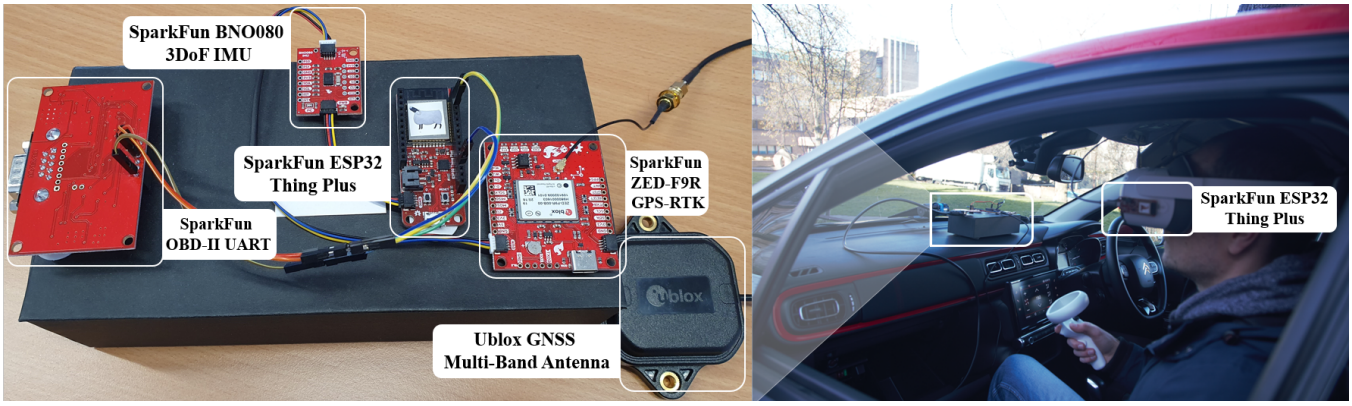

Figure 3: *PassengXR* in-car setup: SparkFun Arduino vehicle sensor boards on car dashboard, and passenger wearing Pico Neo 3 Pro headset with receiver SparkFun Arduino.

4 *PASSENGXR* MOTION PLATFORM

PassengXR is an open platform for enacting passenger XR experiences in-car, with a reference hardware implementation and a Unity-based toolkit. It overcomes key challenges in maintaining headset alignment and compensating for yaw drift, conveying vehicle telemetry to multiple standalone XR headsets for creating motion- and location-based experiences, and recording/replaying vehicle telemetry data to assist in offline development and testing. We first outline the core hardware and software components, and then provide further details around these contributions, and the technical challenges that were overcome.

4.1 Hardware

The hardware components are divided into those that are required per-vehicle (to detect and relay vehicle telemetry) and those that are required per-user (to receive data and run/render the virtual environment).

4.1.1 Vehicle Components. At the centre of the vehicle data sensing is the **SparkFun ESP32 Thing Plus**[5] (~$23), a Microcontroller based on the Espressif ESP32-WROOM chipset. Aimed at IoT applications, these modules are low power, low cost, and are readily programmable using the available Arduino core. This board acts as the server for gathering car sensor data and can be trivially setup to act as a Wi-Fi station and broadcast to available clients (other ESP32 boards) without additional networking infrastructure such as a dedicated access point. They also feature connectivity to other devices via I^2C and Serial Peripheral Interface (SPI) connections, meaning extensive support for a range of existing sensor boards. Connected to the car server are three sensors for providing vehicle telemetry:

Orientation: A *SparkFun BNO080 3DoF IMU*[6] (~$38) provides high sample rate (up to 1000Hz) data regarding the car orientation.

Velocity: A *Sparkfun OBD-II UART*[7] (~$57) board for communicating with the vehicle OBD-II port, enabling polling of the current velocity, at a sample rate determined by the vehicle's capabilities (modern vehicles typically support 15Hz+).

Location: A *Sparkfun ZED-F9R GPS-RTK Dead Reckoning Breakout*[8] (~$290) with Ublox GNSS Multi-Band Antenna[9] (~$73) provides GNSS/global positioning data at 10Hz. Importantly, this sensor supports on-board IMU-based dead reckoning, meaning we can get accurate positional data at relatively high sample rate for consumer GNSS.

The configuration of sensors connected to the car server board is shown in Figure 3. In total, all sensors and the server Arduino board cost approximately $480, and if GNSS is not needed, the cost would be only ~$117.

For shared multi-user experiences in the same vehicle (i.e., users see each others avatar in the experience), an additional device is needed to act as server for Unity due to our reliance on the Mirror networking library (see subsubsection 4.3.3). Whilst that device can be one of the client headsets for P2P networking, acting as both client and server can introduce a performance hit that is undesirable for standalone XR. Consequently, we recommend that for multi-user experiences, either cloud-based hosting is used, or an additional local device (e.g. a laptop) is deployed, which would significantly increase cost. However, no laptop is needed for individual multi-user experiences in the same vehicle, where each user has an independent instance.

4.1.2 User Components. Based on the drift testing, we chose the **Pico Neo 3 Pro**[10] **headset** (~$650) for user devices (though the near identical Neo 3 Link is now available for only ~€449/$499). It provides a good level of processing power for standalone devices, with a Qualcomm XR2 processor and 6GB RAM (broadly equivalent to Meta Quest 2). An advantage of the Pico Neo 3 is that it can be set to a 3DoF tracking mode while the controllers remain positionally tracked in 6DoF - to our understanding this is a unique feature in standalone headsets. This opens up a wider range of user interaction possibilities and also offers the option to use a controller as a reverse positional tracker for the headset.

[5]https://www.sparkfun.com/products/15663
[6]https://www.sparkfun.com/products/14686
[7]https://www.sparkfun.com/products/9555
[8]https://www.sparkfun.com/products/16344
[9]https://www.sparkfun.com/products/15192
[10]https://www.pico-interactive.com/us/neo3.html

The other user component is another **SparkFun ESP32 Thing Plus**, set to be a client and receive vehicle telemetry data broadcast by the vehicle server Thing Plus. The client ESP32 board is connected to the Pico Neo 3 Pro via a micro-USB to USB-C cable. We have estimated the overall latency from motion-to-photon: when a new vehicle telemetry sensor sample is broadcast it is received and processed by the client XR headset in approximately 20ms (roundtrip latency ~40ms), however, this number will vary depending on a number of factors (*e.g.,* latency in sensors such as OBD, sample rates, the available BAUD rate *etc.*). 20ms is half the latency in McGill *et al.* [34] and below the ~50ms that Stauffert *et al.* [45] recommend for a responsive VR experience that does not contribute to cybersickness, and we anticipate that it will be possible to reduce this by ~5-10ms as we move toward receiving telemetry broadcasts using onboard headset WiFi chipsets instead. However this introduces an additional compatibility challenge across XR headsets that we chose to circumvent for simplicity. Moreover, having an additional ESP32 board on the headset itself offers extensibility for researchers in terms of adding additional sensing, such as physiological sensing for motion sickness onset [27], or adding an additional IMU for alignment correction (discussed later). The total per-user cost is approximately $675. Depending on budget and the purpose of the application, this setup can be scaled for multiple users, each with their own Neo 3 Pro and client ESP32 Thing Plus, all sharing the same broadcast data from the one server Arduino.

4.2 Software

The software functionality of *PassengXR* is broadly split into two parts: the Arduino-based vehicle sensors and the Unity-based *Motion Platform* software, that use Google Protocol Buffers (Protobuf) to send/receive data packets.

4.2.1 Vehicle Arduino Sensors. The vehicle server *SparkFun ESP32 Thing Plus* board runs custom Arduino C++ scripts that retrieve sensor updates (IMU, OBD, GNSS) and serialize the vehicle telemetry into a Protocol Buffer[11] format before broadcasting this over Wi-Fi to nearby clients (via known MAC addresses). The client ESP32 receiver board then relays the Protobuf data over USB serial connection to the Unity application on the XR headset.

4.2.2 Unity Motion Platform. The Unity application parses the ProtoBuf telemetry data into a Unity scene-independent *ScriptableObject* (SO) asset called `VehicleProtobufSensor` which exposes multiple custom base data providers, such as `Orientation`, `Velocity` and `WorldPosition`. Using Unity XR plugins, all headset and controller input is similarly used to populate an `XRDeviceSensor` exposing it's own `Orientation` and `Position` data providers. These providers are then fed into the `MotionPlatformConfiguration` SO which in turn is assigned to the `MotionPlatform` which enacts alignment, drift correction, XR input and vehicle reference frame movements within the Unity scene based on this data.

4.3 Key Components and Contributions

4.3.1 Flexible and Configurable Motion Platform. A key contribution of *PassengXR* over other platforms is the ease with which designers can configure how different sources of motion data are used. The `MotionPlatformConfiguration` defines how the XR headset alignment should be maintained, and how the virtual reference frame of the vehicle should match the experienced motion in reality. This is a serialized SO, meaning developers can define multiple configurations, and easily swap out what configuration is active on the `MotionPlatform`. Sensor data providers (e.g. vehicle world position, orientation, velocity) are extensible (i.e. adding compatibility for new hardware and sensor types is straightforward), and these providers are also *chainable*, meaning results can be passed from provider to provider. This makes it trivial to apply transformations to incoming sensor data, such as yaw alignment corrections, or to alter the perception of experienced motion etc. prior to these results being translated into vehicle reference frame motions by the platform. The platform works with Unity's generic XR plugins and `XRRig` for enacting user head/controller/hand movements, meaning that any Unity XR-compatible headset can be supported in theory (if not in practice, based on our drift results). Using the `MotionPlatformConfiguration`, the designer can configure their Unity environment so that their chosen virtual objects (e.g., a virtual car model) have their position, orientation and movement controlled from the incoming `VehicleTelemetry` and headset data.

4.3.2 IMU-Related Alignment and Drift Compensation. Our platform supports several approaches that practitioners can adopt to identify and correct the forward bearing of the passenger, and so maintain an XR headset's alignment with the vehicle and avoid drift. These include: an automatic alignment method that detects shared acceleration profiles between vehicle and headset; the use of external tracked anchors (*e.g., QR codes or tracked XR controllers*) that indicate the forward bearing; and a common manual alignment method. Descriptions of these approaches and the challenges that were faced in developing them, are discussed in Section 5.

4.3.3 Sense and Share Vehicle and Multi-User Movement Across Network. Previous vehicular XR platforms have been designed for a single user, often using PC-based VR setups that cannot scale economically. Using WiFi broadcast between the vehicle (server) and headset (client) SparkFun ESP32 Thing Plus boards, *PassengXR* supports up to 10 passengers (the maximum number of connections under ESP32) receiving the same vehicle telemetry concurrently. This means that multiple headsets in the same vehicle can wirelessly receive their own copy of the car telemetry to be used as needed, without adding more sensors to the vehicle or needing an additional server device such as a PC. This mean each person can have their own individual instance of the same experience (such as the Matched Physical and Virtual Movement example application below), or have the movement of the car conveyed in a way that suits their preferences to reduce motion sickness [34].

Using the *Mirror* networking API[12], *PassengXR* also supports shared multi-user experiences in the car for the first time. Any device running the `MotionPlatform` can act as a host and nearby clients can connect and each shares a synchronised `Transform` (position, rotation and scale) of each aspect of the XR Rig (headset, controllers). Each user can then see the others' actions in the same virtual space. The broadcast vehicle data and networked users provides new opportunities for in-car XR, such as multiple passengers

[11] https://developers.google.com/protocol-buffers

[12] https://mirror-networking.com/

collaborating on a task, watching a movie together or playing a multiplayer game. Outside of cars, the setup could also be used on public transport such as planes or bus tours (as we illustrate in the Demonstrators section), to give groups of passengers a shared location-based commercial experience.

4.3.4 Recording and Playback for Desk-Based Development. All vehicle and headset sensor data coming into the *Motion Platform* can be serialized, timestamped and recorded to JSON files. Practitioners can use this data to carry out *post hoc* analysis of user and vehicle behaviour following a drive. However, this data is primarily intended to be *played back* within the Unity application. The motion platform is designed in a way that *the same* `MotionPlatformConfiguration` can run using either live sensor data coming from the vehicle in real-time, or from data coming from pre-recorded files. The outcome of these within the platform is identical: all Unity objects, reference frames, environments *etc.* move and behave the same way from playback data as they did during the actual drive.

PassengXR, therefore, lets practitioners design, adjust and test new vehicular experiences in a lab/office using *real data* and without the need to access a real vehicle. It is costly - often prohibitively so - to acquire and use real vehicles in HCI research. It can also be time-consuming and practically challenging to find or create controlled (and safe) driving environments in which to test new interface designs, or access different types of road layouts [47] (urban, country, winding, interstate *etc.*). Lab-based simulators can only go so far in recreating the movement of a vehicle or the surrounding environment, and often involve no movement at all. Through *PassengXR* playback functionality, we can widen access equality and make it easier, cheaper and faster for practitioners to create and test new, ecologically valid in-car XR experiences, as well as adjust tracking, alignment and conveyances of motion before deployment to a real car. Playback files can also be shared amongst the community to create a pool of different routes, road types, locations *etc*, and we provide several pre-recorded datasets to support practitioners who have no means to record their own.

As a practical example for using playback, we used it as part of our work on developing the acceleration-based automatic correction of a headset's forward bearing. We could playback both vehicle and headset movements and test different algorithms to detect shared accelerations between the two and apply corrections to the headset's rotational position to keep it better aligned with the forward bearing of the vehicle.

4.3.5 Comparatively Low Cost, Off-the-Shelf and Extensible. The vehicle sensors are all Arduino (SparkFun) and can be bought for $480 (or only $117 if GPS is not needed). Each user requires one ESP32 board ($23) and a suitable XR headset such as the Pico Neo 3 Pro (~£$650) or cheaper Neo 3 Link (~£399/€449/$499). Therefore, researchers can have a working sensing and display setup for $640-$1000, with additional per-user cost of $640. This is a considerably lower investment than other platforms/toolkits that require a per-user PC/Laptop (~$1000+) and PC VR headset (~$400, up to thousands of dollars for Varjo headsets), as well as Wi-Fi router (~$50), and potentially a vehicle with compatible telemetry built-in (e.g., high-end Audi models). All the components are readily available from online stores and require minimal technical expertise to connect (little soldering, mostly plug-and-play).

4.3.6 Open Code and Datasets for Creating In-Car XR Experiences. All Arduino and Unity code, along with documentation and setup/implementation instructions, is available through the ViAjeRo project website (https://viajero-project.org/). We also provide three pre-recorded driving datasets - including vehicle telemetry (IMU, GNSS, OBD-II) and headset orientation - from different routes: motorway, urban city, and country roads in the UK.

5 TECHNICAL CHALLENGES

5.1 Headset Pose Within Car Reference Frame

Fundamental to the operation of a passenger XR experience is the ability to separately track the XR headset's local orientation (and optionally position/pose) relative to the reference frame of the moving vehicle. This is challenging because XR headsets rely on in-built IMUs to track headset orientation changes, however these will also sense vehicle orientation changes. Moreover, 6DoF headsets rely on Visual Inertial Odometry (*VIO* [42]) / Simultaneous Localization and Mapping (SLAM [7, 31]), fusing this IMU data with optical data about the local environment [53] to track the relative or global position of the headset. However this visual information is largely conflicting - capturing both the static and stable vehicle interior, and also the changing, moving exterior environment - which can undermine inside-out positional tracking.

Broadly, there are two approaches to correcting this issue [33, 35]. The first is to subtract the vehicle IMU (*vIMU*) yaw angle from the headset IMU (*hIMU*) yaw angle, effectively removing the influence of the vehicle motion on headset tracking. This approach is what we suggest is necessary for correcting 6DoF tracking implementations. However, where 3DoF tracking is the target, such an approach has notable limitations e.g. subtle discrepancies and latency in conveying vehicle orientation changes to the headset can induce a sensation of micro-stutters during a car turn. Instead, we suggest to allow 3DoF headsets to move freely based on the *hIMU*-sensed head and vehicle orientation changes, and zero/align the 3DoF headset to the vehicle reference frame forward, and separately enact vehicle orientation changes on the reference frame alone.

How these corrections are practically enacted however can vary significantly based on the available (and crucially, accessible) sensing and tracking capabilities of any given XR device *i.e.* 6DoF headsets with camera-based visual inertial tracking, 3DoF headsets with IMU tracking alone. *PassengXR* supports several routes towards correcting the headset pose, building on proposals outlined by McGill *et al.* [33] as well as approaches used in research [13, 15, 17, 34], and we outline some of the most common XR headset targets we have encountered thus far, and how our platform can support their correct operation in moving vehicles. Note that we exclude solutions that rely on outside-in tracking (e.g. using Optitrack or ART cameras mounted in the vehicle) - our focus is on cost-effective self-contained inside-out approaches.

5.1.1 ***6DoF Headset with Correctable Tracking.*** The optimal case is where an XR headset provides access to positional tracking data. For example, corrections to *hIMU* can be applied based on *vIMU* prior to the *hIMU* data being ingested by the *VIO*/positional tracking implementation. Any yaw drift can then be accounted for by the headset positional tracking algorithm as normal.

Such an implementation would however require very low latency *vIMU* data to be available to the headset, particularly given the reliance on asynchronous reprojection approaches such as Asynchronous TimeWarp (ATW)[13] that apply minute corrections to imagery based on just-in-time *hIMU* data. Whilst *PassengXR* would support this, to the best of our knowledge no consumer XR devices provide this level of access to their tracking implementations.

End Result: 6DoF headset with positional tracking in-vehicle.

5.1.2 ***6DoF Headset with Separate Visual and Inertial Pose.*** Where we cannot inject corrections into the tracking implementation, there are two alternative approaches. Firstly, if we can query the tracking to separately get an orientation based on *hIMU*, and a position and orientation based on Visual Odometry (*VO*)/Visual SLAM alone, then we can still apply the *VO* positional data to the resultant headset pose (effectively retaining 6DoF tracking), and correct the *hIMU*-based orientation separately using *vIMU* data as before. *hIMU* drift can be detected and corrected by zero-ing to the *VO* orientation and acting on any significant differences here.

Variations on this approach include headsets where some optical tracking data can be accessed e.g. point-cloud data (ignoring points at a depth beyond the vehicle interior), again providing a visual-only estimate of position and orientation within the vehicle. However, we note once more that no consumer XR device currently provides this level of access to the outputs of their tracking implementations - typically only providing either IMU-based 3DoF, or positional tracking-based 6DoF pose, which is prone to errors/jitter as previously discussed. Where there is access to the cameras driving the tracking (often prevented for reasons of privacy) or where additional headset-mountable cameras (such as the ZED mini[14]) can be utilized, this solution could be implemented by practitioners. However, we do not recommend pursuing this approach currently, given the performance challenges - requiring a low latency high accuracy high sample rate *VO* approach that does not additionally impact the performance of the standalone headset.

End Result: 6DoF headset with positional tracking in-vehicle.

5.1.3 ***6DoF/3DoF with Tracking of External Anchors.*** Where a headset does not provide sufficient access to their tracking implementation or cameras, but does have a developer-accessible capacity to track the position and orientation of external objects (e.g. controllers) or markers (e.g. QR codes via ARCore/Vuforia), then this offers the possibility of supporting 3DoF tracking with alignment and drift correction. These trackable anchors can be placed at a known forward reference anchor in the car cabin, such as on the dashboard. The passenger's orientation can then be corrected based on the detected pose of the anchor relative to the headset, compensating for *hIMU* drift.

The Pico Neo 3 Pro is a working example of this, and is currently our recommended choice - it can be set to 3DoF mode (disabling positional tracking) and yet the controllers remain positionally tracked in 6DoF. This lets practitioners place the controller within the passenger's field of view to use as a visible marker for correct forward bearing and, potentially, as a means of lower-precision relative positional tracking of the headset. We have also tested similar solutions using QR codes tracked by the front-mounted camera of the Pico G2 4K Enterprise Edition headset, and in both cases we could reliably configure *PassengXR* to track and correct yaw drift based on this ground truth.

End Result: 3DoF Headset that maintains correct orientation relative to car reference frame.

5.1.4 ***3DoF-Only Headset with No Optical Sensing.*** Headsets that are restricted to 3DoF orientation tracking (such as the HTC Vive Flow[15] used by Holoride [18]) have no means of establishing an external reference point for the forward bearing, having omitted optical sensing for e.g. reasons of cost or privacy. In such cases, the simplest approach is to get the user to align/zero the headset whilst looking straight ahead in the car, and periodically prompt the user to re-calibrate based on the anticipated yaw drift over time for a given headset based on the experienced vehicle motions. In *PassengXR*, manual re-alignment, such as utilized in [34]), can be trivially enacted through e.g. pressing a button on an XR controller. However, requiring manual periodic re-alignment by the user is problematic - it is disruptive to the experience, and moreover prolonged periods where their perception of car motion is subtly mis-aligned with what is experienced in reality introduces the risk of sensory mis-match that could provoke motion sickness.

As part of *PassengXR*, we have explored non-visual alignment and drift detection relying on *vIMU* and *hIMU* data. One promising approach is to detect common accelerations (e.g. the linear accelerations of the car moving forward) that are evident on both *hIMU* and *vIMU* - provided there is little-to-no head movement, the accelerometer vector of both devices should represent the same forces - those of gravity, and of the vehicle acceleration. Consequently, we can calculate the angular difference between the two vectors, and compare this with the current difference between the zeroed/previously aligned *hIMU* and *vIMU*, giving us an estimate of the extent to which the *hIMU* and *vIMU* have drifted apart. In our own testing we used playback of real vehicle and headset IMU data within *PassengXR* and identified accelerometer vectors of common magnitude (acceleration experienced by both), calculating the angular difference (yaw) between the two vectors, and using this to re-align the headset. When clearly identified, the angular difference was 5-10°, and we expect this can be refined further based on identifying clear vehicle accelerations undertaken with a lack of head movement, or even creating the circumstances for such events in an autonomous vehicle e.g. directing the user to look at a fixed point when the car knows an acceleration is about to occur.

End Result: 3DoF Headset that maintains correct orientation (with a degree of mis-alignment) relative to car reference frame.

5.1.5 Correcting IMU Drift / Mis-Alignment Once Detected. In all cases where an XR headset can experience yaw drift and a resultant mis-alignment relative to the vehicle reference frame, we have outlined how we can detect this drift/mis-alignment. However, there is then the question of how re-alignment can be enacted. *PassengXR* supports two approaches here. Firstly, we can *automatically periodically re-align based on a threshold value* e.g. if the yaw drift is greater than x, then we trigger re-alignment. Our implementation

[13] https://developer.oculus.com/documentation/native/android/mobile-timewarp-overview/

[14] https://www.stereolabs.com/zed-mini/

[15] https://www.vive.com/uk/product/vive-flow/overview/

simply blinks the XR headset view temporarily (fade to black), and then re-aligns the headset orientation with the vehicle reference frame during the blink. For small corrections, this should be imperceptible (as demonstrated by Langbehn *et al.* [24] for redirected walking). The same approach is used for manual alignment when triggered. However, such an approach could be disruptive if the mis-alignment is sufficiently large.

Consequently, we also provide an implementation of a *real-time, continuous yaw alignment correction*, instead using rotational gain [38] as applied to the user's head orientation to apply yaw corrections. For example, if the headset is known to be mis-aligned from the vehicle forward by +x° on the yaw, then we induce small amounts of rotational gain for head movements in the opposite direction to correct the alignment when the user moves their head (or similarly decrease the gain ratio below 1 for movements in the same direction). Whilst we have not yet evaluated the impact of head-based rotational gain when experienced in a moving vehicle (e.g. in terms of motion/simulator sickness), prior research into rotational gain [38] would suggest that there is a high degree of tolerance here, and consequently for 3DoF headsets we can begin to move away from (potentially disruptive) blink-based corrections towards continuous subtle re-alignment - potentially important for longer journeys, or those where the the car experiences frequent turns that will more quickly induce mis-alignment.

5.2 Location-Based XR Experiences

Location-based experiences (LBEs) are a key component of many envisaged passenger XR scenarios, for example augmented tourism [3], location-based gaming [48] etc. To enact an LBE, we need to know where in the world the vehicle is over time, and in *PassengXR* this is established using GNSS positioning. However, despite our use of a high sample rate (10Hz+) commercial untethered dead reckoning GNSS chipset (i.e. one which utilizes Kalman filtering and fusion of additional IMU and velocity data to predict vehicle position in between GNSS samples), we found that such a solution was not sufficient to portray an LBE in XR.

Kalman filtering used in this context effectively estimates/ predicts the vehicle position based on the combination (typically) of low sample rate but accurate data (GNSS) and high sample rate but noisy data (IMU, GNSS velocity, wheel tick etc.). A typical Kalman filtering-based approach however undermines a key constraint of passenger XR - that motion that is physically perceived (*e.g.,* by the vestibular system) is also visually perceived, with minimal sensory mismatch, as mismatches often result in motion sickness [34, 40]. A Kalman filter introduces the possibility of more often perceiving motion which is not physically occurring, as the predicted position would be based on low sample rate/noisy GNSS data, and the high sample rate IMU/OBD data. Very low-latency updates from IMU -> VR are critical to maintain comfort, and occasional visual jumps as the GNSS re-aligns are easy to mask perceptually (*e.g.,* when eyes blink). A continuous fusion risks people getting continuously out of sync with motion (even if slight) and this is a more likely cause of VR sickness, although such filters could be tuned to reduce sensory mismatch. Consequently, we noted that an alternate approach was necessary, one that ensured that the visually perceived motion (e.g. optic flow) provided by our XR application *always* matched what motion was physically experienced.

We take a three step approach: 1) during initial vehicle movement, we align the forward vector of the vehicle reference frame to the current GNSS forward bearing; 2) from this point onwards, we enact real-time vehicle reference frame world position updates based on the combination of IMU orientation and OBD-II velocity data i.e. using the highest sample rate, lowest latency vehicle telemetry we have available; 3) we monitor the difference between the GNSS position and our current estimated IMU+OBD2 position, and if these drift apart by a customisable threshold, then we use a blink-and-realign approach similar to our drift correction as discussed earlier. In this way, we can ensure that vehicle motion is always accurately portrayed to the XR user, with no additional interpolation or unnecessary prediction, and consequently no visual perception of unreal movements, whilst being able to deliver LBEs.

We argue this approach is sufficient to support a breadth of LBEs in any vehicle, without necessitating additional vehicle environment sensing such as LiDAR or additional tracking cameras. However, this approach prioritises perception of motion over location accuracy. If this trade-off is not suitable for a particular application (e.g. driver AR lane assistance), for example if the practitioner needs to prioritise accuracy instead and is willing to tolerate some perception of unreal visual motion, then approaches using additional vehicle localisation sensing (such as the depth camera/LiDAR sensing utilized in XR-OOM [13]) would be preferable.

6 PASSENGER XR DEMONSTRATORS

In this section we describe three use cases, including three demonstration applications (see accompanying video), that illustrate some of the different content or experience types that *PassengXR* can produce. In each example, we explain the application and how the motion platform was configured to achieve the functionality. This is a "Demonstration" approach to evaluating a toolkit [26], including both *novel examples* - to demonstrate new possibilities - as well as *replicated examples* - to show how existing but practically inaccessible functionality is possible through the toolkit. We also "go beyond" [26] the demonstrations with high level guidance on how to create the experiences in the Motion Platform.

6.1 Matched Virtual and Physical Movement

This application builds on previous work, such as CarVR [17] and Holoride [18], and is intended as a real-time passenger experience in a real car. A single user is placed in a virtual scene in space, sitting inside a spaceship cockpit and surrounded by robot drones (see Figure 4). The lateral movement and rotation of the spaceship matches the movement of the car through the real world, and as the ship moves through space, nearby drones fire at the user, who has to fire back and destroy the drones using a laser cannon. The ship is set as the reference frame by adding the `ReferenceFrame` script to the object. The `MotionPlatformConfiguration` is configured to set the world position of the reference frame in Unity (`UnityWorldPositionProvider`) based on the orientation and velocity readings from the `VehicleProtobufSensor` vehicle telemetry. The viewpoint in the spaceship is controlled by the VR headset by setting the `HeadsetLocalOrientation` to an `XRDeviceSensor`

Figure 4: Illustration showcasing the *Matched Virtual and Physical Movement* application (top), where the movement of the virtual spaceship (bottom) reference frame matches the movement of the real car the passenger is inside.

Figure 5: The *Movement-Locked Content for Productivity* application, where a single user interacts with web-based word processor, pdf and video content locked to the reference frame of the virtual vehicle they are sat inside. A digitised city route updates around the vehicle based on the real location.

in the Configuration with an appropriate alignment approach set. The cannon is aimed via the rotation of the VR controller, and fired with the trigger button. We used assets from the Unity asset store to create the experience [1, 8–10, 20, 46].

6.2 Movement-Locked Content for Productivity

A passenger is commuting to their office in their autonomous vehicle and wants to do work on the way. There are two facets of *PassengXR* demonstrated in this application. Firstly, we lock productivity applications to the reference frame, so that they always remain in front of the passenger (Figure 4). Secondly, we display the movement of the real car through the city, by rendering a digital recreation of the streets via Mapbox[16]. This peripheral motion may help reduce motion sickness [4, 5, 34] and help the passenger's awareness of the journey progress. A digital car is rendered around the user and set as the reference frame (`ReferenceFrame`). Three 3DWebView[17] `CanvasWebViewPrefab` objects are children of the car, placed in a horizontal arc at head height inside the car cabin (showing a pdf, a Word document, and a YouTube video, see Figure 5). For the city, a Mapbox `CitySimulatorMap` prefab renders a set of tiles that contain real street and building data based on a latitude and longitude pair, centred on a position in Unity world space. A script `GPSUpdater` is added to the prefab, which takes the `VehicleProtobufSensor` as a variable, and uses the GNSS data from vehicle telemetry to update the latitude and longitude of the rendered tiles. The `MotionPlatform` sets the `WorldRotation` of the reference frame based on the `VehicleProtobufSensor` orientation data and aligns to GNSS bearing, so that the virtual vehicle, and the user within it, rotate correctly based on the real car data.

6.3 Shared Location-Based Bus Tour

As *PassengXR* is capable of supporting multiple concurrent users (Figure 6), this application illustrates a use case where two people are viewing the same virtual experience, and are able to see each other in the space (Figure 6). We imagine a scenario where two people take a bus tour through a city, and the bus uses the motion platform to provide location- and perspective-correct digital AR overlays on historical buildings, to enhance the experience. In the example we use a digital tour bus and city environment to illustrate the multi-user functionality, however, the experience can also be built for a real bus ride using *PassengXR* sensors and software.

The `BaseNetworkManager` class establishes a server on a host Windows PC, and clients on Android headsets, which connect to the PC IP address using *Mirror*'s `KcpTransport`. The class sequentially instantiates an `XR Rig Shared` player prefab object for each connected client at set seating locations within the scene, based on an array of objects containing *Mirror* `NetworkStartPosition` scripts. The `XRRigShared` prefab includes scripts for handling headset and controller input, as well as *Mirror* `NetworkIdentity` and `NetworkTransform` scripts (and `NetworkTransformChild` scripts for child objects). These are used by the *Mirror* server to share the

[16] https://www.mapbox.com/
[17] https://developer.vuplex.com/webview/overview

Figure 6: Image of the *Shared Location-Based Bus Tour* application, where two networked users share the same experience in a mock city bus tour, sitting inside a reference frame bus and viewing Augmented Reality-style information overlaid on a historic building as the bus moves along the street.

Unity `Transform` updates of each XR Rig to the other user. During the demonstration, the two users are sat next to each other in the `ReferenceFrame` virtual tour bus whose movement along the street is updated using the `FusedWorldPosition` provider based on the previously outlined LBE approach. As the bus passes the building, location-based triggers cause AR-like overlays to appear, and the users can e.g., point to parts of the scenery to draw their partner's attention to it. The city model was supplied by Glasgow City Council and we used a bus asset from the Unity store [11].

7 FUTURE IMPROVEMENTS & LIMITATIONS

3DoF Headset Tracking We have chosen to use only 3DoF headset tracking, as current inside-out positional tracking cannot properly function in regular driving environments, and external positional trackers (e.g. using outside-in tracking enacted via Optitrack or ART cameras) add cost and complexity to in-car research platforms. Because of this decision there is an increased likelihood that passengers will experience increased motion sickness, and decreased immersion [6] due to a lack of translational head movement. Recent developments in VR headsets and our own platform will surely address these issues allow for more complex types of applications that can be fully experienced (*e.g.,* exploring 3D structures).

Supported XR Headsets We have so far focused on support for - and testing of - VR headsets, as they have wider uptake in commercial and research passenger XR endeavours. However, AR has a long history in supporting drivers and recent research has suggested that the technology is also well-suited to passenger experiences [48, 50]. Unity can build applications for Windows Mixed Reality (WMR) platforms, and we are currently in the process of testing *PassengXR* with AR headsets such as Hololens[18] and NReal[19], and we anticipate that our proposed solutions will in time work on these platforms.

Vehicle Velocity / Linear Accelerations Our approach uses the OBD-II port to poll vehicle velocity. This port was made mandatory across many countries over the past 20 years, however the performance of the OBD-II protocol is highly variable - for example, in our testing a 2006 model BMW Mini could achieve only 6Hz when polling the velocity PID, compared to closer to 20Hz in 2019 model Citroen C3. Consequently, where perception of linear accelerations is paramount (e.g. motion sickness research), we recommend thoroughly examining available vehicle options to prioritise high sample rate velocity polling. For vehicles without in-built measures of velocity (e.g. trains), consideration will need to be given to other forms of sensing e.g. using the IMU accelerometer to visually portray motion based on acceleration rather than absolute velocity, or relying on sensor-fused GNSS velocity and IMU linear accelerations to estimate velocity.

Sensing of External Environment We currently have no in-built support for vehicle-based sensing of the external environment (e.g. LiDAR sensing). However, potential users of our platform can utilize any Unity-compatible sensing to *e.g.,* provide support for detecting and appropriating real objects around the car for engaging XR experiences, such as demonstrated by Togwell *et al.* [48] and their use of the ZED2 camera[20] to track other vehicles.

Collaboration & Physical Restrictions Passengers have fixed positions with limited rotational freedom, and so cannot physically face, or turn to face, each other. There is also restricted freedom of arm movement due to nearby seats, people, doors *etc.* Passengers may also want to collaborate with people located in remote offices who have full freedom of movement. To overcome these limitations, we are extending the platform to support custom user orientations and relative positions and heights, so that, for example, four car passengers can interact as if seated round a table, or a seated passenger can collaborate with a standing office colleague, all whilst still allowing each passenger individually perceive the correct vehicle motion relative to themselves. We are also exploring the use of redirected arm movements [54] to let passengers feel as if they are physically interacting in spaces not confined by the vehicle interior. Another potential workaround is the use of perception manipulation techniques [2, 25, 51], that visually distort the virtual environment to enable people to believe they are in a much bigger physical environment or to improve collaboration strategies [19, 43, 44].

Improved Headset Tracking We continue to improve the automatic headset alignment and drift correction approaches. In particular, we see significant promise in the use of rotational gain to perform imperceptible corrections. We are also working towards a 6DoF tracking solution utilising headset point-cloud data, using only the static interior of the car for tracking reference points.

Software & User Evaluation Our platform does not provide the same extent of functionality as commercial platforms such as

[18] https://www.microsoft.com/en-us/hololens
[19] https://www.nreal.ai/
[20] https://www.stereolabs.com/zed-2/

Holoride's Elastic SDK[21], which has been developed in partnership with vehicle manufacturers to make full use of the sensor data available in high-end models. However, our approach does broadly work with any modern car without restriction or licensing terms. Regarding practitioner use of *PassengXR*, we have not yet conducted any usability evaluations of the toolkit. However, we have designed it to be largely 'drag-and-drop' through the Motion Platform Configuration and Scriptable Sensors, and have documented component inspector views to be more explanatory. We have also chosen simple off-the-shelf sensors that are widely used and available, and trivially replaceable. Consequently, we believe that *PassengXR* strongly supports replication and re-use by others. We chose to conduct an evaluation-by-demonstration [26] and report key technical challenges that are core to vehicular XR research, as these would be of significant utility to the community. While we have provided measurements of system latency, headset drift and drift compensation error, the paper lacks a rigorous technical evaluation, and our measurements should be considered advisory, as they will vary by device and implementation.

8 WIDER CONSIDERATIONS

Having explained *PassengXR*, we believe it is important to discuss important issues and barriers that impact the wider goal of vehicular XR research.

8.1 Open Headset Platforms and Tracking

The challenge of achieving reliable 6DoF positional tracking in moving vehicles is not trivial, especially in a way that is open, as well as economical and scalable enough to allow practitioners of varying means to develop experiences. Outside-in tracking is costly, needs additional hardware infrastructure and needs robust mounting inside vehicles to maintain accurate tracking. This means that, much as has happened to consumer roomscale VR, we expect that inside-out tracking is the best route forward for supporting 6DoF XR passenger experiences.

However, whilst we have outlined how 6DoF inside-out XR headsets could work in vehicles, in practice there are a number of obstacles to achieving this. Inside-out tracking on headsets is typically a black-box whose calculations, filters and adjustment are tailored to the typical indoor usage scenarios. This leads to highly problematic behaviour in vehicles, as seen in the significant drift and inappropriate 'corrections' performed by the Meta Quest 2 in our testing. Therefore, we argue that headset manufacturers need to provide more open access to their tracking algorithms, to allow researchers and developers to adjust them and combat the unique sensory challenges in vehicles. Such moves would also benefit other communities of practitioners and users, for example those working on XR simulated motion platforms could more tightly integrate experienced motions into cutting-edge consumer headsets.

8.2 Standardisation of Passenger XR

We also call for similar openness from platform holders, to provide accessible tools and make it easier for designers to create experiences. Platforms such as Holoride can provide robust software development kits as well as rich sensing apparatus, but the software is licensed, limited to a small number of cars, restricted to commercial entities and has unclear distribution/deployment options. It is entirely feasible that other travel companies, such as airlines, tour bus operators etc., could also produce proprietary walled gardens for creating XR passenger experiences. And so there is a need for open platforms that support any car, and in-time any vehicle, for any XR headset and as wide a suite of sensors as possible. Our hope is that we can contribute to open standards for reporting vehicle telemetry and information regarding the external environment to all passengers across a variety of modes of transport. Such a move would benefit passenger XR in particular, but could also support other vehicular experiences e.g. supporting per-passenger SoundsRide [22]. *PassengXR* is the first attempt towards this goal and we call on others to work towards it.

[21] https://www.holoride.com/elastic-sdk

9 CONCLUSION

This paper presented *PassengXR*: a low-cost, open-source toolkit built using Unity to help practitioners create multi-user passenger XR experiences in any vehicle. Using the *PassengXR* Motion Platform, vehicle telemetry can be read from off-the-shelf ESP32 IoT boards and used to incorporate, convey, or ignore the movement of a passenger's vehicle in an XR headset experience. This opens up a wealth of possibilities for passengers to engage in immersive entertainment, utilize virtual workspaces, or even deploy customisable visual motion cues to counteract motion sickness. We also described a process to systematically measure the extent of XR headset orientation drift during real driving, and outlined several approaches to mitigating drift and maintain XR headset alignment with the forward bearing of the vehicle. Through the descriptions of reference hardware and open software components, our provided alignment approaches, and our three demonstrator applications for vehicle-locked content, motion-based entertainment, and multi-user experiences, we have provided the community with a platform for furthering, and democratising, research into vehicular XR.

ACKNOWLEDGMENTS

This research received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (#835197, *ViAjeRo*). We would like to thank Glasgow City Council for providing the 3D model of central Glasgow used in the Shared Bus Tour example.

REFERENCES

[1] Adequate. 2022. *Space Droid.* https://assetstore.unity.com/packages/3d/vehicles/space/space-droid-32200.

[2] Mahdi Azmandian, Mark Hancock, Hrvoje Benko, Eyal Ofek, and Andrew D Wilson. 2016. Haptic retargeting: Dynamic repurposing of passive haptics for enhanced virtual reality experiences. In *Proceedings of the 2016 chi conference on human factors in computing systems*. 1968–1979.

[3] Costas Boletsis and Dimitra Chasanidou. 2018. Audio Augmented Reality in Public Transport for Exploring Tourist Sites. In *Proceedings of the 10th Nordic Conference on Human-Computer Interaction* (Oslo, Norway) *(NordiCHI '18)*. Association for Computing Machinery, New York, NY, USA, 721–725. https://doi.org/10.1145/3240167.3240243

[4] Hyung-jun Cho and Gerard J. Kim. 2020. RoadVR: Mitigating the Effect of Vection and Sickness by Distortion of Pathways for In-Car Virtual Reality. In *26th ACM Symposium on Virtual Reality Software and Technology* (Virtual Event, Canada) *(VRST '20)*. Association for Computing Machinery, New York, NY, USA, Article 70, 3 pages. https://doi.org/10.1145/3385956.3422115

[5] Hyung-Jun Cho and Gerard J. Kim. 2022. RideVR: Reducing Sickness for In-Car Virtual Reality by Mixed-in Presentation of Motion Flow Information. *IEEE Access* 10 (2022), 34003–34011. https://doi.org/10.1109/ACCESS.2022.3162221

[6] James J. Cummings and Jeremy N. Bailenson. 2016. How Immersive Is Enough? A Meta-Analysis of the Effect of Immersive Technology on User Presence. *Media Psychology* 19, 2 (2016), 272–309. https://doi.org/10.1080/15213269.2015.1015740 arXiv:https://doi.org/10.1080/15213269.2015.1015740

[7] H. Durrant-Whyte and T. Bailey. 2006. Simultaneous localization and mapping: part I. *IEEE Robotics Automation Magazine* 13, 2 (2006), 99–110. https://doi.org/10.1109/MRA.2006.1638022

[8] Infima Games. 2022. *Low Poly Space Station.* https://assetstore.unity.com/packages/3d/environments/sci-fi/low-poly-space-station-63555.

[9] Sakari Games. 2022. *Space Fighter.* https://assetstore.unity.com/packages/3d/vehicles/space/space-fighter-104.

[10] GAPH. 2022. *FX Explosion Pack.* https://assetstore.unity.com/packages/vfx/particles/fire-explosions/fx-explosion-pack-30102.

[11] GEST. 2022. *City Tour Bus.* https://assetstore.unity.com/packages/3d/vehicles/land/city-tour-bus-118264.

[12] Florin-Timotei Ghiurău, Mehmet Aydın Baytaş, and Casper Wickman. 2020. ARCAR: On-Road Driving in Mixed Reality by Volvo Cars. In *Adjunct Publication of the 33rd Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '20 Adjunct)*. Association for Computing Machinery, New York, NY, USA, 62–64. https://doi.org/10.1145/3379350.3416186

[13] David Goedicke, Alexandra W.D. Bremers, Sam Lee, Fanjun Bu, Hiroshi Yasuda, and Wendy Ju. 2022. XR-OOM: MiXed Reality Driving Simulation with Real Cars for Research and Design. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI '22)*. Association for Computing Machinery, New York, NY, USA, Article 107, 13 pages. https://doi.org/10.1145/3491102.3517704

[14] David Goedicke, Alexandra W.D. Bremers, Hiroshi Yasuda, and Wendy Ju. 2021. *XR-OOM: Mixing Virtual Driving Simulation with Real Cars and Environments Safely.* Association for Computing Machinery, New York, NY, USA, 67–70. https://doi.org/10.1145/3473682.3480266

[15] David Goedicke, Jamy Li, Vanessa Evers, and Wendy Ju. 2018. VR-OOM: Virtual Reality On-ROad Driving SiMulation. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–11. https://doi.org/10.1145/3173574.3173739

[16] Jonas Haeling, Christian Winkler, Stephan Leenders, Daniel Keßelheim, Axel Hildebrand, and Marc Necker. 2018. In-Car 6-DoF Mixed Reality for Rear-Seat and Co-Driver Entertainment. In *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. 757–758. https://doi.org/10.1109/VR.2018.8446461

[17] Philipp Hock, Sebastian Benedikter, Jan Gugenheimer, and Enrico Rukzio. 2017. CarVR: Enabling In-Car Virtual Reality Entertainment. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. Association for Computing Machinery, New York, NY, USA, 4034–4044. https://doi.org/10.1145/3025453.3025665

[18] Holoride. 2022. *Holoride.* https://www.holoride.com/.

[19] Hiroshi Ishii and Minoru Kobayashi. 1992. Clearboard: A seamless medium for shared drawing and conversation with eye contact. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 525–532.

[20] jandd661. 2022. *Asteroid Field Creator 1.5.* https://assetstore.unity.com/packages/tools/level-design/asteroid-field-creator-1-5-143661.

[21] Richie Jose, Gun A. Lee, and Mark Billinghurst. 2016. A Comparative Study of Simulated Augmented Reality Displays for Vehicle Navigation. In *Proceedings of the 28th Australian Conference on Computer-Human Interaction* (Launceston, Tasmania, Australia) *(OzCHI '16)*. Association for Computing Machinery, New York, NY, USA, 40–48. https://doi.org/10.1145/3010915.3010918

[22] Mohamed Kari, Tobias Grosse-Puppendahl, Alexander Jagaciak, David Bethge, Reinhard Schütte, and Christian Holz. 2021. SoundsRide: Affordance-Synchronized Music Mixing for In-Car Audio Augmented Reality. In *The 34th Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '21)*. Association for Computing Machinery, New York, NY, USA, 118–133. https://doi.org/10.1145/3472749.3474739

[23] Ryo Kodama, Masahiro Koge, Shun Taguchi, and Hiroyuki Kajimoto. 2017. COMS-VR: Mobile virtual reality entertainment system using electric car and head-mounted display. In *2017 IEEE Symposium on 3D User Interfaces (3DUI)*. 130–133. https://doi.org/10.1109/3DUI.2017.7893329

[24] Eike Langbehn, Frank Steinicke, Markus Lappe, Gregory F. Welch, and Gerd Bruder. 2018. In the Blink of an Eye: Leveraging Blink-Induced Suppression for Imperceptible Position and Orientation Redirection in Virtual Reality. *ACM Trans. Graph.* 37, 4, Article 66 (jul 2018), 11 pages. https://doi.org/10.1145/3197517.3201335

[25] Eike Langbehn, Frank Steinicke, Markus Lappe, Gregory F Welch, and Gerd Bruder. 2018. In the blink of an eye: leveraging blink-induced suppression for imperceptible position and orientation redirection in virtual reality. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–11.

[26] David Ledo, Steven Houben, Jo Vermeulen, Nicolai Marquardt, Lora Oehlberg, and Saul Greenberg. 2018. Evaluation Strategies for HCI Toolkit Research. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–17. https://doi.org/10.1145/3173574.3173610

[27] Gang Li, Ogechi Onuoha, Mark McGill, Stephen Brewster, Chao Ping Chen, and Frank Pollick. 2021. Comparing Autonomic Physiological and Electroencephalography Features for VR Sickness Detection Using Predictive Models. In *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*. 01–08. https://doi.org/10.1109/SSCI50451.2021.9660126

[28] Jingyi Li, Ceenu George, Andrea Ngao, Kai Holländer, Stefan Mayer, and Andreas Butz. 2021. Rear-Seat Productivity in Virtual Reality: Investigating VR Interaction in the Confined Space of a Car. *Multimodal Technologies and Interaction* 5, 4 (2021). https://doi.org/10.3390/mti5040015

[29] Jingyi Li, Agnes Reda, and Andreas Butz. 2021. *Queasy Rider: How Head Movements Influence Motion Sickness in Passenger Use of Head-Mounted Displays.* Association for Computing Machinery, New York, NY, USA, 28–38. https://doi.org/10.1145/3409118.3475137

[30] Philipp Maruhn, André Dietrich, Lorenz Prasch, and Sonja Schneider. 2020. Analyzing Pedestrian Behavior in Augmented Reality — Proof of Concept. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. 313–321. https://doi.org/10.1109/VR46266.2020.00051

[31] Mark McGill, Jan Gugenheimer, and Euan Freeman. 2020. A Quest for Co-Located Mixed Reality: Aligning and Assessing SLAM Tracking for Same-Space Multi-User Experiences. In *26th ACM Symposium on Virtual Reality Software and Technology* (Virtual Event, Canada) *(VRST '20)*. Association for Computing Machinery, New York, NY, USA, Article 26, 10 pages. https://doi.org/10.1145/3385956.3418968

[32] Mark Mcgill, Aidan Kehoe, Euan Freeman, and Stephen Brewster. 2020. Expanding the Bounds of Seated Virtual Workspaces. *ACM Trans. Comput.-Hum. Interact.* 27, 3, Article 13 (may 2020), 40 pages. https://doi.org/10.1145/3380959

[33] Mark McGill, Gang Li, Alex Ng, Laura Bajorunaite, Julie Williamson, Frank Pollick, and Stephen Brewster. 2022. *Augmented, Virtual and Mixed Reality Passenger Experiences.* Springer International Publishing, Cham, 445–475. https://doi.org/10.1007/978-3-030-77726-5_17

[34] Mark McGill, Alexander Ng, and Stephen Brewster. 2017. I Am The Passenger: How Visual Motion Cues Can Influence Sickness For In-Car VR. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. Association for Computing Machinery, New York, NY, USA, 5655–5668. https://doi.org/10.1145/3025453.3026046

[35] Mark McGill, Julie Williamson, Alexander Ng, Frank Pollick, and Stephen Brewster. 2019. Challenges in passenger use of mixed reality headsets in cars and other transportation. *Virtual Reality* 24, 4 (2019), 583–603. https://doi.org/10.1007/s10055-019-00420-x

[36] Farbod N. Nezami, Maximilian A. Wächter, Nora Maleki, Philipp Spaniol, Lea M. Kühne, Anke Haas, Johannes M. Pingel, Linus Tiemann, Frederik Nienhaus, Lynn Keller, Sabine König, Peter König, and Gordon Pipa. 2020. WestDrive X LoopAR: An open-access virtual reality project in Unity for evaluating user interaction methods during TOR. https://doi.org/10.48550/ARXIV.2012.12041

[37] Alexander Ng, Daniel Medeiros, Mark McGill, Julie Williamson, and Stephen Brewster. 2021. The Passenger Experience of Mixed Reality Virtual Display Layouts in Airplane Environments. In *2021 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 265–274. https://doi.org/10.1109/ISMAR52148.2021.00042

[38] Anders Paludan, Jacob Elbaek, Mathias Mortensen, Morten Zobbe, Niels Christian Nilsson, Rolf Nordahl, Lars Reng, and Stefania Serafin. 2016. Disguising rotational gain for redirected walking in virtual reality: Effect of visual density. In *2016 IEEE Virtual Reality (VR)*. 259–260. https://doi.org/10.1109/VR.2016.7504752

[39] Daniel Perez, Mahmud Hasan, Yuzhong Shen, and Hong Yang. 2019. AR-PED: A framework of augmented reality enabled pedestrian-in-the-loop simulation. *Simulation Modelling Practice and Theory* 94 (2019), 237–249. https://doi.org/10.1016/j.simpat.2019.03.005

[40] James T Reason and Joseph John Brand. 1975. *Motion sickness.* Academic press.

[41] Andreas Riegler, Andreas Riener, and Clemens Holzmann. 2021. A Systematic Review of Virtual Reality Applications for Automated Driving: 2009–2020. *Frontiers in Human Dynamics* 3 (2021). https://doi.org/10.3389/fhumd.2021.689856

[42] Davide Scaramuzza and Zichao Zhang. 2020. *Visual-Inertial Odometry of Aerial Robots.* Springer Berlin Heidelberg, Berlin, Heidelberg, 1–9. https://doi.org/10.1007/978-3-642-41610-1_71-1

[43] Maurício Sousa, Rafael Kufner dos Anjos, Daniel Mendes, Mark Billinghurst, and Joaquim Jorge. 2019. Warping deixis: distorting gestures to enhance collaboration. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–12.

[44] Maurício Sousa, Daniel Mendes, Rafael K dos Anjos, Daniel Simões Lopes, and Joaquim Jorge. 2019. Negative Space: Workspace Awareness in 3D Face-to-Face Remote Collaboration. In *The 17th International Conference on Virtual-Reality Continuum and its Applications in Industry*. 1–2.

[45] Jan-Philipp Stauffert, Florian Niebling, and Marc Erich Latoschik. 2020. Latency and Cybersickness: Impact, Causes, and Measures. A Review. *Frontiers in Virtual Reality* 1 (Nov. 2020), 582204. https://doi.org/10.3389/frvir.2020.582204

[46] Ebal Studios. 2022. *Hi-Rez Spaceships Creator Free Sample*. https://assetstore.unity.com/packages/3d/vehicles/space/hi-rez-spaceships-creator-free-sample-153363.

[47] Jason Thompson, Mark Stevenson, Jasper S Wijnands, Kerry A Nice, Gideon DPA Aschwanden, Jeremy Silver, Mark Nieuwenhuijsen, Peter Rayner, Robyn Schofield, Rohit Hariharan, and Christopher N Morrison. 2020. A global analysis of urban design types and road transport injury: an image processing study. *The Lancet Planetary Health* 4, 1 (2020), e32–342. https://doi.org/10.1016/S2542-5196(19)30263-3

[48] Henry Togwell, Mark McGill, Graham Wilson, Daniel Medeiros, and Stephen Brewster. 2022. In-cAR Gaming: Exploring the use of AR headsets to Leverage Passenger Travel Environments for Mixed Reality Gameplay. In *to appear in Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems (CHI '22)*. Association for Computing Machinery, New York, NY, USA, tbc. https://doi.org/10.1145/3491101.3519741

[49] Akira Utsumi, Tsukasa Mikuni, and Isamu Nagasawa. 2019. Effect of On-Road Virtual Visual References on Vehicle Control Stability of Wide/Narrow FOV Drivers. In *Proceedings of the 11th International Conference on Automotive User Interfaces and Interactive Vehicular Applications: Adjunct Proceedings* (Utrecht, Netherlands) *(AutomotiveUI '19)*. Association for Computing Machinery, New York, NY, USA, 297–301. https://doi.org/10.1145/3349263.3351331

[50] S. WANG, V. CHARISSIS, R. LAGOO, J. CAMPBELL, and D. K. HARRISON. 2019. Reducing Driver Distraction by Utilizing Augmented Reality Head-Up Display System for Rear Passengers. In *2019 IEEE International Conference on Consumer Electronics (ICCE)*. 1–6. https://doi.org/10.1109/ICCE.2019.8661927

[51] Graham Wilson, Mark McGill, Matthew Jamieson, Julie R Williamson, and Stephen A Brewster. 2018. Object manipulation in virtual reality under increasing levels of translational gain. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–13.

[52] Dohyeon Yeo, Gwangbin Kim, and SeungJun Kim. 2019. MAXIM: Mixed-reality Automotive Driving XIMulation. In *2019 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*. 460–464. https://doi.org/10.1109/ISMAR-Adjunct.2019.00124

[53] Khalid Yousif, Alireza Bab-Hadiashar, and Reza Hoseinnezhad. 2015. An overview to visual odometry and visual SLAM: Applications to mobile robotics. *Intelligent Industrial Systems* 1, 4 (2015), 289–311.

[54] André Zenner, Hannah Maria Kriegler, and Antonio Krüger. 2021. *HaRT - The Virtual Reality Hand Redirection Toolkit*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3411763.3451814