RetroSphere: Self-Contained Passive 3D Controller Tracking for Augmented Reality

ANANTA NARAYANAN BALAJI[*], National University of Singapore, Singapore
CLAYTON KIMBER, Google, Ireland
DAVID LI, University of Maryland, College Park, USA
SHENGZHI WU, Google, Mountain View, USA
RUOFEI DU, Google, San Francisco, USA
DAVID KIM, Google, Zurich, Switzerland

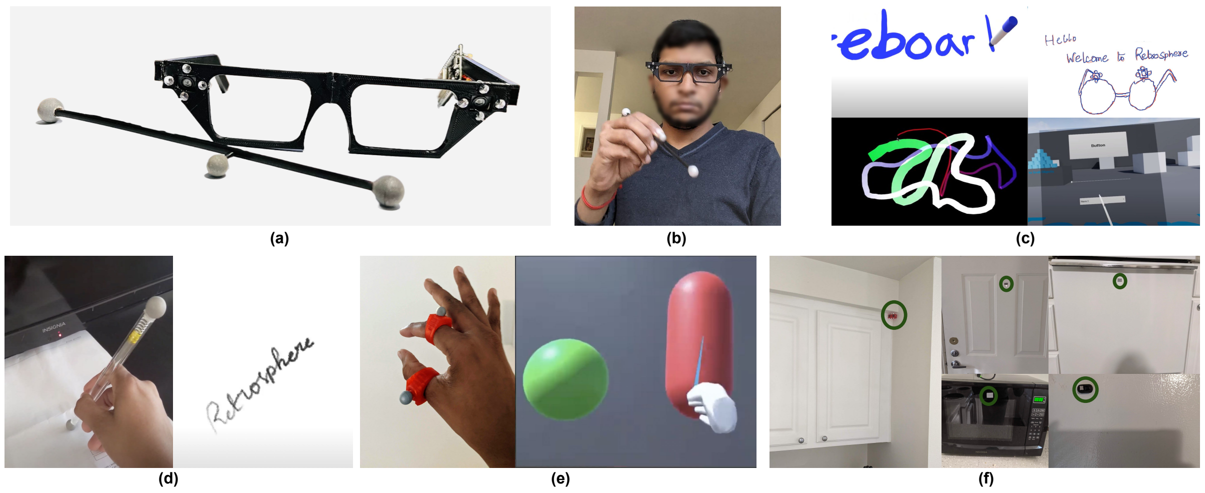

Fig. 1. (a) RetroSphere provides a low-power 6DoF tracking solution for AR glasses using an accompanying passive retroreflective stylus. (b) Users interact with 3D VR/AR user interfaces via RetroSphere enabled glasses and accompanying retroreflective stylus. (c) RetroSphere can be used for writing or drawing on any surface, 3D free-form painting, and 3D menu controls. (d) “RetroPen” is an AR Pencil prototype for seamless AR drawing experiences with RetroSphere enabled glasses. (e) Our “RetroRing” prototype consists of two retroreflective finger rings: one worn on the index finger and the other worn on the thumb. The two finger rings can be used together to interact with 3D user interfaces through finger gestures such as tap and pinch. (f) We designed “RetroSense”, a RetroSphere integrated prototype for smart home sensing using retroreflective markers.

[*]This work was partly done while the author was interning at Google.

Authors’ addresses: Ananta Narayanan Balaji, ananta@comp.nus.edu.sg, National University of Singapore, Singapore; Clayton Kimber, claytonk@google.com, Google, Ireland; David Li, dli7319@terpmail.umd.edu, University of Maryland, College Park, USA; Shengzhi Wu, wushengzhi891215@gmail.com, Google, Mountain View, USA; Ruofei Du, ruofei@google.com, Google, San Francisco, USA; David Kim, kidavid@google.com, Google, Zurich, Switzerland.

Advanced AR/VR headsets often have a dedicated depth sensor or multiple cameras, high processing power, and a high-capacity battery to track hands or controllers. However, these approaches are not compatible with the small form factor and limited thermal capacity of lightweight AR devices. In this paper, we present RetroSphere, a self-contained 6 degree of freedom (6DoF) controller tracker that can be integrated with almost any device. RetroSphere tracks a passive controller with just 3 retroreflective spheres using a stereo pair of mass-produced infrared blob trackers, each with its own infrared LED emitters. As the sphere is completely passive, no electronics or recharging is required. Each object tracking camera provides a tiny Arduino-compatible ESP32 microcontroller with the 2D position of the spheres. A lightweight stereo depth estimation algorithm that runs on the ESP32 performs 6DoF tracking of the passive controller. Also, RetroSphere provides an auto-calibration procedure to calibrate the stereo IR tracker setup. Our work builds upon Johnny Lee's Wii remote hacks and aims to enable a community of researchers, designers, and makers to use 3D input in their projects with affordable off-the-shelf components. RetroSphere achieves a tracking accuracy of about 96.5% with errors as low as ~3.5 cm over a 100 cm tracking range, validated with ground truth 3D data obtained using a LIDAR camera while consuming around 400 mW. We provide implementation details, evaluate the accuracy of our system, and demonstrate example applications, such as mobile AR drawing, 3D measurement, etc. with our Retrosphere-enabled AR glass prototype.

CCS Concepts: • **Human-centered computing → Interaction devices**; **Mobile devices**; • **Hardware → Sensor devices and platforms**.

Additional Key Words and Phrases: Retroreflectors, Augmented reality, Virtual reality, Infrared marker tracking, Augmented reality glasses

1 Introduction

In recent years, spatial interaction has become mainstream with support being provided by both Android and iOS through their AR SDKs [6, 7, 32]. AR SDKs provide smartphones with access to perception tools to enable interactive AR applications [31], such as measurement apps [1], AR gaming apps [19], AR shopping apps [5], and social media apps with built-in AR experiences [22]. Lightweight AR glasses are also being developed by many companies [11, 13, 15] and research teams [49]. However, AR experiences supported by lightweight devices don't provide dedicated 3D input sensors, unlike most VR systems. For example, mobile AR applications usually requires tracking visual features from either the environment or a fiducial marker using the rear-facing camera. More sophisticated AR headsets, such as Microsoft HoloLens, have a dedicated depth sensor, a strong processor, and high battery capacity to track a controller or hands in 3D [14, 58, 65]. In contrast, many VR systems have active 6DoF controllers that allow for 6DoF tracking as well as buttons and joystick-based input. However, these approaches utilize a significant amount of computation power, require a high battery capacity on the device, or are large and unsuitable for use while engaging in everyday activities with future AR glasses. With the current AR/VR trackers, power consumption arises from two crucial factors - (1) CPU executing the controller tracking algorithm and (2) Camera hardware.
1. CPU Power - Running a traditional camera continuously for 3D tracking requires the addition of compute-intensive image processing pipelines running on the chips. For AR glasses, such as Ray-Ban Stories, Snap Spectacles, Nreal, North, Vuzix Blade, or Google Glass, running these algorithms continuously would incur a high CPU load and more quickly drain the battery of such lightweight devices. For instance, an ARM CPU executing a simple OpenCV-based canny edge detection consumes around *870-1200 mW* on the CPU of an iPhone 8 Plus CPU (Apple A11 Bionic) and *1400 mW* on an Odroid-XU3 board [38].
2. Camera Power - There is also significant power consumption by the camera hardware in the smartphone itself. In an earlier work [60], it has been shown that the power consumption of a Nexus 6 smartphone camera varies between *800 mW* to *3000 mW* depending on the image resolution and frame rate. LIDAR scanners on

2474-9567/2022/12-ART157
https://doi.org/10.1145/3569479

smartphones and tablets used for depth sensing also consume substantial power. We found out that the LIDAR scanner in the iPad Pro 12 consumes an average of *4000 mw*.

In the future, we envision users interacting with their lightweight AR glasses or other 3D systems spontaneously throughout the day, where 3D stylus interaction always runs in the background and becomes an always-available input mechanism to turn on the display, invoke applications, point to objects in the real world, etc. Due to form-factor constraints and limited thermal capacity, these devices will have very strict requirements concerning high compute and power consumption with always-on RGB cameras.

In this paper, we present RetroSphere, a low-power and low-compute self-contained 6DoF tracker that can provide spatial input on almost any device. Our work builds upon Johnny Lee's Wii remote hacks [41] and aims to enable a community of researchers, designers, and makers to integrate 3D input into their projects with affordable off-the-shelf components. Similar to Lee's Wii remote hacks, we make use of low-cost and mass-produced infrared blob tracking cameras equipped with co-located infrared (IR) illuminators (consisting of 4 low-power IR LEDs) to track a stylus-like controller with just 3 retroreflective spheres. We place the IR cameras in a stereo configuration, allowing us to triangulate the exact 6DoF positioning of the target. The IR cameras directly output the 2D coordinates of tracked IR illumination sources, while the 3D triangulation of the 2D points is processed on a tiny, low-power ESP32 microcontroller. Using retroreflective markers allows our 3D tracking approach to be integrated into devices of various form factors, such as a wand, pen, wristband, ring, or customized 3D-printed objects. With AR glasses being the hope for the next wave of everyday tech devices, RetroSphere will enable practitioners to make use of affordable off-the-shelf hardware components to support low-power and low-cost 3D input.

The main contributions of our work are:

- We present (1) RetroSphere - a self-contained low-power and low-compute 6DoF controller tracker and (2) a passive/power-free stylus-like controller for providing low-cost 3D input in resource-constrained devices. We open-source[1] our design which uses affordable, off-the-shelf hardware consisting of stereo infrared trackers with co-located emitters, that can be prototyped, 3D-printed, and attached to any device, such as a smartphone or AR Glasses for tracking passive pen-like stylus with three tiny retroreflective spheres.
- We present an automatic calibration technique for the stereo infrared trackers with the users randomly waving their stylus in mid-air. In addition, we demonstrate how only three marker coordinates available from stereo infrared trackers are enough to achieve an on-device, low-compute depth estimation algorithm in RetroSphere. Using an on-device neural network-based occlusion correction technique, RetroSphere also solves marker occlusion caused by our hands while operating the RetroSphere stylus.
- We evaluate our RetroSphere-enabled AR glasses prototype with 20 participants and find that RetroSphere provides high-accuracy tracking with low errors of approximately 3.5% at depth estimation and 4.5% at orientation against ground-truth 3D measurements from a commercial LIDAR sensor. We also evaluated the impact of the various factors such as motion, hand movement speeds, lighting conditions and marker occlusion on the performance of RetroSphere.
- We demonstrate a variety of mobile AR applications enabled by RetroSphere including in-situ 3D drawing, 3D user interfaces, and real-time 3D measurements. Finally, we demonstrate three use-cases realized with our RetroSphere hardware - (a) AR Pencil with pressure sensing, (b) RetroSphere finger attachments for UI control in future AR glasses, and (c) monitoring smart-home appliances with RetroSphere.

[1]RetroSphere Project webpage - https://retrosphere.github.io/ and RetroSphere's open source hardware design - https://github.com/AnantaBalaji/RetroSphere

2 Background and Related Works

In recent years, real-time spatial tracking of pen-like (3 to 6-DoF) controllers has been actively investigated for providing more expressive interactions in VR and AR applications. VR and AR controller sensing approaches can be broadly categorized into (a) camera-based, (b) electromagnetic, and (c) laser/MEMS-based tracking. We summarize current state-of-the-art controller tracking techniques organized based on power consumption and computational requirements in the sections below.

2.1 Camera-based tracking

The most popular VR-controller tracking approach is to apply object tracking approaches to the image obtained from RGB or infrared cameras. Leap Motion [33] uses a hand gesture recognition system with stereo infrared cameras mounted on the head-mounted displays. Oculus Touch [17] uses a set of handheld input devices equipped with a ring of infrared LEDs to allow for real-time 3D tracking. Since controllers are equipped with IR LEDs, they can only run continuously for 37 hours [2]. Oculus Quest [16] and Rift make use of 4-5 cameras on the head mounted display(HMD) to track points in the environment and infrared LEDs on the controllers. HTC Vive Focus [8] makes use of a pair of VR controllers that can be tracked with cameras attached to the HMD display. POL360 [37] is a universal motion controller that makes use of light polarization for 6DoF tracking. However, it can only operate for 4 hours with a 1200 mAh battery. All the above-mentioned tracking hardware as well as compute approaches require high power overhead and computational resources, thereby making them unsuitable for resource-constrained AR devices.

Tracking approach	**Is controller passive?**	**Power consumed by electronics on the smartphone**	**Computation involved in tracking**	**Form-factor**	**Tracking range(cm)**	**Power consumption(mW)**
Leap Motion [33]	No	High	High	N/A	80	2500
Oculus Touch [17]	No	High	High	Handheld	300	N/A
POL360 [37]	No	High	High	Handheld	200	1270
Auraring [51]	No	Low	Low	Ring	10	75.64
Laser/MEMS tracking [46]	No	High	High	Simple Pen/stylus-like	100	1000
ARPen [59]	Yes	High	High	Pen-like with dodecahedron markers	40	5000
DodecaPen [64]	Yes	High	High	Pen-like with cube markers	40	5000
RetroSphere	***Yes***	***Low***	***Very Low***	***Simple Pen/stylus-like***	***100***	***400***

Table 1. Comparison of RetroSphere against state-of-the-art 3D tracking approaches.

In recent years, there have been a few techniques for real-time tracking of a controller with a mobile phone. 2D binary square fiducial trackers [47, 56] have become building blocks for many AR applications. The four corners of markers can be recognized using a mobile phone camera and used to obtain the 6DoF position and orientation. DodecaPen [64] leverages the aforementioned approach and makes use of a 3D-printed dodecahedron pen with each side having a fiducial marker. However, DodecaPen can only work with a fixed camera. ARPen [59] leverages a similar form factor as DodecaPen and uses Apple's ARKit to track a cube of fiducial markers attached to the end of the controller with a phone camera. ARPen only works well in scenarios where only very small movements are involved. In addition, their tracking approach also involves heavy computation and would impose high power consumption on the mobile phone. We compare RetroSphere against ARPen in section 4 and show that RetroSphere achieves 10×**power savings** over ARPen. Ad hoc UI [31] learns features of everyday objects on-the-fly with a phone camera and tracks their 6DoF poses, yet requiring higher compute than DodecaPen and ARPen on the mobile device.

2.2 Electromagnetic controller tracking

Lyons [44] demonstrated a DC 2D electromagnetic tracking approach with a Google Cardboard device. This approach senses a permanent magnet with the help of a magnetometer and other smartphone sensors. SynchroWatch [53] makes use of a permanent magnet attached to a thumb ring to identify finger gestures. AC magnetic tracking systems have reported sub-millimeter level tracking accuracy [20] and can work for long ranges of over 100 cm. They are used in current VR headsets such as Magic Leap and Razer Hydra. However, these high-power systems have a battery life of only 10 hours [21]. Most recently, AuraRing [51] made use of low-power short-range AC communications to precisely track a magnetic ring from a wristband. Although Auraring only consumes 2.34mW on the ring controller and 73.3mW on the wrist tracking hardware, it can only support distances of up to 10cm.

2.3 Laser/MEMS-based controller tracking

Laser imaging systems have been used for precise 3D sensing for tracking in VR/AR headsets. Milanoviċ's laser imager [46] makes use of a MEMS mirror scanning module to send low-power pulses within the field of view and uses a single photodiode to measure the reflected light. However, laser imaging systems require a MEMS mirror and a light source to be placed on the mobile phone to perform 3D tracking of the retroreflective controller. This setup is very hard to achieve on a resource-constrained device.

All the above controller tracking approaches cannot provide low-power and low-compute 3D tracking. Table 1 compares our approach against state-of-art tracking approaches in terms of form-factor, power consumption, amount of computation involved with tracking, tracking range and power consumption.

2.4 Infrared camera-based tracking

Energy-efficient infrared camera-based tracking has been extensively investigated in the past. Zsense [63] uses infrared sensors and emitters arranged in specialized patterns to perform 2D gesture sensing with a low power consumption of 60.2mW. Wavesense [62] performs hand tracking up to 20cm with infrared sensors and emitters integrated into a commercial VR headset with just 69.7mW per second . RetroSphere draws inspiration from Johnny Lee's [41] projects on interaction techniques supported or enabled by the Wiimote infrared blob tracking cameras. Johnny Lee made use of Nintendo Wii infrared cameras for highly interactive applications, such as 2D finger tracking, interactive whiteboard displays powered by an IR-emitting light pen, and head tracking for desktop VR displays. Wiimote IR trackers have been shown to be effective at outdoor target positioning even under extremely sunny weather conditions with the help of IR beacons with specially encoded patterns [43].

Rigid constellations of retroreflective spheres have been shown to provide effective 6DoF tracking with multi-view infrared cameras [45]. De Amici et al. [25] propose a more affordable sub $1000 alternative to professional motion tracking systems using multiple Wiimotes to track the 6DoF transform of retroreflective markers with an average error of 5 mm. Scherfgen and Herpers [54] demonstrate that stereo calibration can be performed with a reflective calibration board, while Petrič et al. [52] propose multi-camera calibration with a robotic arm. Kim et al. [40] extend the capabilities described above with a pair of high-resolution infrared cameras to efficiently estimate the depth of contours between retroreflective and regular materials.

While RetroSphere shares a similar stereo camera configuration, our work focuses on a lightweight software and hardware architecture that is suited for mobile applications. We further demonstrate interactive real-time use cases with AR application examples.

One major advantage of marker-based tracking approaches is the lack of electronics associated with the controller. The controller can be a pen with just a retroreflective sphere. The other major advantage is that the tracking approach only requires stereo infrared blob tracking cameras and a few IR LEDs for illuminating the

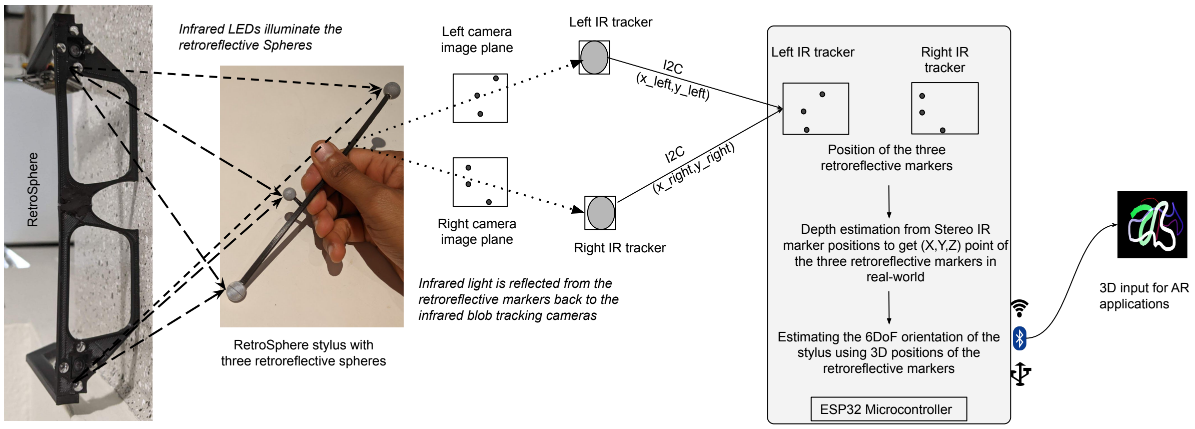
RetroSphere
Infrared LEDs illuminate the retroreflective Spheres
RetroSphere stylus with three retroreflective spheres
Left camera image plane
Right camera image plane
Left IR tracker
Right IR tracker
I2C (x_left,y_left)
I2C (x_right,y_right)
Infrared light is reflected from the retroreflective markers back to the infrared blob tracking cameras
Left IR tracker
Right IR tracker
Position of the three retroreflective markers
Depth estimation from Stereo IR marker positions to get (X,Y,Z) point of the three retroreflective markers in real-world
Estimating the 6DoF orientation of the stylus using 3D positions of the retroreflective markers
ESP32 Microcontroller
3D input for AR applications

Fig. 2. High-level overview of RetroSphere.

scene. This hardware setup reduces the computation significantly since the retroreflective marker detection and tracking can be offloaded in a dedicated low-power microcontroller.

In our paper, we demonstrate RetroSphere. A 3D tracker, which makes use of an off-the-shelf attachment for smartphones and AR glasses comprising of stereo infrared cameras equipped with infrared illuminators (with only 4 infrared LEDs for each IR camera) to perform 6DoF tracking of a pen/stylus-like controller with 3 retroreflective spheres. Our approach completely gets rid of the computation overhead involved in the 6DoF tracking of the controller since the entire compute required for 6DoF tracking can be completely done using a low-power, tiny ESP32 microcontroller on the hardware attachment.

3 RetroSphere 6DoF Tracking Solution

The main design goals of RetroSphere are:

(1) **Low-power tracking hardware** - RetroSphere's tracking hardware is designed for very low-power consumption to always run in the background for continuous 3D tracking. In addition, the RetroSphere stylus is completely passive and doesn't require any power.
(2) **Accurate 3D input** - RetroSphere's 3D tracking algorithm is built to achieve highly accurate 6DoF tracking for 3D AR inputs.
(3) **Ease of integration** - RetroSphere is made with low-cost and affordable hardware components which can be easily integrated into future AR glasses.
(4) **DIY off-the-shelf hardware** - RetroSphere is self-contained and can be easily mounted on existing AR glasses or lightweight devices to provide cheap and accurate 3D input.

3.1 High-level overview of RetroSphere

RetroSphere consists of a passive stylus-like controller with three retroreflective spheres. The retroreflective spheres, when illuminated with an infrared light source, reflect back the light in the same direction. An infrared tracking camera placed near the infrared light source captures the reflected light from the retroreflective sphere and identifies the 2D position of the spheres present in the RetroSphere controller. RetroSphere utilizes a stereo infrared tracker setup - each equipped with an infrared scene illuminator. Fig. 2 shows a scene illuminated by infrared LED emitters from the stereo infrared cameras and the retroreflective spheres captured in the left and

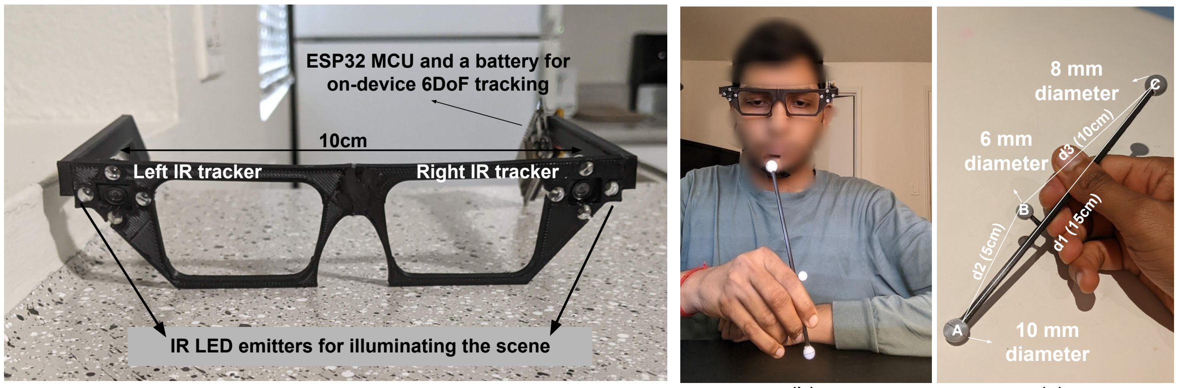

Fig. 3. Example prototypes built with RetroSphere system. (a) A RetroSphere AR glasses prototype with sensors mounted on a 3D-printed frame. (b) A user is wearing our RetroSphere glasses for tracking the 6DoF poses of the RetroSphere stylus. (c) Our RetroSphere stylus consists of three retroreflective markers, forming an obtuse triangle, with each having a different diameter.

the right infrared tracker image. As shown in Fig. 3(a), we attached RetroSphere to an AR Glass-like prototype. The step-by-step workflow of RetroSphere, shown in Fig. 2, is explained as follows:

(1) The user performs 3D input with the RetroSphere stylus in their hand, while infrared LEDs around the cameras illuminate the retroreflective spheres present in our RetroSphere stylus.
(2) The spheres reflect back the infrared illumination, which is captured by the infrared stereo tracking cameras.
(3) The 2D coordinates of the three spheres are estimated by each camera and sent to the ESP32 microcontroller via I2C.
(4) The 2D coordinates of the three spheres from the left and right cameras are matched on the microcontroller.
(5) Our simple depth estimation approach running on an ESP32 microcontroller (explained in section 3.3) triangulates the 3D position (x, y, z) of each sphere on the stylus based on their 2D positions.
(6) Once the 3D positions of all three retroreflective spheres are estimated, we can calculate the 6DoF orientation of the stylus making use of the geometry in which the spheres are placed at the stylus.

The 6DoF tracked RetroSphere stylus can be used in combination with AR SDKs, such as Unity3D, ARCore SDK [6], or ARKit XR [4] to drive highly interactive AR applications on resource-constrained smartphones or AR glasses.

3.2 Hardware details of RetroSphere

RetroSphere hardware - Fig. 3(a) shows the prototype of our Retrosphere-enabled AR glass prototype. RetroSphere uses Stereo PixArt PAJ7025R3 infrared blob tracking cameras [18] which have a very low active current consumption of 6 mA, high object tracking resolution of 4096×4096, and can track up to 16 IR objects at a time. The IR trackers support a maximum sampling rate of 200Hz. However, we sample the IR trackers at 100Hz to save power. We ran an initial study and varied the distance between the trackers from 5cm to 15cm in steps of 1cm and found that both the IR trackers observe all three markers in the Retroreflective stylus reliably between 5cm - 13cm. For distances above 13cm, a few markers were unobserved in either of the cameras and this would greatly affect the performance of our 6DoF tracking algorithm. The IR trackers in our AR glass prototype were separated by a distance of 10 cm. The infrared emitters consist of a circular array of four Optek OP294 [3] infrared LEDs with low current consumption of only 5 mA. Our prototype uses an ESP32 microcontroller [10] with 512 KB

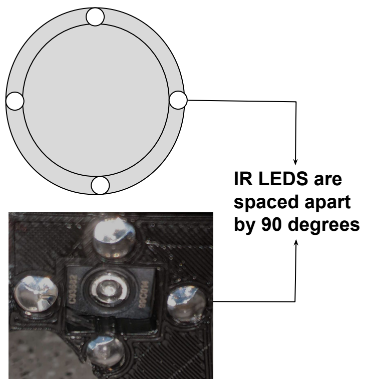

Fig. 4. IR LED positioning in our infrared emitters.

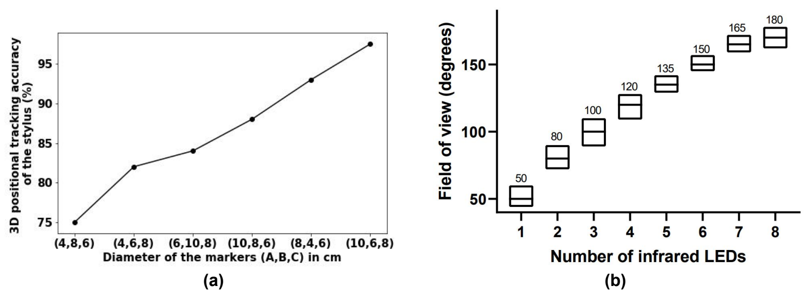

Fig. 5. (a) Diameters of markers A,B,C *vs.* 3D tracking accuracy and (b) Number of infrared LEDs *vs.* field-of-view.

of RAM to perform on-device 6DOF tracking of our RetroSphere stylus. The device is powered via a 1300mAh LiPo battery. The infrared tracking cameras are connected to the ESP32 via I2C ports and powered through 3.3V output from ESP32. The infrared emitters are also powered through the 5V output from ESP32. The AR glass prototype is $14.5 \times 5.05 \times 11cm$ in dimensions and weighs 62 grams.

RetroSphere stylus - The RetroSphere stylus shown in Fig. 3(c) consists of three retroreflective spherical markers (A, B, C) with diameters 10mm, 6mm and 8mm respectively. The stylus is 15cm in length and the distance between A to B (d2) is 5cm and the distance between B to C (d3) is 10cm. The spheres are arranged in the shape of an obtuse triangle. The retroreflective markers are available in diameters of 4mm to 19mm. However, we considered only the four smallest diameters - 4, 6, 8, and 10mm to achieve a smaller form factor. The authors conducted an empirical study that evaluated multiple configurations of marker diameters by varying the size of the markers A, B, and C in the stylus. For each diameter configuration, the 3D tracking accuracy of RetroSphere was measured and the best diameter configuration was chosen as shown in Fig. 5(a). Similarly, the distances between the markers d1, d2, and d3 were chosen through trials to achieve better 6DoF tracking. Although IR trackers in RetroSphere can track retroreflective markers irrespective of diameters, the 6DoF estimation pipeline used in RetroSphere utilizes the geometry of the marker positioning to achieve higher tracking accuracies.

3.3 Infrared LED positioning

In our prototype, we chose to use a circular infrared LED array emitter for illumination since circular LED arrays provide more uniform illumination and a better field-of-view. Here, the field-of-view represents the diagonal dimensions of the measurement field observed by the infrared trackers in the stylus plane. RetroSphere uses fixed IR LED illumination (5mA LED current) and doesn't require special lighting patterns. For determining the number of LEDs required in each infrared emitter array, we arranged eight infrared LEDs in a circular pattern and calculated the approximate field-of-view achieved by the infrared LEDs by switching them on one by one. Fig. 5(b) shows how the field-of-view obtained increases with the number of LEDs. With four infrared LEDs, we achieve our desired field-of-view of 120 degrees which is enough to capture 3D user interactions for smartphone applications. Since the number of IR LEDs also directly affects the power consumption of the device, we choose four LEDs to achieve a satisfactory trade-off between the field-of-view and the power consumption. Fig. 4 shows our final IR LED emitter design in which our infrared LEDs are separated from each other by 90 degrees and achieve a field-of-view of 120 degrees. Each of our IR LEDs is driven by a forward current of 5 mA and can

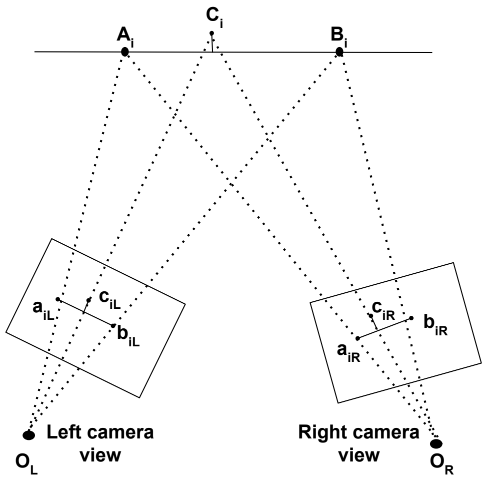

Fig. 6. Illustration of our camera model for tracking our retroreflective stylus.

illuminate our retroreflective controller up to a distance of 120cm. A distance of 1 meter is enough for user interactions with a controller since the average human arm length is only 63.5 cm and ranges between 40cm to 105cm [12].

3.4 RetroSphere 6DoF tracking algorithm

The 6DoF tracking approach used in RetroSphere consists of three stages as follows:

(1) An automatic calibration procedure for calibrating the stereo infrared camera setup.
(2) A simple stereo depth estimation approach to compute the 3D position of the three retroreflective spheres present in the stylus.
(3) A 6DoF estimation algorithm to estimate the orientation of RetroSphere stylus.

3.4.1 ***Automatic calibration Procedure*** - For calibrating the stereo infrared cameras, the most well-known solution would be to use the canonical chessboard pattern-based calibration [26]. However, such a calibration setup is quite cumbersome for novice users to replicate when a calibration drift occurs. Thus, RetroSphere adopts an automatic calibration procedure from Wang *et al.* [61], in which the stereo camera setup is calibrated when the RetroSphere stylus is waved in mid-air for 30 seconds. If in a stereo camera setup, both cameras synchronously observe a 1D object (*e.g.*, a stylus) undergoing at least 6 times general motions, the intrinsic parameters and the relative pose of both the cameras can be calibrated [61]. The original automatic calibration [61] has been designed for RGB cameras and a stylus with three collinear markers on the same straight line. We developed a modified version of the automatic calibration procedure [61] which makes use of just the 3 marker coordinates as well as geometrically non-collinear marker positions locations in our stylus. Fig. 7 shows the high-level overview of our automatic calibration procedure. The steps involved in the calibration procedure are summarized below:
(a) Waving the stylus in mid-air for 30 seconds The user waves the stylus in mid-air for 30 seconds, as the ESP32 continuously observes the 2D positions of three collinear retro-reflective spherical markers (A, B, C) of the RetroSphere stylus from both left and right IR tracking cameras.

(b) RetroSphere camera model - As shown in Fig. 6 representing the standard pinhole camera model of our

RetroSphere hardware, the controller is shown as the line segment ABC (L_{ABC}) with the 2D and 3D marker locations represented as $[x, y]^T$ and $[X, Y, Z]^T$ respectively.

Let a, b, c represent the 3 marker locations of the retroreflective controller being captured by the infrared cameras. Given the image points $\{a_{ij}, b_{ij}, c_{ij} \mid j = 1, 2, ...n, i = L, R\}$ of the retroreflective controller from the left and right infrared trackers under the i^{th} image frame. Our calibration algorithm aims to compute the metric projection matrix under the left camera coordinate system as follows:

$$\begin{cases} \mathbf{P_L^{(e)}} = \mathbf{K_L}|\mathbf{R_L}|\mathbf{t_L} \\ \mathbf{P_R^{(e)}} = \mathbf{K_R}|\mathbf{R_R}|\mathbf{t_R} \end{cases} \tag{1}$$

The left and right camera matrices can be obtained linearly. Firstly, the vanishing points of the markers in the controller are computed. We then compute the infinite homographies between the cameras. Using the infinite homographies, the affine projection matrix and the metric projection matrix can be computed. Unlike existing calibration methods, this method does not require a calibrated base camera and can be done automatically without the need for a calibration board or object.

(c) Affine calibration - As the user waves the stylus in mid-air for 30 seconds, the correspondence of the image points $\{a_{ij}, b_{ij}, c_{ij} \mid j = 1, 2, ...n, i = L, R\}$ can be established by identifying the unique marker size for each marker in our retroreflective stylus. As we know the geometry of the retroreflective stylus, the vanishing points($v_{i,j}$) of the line L_{ABC} in both left and right cameras can be obtained. The simple ratio of the marker positions A, B, and C is given by,

$$\text{simple}(\mathbf{A_j}, \mathbf{B_j}, \mathbf{C_j}) = d_1/d_2 \tag{2}$$

where $d_1 = \|A - C\|$ and $d_2 = \|B - C\|$. The cross ratio of the points $\left\{\mathbf{A_j}, \mathbf{B_j}, \mathbf{C_j}, \mathbf{V_{j\infty}}\right\}$ is also d_1/d_2 where $\mathbf{V_{j\infty}}$ is the infinite point of line ABC. Since the perspective transformation preserves the cross ratio, the vanishing points can be obtained from the linear constraints on v_{ij} as follows:

$$\text{cross}(\mathbf{A_j}, \mathbf{B_j}, \mathbf{C_j}, \mathbf{V_{ij}}) = d_1/d_2 \tag{3}$$

The homographies and the image points can be obtained from the image point correspondence between vanishing points and the infinite homographies. With the homographies and the image points, the projective reconstruction of the 2D points and the camera can be easily computed using the technique of projective reconstruction with planes. Solving the affine camera matrices computed with homographies provides the affine reconstruction of the 2D marker locations $\left\{\mathbf{A_j^{(a)}}, \mathbf{B_j^{(a)}}, \mathbf{C_j^{(a)}}\right\}$.

(d) Metric calibration - The affine projection matrix is used to compute the metric projection matrices. The metric reconstruction of the image points can be solved to compute $\mathbf{K_0}$, the intrinsic parameter of the left infrared camera.

As we already know that $\left\|\mathbf{A_j^{(e)}} - \mathbf{C_j^{(e)}}\right\| = d_1; \left\|\mathbf{B_j^{(e)}} - \mathbf{C_j^{(e)}}\right\| = d_2$, we can obtain the linear constraints for obtaining K_0 as follows:

$$\begin{cases} \left(\mathbf{C}_j^{(a)} - \mathbf{A}_j^{(a)}\right)^T \overline{\omega} \left(\mathbf{C}_j^{(a)} - \mathbf{A}_j^{(a)}\right) = d_1^2 \\ \left(\mathbf{C}_j^{(a)} - \mathbf{B}_j^{(a)}\right)^T \overline{\omega} \left(\mathbf{C}_j^{(a)} - \mathbf{B}_j^{(a)}\right) = d_2^2 \end{cases} \tag{4}$$

where $\overline{\omega} = \mathbf{K_0^{-T}K_0^{-1}}$. We can obtain $\overline{\omega}$ from solving 18 and $\mathbf{K_0}$ is obtained from the Cholesky decomposition of $\overline{\omega}^{-\mathbf{1}}$. From $\mathbf{K_0}(\mathbf{K_L})$, we can obtain the intrinsic parameters $\mathbf{K_R}$, the rotation matrices, and translation matrices using QR decomposition of the metric projection matrix(10).

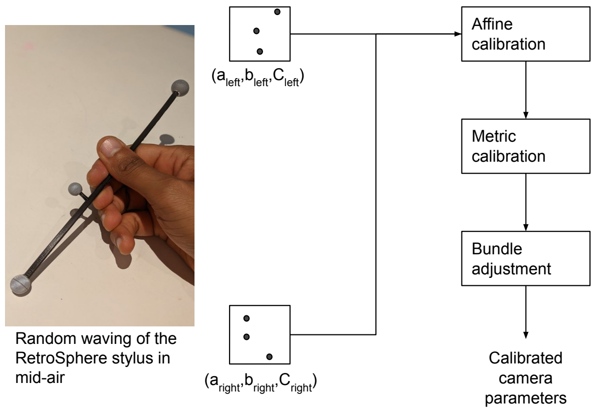
Random waving of the RetroSphere stylus in mid-air
$(a_{left}, b_{left}, C_{left})$
$(a_{right}, b_{right}, C_{right})$
Affine calibration
Metric calibration
Bundle adjustment
Calibrated camera parameters

Fig. 7. Auto-calibration algorithm employed in RetroSphere.

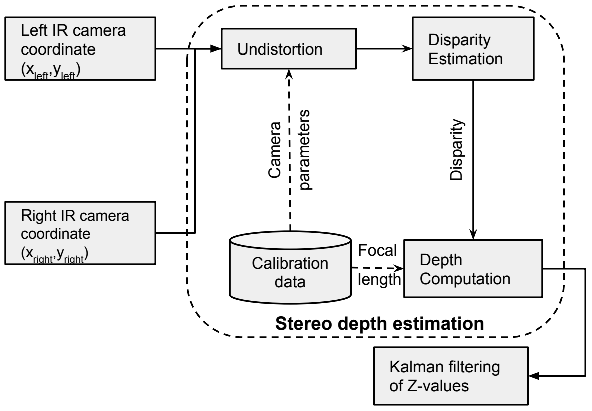
Left IR camera coordinate (x_{left}, y_{left})
Right IR camera coordinate (x_{right}, y_{right})
Undistortion
Disparity Estimation
Camera parameters
Disparity
Calibration data
Focal length
Depth Computation
Stereo depth estimation
Kalman filtering of Z-values

Fig. 8. Overview of RetroSphere's Stereo depth estimation pipeline.

In our current prototype, we make use of the button on ESP32 to begin a calibration session of 30 seconds during which the user waves the RetroSphere stylus in mid-air while they are holding their head still. The automatic calibration procedure achieves an average RMS re-projection error of ≈ 1.4 pixels for a 4096×4096 IR tracker resolution.

3.4.2 ***Stereo depth estimation pipeline*** Fig. 8 shows a high-level overview of our stereo depth estimation pipeline. Firstly, the left and right infrared cameras capture the 2D position of the three retroreflective spheres (A, B, C) present in the stylus with respect to their own camera planes denoted by $(x_{\text{left}}, y_{\text{left}})$ and $(x_{\text{right}}, y_{\text{right}})$ respectively for each marker. For each marker, the IR trackers provide both the 2D coordinates as well as the size of the marker. Since each marker is of a different diameter, the size parameter of each detected marker in the IR tracker helps to successfully match correspondences between each marker.

For each pair of matched markers, $(x_{\text{left}}, y_{\text{left}})$ and $(x_{\text{right}}, y_{\text{right}})$ are undistorted with the help of optimal camera matrix and camera intrinsic parameters obtained from stereo camera calibration. Since there is point correspondence between the two coordinates $(x_{\text{left}}, y_{\text{left}})$ and $(x_{\text{right}}, y_{\text{right}})$, we directly compute disparity between 2D coordinates of the each retroreflective sphere in the left and right infrared camera. The disparity d is computed as the sum of absolute differences between the two coordinates,

$$d = \left\| x_{\text{left}} - x_{\text{right}} \right\| + \left\| y_{\text{left}} - y_{\text{right}} \right\| \tag{5}$$

Once the disparity is obtained, we compute the 3D position of each marker using the following perspective projection equation [35]:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/f & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z/f \end{bmatrix} \tag{6}$$

$$Z = f \cdot B/d \tag{7}$$

where f is the focal length obtained from camera calibration (in pixels), B is the baseline separation between the stereo cameras (in centimeters) and d is the disparity (in pixels). Using the above equations, the disparity can be reprojected back to real-world 3D point (x, y, z). Our depth estimation procedure requires very little compute and can be easily implemented and run on a tiny microcontroller, such as ESP32. At the end of our depth estimation procedure, we will obtain the 3D positions of all the three retroreflective spheres present in our RetroSphere stylus.

Kalman filtering of raw depth values - RetroSphere is designed for user interaction with a smartphone or lightweight AR glasses in which the user is always very close to the infrared cameras. For smooth 3D user interaction, unpredictable jitter caused by sensor noise and depth estimation errors is not desired. Therefore, we filter the raw depth values with a Kalman filter [42], which has been shown to perform well for depth stabilization [23]. Kalman filter works by recursively predicting the next z-value state and correcting it with only the present z-value and a previously measured estimate of the z-value. It does not require any additional history of the system and can run faster in real-time. The Kalman filter further smoothens the depth assuming that the

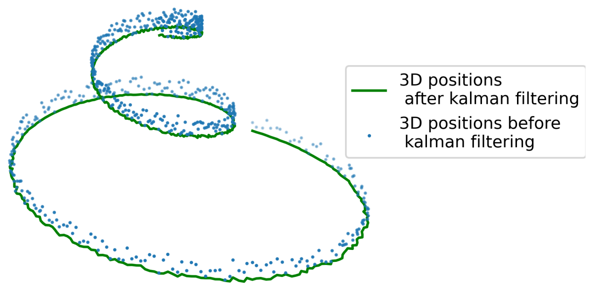

Fig. 9. 3D positions of our controller before and after Kalman filtering of depth values.

depth of the retroreflective stylus changes slightly between two successive frames. Figure 9 shows a 3D plot of our RetroSphere stylus before and after Kalman filtering. We find that the Kalman filter smoothens the depth value and improves the visualization of our stylus movements.

3.4.3 *Estimating 6DoF orientation of the RetroSphere stylus*

From the stereo depth estimation pipeline, we obtain the reconstructed 3D positions of all three retroreflective markers (A, B, C) present in the RetroSphere stylus. We need to measure the rotation and translation matrices to perform 6DoF tracking of the stylus. Theoretically, we know that at least three corresponding marker pairs (an object marker and the corresponding reconstructed marker) must be known for the successful determination of the rotation. In the RetroSphere Stylus, we have 3D positions of exactly three markers thereby aiding in easier 6DoF tracking. The reconstructed markers are then matched with their corresponding object markers using the size as well as the triangular geometry of the markers (A, B, C) in the RetroSphere stylus. We compute the rotation $(\mathbf{R})$ and translation $(\mathbf{t})$ matrices from the three marker positions(real-world coordinate frame) and compare them with the object markers (coordinate frame of the stylus). If for every marker pair

$$||y_i - (\mathbf{R}x_i + \mathbf{t})|| < \text{tolerance}, \ \forall\, i \in \{1, 2, 3\}, \tag{8}$$

then the determined transformation parameters are correct and the 6DoF pose of the RetroSphere stylus could be successfully determined. Here, y_i refers to the 3D position of i_{th} marker and x_i refers to position of the the i_{th} object marker.

3.5 Neural network-based Occlusion correction

With traditional triangulation, all three retroreflective markers must be in view of each camera to estimate the 6DoF orientation of the controller. If even a single marker position is missing due to hand occlusion, the

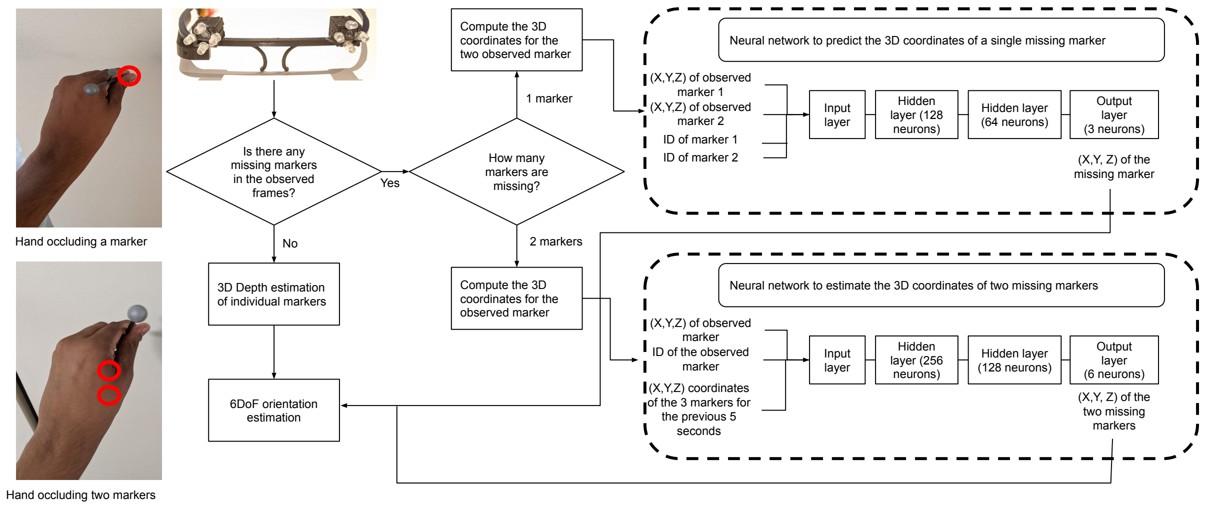
Hand occluding a marker
Hand occluding two markers
Is there any missing markers in the observed frames?
Yes
No
How many markers are missing?
1 marker
2 markers
Compute the 3D coordinates for the two observed marker
Compute the 3D coordinates for the observed marker
3D Depth estimation of individual markers
6DoF orientation estimation
Neural network to predict the 3D coordinates of a single missing marker
(X,Y,Z) of observed marker 1
(X,Y,Z) of observed marker 2
ID of marker 1
ID of marker 2
Input layer
Hidden layer (128 neurons)
Hidden layer (64 neurons)
Output layer (3 neurons)
(X,Y, Z) of the missing marker
Neural network to estimate the 3D coordinates of two missing markers
(X,Y,Z) of observed marker
ID of the observed marker
(X,Y,Z) coordinates of the 3 markers for the previous 5 seconds
Input layer
Hidden layer (256 neurons)
Hidden layer (128 neurons)
Output layer (6 neurons)
(X,Y, Z) of the two missing markers

Fig. 10. Neural network-based occlusion prevention algorithm for estimating the 3D coordinates of markers being occluded by hand.

triangulation algorithms would not be able to provide the 6DoF orientation. As RetroSphere relies on small retroreflective markers, individual markers can be briefly occluded while operating RetroSphere. To enable seamless triangulation, we develop a neural network-based occlusion correction procedure.

The RetroSphere stylus is designed in a way that the outer marker facing the AR glasses can never be occluded. RetroSphere stylus is meant to be used with only one hand like other AR/VR controllers. Under this assumption, there are only cases of marker occlusion to be handled - (a) A single marker is occluded and (b) Two markers are being occluded. We employ a lightweight neural network-based 3D coordinate estimation approach for the occluded 3D markers. Since the geometrical relationship between the markers is well established in our approach, neural networks can very well estimate the 3D coordinates of the occluded markers from the observed/unoccluded markers. Our occlusion correction approach does not support occlusions caused by both hands since our stylus is only meant for single-handed usage. The details of the occlusion correction approach (see Fig. 10) used in RetroSphere are summarized below:

(1) The AR glasses integrated with RetroSphere hardware capture the marker positions and find how many markers are observed in the frame.
(2) ***If all the three markers are visible***, there is no occlusion and the 3D coordinates of all the markers are estimated using our stereo depth estimation pipeline. From the 3D coordinates of the markers, the 6DoF orientation of the stylus can be estimated.
(3) ***If only two markers are visible***, a single marker has been occluded and the 3D coordinates of the two observed markers are estimated using the stereo depth estimation pipeline. We then pass the 3D coordinates of the two observed markers along with their marker IDs to a neural network that estimates the 3D coordinates of the occluded marker. As shown in Fig. 3 (c), the smallest marker (B) is in the middle, the next smallest marker (C) is farther from marker (B) and the larger marker (A) is closer to marker (B). Since we know the 3D coordinates of the two observed markers - (1) If the ratio between the pixel area of the markers is greater than 4, we identify the bigger marker as marker (A) and the smaller marker as marker (C). (2) If the ratio between the pixel area of the markers is between 2 and 3, we identify the bigger marker

as marker (C) and the smaller marker as marker (B). (3) If the ratio between the pixel area of the markers is lesser than 2, we identify the bigger marker as marker (A) and the smaller marker as marker(B). The pixel area thresholds were estimated empirically based on observing the marker coordinates at multiple depths. We use a two-layer fully connected neural network with 128 neurons and 64 neurons. Both the hidden layers use the sigmoid activation function. For the output layer, we employed a linear activation function.

(4) ***If only one marker is visible***, two markers have been occluded and the 3D coordinates of the observed marker are estimated. Given two observed markers, we only need their 3D coordinates to estimate the position of the third marker since the geometry of the RetroSphere stylus allows neural networks to decode the position of the single occluded marker. On the other hand, the same cannot be valid for two occluded markers, since it is hard to estimate the 3D position of the unoccluded markers from the 3D position of a single marker. However, we know the 3D coordinates of markers from the previous time frames and thus, make use of it to predict the projectile of the hand movements to estimate the 3D coordinates of the two occluded markers. We compute the distance between the 3D coordinates of the observed marker and the 3D coordinates of all the three markers observed in the previous frame. The marker ID of the closest marker in the previous frame is assigned to the observed marker since the position of the markers doesn't change significantly between frames owing to the fact that human hand movement speed [27] is much slower than the sampling rate(100Hz) of the PixArt IR tracker.Therefore, we pass the 3D coordinates of the observed marker along with its marker ID and the 3D coordinates of all 3 markers for the previous 5 seconds to a neural network that estimates the 3D coordinates of the two occluded markers. We used a two-layer fully connected neural network with 256 neurons and 128 neurons. Both the hidden layers used the sigmoid activation function. For the output layer, we employed a linear activation function.

Training details - For training both of our neural networks, we collected real-time data from 15 participants. Each participant was asked to wear our RetroSphere-enabled AR glasses mockup and randomly wave the stylus in mid-air for 10 minutes. The 3D coordinates of the markers estimated by the RetroSphere stylus in each frame were sent via Bluetooth to a PC. The data from 10 participants were used to train the neural network and the data from the remaining 5 participants were used for testing its accuracy. We discarded all the time frames where markers were missing. The participants recruited for this study were not invited for subsequent evaluations in section 4.1.1.

For training the neural network for correcting single marker occlusion, we randomly dropped one of the markers in each 3D coordinate and passed it as input to the neural network. The maximum occlusion duration was 69s and 72s in the training and test data respectively. The input and output pairs will be the 3D coordinates of the two markers along with their marker IDs and the corresponding 3D coordinate of the dropped marker. During testing, we found out that our neural network model obtained **98.2%,98.6%, and 97.4%** accuracy in predicting the X, Y, Z coordinates respectively of the occluded/dropped marker. For training the neural network for handling two marker occlusion, we randomly dropped two of the markers in each 3D coordinate and passed them as input to the neural network. The input and output pairs will be [3D coordinates of the marker along with its marker ID + 3D coordinates of all 3 markers from the image frames from the previous 5 seconds, the corresponding 3D coordinates of the two dropped markers]. During testing, we found out that our neural network model obtained **95.2%, 95.6%, and 94.2%** accuracy in predicting the X,Y,Z coordinates respectively of the occluded/dropped markers. The models were trained offline using TensorFlow and the trained models were deployed on an ESP32 microcontroller using the TF-Lite library to perform online neural network inference. The high efficiency of our occlusion correction approach is demonstrated against ground truth LIDAR data in section 4.

It is very evident that the compute involved in RetroSphere's 6DoF tracking solution is very sparse and thereby takes only 15 ms (achieving a 66 FPS 6DoF tracking) to run on an ESP32 microcontroller, achieving a low latency desirable for an AR 3D input.

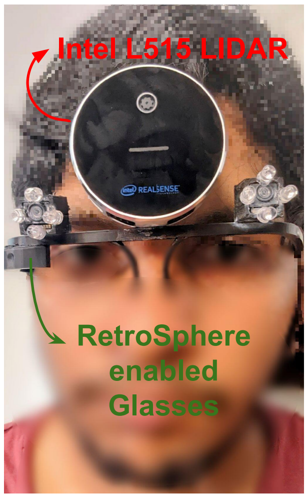

Fig. 11. Experimental setup of RetroSphere-enabled glasses-like prototype attached with an Intel L515 LIDAR sensor for ground truth 3D measurements.

4 Results and Evaluation

4.1 Tracking accuracy

4.1.1 ***Evaluation Procedure*** We evaluate the tracking accuracy of RetroSphere using data collected from users who we invited to try our RetroSphere-enabled AR glasses prototype. We do this to provide a practical estimate of RetroSphere's performance under real-world conditions involving different head sizes, flexibility, and changes in environmental factors. We recruited 20 participants (11 M, 4 F) with different head sizes to wear our AR glasses prototype while relaxing in a room to interact with a Unity-based 6DoF visualizer on the monitor. The head circumferences of the participants ranged between 55-62 cm and our AR glasses mockup fit all the participants comfortably. The hand length ranged between 61-99 cm with an average hand length of 65 cm. Five recruited participants had hand lengths greater than 90 cm.The AR glasses mockup was calibrated once using our auto-calibration procedure before the experiments and the participants were not required to calibrate the AR glasses.

Fig. 11 shows our experimental setup in which the user is wearing RetroSphere-enabled AR glasses and a small Intel L515 LIDAR (weighing 360g), which provides the required ground truth 3D information of the retroreflective controller to evaluate RetroSphere. The AR glasses prototype in Fig. 11 is slightly modified from the actual prototype in Fig. 3(a) to make sure the LIDAR sensors can be held firmly on the prototype. The distance between the IR trackers is still kept at 10cm to be similar to the actual prototype.The LIDAR Camera is placed on the same vertical axis as the AR glasses with our RetroSphere cameras. We used a combination of blu tac and glue to secure the LIDAR sensor on our modified glass prototype.

The data collection was carried out for 15 minutes with each participant. To ensure there is sufficient battery life, the AR glasses were powered with a 1300 mAh LiPo battery. The participants were asked to naturally move their controllers for the first 5 minutes while being explicitly asked to reach the maximum possible distance between the AR Glasses on their heads and the controller on their hands. During the remaining 10 minutes, participants spent 3 minutes on the 3D user interface application (Fig. 22(d)), 3 minutes on the drawing application (Fig. 22(a)), and 4 minutes on the mid-air visualization application (Fig. 22(c)). After the data collection, the participants also

6DoF Parameters	Mean error	6DoF Parameters	Mean error
X (mm)	3.2	**Pitch (deg)**	4.65
Y (mm)	4.3	**Yaw (deg)**	6.95
Z (mm)	12	**Roll (deg)**	4.85
Position tracking error (mm)	18.5		
Orientation tracking error (deg)	5.85		

Table 2. The average position and orientation error among the 20 participants.

Depth sensing controllers	Average error
Kinect [39]	1% (4 cm)
Magic Leap [24]	1% (5 cm)
Intel RealSense D435 [9]	2 mm
POL360 [37]	0.691 cm
Oculus Quest [55]	3.5 mm
RetroSphere	**2.4% (~1.2 cm)**

Table 3. Depth-sensing accuracy of RetroSphere against state-of-art 3D input tracking.

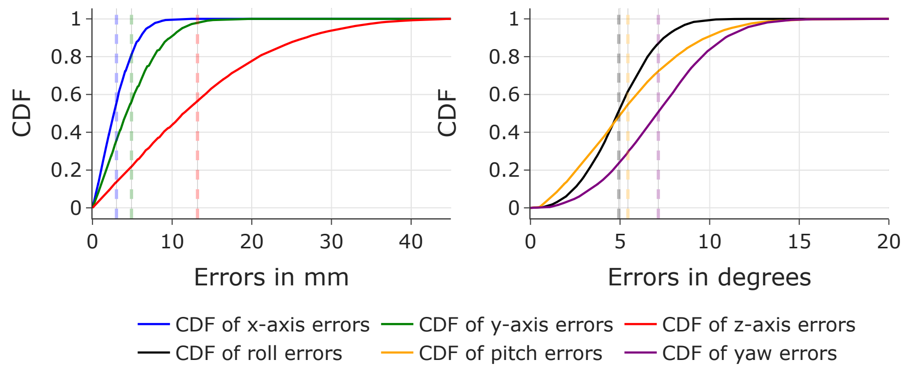

Fig. 12. CDF of 6DoF position of the controller based on data from 20 participants. The dotted lines represent the mean error in this graph.

tried out mid-air drawing with the AR glasses prototype for 5 minutes in three different environments - (1) an indoor environment with natural lights on, (2) an outdoor environment while standing/sitting, and walking, and (3) a dark room with lights off. During this time, the ground-truth 6DoF pose of the controller estimated from the LIDAR and the 6DoF pose of the controller measured by RetroSphere was recorded by a Python program. Altogether, this resulted in 3.5 hours of synchronized 6DoF tracking measurements of the ground truth LIDAR and RetroSphere.

4.1.2 Results

• **Mean errors :** We compared the 6DoF controller tracking accuracy of RetroSphere against the ground truth 6DoF pose obtained from the Intel L515 LIDAR. Table 2 shows the average 6DoF tracking errors of RetroSphere against ground truth 6DoF measurements. Positional tracking error is the euclidean distance between (x, y, z) measurements from ground truth and RetroSphere whereas Orientation tracking error is the euclidean distance between rotation matrices from ground truth and RetroSphere. The results show that RetroSphere has a mean error of 18.5 mm and 5.85 degrees for position and orientation respectively. Depth(Z) errors are higher than the X, and Y errors since the IR trackers are known for accurate tracking of 2D coordinates of the markers. On the other hand, depth tracking accuracy heavily depends on our low compute depth estimation pipeline. The orientation errors are slightly higher because the RetroSphere stylus uses only three retroreflective markers to estimate orientation.

• **CDF of errors :** The complete error distribution of the 6DoF parameters $(x, y, z, \text{roll}, \text{pitch}, \text{yaw})$ using data

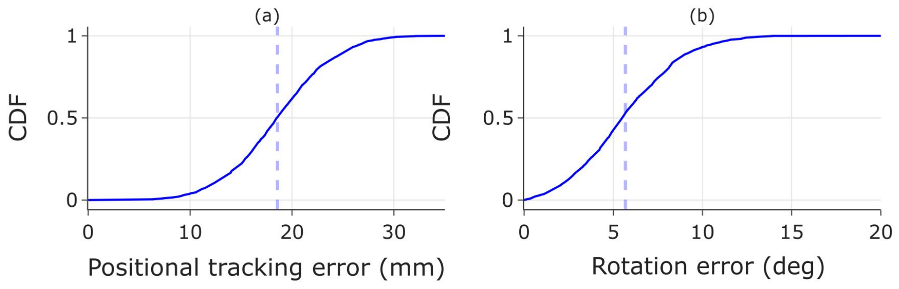

Fig. 13. CDF of position and rotation errors of the controller based on data from 20 participants. The dotted lines represent the mean error in this graph.

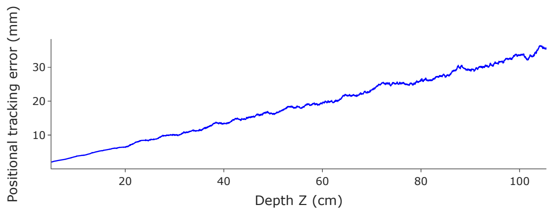

Fig. 14. Euclidean positional errors in Cartesian coordinates between RetroSphere and LIDAR ground truth measurements based on data from 20 participants.

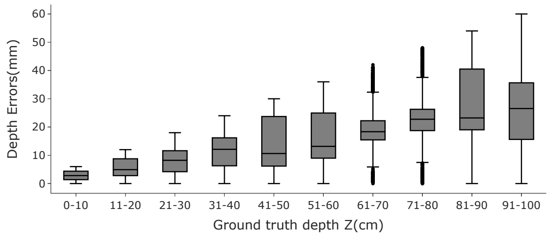

Fig. 15. Box-plots of depth errors against ground truth depth measurements from LIDAR based on data from 20 participants.

aggregated from all 20 participants is expressed using a cumulative distribution function shown in Fig. 12. Fig. 13 shows the cumulative distribution of position and orientation errors The CDF shows that the mean position and orientation tracking error of RetroSphere is very small and well within the acceptable range.

• **Depth errors:** RetroSphere also performs very well at depth (z) estimation. Fig. 15 shows the box plot of depth errors in RetroSphere against ground truth depth measurements ranging from 0 to 100cm. We observe that the errors increase somewhat linearly, indicating that RetroSphere performs almost equally across the full sensing range of 0 to 100cm. We found that the average depth error ranged between 1.5-2.5% - with a worst-case error of 55mm (5.5%) and a mean error of 22mm (2.2%) at the maximum depth of 100 cm.

• **Positional tracking error:** Fig. 14 expresses the positional tracking error as a function of the ground truth depth (Z). We see that the positional tracking error also increases linearly as the depth. This finding reaffirms that RetroSphere performs equally across the full sensing range of 0-100 cm. In addition, Fig. 14 helps developers choose a minimum size for buttons, sliders, dropdowns, or objects based on the desired depth in the 3D virtual UI to prevent any failure while interacting with the RetroSphere controller.

• **Relative motion tracking:** Although tracking accuracy is a very significant metric of RetroSphere's performance, it is also important for RetroSphere to track relative motion especially when RetroSphere is being used as an input device. Fig. 16 shows the ground truth and the estimated position of the controller over a time window of 3 minutes for two different participants. The data of participant 1 on the left has a mean position tracking error of 3.3 mm whereas the data of participant 2 on the right has a slightly higher position tracking error of 4.6 mm. Although the errors are slightly higher for participant 2, we can still observe that the relative motion tracked by RetroSphere still aligns extremely well with the ground truth measurements. We also visualized random mid-air movements with the RetroSphere stylus made by a single participant. Fig. 21 shows 3D plots of four of

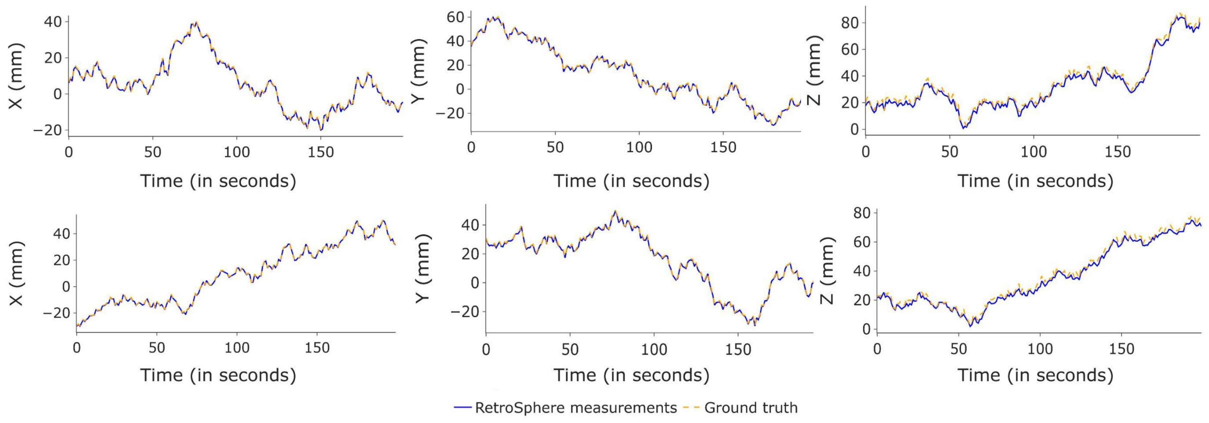

Fig. 16. Comparison of 3D positional tracking between RetroSphere and ground truth LIDAR for three minutes by two different participants.

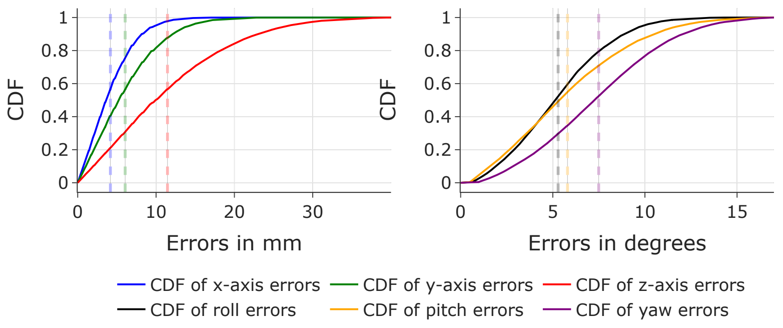

Fig. 17. CDF of 6DoF position of the controller from frames where occlusion has occurred. The dotted lines represent the mean error in this graph.

the mid-air interactions (made by a single participant) measured by LIDAR and RetroSphere. We observe that 3D data obtained from RetroSphere has very high correspondence with the 3D coordinates obtained from LIDAR.

• **Efficieny of neural network-based occlusion correction :** To demonstrate the efficiency of our occlusion prevention algorithm, we detected all the frames from RetroSphere where one or more markers were occluded. We found out that approximately 5% of the data collected from our user study had one or two markers occluded by hand occlusion. We then compared the 6DoF position/orientation estimated by our neural network-based occlusion correction approach against ground truth LIDAR measurements for all the frames where occlusion had occurred. Fig. 17 shows the CDF of errors for all the 6DoF parameters estimated by our neural network-based occlusion correction approach. We see that the mean errors being observed in the CDF are very similar to the CDF of errors obtained with all the data as shown in Fig. 12. We found a maximum and average occlusion duration of 75s and 25s respectively, in the collected data. The 3D positional tracking accuracies were around 90-95% with an occlusion duration of 55s. However, the accuracy dropped to 88-93% and 86-91% for occlusion duration between 55-67s and 67-75s respectively. Although our occlusion correction approach can still guarantee better accuracies at longer occlusion duration, the residual errors from the previous frames slightly reduce the accuracy under longer occlusion durations.

• **Effect of environmental changes on RetroSphere's tracking accuracy:** To study the accuracy of Ret-

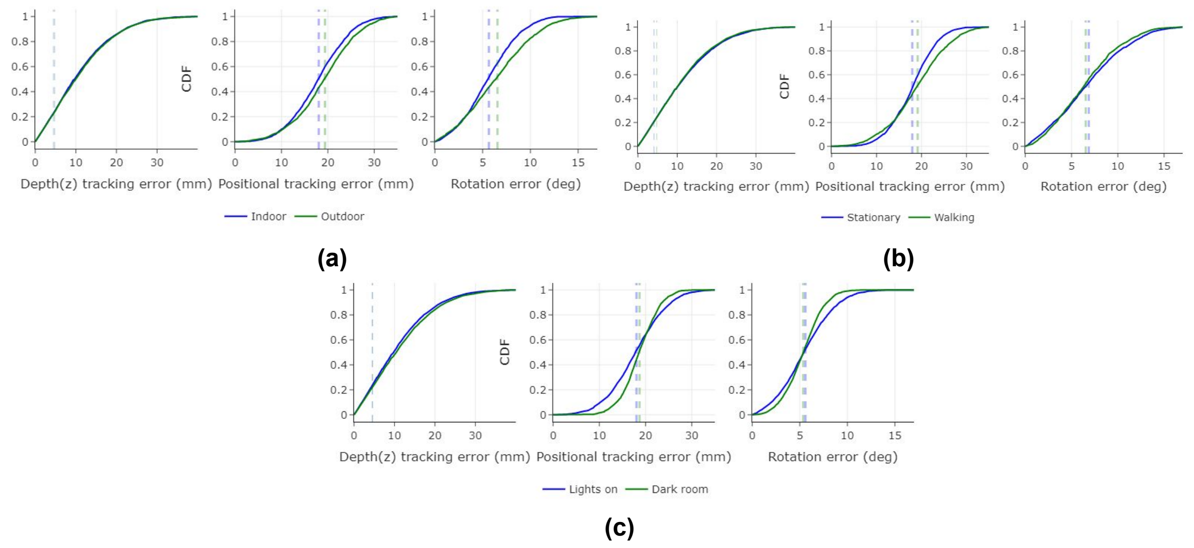

Fig. 18. CDF of Depth(z), position and orientation errors of the controller under various environmental conditions for 20 participants. The dotted lines represent the mean error in this graph. (a) Stationary *vs.* walking (b) Indoor *vs.* outdoor and (c)Lights on *vs.* Dark room.

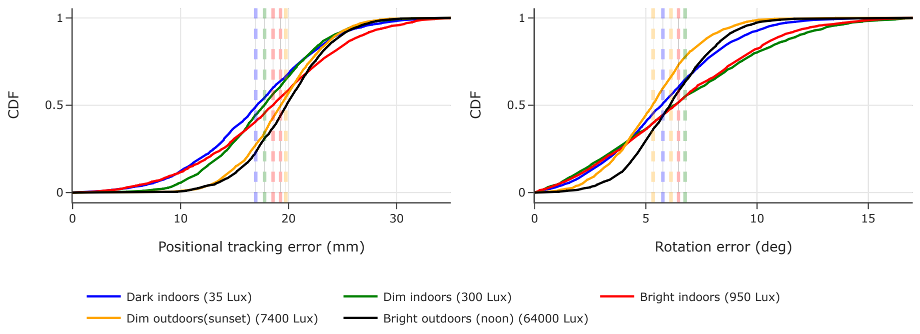

Fig. 19. CDF of position and orientation errors of the controller under various lighting conditions for 10 participants.

roSphere's tracking under varying environments, we collected data from all the 20 participants under four conditions: (1) an indoor room with lights on (*Indoor/Lights on*), (2) an outdoor location while sitting/standing (*Stationary*), (3) an outdoor location while walking (*Walking/Outdoor*) and (4) a dark room with lights off (*Dark room*). Figure 18 shows the CDF of position and orientation errors of 6DoF parameters estimated by RetroSphere against ground truth LIDAR measurements under all four environments considered in our user study. We can observe that the CDF is similar for all four environments. RetroSphere works well in both stationary indoor/outdoor environments as well as moving environments.

• **Effect of lighting conditions:** To demonstrate the efficiency of RetroSphere's tracking accuracy under

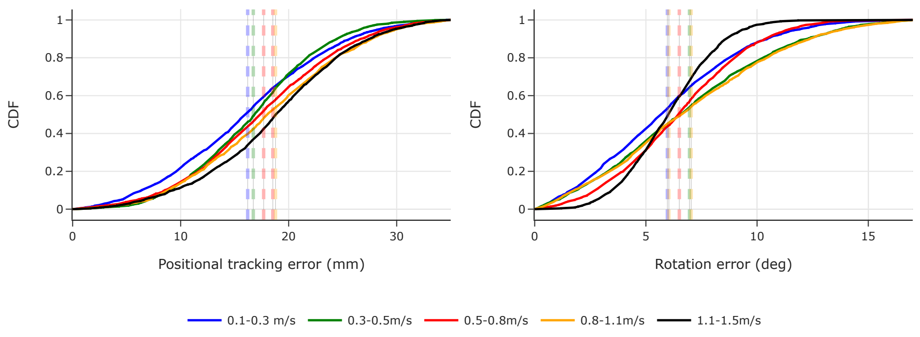

Fig. 20. CDF of position and orientation errors of the controller under various hand velocities for 10 participants.

varying lighting conditions, we collected data from 10 participants. Each participant used RetroSphere under five conditions for five minutes each: (1) dark indoor room at night (≈ 35 lux), (2) dim indoor room with lights off in evenings(≈ 300 lux), (3) bright indoor room with lights on (≈ 950 lux), (4) dim outdoors in the evening (≈ 7400 lux) and (5) bright outdoors during the noon (≈ 64000 lux). The light intensities were measured using the ambient light sensor using a Google Pixel 3 XL phone. Fig. 19 shows the CDF of position and orientation errors of 6DoF parameters estimated by RetroSphere against ground truth LIDAR measurements under all five lighting conditions considered in our user study. We observe that the CDFs are very similar for all five lighting conditions indicating the robustness of IR cameras under various lighting conditions.

• **Effect of hand movement velocities:** To study the accuracy of RetroSphere's tracking under different hand movement speeds, we collected data from 10 participants for 5 minutes. During data collection, we asked the participants to hold the RetroSphere stylus and move their hands - beginning with slow hand speed and steadily progressing towards their maximum hand speed. The participants were asked to wear an Empatica E4 wristwatch to collect IMU data from which ground truth hand velocities can be estimated. The hand speeds ranged from 0.1-1.5m/s. Figure 20 shows the CDF of position and orientation errors of 6DoF parameters estimated by RetroSphere against ground truth LIDAR measurements under different hand movement speeds. This is due to the fact that the 66 FPS tracking rate of RetroSphere is faster than the hand movement speed.

• **Maximum tracking range supported by RetroSphere:** We placed the RetroSphere glasses frames stationary on the desk and kept moving the stylus away from the RetroSphere glasses to identify the maximum tracking distance. We found that the maximum tracking distance currently supported by RetroSphere stylus is up to 150cm. After moving the stylus beyond 150cm, the IR trackers were unable to track the markers since the IR illumination provided by the IR LEDs is lost at longer distances.

4.2 Comparison of RetroSphere depth errors against state-of-the-art AR/VR controllers

Table 3 shows RetroSphere's depth errors against depth errors obtained from state-of-the-art AR/VR controllers. Although the depth errors of RetroSphere may be slightly higher than those of other state-of-the-art VR/AR controllers, RetroSphere consumes very little power and can be easily retrofitted to lightweight devices. Kinect [39] uses an infrared laser projector to measure depth with structured light patterns incurring a high power consumption of 5 Watts. Magic Leap [24] has a dedicated depth sensor and infrared emitters resulting in high power consumption. The Intel RealSense Depth Camera [9] consumes around 3.5W. The light source

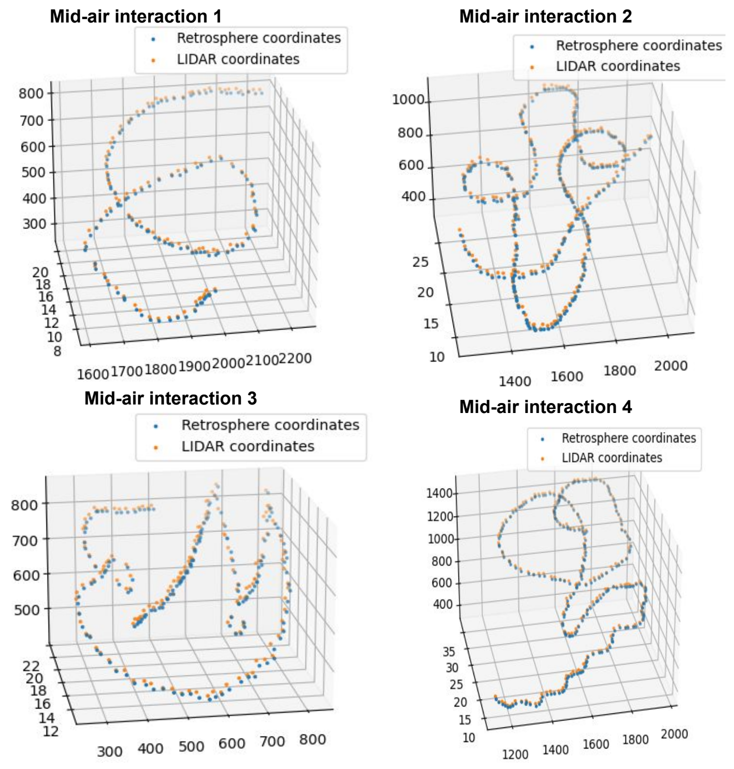

Fig. 21. 3D points cloud captured with RetroSphere and Intel RealSense LiDAR L515 (ground truth).

required for POL360 [37] would impose a 1 Watt power overhead. Hand tracking in Oculus Quest [55] supports only a battery life of 2 hours. On the other hand, RetroSphere achieves an average depth error of as low as 1.2 cm (2.4%) over a tracking range of 100 cm while consuming only 400mW. This includes the power consumption of the tracking cameras, IR LEDs, and the microcontroller.

4.3 Latency and Power Consumption of RetroSphere

The RetroSphere controller is completely passive and free of any electronics, while the smartphone sensor attachment is powered through 5V USB output. RetroSphere has three main components. The power consumption is as follows:

(1) Stereo infrared illuminators, each with 4 IR LEDs. OP294 IR LEDs consume around 5 mA. In total, eight IR LEDs consume around 40 mA.
(2) Stereo Pixart PAJ7025R3 infrared tracking cameras consume about 8 mA with each camera having a current consumption of 4 mA@200Hz.
(3) ESP32 microcontroller consumes ≈30 mA @ 80MHz.

• **Power :** We measured the power consumption of RetroSphere using a Monsoon power monitor. RetroSphere consumes 80 mA during operation and 2 mA during the idle state. During the idle state, the ESP32 goes into a light sleep with an average current consumption of 0.8 mA. Therefore, RetroSphere only consumes 80 mA X 5 Volts = 400 mW during operation and can be easily used as an off-the-shelf hardware attachment for lightweight AR devices.

	Power Consumption (mW)				**Latency (ms)**			
	Input Resolution (Megapixels)	**Power (mW)**	**Mask kernel pixel width**	**Power (mW)**	**Input Resolution (Megapixels)**	**Latency (ms)**	**Mask kernel pixel width**	**Latency (ms)**
Dodecapen / ARPen	0.2	4200 (10X)	2	4200 (10X)	*0.2*	22	*2*	*22*
	0.4	4800 (12X)	4	4325 (11X)	0.4	29	4	26
	0.6	5000 (12.5X)	6	4500 (11X)	0.6	36	6	29
	0.8	*5400 (13.5X)*	*8*	*4600 (11.5X)*	0.8	42	*8*	34
	1.0	5600 (14X)	10	4800 (12X)	1.0	48	10	38
RetroSphere	***Idle Mode***		***0***		***15***			
	Active mode		***400***					

Table 4. Comparison of RetroSphere and ARPen in terms of power and latency. RetroSphere is at least 10X power-efficient and 1.5X faster than ARPen.

RetroSphere consumes very little power compared to the mainstream commercial VR headsets. As for passive 3D input solutions for smartphones, Dodecapen [64] and ARPen [59] achieve the state-of-art performances. However, ARPen requires a smartphone camera and associated image processing compute for tracking a pen-like stylus with cubic fiducial markers. To estimate the power consumption of ARPen, we ran the open-source ARPen application on an iPhone 12 and measured only the power consumed by the smartphone's CPU and camera for ARPen tracking under different tracking configurations. Table 4 shows that RetroSphere achieves $\frac{1}{12.5}\times \sim \frac{1}{10}\times$ power compared to ARPen.

• **Latency :** RetroSphere's 6DoF tracking runs on-device on the ESP32 microcontroller running at 80MHz. The time consumed by the 2 stages in RetroSphere 6DoF tracking are- (a) 3D position estimation of the three retroreflective markers present in the stylus takes 9.5 ms and (c) 6DoF estimation of the stylus consumes 5.5ms. In total, RetroSphere achieves a low latency of only 15 ms (supporting a frame rate of 66 FPS similar to the 60 FPS tracking rate supported by Oculus Quest 2[2]). We ran ESP32 at a low clock frequency to save power and latency can be improved further by running the microcontroller at a higher clock frequency. Table 4 shows that ARpen achieves a minimum latency of only 22ms while running on a powerful smartphone. The sparse computation involved in RetroSphere 6DoF tracking helps in achieving better latency even when being run on a microcontroller.

5 Applications and Use-cases

5.1 AR glasses application Demonstrations with RetroSphere

RetroSphere can provide low-cost and reliable 3D input in a self-contained form factor, which is well suited for AR applications running on a smartphone or AR glasses. In this section, we demonstrate a few Unity applications that run on a laptop while our RetroSphere-enabled AR glasses continuously track the 6DoF position of the RetroSphere stylus and send them to the laptop via Bluetooth at a tracking rate of 66 FPS. The BLE connection was configured to provide a throughput of 1MB/s. In the future, we could use RetroSphere's output with applications running directly on AR glasses.

5.1.1 ***AR 2D/3D drawing*** RetroSphere can transform any flat surface into a 2D digital drawing or writing canvas by leveraging the distance between the stylus and the closest surface. Fig. 22 (b) demonstrates a Unity application in which the user writes on the flat surface of a table with our RetroSphere stylus and the Unity application running on the laptop visualizes the 6DoF position of the stylus along with the digitized text "whiteboard".

[2]Hand tracking in Oculus Quest 2: https://uploadvr.com/quest-2-high-freqency-hand-tracking/

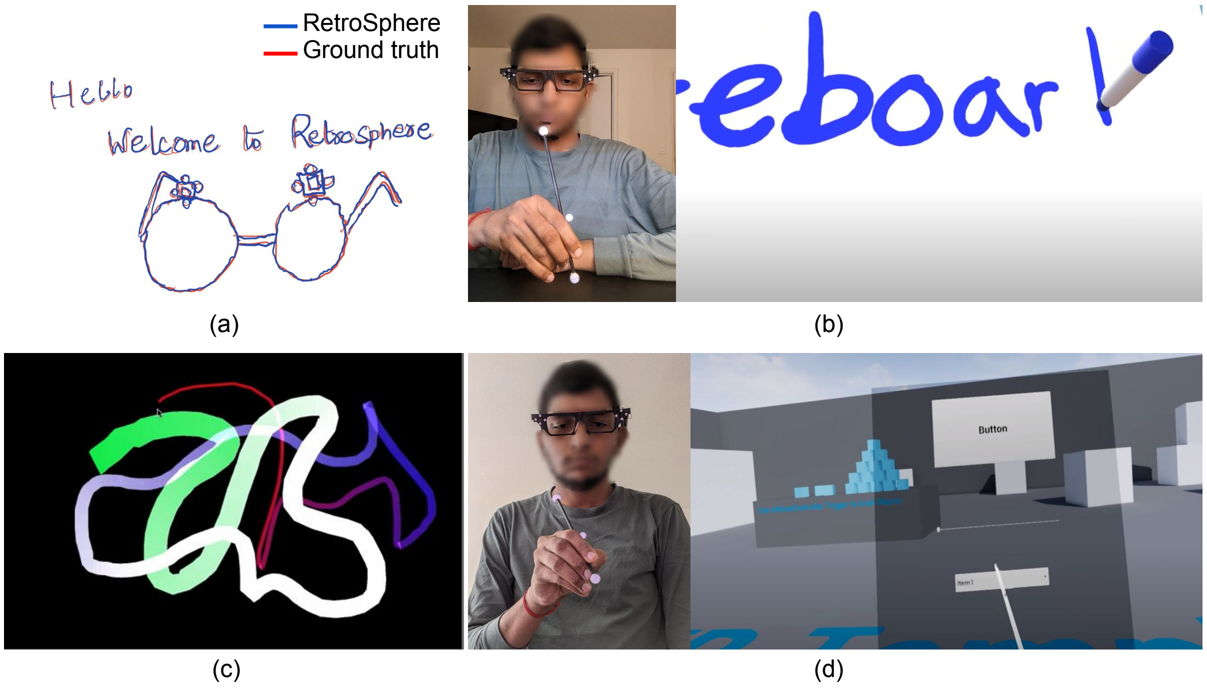

Fig. 22. (a) A hand-written note taken using RetroSphere overlaid against the ground truth note. (b) RetroSphere can enable the user to take notes or draw on almost any flat surface or a wall. (c) Unity visualization app for visualizing 3D mid-air drawings when using RetroSphere. and (d) RetroSphere can enable AR glasses to interact easily with 3D user interfaces in the virtual world.

Fig. 22 (a) shows a sample digital note taken by RetroSphere against the ground truth note and we can observe that they have very high correspondence, Since this application only requires 2D input, the depth is kept fixed at the writing surface.

Digital drawing can also be extended to free-space 3D and used in XR applications such as Geollery [29, 30] and CollaboVR [36]. RetroSphere provides absolute 6DoF position tracking of the controller relative to the sensor, thus allowing the user to paint or write at different depths. Fig. 22 (c) demonstrates an application in which the user can perform mid-air drawings and the Unity application visualizes strokes made using the RetroSphere stylus with depth represented using a depth colormap. In the future, RetroSphere could also enable volumetric 3D sculpting [34] for drawing 3D cartoons and objects.

5.1.2 ***3D user interfaces*** 3D input can be used to press 3D buttons and control other spatial elements, such as sliders and dials. Fig. 22 (c) shows a Unity application that places buttons, sliders, and dropdowns at different depths in a virtual room and the user makes use of his RetroSphere stylus to interact with the 3D user interface. We make use of the rate of change in the depth to identify button press or select a slider/dropdown menus using the RetroSphere stylus. Fig. 14 shows the average positional tracking errors at multiple depths and developers can make use of Fig. 14 to determine the minimum size of 3D UI elements based on their desired depth placements - thereby getting rid of stylus interaction failures. For instance, if a developer wants to place a slider at 50cm depth in the UI, he/she can keep the size of the slider to be at least 1.5×1.5 cm since the positional tracking errors were 1.5cm at a depth of 50cm.

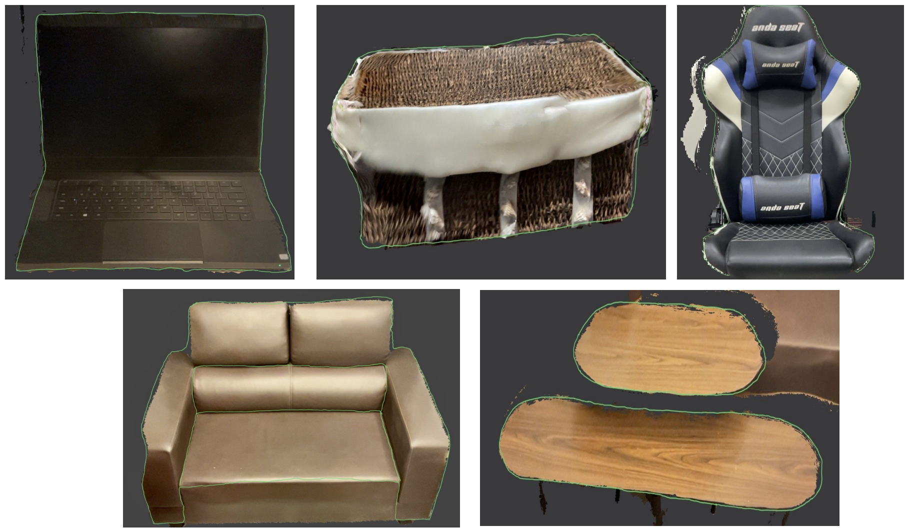

Fig. 23. Shape tracing of various objects by RetroSphere and a LIDAR for ground truth. The green line represents the edges traced by the RetroSphere controller and the ground truth 3D image is obtained from the LIDAR camera.

5.1.3 ***3D AR measurements*** With the same experimental setup as shown in Fig. 11, we made use of the RetroSphere stylus to trace the shape of indoor objects by moving the RetroSphere stylus along the edges of the object. The 3D coordinates of the RetroSphere stylus were measured by LIDAR and RetroSphere simultaneously. We recruited 3 participants to use our RetroSphere stylus to trace the shapes of 5 objects. The 8mm marker in the RetroSphere stylus was used to trace the shape of the objects. Fig. 23 shows 3D plots of the object shapes traced by the RetroSphere stylus overlaid on the ground truth 3D image obtained from the LIDAR. Again, we find that the 3D shape traced by RetroSphere highly corresponds with the 3D shape of the object. Fig. 24 shows a box-plot of RetroSphere's positional tracking errors against LIDAR ground truth depths. We observe that the depth errors increase as the size of the objects increase (from a small laptop to a big table). The average depth error for the table was close to 6.5 mm whereas the average depth error for the laptop was 1.4 mm. The worst-case depth error

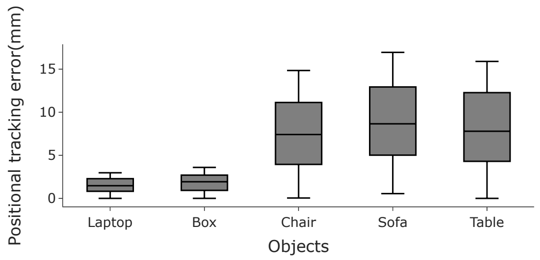

Fig. 24. Positional tracking errors in millimeters measured against LIDAR readings.

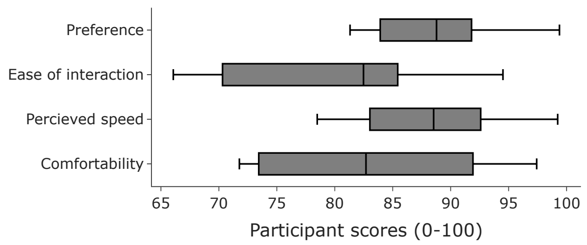

Fig. 25. Subjective measures - comfort, perceived speed, ease of interaction, and user preference for RetroSphere from 20 participants.

was ≈ 1.57 cm which occurred for the sofa. This shows that RetroSphere can easily trace 3D object shapes with high accuracy and can serve as a virtual tape for 3D measurements of real-world objects.

5.2 Qualitative user study

We conducted a short, preliminary qualitative user study with 20 participants (14M, 6F) to score RetroSphere on a scale of 1-100 on four measures - preference, ease of interaction, comfort, and perceived speed. The participants were asked to wear our RetroSphere-enabled AR glasses prototype and try out the various desktop based Unity AR applications we had demonstrated earlier in section 5.1. The participants tried each of the applications for 2 minutes. The participants were then given an Oculus Quest 2 to wear and try out Tilt Brush [3] 3D painting app for 5 minutes. We chose TiltBrush since it is one of the better known VR painting apps and seemed suitable for the comparison with our drawing/writing experiences in RetroSphere. In our post-study questionnaire, the participants were asked to provide scores between 0-100 with a score of 0 meaning the participant strongly disagrees that RetroSphere is better than Oculus Quest 2 and a score of 100 meaning the participant strongly agrees that RetroSphere is better than Oculus Quest 2. The results of our post-study questionnaire are presented in Fig. 25.

• **Preference**: We asked the participants to rate their preference of RetroSphere over Oculus Quest 2 for everyday use. The study revealed that RetroSphere was significantly more preferred (mean score = 90) than an Oculus Quest 2. This was expected because users prefer to wear glasses/tiny controllers over bulky controllers/headsets.

• **Ease of interaction**: We also studied the ease of pointing, by asking participants to rate how easier it was to point at a button/slider/dropdown in the 3D user interface with RetroSphere. We found that it was relatively easier (mean score = 83) to point at a 3D UI item with the RetroSphere stylus.

• **Perceived speed**: We asked participants to rate their perception of the RetroSphere stylus movements. It was revealed that RetroSphere was quite fast (mean score = 90).

• **Comfort**: A good AR input device should be comfortable enough for users to carry along everywhere. Here, we asked the users to rate the comfort of RetroSphere against Oculus Quest 2. The study results show that the users found RetroSphere relatively comfortable to use with a mean score of 86.

We acknowledge that our current user study is very preliminary since we still haven't demonstrated true AR/VR experiences with RetroSphere unlike Oculus Quest 2 where the VR experiences are viewed directly on the headset. In future, we plan to conduct larger scale user studies with RetroSphere attached to AR glasses for a fair comparison to commercial headworn devices.

6 Application prototypes designed with RetroSphere

In this section, we present three application prototypes realized with RetroSphere hardware for everyday applications.

6.1 **RetroPen - A pressure-sensitive retroreflective marker based AR pencil**

We explored the design of a pressure-sensitive passive AR pencil, 'RetroPen'. Fig. 26 (a) shows our RetroPen hardware that can be used with AR glasses with the RetroSphere tracker. RetroPen consists of two cylindrical tubes - an inner tube and the outer tube. One end of the inner tube is attached to a retroreflective marker (marker 1 with 3mm radius) and the other end is inserted into the outer tube. One end of the outer tube is attached to a bigger retroreflective marker (marker 2 with 6mm radius)). There is a spring inside the outer tube to facilitate the inner tube to slide down when the pen is pressed. Both the markers are of different sizes. The distance between

[3]"TiltBrush" VR painting app - https://www.tiltbrush.com/

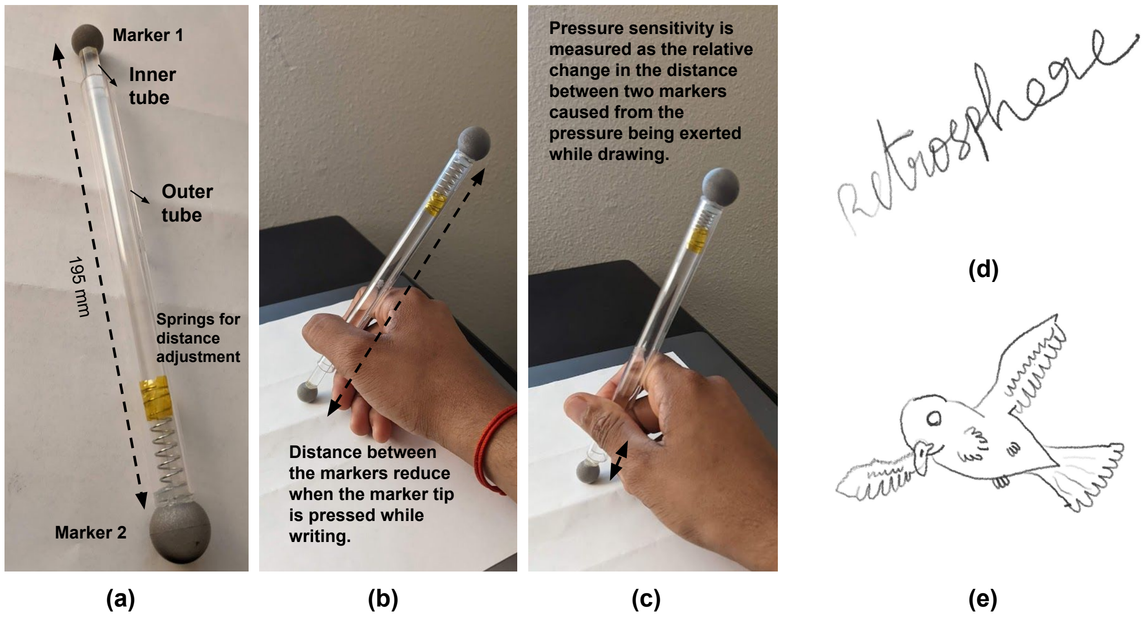
Marker 1
Inner tube
Outer tube
195 mm
Springs for distance adjustment
Marker 2
Distance between the markers reduce when the marker tip is pressed while writing.
Pressure sensitivity is measured as the relative change in the distance between two markers caused from the pressure being exerted while drawing.
Retrosphere
(a)
(b)
(c)
(d)
(e)

Fig. 26. (a) RetroPen - AR Pencil prototype for seamless AR drawing experiences. (b) When RetroPen is pressed on the paper for writing, the distance between the two markers changes indicating that the user has started to write. (c) RetroPen provides pressure sensitivity by calibrating hand pressure against the distance between the markers. (d) The word 'RetroSphere' written with increasing hand pressure while writing with RetroPen. The strengths of the pressure are visualized as the stroke intensities. (e) A bird and text drawn with our RetroPen prototype.

the markers in our current prototype is 195mm in our current prototype. We summarize the features supported by RetroPen below:

(1) **Writing tip and eraser** - Since both the markers are of unique size, RetroPen uses marker 1 for writing and marker 2 for erasing. The user can quickly flip the pen to erase and flip back to use marker 1 to start drawing/writing.
(2) **Writing/Drawing detection** - As the user presses the pen against the surface/paper for writing, the inner tube is pushed against the spring causing a slight decrease in the distance between the two markers as shown in Fig. 26 (b). This distance change is picked up by our RetroSphere which can uniquely identify the 3D coordinates of the two markers and detects writing to open the appropriate application on a mobile phone or AR glasses.
(3) **Pressure sensitivity** - In RetroPen, Hand pressure is proportional to the distance between the markers since the higher the hand pressure, the lesser the distance between the markers would be. Our RetroSphere hardware can easily detect the distance between the markers and calibrate it against stroke intensity. The pressure variations from low to high while writing the word "RetroSphere" is visualized with varying stroke intensities using our Unity-based PC software in Fig. 26 (d). The 3D coordinates of the markers are being sent via Bluetooth LE from ESP32 microcontroller in our RetroSphere glasses prototype to the PC for visualization.
(4) **Tilt sensitivity** - Since we have two markers in RetroPen, our RetroPen hardware can detect tilt with the help of the 5DoF orientation available from the two controllers. This will also help artists to paint

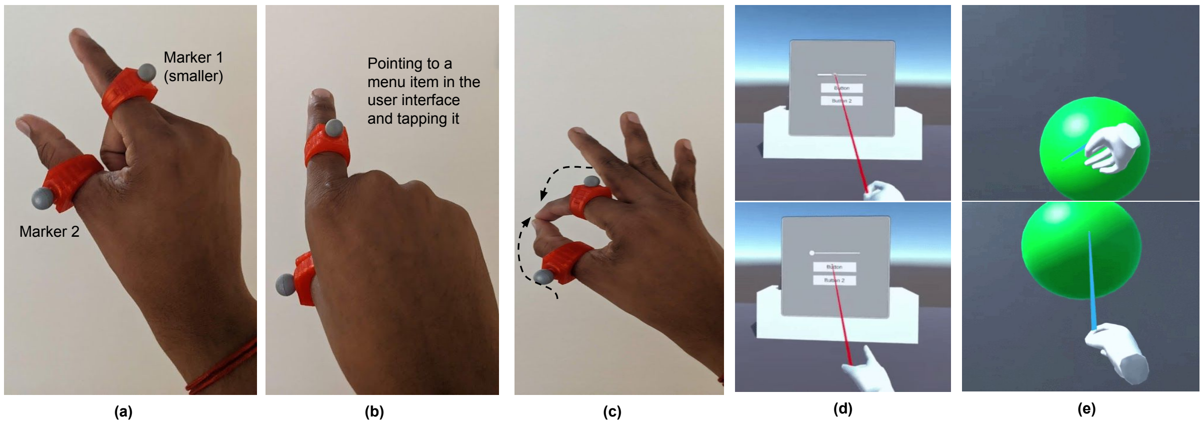
Marker 1
(smaller)
Marker 2
Pointing to a menu item in the user interface and tapping it
(a)
(b)
(c)
(d)
(e)

Fig. 27. (a) Our RetroRing prototype consists of two retroreflective finger rings: one worn on the index finger and the other worn on the thumb. (b) RetroRing on the index finger supports user interactions like taps, double taps, and swipes. (c) The two-finger rings can be used together to make a pinch gesture. (d) RetroRing on the index finger can be used to control AR menu objects like buttons and sliders. (e) RetroRing on both the index and thumb fingers can be used to make a pinch gesture for selecting virtual objects and dragging them in VR/AR.

their drawings seamlessly. Fig. 26 (e) shows a bird being drawn using RetroPen and our Retrosphere Glass prototype by one of the authors. We can see that RetroSphere can handle minute pressure variations while drawing and provides a seamless writing/drawing experience.

RetroPen can provide most of the features being provided by current state-of-the-art digital styluses for tablets. In addition, RetroSphere also has a very low latency of 15ms which is comparable to the state-of-the-art Apple Pencil 2[4] which has a latency of just 9ms. In the future, RetroPen can be modified with markers resembling the actual pen tip and eraser to provide a more natural writing experience for users. Since RetroPen is also passive, it would make a great candidate as an AR pencil accessory for future AR glasses.

6.2 RetroRing - Retroreflective rings for seamless interaction with AR/VR user interfaces

Unlike commercial VR headsets such as Oculus Quest 2[5] which can afford higher compute and large batteries to provide hand tracking, AR glasses will not be able to afford high power consumption as well as compute due to the lightweight glasses form-factor. For instance, Oculus Quest 2 supports two bulky controllers as well as hand tracking for interaction with the VR user interface and only supports a battery life of 2-3 hrs [6]. With significantly smaller batteries and the promise of all-day battery life, AR glasses need to be frugal in consuming power. Recent works [50, 57] have shown that retroreflective finger rings/gloves can facilitate high-speed finger tracking. Thus, we designed 'RetroRing' - a ring-based retroreflective marker tracking for providing hand interactions with RetroSphere enabled AR glasses.

Fig. 27 (a) shows our RetroRing prototype worn on a user's fingers. Two RetroRings are being used in our prototype - one being worn on the index finger (marker 1 with radius 3mm) and the other being worn on the thumb (marker 2 with radius 4mm). The RetroRing on the index finger is mainly used for user interaction with

[4]Apple pencil 2 specifications - https://www.macrumors.com/guide/apple-pencil/
[5]Oculus Quest 2 specifications - https://en.wikipedia.org/wiki/Oculus_Quest_2
[6]Battery life of Oculus Quest 2 - https://knowtechie.com/whats-the-battery-life-of-the-oculus-quest-2

the AR user interface. As the user points the index finger on the UI menu (see Fig. 27 (b), we cast a ray from the 3D coordinates of the marker in the user's fingers to the menu item to select the menu item. The RetroRing on the index finger can support four interactions as follows:

(1) **Tap -** The user can tap on a menu button to select it with the index finger. We identify taps with the help of the rate of change of velocity of the RetroRing's depth. As the user taps on a menu item, the finger moves forward which is captured as an increase in depth by the RetroSphere hardware. We have both the change in depth of the RetroRing as well as the time from the RetroSphere hardware to calculate finger velocity. Since taps are done with a certain velocity, we use a threshold of 1m/s [48] to detect taps made with the index finger. Taps are used to select menu items like buttons, dropdowns, and checkboxes.
(2) **Double Tap** - Since we can detect each tap, double taps can be identified by setting a threshold on the time between two consecutive taps. If two consecutive taps occur within a threshold of 50ms, we detect a double tap in our current implementation.
(3) **Swipe** - We can detect both left as well as right swipes with the help of change in the 3D coordinates of the RetroRing between consecutive frames.
(4) **Long Press/Hold** - As the user points the finger on an object or menu in the AR user interface, RetroSphere detects a long press or hold by identifying the stationary RetroRing 3D coordinates in consecutive frames. We detect a long press if the user holds for 1 second in our current implementation.

Fig. 27 (d) shows our Unity-based PC software which has buttons and sliders for users to interact. The user taps with RetroRing on the index finger to click the buttons. Holding the finger on the slider knob selects the slider and the user moves left/right to select the appropriate value on the slider. As shown in Fig. 27 (c), we also make use of a RetroRing in the thumb along with the RetroRing in the index finger to perform **pinch and drag**. When the user performs a pinch with their thumb and index finger, the distance between the 3D coordinates of both the markers decreases and can be used to detect a pinch. In our implementation, the distance was set to smaller than 3.5cm to detect a pinch. Once the user pinches a virtual object, the object can be dragged across the virtual UI. Fig. 27 (e) shows a user interacting with a virtual sphere visualized using Unity PC software and pinches on the sphere to select and drag it. We can see that RetroSphere supports all the basic hand interactions required for VR /AR user interfaces.

6.3 RetroSense - Smart home sensing with Retroreflective markers

Retroreflective markers have been known to work well for city-scale sensing [66] with long-range laser vibrometry. Recently, infrared tags [28] have been embedded into real-world objects through 3D printing to facilitate tangible interactions for augmented reality applications. Retroreflective markers can also be attached to real-world objects like doors, home appliances, etc. to support smart home sensing applications. To this aim, we designed 'RetroSense' - a RetroSphere integrated prototype for smart home sensing using retroreflective markers. Fig. 28 (a) shows our RetroSense smart home hardware consisting of stereo Pixart infrared trackers and 8 infrared LEDs to illuminate the scene. RetroSense can be attached to any corner of the room as shown in Fig. 28 (b) to provide smart control of retroreflective markers.

Fig. 28 (c) shows a 2×4 keypad with each key coated with retroreflective tape. When the finger presses the key, the marker on the key is occluded from RetroSense and it triggers a keystroke. We can easily identify the ID of the keystroke by selecting the top-most occlusion on a partially occluded keypad. The keys can be either configured as switches for smart appliances control like switching on/off lights and fans or as a messaging interface on real-world objects. Fig. 28 (d) shows a slider with a retroreflective marker. Depending on the slider position, the 3D coordinates of the marker on the slider is measured using RetroSense to identify the slider position and appropriate functions can be configured for each of the slider position. Retroreflective markers can also be attached to home appliances such as microwaves to check whether cooked food has been left inside

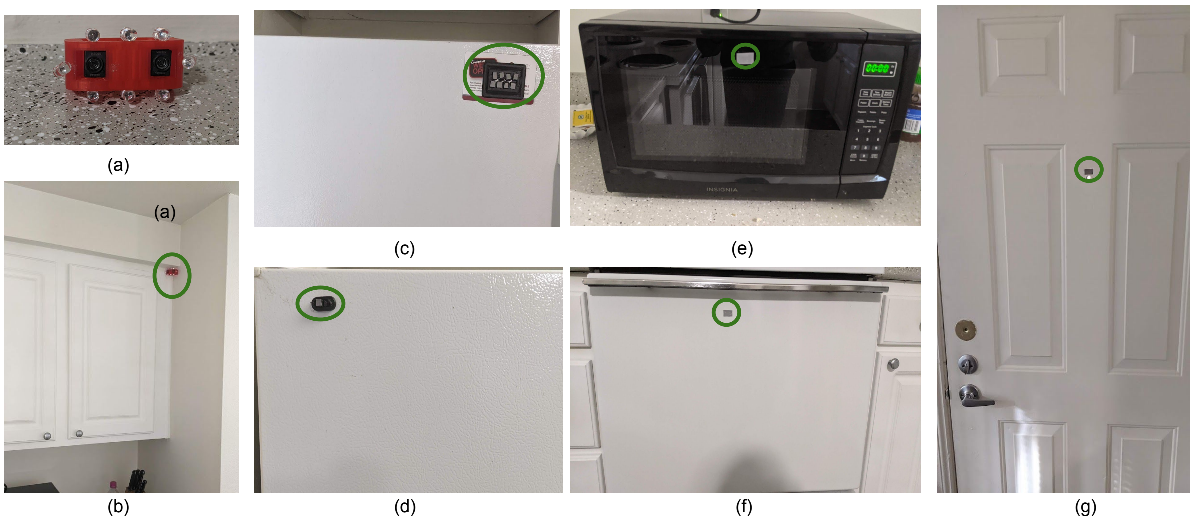

Fig. 28. (a) "RetroSense" - RetroSphere enabled hardware prototype for smart home sensing (b) RetroSense being attached to the top of the ceiling for sensing retroreflective markers in a room. (c) A 2X4 keypad covered with retroreflective tape can control various appliances (d) A retroreflective slider to be used with RetroSense hardware (e) Retroreflective tape is attached to the microwave oven to check if it has been used. (f) Retroreflective tape is attached to the oven to track when the oven door was used. (g) Retroreflective tape is attached to the door to sense door opening/closing and trigger appropriate smart appliance control.

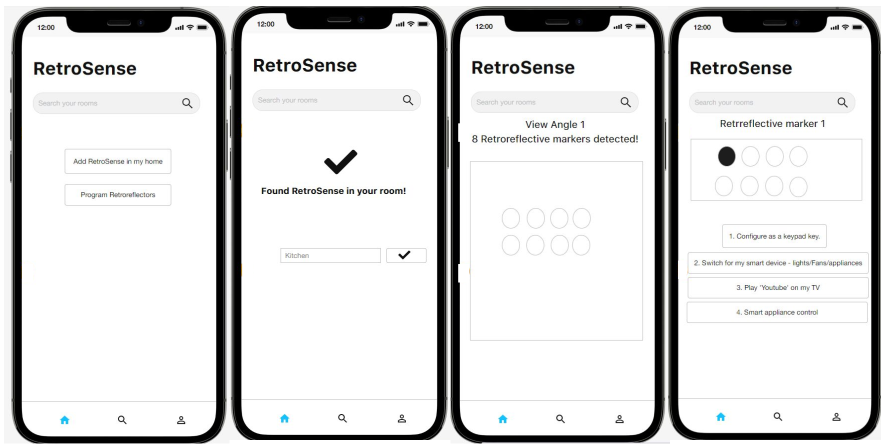
12:00
RetroSense
Search your rooms
Add RetroSense in my home
Program Retroreflectors
12:00
RetroSense
Search your rooms
Found RetroSense in your room!
Kitchen
12:00
RetroSense
Search your rooms
View Angle 1
8 Retroreflective markers detected!
12:00
RetroSense
Search your rooms
Retrreflective marker 1
1. Configure as a keypad key.
2. Switch for my smart device - lights/Fans/appliances
3. Play 'Youtube' on my TV
4. Smart appliance control

Fig. 29. Smartphone user interface for configuring our RetroSense hardware prototype

(see Fig. 28 (e) and Fig. 28 (f)). For example, if the user started cooking and left the kitchen without picking out the food inside, RetroSense can trigger a notification to the user's device via Wi-Fi if the door hasn't been opened after a long period.

Retroreflective markers being attached to the doors, as shown in Fig. 28 (g), can be used to detect the user's entry similar to an ambient sensor and switch off smart appliances like lights and fans via Wi-Fi control. Although we make use of Retroreflective tapes, future markers can be made with retroreflective paint on the objects directly. In the future, we also plan to actuate the RetroSense hardware around the room to provide a wider coverage of the room. Fig. 29 shows our smartphone UI for configuring RetroSense to program the function of each key in a 2×4 keypad.

7 Limitations and Future Work

We believe that RetroSphere will serve as an easy-to-use prototyping tool to enable 3D input on lightweight devices. However, RetroSphere comes with a few limitations inherent to our design choices to make a lightweight 6DoF tracker and a small, electronics-free 3D input controller.

- **Line of sight.** Direct line of sight between the object tracking cameras and the retroreflective controller is required since we rely on a vision-based system.
- **Other IR sources.** Highly reflective surfaces can create additional reflections of the infrared emitters. Interference from infrared sources such as candles can lead our system to track objects other than the input device. However, new IR objects detected in a scene could be easily ignored by constraining the valid marker position to a predefined distance (e.g., arm length).
- **No buttons.** RetroSphere's input controller is devoid of electronics to remove the need for charging and achieve a small form factor. We describe workarounds for this limitation below in subsection 7.2.

7.1 Form-factor of RetroSphere controller

The size of the retroreflective marker makes its use in various form factor input accessories particularly flexible. Our system relies on an 8mm, 10mm, and 6mm sphere, however, smaller spheres or retroreflective stickers can also be used. 3D input could be enabled in a wearable form factor such as a wristband.

7.2 Buttons for RetroSphere control

In this work, we focused on the tracking part of 3D input. Hence, our passive controller does not have a physical button. For our demos, we designed our user experiences which rely on distance thresholds between the controller and the 3D user interface or rely on touchscreen buttons on the phone. Dynamic stroke-based gestures could be used as well.

A mechanical electronics-free button could be implemented with an additional marker on the controller that is occluded until a spring-loaded button is pressed that uncovers the marker. The sudden appearance of the additional marker could be interpreted as a "click".

7.3 Integration of RetroSphere and camera into AR glasses

Although RetroSphere provides cheap and accurate 3D input for AR glasses, infrared trackers cannot provide real-world images like a traditional camera. In the future, we plan to integrate both RetroSphere and traditional camera hardware into the glasses for more immersive AR experiences. Such integration will provide an always-on low-power RetroSphere hardware tracking for 3D input and on-demand camera hardware for improved user experiences based on the application needs.

8 Conclusion

We presented RetroSphere - a low-cost, low-compute, and power-efficient 3D input device for resource-constrained AR devices such as smartphones and AR glasses. With just a stereo pair of infrared blob tracking cameras, IR illuminators, and a microcontroller, RetroSphere provides passive 3D inputs with an average depth accuracy of ≈ 96.5%. RetroSphere's sparse 6DoF estimation pipeline requires very little compute and can run on a tiny ESP32 microcontroller. RetroSphere easily integrates into any device with just affordable electronics and power consumption as low as 400 mW. We demonstrated a variety of applications enabled by RetroSphere, such as 2D/3D drawing in AR, hand gestures, 3D AR measurements, and smart home sensing. A preliminary evaluation of RetroSphere with 20 participants also showed promising evidence supporting its usability. By open-sourcing both our hardware and software, we strongly believe that RetroSphere will empower a community of researchers, designers, and practitioners to integrate 3D input into their projects with affordable off-the-shelf components.

References

[1] [n.d.]. ARPlan 3D: Tape Measure, Ruler, Floor Plan Creator - Apps on Google Play.

[2] 2019. Re: Touch Controllers Battery Life Test - I Don't Believe It. https://forums.oculusvr.com/t5/General/Touch-Controllers-Battery-Life-Test-I-don-t-believe-it/m-p/745201#M346926

[3] 2020. Plastic Infrared Emitting Diode OP294. https://www.ttelectronics.com/products/categories/optoelectronics/optoelectronics/op294/

[4] 2021. About ARKit XR Plugin | ARKit XR Plugin | 4.1.5. https://docs.unity3d.com/Packages/com.unity.xr.arkit@4.1/manual/

[5] 2021. Amazon AR View. https://www.amazon.com/adlp/arview

[6] 2021. ARCore API Reference. https://developers.google.com/ar/reference

[7] 2021. Augmented Reality - Apple Developer. https://developer.apple.com/augmented-reality/

[8] 2021. Controller | VIVE™. https://www.vive.com/eu/accessory/controller/

[9] 2021. Depth Camera D435. https://www.intelrealsense.com/depth-camera-d435/

[10] 2021. ESP32 Wi-Fi & Bluetooth MCU I Espressif Systems. https://www.espressif.com/en/products/socs/esp32

[11] 2021. Focals by North. https://www.bynorth.com/

[12] 2021. How Long Is the Average Human Arm? https://www.reference.com/science/long-average-human-arm-62c7536c5e56f385

[13] 2021. MAD Gaze - AR Smart Glasses. https://www.madgaze.com/

[14] 2021. Microsoft HoloLens | Mixed Reality Technology for Business. https://www.microsoft.com/en-us/hololens

[15] 2021. Nreal - Building Mixed Reality for Everyone. https://www.nreal.ai/

[16] 2021. Oculus Quest 2. https://en.wikipedia.org/w/index.php?title=Oculus_Quest_2&oldid=1014363476 Page Version ID: 1014363476.

[17] 2021. Oculus Quest Accessories & Parts | Oculus. https://www.oculus.com/quest/accessories/

[18] 2021. PixArt Imaging Inc. - PRODUCTS - PAJ7025R3. https://www.pixart.com/products-detail/46/PAJ7025R3

[19] 2021. Pokemon GO. https://pokemongolive.com/post/main/

[20] 2021. Polhemus Electromagnetic Tracking Systems. https://polhemus.com/applications/electromagnetics/

[21] 2021. Polhemus G4. https://polhemus.com/motion-tracking/all-trackers/g4

[22] 2021. Spark AR Studio - Create Augmented Reality Experiences | Spark AR Studio. https://sparkar.facebook.com/ar-studio/

[23] Abdenour Amamra and Nabil Aouf. 2018. GPU-Based Real-Time RGBD Data Filtering. *J. Real-Time Image Process* 14, 2 (2018), 323–340. https://doi.org/10.1007/s11554-014-0453-7

[24] Zhao Chen, Vijay Badrinarayanan, Gilad Drozdov, and Andrew Rabinovich. 2018. Estimating Depth From Rgb and Sparse Sensing. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 167–182. https://doi.org/10.1007/978

[25] Stefano De Amici, Andrea Sanna, Fabrizio Lamberti, and Barbara Pralio. 2010. A Wii Remote-Based Infrared-Optical Tracking System. *Entertainment Computing* 1, 3-4 (2010), 119–124. https://doi.org/10.1109/ICVR.2008.4625155

[26] Arturo De la Escalera and Jose María Armingol. 2010. Automatic Chessboard Detection for Intrinsic and Extrinsic Camera Parameter Calibration. *Sensors* 10, 3 (2010), 2027–2044. https://doi.org/10.1145/2087756.2087837

[27] Kurt M. DeGoede, James A. Ashton-Miller, Jimmy M. Liao, and Neil B. Alexander. 2001. How Quickly Can Healthy Adults Move Their Hands to Intercept an Approaching Object? Age and Gender Effects. *The Journals of Gerontology: Series A* 56, 9 (09 2001), M584–M588. https://doi.org/10.1093/gerona/56.9.M584 arXiv:https://academic.oup.com/biomedgerontology/article-pdf/56/9/M584/9732755/M584.pdf

[28] Mustafa Doga Dogan, Ahmad Taka, Michael Lu, Yunyi Zhu, Akshat Kumar, Aakar Gupta, and Stefanie Mueller. 2022. InfraredTags: Embedding Invisible AR Markers and Barcodes Using Low-Cost, Infrared-Based 3D Printing and Imaging Tools. In *CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI '22)*. Association for Computing Machinery, New York, NY, USA, Article 269, 12 pages. https://doi.org/10.1145/3491102.3501951

[29] Ruofei Du, David Li, and Amitabh Varshney. 2019. Geollery: a Mixed Reality Social Media Platform. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI, 685)*. ACM, 13 pages. https://doi.org/10.1145/3290605.3300915

[30] Ruofei Du, David Li, and Amitabh Varshney. 2019. Project Geollery.com: Reconstructing a Live Mirrored World With Geotagged Social Media. In *Proceedings of the 24th International Conference on Web3D Technology (Web3D)*. ACM, 1–9. https://doi.org/10.1145/3329714.3338126

[31] Ruofei Du, Alex Olwal, Mathieu Goc, Shengzhi Wu, Danhang Tang, Yinda Zhang, Jun Zhang, David Tan, Federico Tombari, and David Kim. 2022. Opportunistic Interfaces for Augmented Reality: Transforming Everyday Objects Into Tangible 6DoF Interfaces Using Ad Hoc UI. In *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems (CHI)*. ACM. https://doi.org/10.1145/3491101.3519911

[32] Ruofei Du, Eric Turner, Maksym Dzitsiuk, Luca Prasso, Ivo Duarte, Jason Dourgarian, Joao Afonso, Jose Pascoal, Josh Gladstone, Nuno Cruces, Shahram Izadi, Adarsh Kowdle, Konstantine Tsotsos, and David Kim. 2020. DepthLab: Real-Time 3D Interaction With Depth Maps for Mobile Augmented Reality. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology (UIST)*. ACM, 15 pages. https://doi.org/10.1145/3379337.3415881

[33] Youchen Du, Shenglan Liu, Lin Feng, Menghui Chen, and Jie Wu. 2017. Hand Gesture Recognition With Leap Motion. arXiv:1711.04293 [cs.CV]

[34] Tinsley A. Galyean and John F. Hughes. 1991. Sculpting: an Interactive Volumetric Modeling Technique. In *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '91)*. Association for Computing Machinery, New York, NY, USA, 267–274. https://doi.org/10.1145/122718.122747

[35] Richard Hartley and Andrew Zisserman. 2003. *Multiple View Geometry in Computer Vision*. Cambridge University Press. https://doi.org/10.1017/CBO9780511811685

[36] Zhenyi He, Ruofei Du, and Ken Perlin. 2020. CollaboVR: a Reconfigurable Framework for Multi-User to Communicate in Virtual Reality. In *2020 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, 542–554. https://doi.org/10.1109/ISMAR50242.2020.00082

[37] Hyouk JANG, Juheon CHOI, and Gunhee Kim. 2019. POL360: a Universal Mobile VR Motion Controller Using Polarized Light. In *25th ACM Symposium on Virtual Reality Software and Technology* (Parramatta, NSW, Australia) *(VRST '19)*. Association for Computing Machinery, New York, NY, USA, Article 20, 10 pages. https://doi.org/10.1145/3359996.3364262

[38] Hyeonseok Jung, Kyoseung Koo, and Hoeseok Yang. 2019. Measurement-Based Power Optimization Technique for OpenCV on Heterogeneous Multicore Processor. *Symmetry* 11 (12 2019), 1488. https://doi.org/10.3390/sym11121488

[39] K. Khoshelham. 2. Accuracy Analysis of Kinect Depth Data. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 3812 (2), 133–138. https://doi.org/10.5194/isprsarchives-XXXVIII-5-W12-133-2011

[40] David Kim, Shahram Izadi, Jakub Dostal, Christoph Rhemann, Cem Keskin, Christopher Zach, Jamie Shotton, Timothy Large, Steven Bathiche, Matthias Niessner, et al. 2014. Retrodepth: 3D Silhouette Sensing for High-Precision Input on and Above Physical Surfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 1377–1386. https://doi.org/10.1145/2556288.2557336

[41] Johnny Chung Lee. 2008. Hacking the Nintendo Wii Remote. *IEEE Pervasive Computing* 7, 3 (2008), 39–45. https://doi.org/10.1109/MPRV.2008.53

[42] Q. Li, R. Li, K. Ji, and W. Dai. 2015. Kalman Filter and Its Application. In *2015 8th International Conference on Intelligent Networks and Intelligent Systems (ICINIS)*. 74–77. https://doi.org/10.1109/ICINIS.2015.35

[43] Michael Lin and Kooktae Lee. 2020. Outdoor Target Positioning Using Wii Remote IR Camera and Signal Modulation. *Sensors* 20, 8 (2020). https://doi.org/10.3390/s20082163

[44] Kent Lyons. 2016. 2D Input for Virtual Reality Enclosures With Magnetic Field Sensing. In *Proceedings of the 2016 ACM International Symposium on Wearable Computers* (Heidelberg, Germany) *(ISWC '16)*. Association for Computing Machinery, New York, NY, USA, 176–183. https://doi.org/10.1145/2971763.2971787

[45] Audiovisuelle Medien. 2006. Implementation of a Low Cost Marker Based Infrared Optical Tracking System. (2006). https://doi.org/10.1109/VSMM.2001.969693

[46] Veljko Milanović, Abhishek Kasturi, James Yang, and Frank Hu. 2017. A Fast Single-Pixel Laser Imager for VR/AR Headset Tracking. In *MOEMS and Miniaturized Systems XVI*, Wibool Piyawattanametha and Yong-Hwa Park (Eds.), Vol. 10116. International Society for Optics and Photonics, SPIE, 91 – 99. https://doi.org/10.1117/12.2253425

[47] Víctor Mondéjar-Guerra, Sergio Garrido-Jurado, Rafael Muñoz-Salinas, Manuel J. Marín-Jiménez, and Rafael Medina-Carnicer. 2018. Robust Identification of Fiducial Markers in Challenging Conditions. *Expert Systems With Applications* 93 (2018), 336–345. https://doi.org/10.1016/j.eswa.2017.10.032

[48] Ryuhei Okuno, Masaru Yokoe, Kenzo Akazawa, Kazuo Abe, and Saburo Sakoda. 2006. Finger Taps Movement Acceleration Measurement System for Quantitative Diagnosis of Parkinson's Disease. *Conference Proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference* Suppl (02 2006), 6623–6. https://doi.org/10.1109/IEMBS.2006.260904

[49] Alex Olwal, Kevin Balke, Dmitrii Votintcev, Thad Starner, Paula Conn, Bonnie Chinh, and Benoit Corda. 2020. Wearable Subtitles: Augmenting Spoken Communication With Lightweight Eyewear for All-Day Captioning. In *UIST '20: Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. ACM. https://doi.org/10.1145/3379337.3415817

[50] Luis Paredes, Sai Swarup Reddy, Subramanian Chidambaram, Devashri Vagholkar, Yunbo Zhang, Bedrich Benes, and Karthik Ramani. 2021. FabHandWear: an End-to-End Pipeline From Design to Fabrication of Customized Functional Hand Wearables. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol* 5, 2, Article 76 (jun 2021), 22 pages. https://doi.org/10.1145/3463518

[51] Farshid Salemi Parizi, Eric Whitmire, and Shwetak Patel. 2019. AuraRing: Precise Electromagnetic Finger Tracking. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol* 3, 4, Article 150 (2019), 28 pages. https://doi.org/10.1145/3369831

[52] T Petrič, A Gams, A Ude, and L Žlajpah. 2010. Real-Time 3D Marker Tracking With a WIIMOTE Stereo Vision System: Application to Robotic Throwing. In *19th International Workshop on Robotics in Alpe-Adria-Danube Region (RAAD 2010)*. IEEE, IEEE, 357–362. https://doi.org/10.1007/978-981-10-4086-_68

[53] Gabriel Reyes, Jason Wu, Nikita Juneja, Maxim Goldshtein, W. Keith Edwards, Gregory D. Abowd, and Thad Starner. 2018. SynchroWatch: One-Handed Synchronous Smartwatch Gestures Using Correlation and Magnetic Sensing. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol* 1, 4, Article 158 (2018), 26 pages. https://doi.org/10.1145/3161162

[54] David Scherfgen and Rainer Herpers. 2009. 3D Tracking Using Multiple Nintendo Wii Remotes: a Simple Consumer Hardware Tracking Approach. In *Proceedings of the 2009 Conference on Future Play On@ GDC Canada*. 31–32. https://doi.org/10.1145/1639601.1639620

[55] Leia C Shum, Bulmaro A Valdés, and HF Machiel Van der Loos. 2019. Determining the Accuracy of Oculus Touch Controllers for Motor Rehabilitation Applications Using Quantifiable Upper Limb Kinematics: Validation Study. *JMIR Biomedical Engineering* 4, 1 (2019), e12291.

[56] Misha Sra, Sergio Garrido-Jurado, Chris Schmandt, and Pattie Maes. 2016. Procedurally Generated Virtual Reality From 3D Reconstructed Physical Space. In *Proceedings of the 22nd ACM Conference on Virtual Reality Software and Technology* (Munich, Germany) *(VRST '16)*. Association for Computing Machinery, New York, NY, USA, 191–200. https://doi.org/10.1145/2993369.2993372

[57] Tomohiro Sueishi and Masatoshi Ishikawa. 2021. Ellipses Ring Marker for High-Speed Finger Tracking. In *Proceedings of the 27th ACM Symposium on Virtual Reality Software and Technology* (Osaka, Japan) *(VRST '21)*. Association for Computing Machinery, New York, NY, USA, Article 31, 5 pages. https://doi.org/10.1145/3489849.3489856

[58] Jonathan Taylor, Vladimir Tankovich, Danhang Tang, Cem Keskin, David Kim, Philip Davidson, Adarsh Kowdle, and Shahram Izadi. 2017. Articulated Distance Fields for Ultra-Fast Tracking of Hands Interacting. *ACM Transactions on Graphics (TOG)* 36, 6 (2017), 1–12. https://doi.org/10.1145/3130800.3130853

[59] Philipp Wacker, Oliver Nowak, Simon Voelker, and Jan Borchers. 2019. ARPen: Mid-Air Object Manipulation Techniques for a Bimanual AR System With Pen & Smartphone. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3290605.3300849

[60] Haoxin Wang, BaekGyu Kim, Jiang Xie, and Zhu Han. 2019. How Is Energy Consumed in Smartphone Deep Learning Apps? Executing Locally vs. Remotely. In *2019 IEEE Global Communications Conference (GLOBECOM)*. 1–6. https://doi.org/10.1109/GLOBECOM38437.2019.9013647

[61] L. Wang, F. C. Wu, and Z. Y. Hu. 2007. Multi-Camera Calibration With One-Dimensional Object Under General Motions. In *2007 IEEE 11th International Conference on Computer Vision*. 1–7. https://doi.org/10.1109/ICCV.2007.4408994

[62] Anusha Withana, Tharindu Kaluarachchi, Chanaka Singhabahu, Shanaka Ransiri, Yilei Shi, and Suranga Nanayakkara. 2020. WaveSense: Low Power Voxel-Tracking Technique for Resource Limited Devices. In *Proceedings of the Augmented Humans International Conference* (Kaiserslautern, Germany) *(AHs '20)*. Association for Computing Machinery, New York, NY, USA, Article 19, 7 pages. https://doi.org/10.1145/3384657.3384790

[63] Anusha Withana, Roshan Peiris, Nipuna Samarasekara, and Suranga Nanayakkara. 2015. ZSense: Enabling Shallow Depth Gesture Recognition for Greater Input Expressivity on Smart Wearables. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (Seoul, Republic of Korea) *(CHI '15)*. Association for Computing Machinery, New York, NY, USA, 3661–3670. https://doi.org/10.1145/2702123.2702371

[64] Po-Chen Wu, Robert Wang, Kenrick Kin, Christopher Twigg, Shangchen Han, Ming-Hsuan Yang, and Shao-Yi Chien. 2017. DodecaPen: Accurate 6DoF Tracking of a Passive Stylus. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology* (Québec City, QC, Canada) *(UIST '17)*. Association for Computing Machinery, New York, NY, USA, 365–374. https://doi.org/10.1145/3126594.3126664

[65] Shengzhi Wu, Daragh Byrne, Ruofei Du, and Molly Steenson. 2022. "Slurp" Revisited: Using Software Reconstruction to Reflect on Spatial Interactivity and Locative Media. In *Proceedings of the Designing Interactive Systems Conference (DIS)*. ACM. https://doi.org/10.1145/3532106.3533464

[66] Yang Zhang, Sven Mayer, Jesse T. Gonzalez, and Chris Harrison. 2021. Vibrosight++: City-Scale Sensing Using Existing Retroreflective Signs and Markers. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 410, 14 pages. https://doi.org/10.1145/3411764.3445054

9 Appendix

9.1 Power consumption benchmarks from a smartphone camera

The power consumed by the camera hardware depends on two main factors - image resolution and frame rate. In an earlier work [60], it has been shown that the power consumption of a Nexus 6 smartphone camera gradually

Preview resolution		Camera capture frame rate	
Acquired image resolution	Power consumption (mW)	Camera capture frame rate (FPS)	Power consumption (mW)
640×480	3000	2	800
1280×720	3500	5	1200
1920×1080	4000	10	1600
2592×1458	4500	15	1800
3200×1800	5000	24	2300
4160×3120	5500	30	2600

Table 5. Power consumption benchmarks using a Nexus 6 smartphone [60]

increases from 3000 mW (640 × 480 resolution) to 5000 mW (4160 × 3120 resolution) as the image preview resolution is increased. As shown in Table 5, the average power consumption increases from 800 mW (2 FPS) to 2600 mW (30 FPS) as the framerate increases.

9.2 Mathematical details of our automatic calibration procedure

(b) RetroSphere camera model - As shown in Fig. 6 representing the standard pinhole camera model of our RetroSphere hardware, the controller is shown as the line segment ABC (L_{ABC}) and the 2D as well as 3D marker locations are represented as $[x, y]^T$ and $[X, Y, Z]^T$ respectively. The homogeneous vectors of the 2D and 3D points are represented as $[x, y, 1]^T$ and $[X, Y, Z, 1]^T$ respectively. The perspective projection of the 2D coordinates to its corresponding 3D points is represented as

$$s[x, y, 1]^T = \mathbf{K}|\mathbf{R}|\mathbf{t}|[X, Y, Z, 1]^T and\ \mathbf{K} = \begin{bmatrix} \alpha & \gamma & x_0 \\ 0 & \beta & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{9}$$

where s is the scale factor and $P = k\ |R\ |t$ is the camera matrix with $R\ |t$ being the rotation and translation matrices (extrinsic matrix) to transform the camera coordinates to the real-world coordinates. K is the intrinsic matrix of the camera and $[x_0, y_0]$ is the principal point.

Let a, b, c represent the 3 marker locations of the retroreflective controller being captured by the infrared cameras. Given the image points $\{a_{ij}, b_{ij}, c_{ij}\ | j = 1, 2, ...n, i = L, R\}$ of the retroreflective controller from the left and right infrared trackers under the i^{th} image frame. Our calibration algorithm aims to compute the metric projection matrix under the left camera coordinate system as follows:

$$\begin{cases} \mathbf{P}_{\mathbf{L}}^{(\mathbf{e})} = \mathbf{K}_{\mathbf{L}}|\mathbf{R}_{\mathbf{L}}|\mathbf{t}_{\mathbf{L}} \\ \mathbf{P}_{\mathbf{R}}^{(\mathbf{e})} = \mathbf{K}_{\mathbf{R}}|\mathbf{R}_{\mathbf{R}}|\mathbf{t}_{\mathbf{R}} \end{cases} \tag{10}$$

The left and right camera matrices can be obtained linearly. Firstly, the vanishing points of the markers in the controller are computed. We then compute the infinite homographies between the cameras. Using the infinite homographies, the affine projection matrix as well as the metric projection matrix can be computed. Unlike existing calibration methods, this method does not require a calibrated base camera and can be done automatically without the need for a calibration board or object.

(c) Affine calibration - As the user waves the stylus in mid-air for 30 seconds, the correspondence of the image points $\{a_{ij}, b_{ij}, c_{ij}\ | j = 1, 2, ...n, i = L, R\}$ can be established by identifying the unique marker size for each marker in our retroreflective stylus. As we know the geometry of the retroreflective stylus, the vanishing points($v_{i,j}$) of the line L_{ABC} in both left and right cameras can be obtained. The simple ratio of the marker positons A,B and C is given by,

$$\mathrm{simple}(\mathbf{A_j}, \mathbf{B_j}, \mathbf{C_j}) = d_1/d_2 \tag{11}$$

where $d_1 = \|A - C\|$ and $d_2 = \|B - C\|$. The Cross ratio of the points $\{\mathbf{A_j}, \mathbf{B_j}, \mathbf{C_j}, \mathbf{V_{j\infty}}\}$ is also d_1/d_2 where $\mathbf{V_{j\infty}}$ is the infinite point of line ABC. Since the perspective transformation preserves the cross ratio, the vanishing points can be obtained from the linear constraints on v_{ij} as follows:

$$\mathrm{cross}(\mathbf{A_j}, \mathbf{B_j}, \mathbf{C_j}, \mathbf{V_{ij}}) = d_1/d_2 \tag{12}$$

The infinite homography between the left and right camera satisfies the below equations:

$$\mathbf{H_{R\infty}}\ \mathbf{v_{Lj}} \ = \ \lambda_{Rj}\mathbf{v_{Rj}},\ \ (j = 1, 2, ..., n) \tag{13}$$

From the above equation, the unknown scale factor λ_{Rj} can be elliminated to obtain:

$$[\mathbf{v_{Rj}}] \times \mathbf{H_{L\infty}}\mathbf{v_{Rj}} \ = \ 0,\ \ (j = 1, 2, ..., n) \tag{14}$$

The linear equations 14 can be solved to determine the infinite homographies. With the homographies and the image points, the projective reconstruction of the 2D points and the camera can be easily computed using the technique of projective reconstruction with planes. The affine camera matrices are given as follows:

$$\begin{cases} \mathbf{P_L^{(a)}} \ = \ [\mathbf{H_{L\infty}}|\mathbf{e_L}] \\ \mathbf{P_R^{(a)}} \ = \ [\mathbf{H_{R\infty}}|\mathbf{e_R}] \end{cases} \tag{15}$$

and the affine reconstruction of the 2D marker locations will be $\{\mathbf{A_j^{(a)}}, \mathbf{B_j^{(a)}}, \mathbf{C_j^{(a)}}\}$.

(d) Metric calibration - The affine projection matrix in 15 will be used to compute the metric projection matrices using:

$$\mathbf{P_i^{(e)}} = \mathbf{P_i^{(a)}} diag(\mathbf{K_0}, 1),\ \ (i = L, R) \tag{16}$$

The metric reconstruction of the image points should satisfy the below equations:

$$\begin{cases} \mathbf{A_j^{(e)}} = \mathbf{K_0^{-1}}\mathbf{A_j^{(a)}} \\ \mathbf{B_j^{(e)}} = \mathbf{K_0^{-1}}\mathbf{B_j^{(a)}} \\ \mathbf{C_j^{(e)}} = \mathbf{K_0^{-1}}\mathbf{C_j^{(a)}} \end{cases} ,\ j = (1, 2, ..., n) \tag{17}$$

where $\mathbf{K_0}$ is the intrinsic parameter of the left infrared camera.

As we already know that $\left\|\mathbf{A_j^{(e)}} - \mathbf{C_j^{(e)}}\right\| = d_1; \|\mathbf{B_j^{(e)}} - \mathbf{C_j^{(e)}}\| = d_2$, we can obtain the linear constraints for obtaining K_0 from 17 as follows:

$$\begin{cases} (\mathbf{C_j^{(a)}} - \mathbf{A_j^{(a)}})^\mathbf{T}\ \overline{\omega}(\mathbf{C_j^{(a)}} - \mathbf{A_j^{(a)}}) = d_1^2 \\ (\mathbf{C_j^{(a)}} - \mathbf{B_j^{(a)}})^\mathbf{T}\ \overline{\omega}(\mathbf{C_j^{(a)}} - \mathbf{B_j^{(a)}}) = d_2^2 \end{cases} \tag{18}$$

where $\overline{\omega} = \mathbf{K_0^{-T}}\mathbf{K_0^{-1}}$. We can obtain $\overline{\omega}$ from solving 18 and $\mathbf{K_0}$ is obtained from the Cholesky decomposition of $\overline{\omega}^{-1}$. From $\mathbf{K_0}(\mathbf{K_L})$, we can obtain the intrinsic parameters $\mathbf{K_R}$, the rotation matrices, and translation matrices using QR decomposition of the metric projection matrix(10).

9.3 Kalman Filtering of Raw depth values

We define a general notation of a dynamic system using incomplete or noisy measurements as follows:

$$x_t = Ax_{t-1} + Bu_t + w_t \tag{19}$$

$$z_t = Hx_t + v_t \tag{20}$$

Linear Kalman filter prediction equations are as follows:

$$\overline{x_t} = Ax_{t-1} + Bu_t + w_t \tag{21}$$

$$\overline{P_t} = AP_{t-1}A^T + Q_t \tag{22}$$

The correction equations are as follows:

$$K_t = \overline{P_t}H^T(H\overline{P_t}H^T + R_k)^{-1} \tag{23}$$

$$x_t = \overline{x_t} + K_t(Z_t - H\overline{x_t}) \tag{24}$$

$$P_t = (1 - K_tH)\overline{P_t} \tag{25}$$

where at time t , x_t is the state variable, $\overline{x_t}$ is the estimate, z_t is the observation of the state x_t, $\overline{P_t}$ is the estimated state error covariance, P_t is the state error covariance, A is the state-transition model, B is the control-input model, H is the observation/measurement model, K_t is the Kalman gain, Q_t is the covariance of process noise, R_k is the covariance of observation noise, w_t is the process noise $w_t \sim N(0, Q_t)$, v_k is the observation noise $v_k \sim N(0, R_k)$ and u_t is the control vector.

For our depth/Z-value stabilization/smoothing, we assume a static sensor and a moving RetroSphere stylus. H and A is set to 1 since we have a scalar correspondence between the Z-value measurements and depth will not be higher than the noise amplitude between two consecutive frames (at t and t-1). We set B=0 since our sensor is fixed. We also assume q_k to be zero and $R_k = {\sigma_k}^2$, the covariance of the measurement noise. Thus our Kalman filter equations will become:

$$\overline{x_t} = x_{t-1} \tag{26}$$

$$\overline{P_t} = P_{t-1} \tag{27}$$

For correction we use,

$$K_t = \overline{P_t}(\overline{P_t} + R_k)^{-1} \tag{28}$$

$$x_t = \overline{x_t} + K_t(Z_t - \overline{x_t}) \tag{29}$$

$$P_t = (1 - K_t)\overline{P_t} \tag{30}$$