TriPad: Touch Input in AR on Ordinary Surfaces with Hand Tracking Only

Camille Dupré, Caroline Appert, Stéphanie Rey, Houssem Saidi, Emmanuel Pietriga

TriPad: Touch Input in AR on Ordinary Surfaces with Hand Tracking Only

Camille Dupré
camille.dupre@universite-paris-saclay.fr
Carl Berger-Levrault &
Université Paris-Saclay, CNRS, Inria
Paris, France

Caroline Appert
caroline.appert@universite-paris-saclay.fr
Université Paris-Saclay, CNRS, Inria
Orsay, France

Stéphanie Rey
stephanie.rey@berger-levrault.com
Berger-Levrault
Toulouse, France

Houssem Saidi
houssem.saidi@berger-levrault.com
Carl Berger-Levrault
Paris, France

Emmanuel Pietriga
emmanuel.pietriga@inria.fr
Université Paris-Saclay, CNRS, Inria
Orsay, France

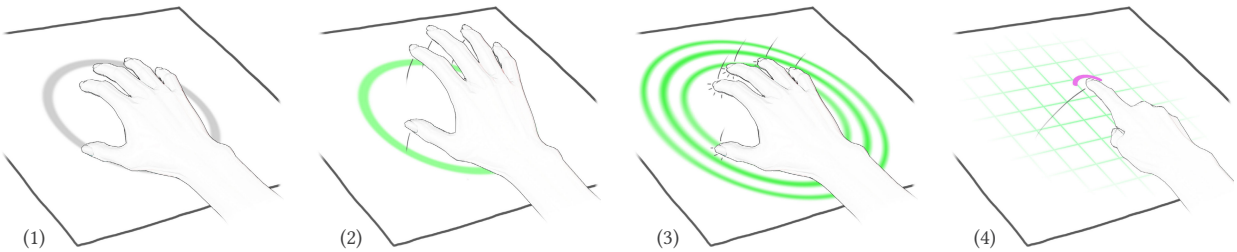


Figure 1: TriPad: touch surface declaration using hand tracking only. (1) Dwelling with all fingers on the target surface: a gray circle appears in the plane defined by the thumb, middle and pinky fingers. (2-3) Performing a quick tap gesture with the whole hand to actually create a TriPad. (4) The surface can now be used for touch input.

ABSTRACT

TriPad enables opportunistic touch interaction in Augmented Reality using hand tracking only. Users declare the surface they want to appropriate with a simple hand tap gesture. They can then use this surface at will for direct and indirect touch input. TriPad only involves analyzing hand movements and postures, without the need for additional instrumentation, scene understanding or machine learning. TriPad thus works on a variety of flat surfaces, including glass. It also ensures low computational overhead on devices that typically have a limited power budget. We describe the approach, and report on two user studies. The first study demonstrates the robustness of TriPad's hand movement interpreter on different surface materials. The second study compares TriPad against direct mid-air AR input techniques on both discrete and continuous tasks and with different surface orientations. TriPad achieves a better speed-accuracy trade-off overall, improves comfort and minimizes fatigue.

CHI '24, May 11–16, 2024, Honolulu, HI, USA
2024. ACM ISBN 979-8-4007-0330-0/24/05…$15.00
https://doi.org/10.1145/3613904.3642323

CCS CONCEPTS

• **Human-centered computing** → **Interaction techniques**; **Mixed / augmented reality**; **Gestural input**.

KEYWORDS

augmented reality; touch input; passive surfaces

ACM Reference Format:
Camille Dupré, Caroline Appert, Stéphanie Rey, Houssem Saidi, and Emmanuel Pietriga. 2024. TriPad: Touch Input in AR on Ordinary Surfaces with Hand Tracking Only. In *Proceedings of the CHI Conference on Human Factors in Computing Systems (CHI '24), May 11–16, 2024, Honolulu, HI, USA.* ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3613904.3642323

1 INTRODUCTION

Mid-air bare-hand input is generally favored over motion controllers for interaction in Augmented Reality (AR) as it leaves the hands unencumbered. Hand tracking has become robust-enough to be featured in mainstream hardware, and the limitations that remain are an active topic of research [15, 16]. But beyond tracking accuracy and robustness, bare-hand input raises questions in terms of interaction design. There is a growing body of research showing that it causes fatigue (*e.g.*, [6, 21]), lacks tactile feedback to confirm input actions (*e.g.*, [32, 37]), and has limited motor precision (*e.g.*, [7, 36]).

These concerns, which get reflected in design guidelines from the industry [2, 38, 39, 49], can be alleviated by using ordinary surfaces

in the user's environment to support touch input. Earlier research has investigated different ways to transform passive surfaces into touchpads. But, because AR headsets – and upcoming generations of AR smartglasses even more so – are essentially *mobile* devices, a technique for turning an ordinary surface into a touchpad should meet specific requirements. It should be versatile and effortless to invoke. It should also work on a variety of surfaces, without instrumenting the environment or the user with additional sensors beyond those embedded in the AR eyewear. Furthermore, the technique should keep additional computations to a minimum, as the AR eyewear typically has a limited power budget.

Techniques found in the literature each meet different subsets of these requirements, but never meet all of them. We introduce TriPad, a novel technique for touch input in Augmented Reality that relies on regular hand tracking only. As illustrated in Figure 1, users declare a new surface by placing their hand in contact with the target surface and making a quick tap gesture to confirm their intent. This low-friction gesture is enough to create the surface plane, which gets positioned and oriented according to the 3D coordinates of three fingertips. Users are then free to use that plane at will to interact with UI panels and other 2D interface components.

Because TriPad only relies on regular hand tracking and does not need to know anything about the surfaces in the environment, it works on a variety of materials, including transparent surfaces such as glass. The technique involves instrumentation of neither the user nor the environment. It only performs simple computations on finger-joint data, which are obtained from the built-in hand tracking API. Additionally, as the embedded cameras used by this API cover an interaction volume that expands beyond the user's field of view, TriPad can be used not only for direct input but for indirect input as well when coupled with the proper interaction model.

After positioning our approach with respect to other opportunistic touch input techniques in AR and beyond, we describe TriPad's two main components: the surface-plane creation technique; and the state machine enabling either direct or "eyes-free" indirect input on those surface planes. We then report on two studies. The first study validates the robustness of TriPad's hand posture and motion interpreter on different materials: wood, glass and a piece of drywall. The second study compares TriPad against direct near- and far-field AR interaction techniques: mid-air touch for objects nearby, and raycasting for objects at a distance. Participants achieve a better speed-accuracy trade-off with TriPad overall. The technique, especially on horizontal surfaces, also improves comfort and minimizes fatigue.

2 RELATED WORK

Touch input on physical surfaces can be achieved in different ways, that each have their advantages and drawbacks: stationary setups that instrument the environment; wearable setups that instrument users with additional sensors, or that rely only on sensors embedded in the AR headset. We first briefly summarize empirical findings about the benefits of touch input for interaction in AR. We then give an overview of prior work about enabling touch interaction on physical surfaces in the user's environment. Finally, we position TriPad with respect to this prior work.

2.1 Benefits of Touch Input in AR

Physical surfaces have potential to improve the overall AR user experience: they provide haptic feedback; they can improve input precision; and they can reduce fatigue. In an early study Lindeman *et al.* [32] observed that passive haptic feedback could improve the manipulation of 2D widgets in immersive environments, and that users preferred interacting with a physical surface rather than in mid-air. Since then, further empirical evidence has been reported about the benefits of physical surfaces for touch input in VR or AR.

When comparing mid-air to surface interaction, one first key issue to consider is accuracy. Because mid-air input lacks any haptic feedback, it can be imprecise. Cheng *et al.* [7] found that a physical tabletop surface improved performance over mid-air interfaces for selection and docking tasks in terms of both speed and accuracy. Medeiros *et al.* [36] also observed significant improvements in terms of performance and user experience for target selection tasks in physically-constrained spaces where surfaces with different orientations are within arms' reach.

These studies also observe benefits of touch input in terms of comfort and fatigue. Direct manipulation of virtual content in mid-air requires that users reach interface widgets with their hands. This forces them to raise their arms, causing fatigue and discomfort. These concerns are raised frequently in the literature about interaction in immersive environments [6, 7, 36, 37, 59]. Hincapié-Ramos *et al.*'s *Consumed Endurance* metric [21] aims at helping designers evaluate their mid-air interaction techniques from this perspective. Movement remapping [36] and support for both direct and indirect input in AR [6, 20] provide alternatives to mid-air direct manipulation that can effectively reduce discomfort.

Beyond discomfort and fatigue, Hsieh *et al.* also raise the issue of social acceptance [23], arguing that mid-air interactions with arms raised can attract unwanted attention. In their survey about interaction techniques for AR smartglasses, Lee *et al.* [29] also observe that touch input, as opposed to voice input or mid-air gesture input, is more socially acceptable.

2.2 Enabling Touch Input on Passive Physical Surfaces

The research works discussed above have mainly contributed empirical evidence about the benefits of surface over mid-air interaction without discussing how to actually enable such surface interaction. Other projects have rather focused on technical solutions to enable touch input on physical surfaces, investigating a wide range of setups that span contexts of use beyond AR eyewear. Technical solutions can be either *stationary* or *wearable*.

Stationary solutions are tied to a single surface or a room. Some instrument the surface itself (walls [44, 61], furniture [46, 48], objects [60]). Other solutions rather observe the surface using remote sensors (visible light [30, 35], infrared [27, 52], time-of-flight [41, 53, 54], thermal [28], *etc.*) coupled with computer vision techniques. The main challenges include differentiating the hand from the surface – which essentially comes down to subtracting the background – and detecting actual contact points between fingers and surface. Of particular relevance, WorldKit [56] is limited to a room but enables users to create interactive surfaces on the fly on walls and furniture. DIRECT [57] combines depth and IR sensors

to enhance tracking performance and removing the need for *a priori* calibration. Recent work has sought further improvements by reducing touch latency using machine learning [9].

Stationary solutions involve setting up equipment such as static cameras, and in many cases performing some calibration before the system can be used. Wearable solutions avoid instrumenting the environment, but face the additional challenge that physical surfaces are not known *a priori* and need to be identified on the fly.

Henderson & Feiner use a camera mounted overhead to track pre-configured optical markers that identify interactive surfaces used for opportunistic control [20]. Ubi Edge [19] also enables opportunistic controls, but focuses on the edges of physical objects detected by a LIDAR camera mounted above a Microsoft HoloLens 2. OmniTouch [18] consists of a pico-projector and depth camera worn on the shoulder. It enables touch input on a variety of surfaces, including the user's own body for on-skin interaction [5], thanks to elaborate finger segmentation and click detection techniques. MRTouch [59] only relies on the sensors embedded in a HoloLens (depth and IR cameras) to detect and track surface planes in real-time. Users are then able to create interactive areas on those surfaces obtained through scene understanding. Though not strictly wearable, KinectFusion [25] also enables a real-time detailed 3D reconstruction of a room. Among other possibilities, it enables multi-touch interaction on both planar and non-planar surfaces, at the cost of a computationally-intensive GPU-based pipeline. But touch input detection does not necessarily mean that the geometry of surfaces needs to be reconstructed, and a variety of other approaches have been developed.

A series of projects have investigated instrumenting the user's fingers or wrist. LightRing [26] consists of two finger-worn sensors. It can detect the 2D movements of a fingertip on a passive surface, but cannot detect finger taps. Gu *et al.* [12] have users wear an IMU (Inertial Motion Unit) on a finger to detect touch contacts using an SVM classification. They can detect touch-down events but do not support drag and long-press interactions. Other finger-worn solutions based on IMU sensors include ActualTouch [47] and FingerTouch [43]. ActualTouch analyzes finger microvibrations to detect contact with a physical surface; the latter rather adopts an analytical approach and can detect tap and drag input events.

Anywhere Surface Touch [42] places sensors below the wrist to observe the fingers and surface from an interesting perspective, at the cost of a fairly clumsy and uncomfortable instrumentation. Acustico [11] detects surface taps with wearable acoustic sensors. The technique can localize taps in 2D, but does not detect movements and is thus limited to discrete interaction. TapID [37] addresses this issue by using an IMU to detect contact while delegating finger tracking to the AR eyewear.

2.3 TriPad's Positioning

A technique for opportunistic [20] touch input designed for AR eyewear should, to be practical, meet all of the following requirements:

- *work in a variety of uninstrumented environments, with a variety of surface materials* (R_1) to support mobile contexts of use;
- *avoid instrumenting users with additional sensors* (R_2), in particular leaving their hands and forearms unencumbered;
- *make it quick and easy to create a new touch input area on-the-fly* (R_3) as too much overhead due to setup effort would deter users from invoking the technique;
- *enable direct and indirect input* (R_4), both to limit fatigue [6] and to enable interaction with surfaces that are not aligned with the virtual content, or surfaces that are in the periphery of the user's field of view [20];
- *keep additional computations for touch input detection to a minimum* (R_5) as the AR eyewear typically has a limited power budget [4, 8, 14, 29].

The above literature review shows that there is a rich body of work on enabling touch interaction with passive surfaces. But each of the techniques discussed earlier has limitations that make it impractical for use with AR eyewear. Stationary techniques are out of scope as they do not meet R_1. Mobile techniques that require users to wear sensors on fingers, wrist or shoulder do not meet R_2 and often have very limited precision. Among the techniques that do not instrument users with additional sensors, only few techniques enable them to create new surfaces on-the-fly (R_3). When they do, they are stationary [56], or only enable direct interaction and work with a limited set of surface materials [9, 59]. Similarly, the few techniques that are expressive enough to support both direct and indirect interaction (R_4) do not meet all requirements, relying on surfaces defined *a priori* [20] or involving finger-worn sensors [43, 47]. Finally, many techniques involve fairly elaborate processing of sensor data, often based on machine-learning, which typically have a non-negligible computation cost (R_5) and often require training.

TriPad meets all of the above requirements, involving simple computations on data readily available from the built-in hand tracking API featured in most recent off-the-shelf AR and MR headsets.

3 TRIPAD

TriPad takes an approach to touch detection different from that of most techniques discussed earlier. Rather than have the system gain an understanding of the user's physical surroundings through elaborate sensor input processing, TriPad relies on the user explicitly declaring the touch surface's position and orientation. Knowing where that virtual plane is in the 3D physical space, detecting finger touch events is then just a matter of measuring the distance between the fingers and the virtual plane.

Figure 1 illustrates the creation of a touch surface and its use to perform a drag gesture. Users dwell with all fingers in contact with the surface to be used, and perform a hand tap gesture. They can then use the surface at will to perform tap and drag gestures with their fingers, the AR eye-wear providing unobtrusive visual feedback about which fingers are considered in contact with the surface.

A key property of this technique is that it only requires finger-joint data from the regular hand-tracking API to work. The virtual plane's position is obtained easily from the 3D coordinates of the thumb, middle and pinky fingertips at dwell time (Figure 1-1). The subsequent tap gesture only serves to confirm the intent to create a touch surface and thus avoid false positives. The 2D virtual plane gets extruded to give it some thickness and a finger is considered in contact whenever its tip collides with the resulting 3D volume.

TriPad thus lets users create touch input areas on any flat physical surface – provided that the hands are tracked accurately – and meets all requirements from Section 2.3. The technique only requires hand tracking data from cameras already embedded in off-the-shelf hardware, and requires instrumenting neither the environment (R_1) nor users (R_2). Since TriPad only analyzes hand movements and finger postures, it works on a wide variety of surface materials including some that many other techniques are struggling with such as glass [59].[1] The surface definition gesture has low friction (R_3), only consisting of a quick dwell-and-tap gesture with the whole hand. A TriPad can be used for direct or indirect input (R_4). Direct input is activated when widgets are spatially aligned with it. Indirect input enables *"eyes-free"* interaction,[2] in which case the hand controls a pointer, as detailed in Section 3.2. Finally, TriPad uses the AR eyewear's regular hand tracking API and limits additional computations (R_5) to 1) computing (once) a plane from three 3D point coordinates, and then 2) measuring (continuously) the spatial configuration of finger joints with respect to that plane. These computations are extremely simple and draw very little power by themselves.

3.1 TriPad Design Considerations

Interaction in AR is about manipulating not only 3D models but 2D content as well (UI widgets, documents, videos, *etc.*) [33]. While freehand gestures are good for 6-DOF manipulations of 3D models, Lee *et al.* report that surface-based hand input better supports interaction with 2D elements [29]. TriPad primarily targets interactions with 2D elements, as by design it restricts gesture input to a plane.

That plane, however, might not always be aligned with the virtual content the user wants to interact with. Some virtual content is bound to a particular physical location beyond the user's reach. Or comfortable surfaces within arms' reach might not have a position and orientation [20] that matches that of the virtual content. TriPad implements indirect input [6] and remapping [36] techniques that enable users to interact with content at any distance and in any orientation. But providing effective feedback is critical, as a TriPad is not necessarily in the user's field of view when they interact with it. Informed by guidelines from the industry [2, 38, 39, 49], we provide both audio and visual cues to the user when creating and interacting with a TriPad.

Feedback when creating and using a TriPad. The system constantly tracks the user's preferred hand and considers any dwell as a possible intention to create a new TriPad. Figure 1 (1-3) details the feedforward and feedback when creating a TriPad. The gray circle indicates that the user's intention has been detected. A small tap gesture with the whole hand will confirm the intent, while any other gesture will simply discard the gray circle. The gray circle will turn green when moving the hand far enough from the surface before coming back to make the tap gesture. On contact, it then fades out with an animated ripple effect [51], confirming that a TriPad is getting created. In the current design, visual feedback is complemented with audio feedback, but only to confirm a TriPad's creation. If the TriPad gets instantiated *"eyes-free"*, the gray circle is shown as a small icon in a corner of the user's field of view. Figure 1-4 illustrates optional feedforward once a TriPad has been created. It consists of a low-contrast grid that smoothly fades out from the center of the TriPad, suggesting that the actual touch area extends beyond the grid. The grid can be kept always visible or hidden. If visible it can be rendered with a higher-contrast color when sensing fingers proximal to the display [1, 22] to convey a sense of activation.

Hand Postures. Informed by the literature [41, 55], the chosen hand postures result from a trade-off between comfort, hand-tracking stability and positional accuracy. Only three fingers are necessary to define a touch pad, hence the technique's name. The posture we had originally considered only required having the thumb, index and middle fingers in contact (similar to 3TC in Gu *et al.*'s design space of tapping postures [12] but involving the thumb instead of the ring finger). This posture suffered from tracking and positional issues in early tests and was not the most comfortable. We eventually opted for a posture involving the whole hand (which would be noted 5PO in Gu *et al.*'s design space), that yielded much better results and was more comfortable. Once a TriPad is created, users can interact with it using one or two fingers. Single-finger interaction involves the index finger, the middle finger remaining in a closed posture. Two-finger interaction involves the index and middle fingers. All other fingers are ignored and can adopt an open or closed posture. Single-finger interaction will typically be achieved using the IPC posture ranked first by participants in Gu *et al.*'s study [12]. Multiple postures can be adopted to interact with two fingers (ranging from 2PC to 5PO). The states and transitions associated with these postures are detailed in Section 3.2.

TriPad Disposal. TriPads can be created and disposed of at will. Disposal can be triggered from a contextual menu attached to the user's non-preferred hand, but could also be mapped to a direct manipulation gesture performed on the TriPad itself, for instance by pinching all fingers together on the surface (*i.e.*, using the metaphor of crumpling a paper sheet).

3.2 State Machines for Direct and Indirect Input

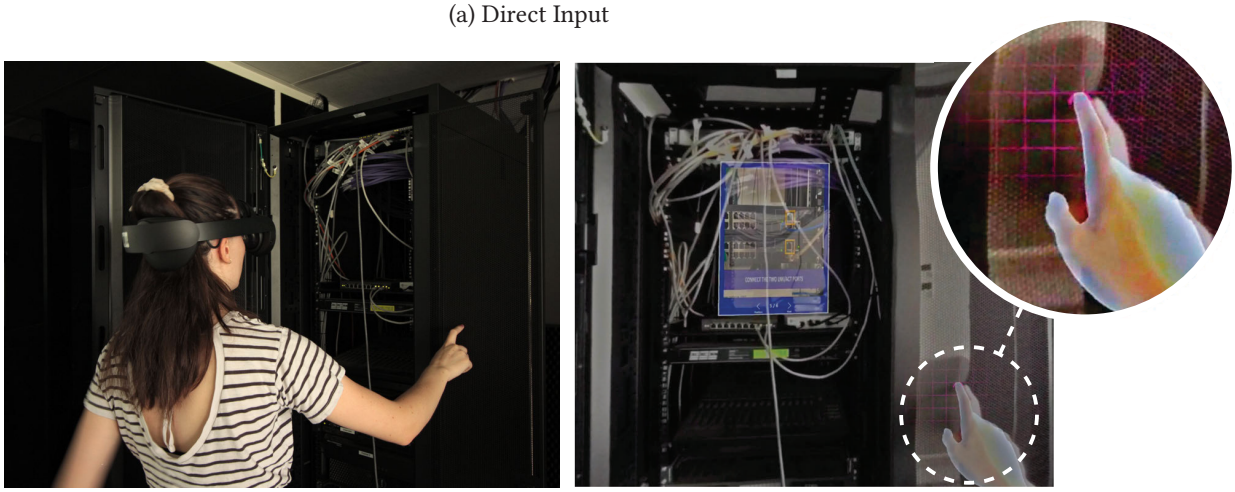
TriPad enables both direct and indirect input, but only one of the two modes can be active at a given moment. Direct input will often be preferred when the virtual content is aligned with the touch surface – except maybe in cases where "fat-finger" occlusion is a problem [50]. The surface is not always aligned with the virtual content, however. For instance users might choose a tabletop to interact with vertical content floating in front of them. We implement the following default behavior:

- Creating a TriPad while looking at a widget activates Direct Input mode provided that the widget is already aligned with the TriPad or that it is floating freely and can be moved in virtual space, *i.e.*, that it is not anchored to a particular position in the physical world. The widget gets smoothly animated to a position and orientation matching that of the TriPad for the duration of the interaction, and is restored to

[1] Though it makes little sense, a TriPad can very well be instantiated by dwelling and tapping in mid-air.

[2] The cameras used for hand tracking in recent and upcoming hardware are often equipped with fisheye lenses that enable tracking the hands in a volume significantly larger than the user's field of view.

(a) Direct Input



(b) Indirect Input

Figure 2: (a) Interacting with a virtual panel lying on a table. The TriPad has been created on the table and enables direct input. (b) Interacting with a virtual panel floating in the air in front of a server rack. The TriPad has been created on a perforated panel next to the rack and enables indirect input.

its original position afterward. Figure 2-a shows an example of TriPad in direct input mode.

- Creating a TriPad while looking at a widget whose position is fixed or while looking at no widget in particular activates Indirect Input mode. Users can then control a cursor that moves in their field of view over and across the whole set of widgets visible to them. Figure 2-b shows an example of TriPad in indirect input mode.

Figure 3 illustrates the two state machines that govern how finger movements on a TriPad trigger input events. In both cases the IDLE state corresponds to no finger in contact.

Direct Input: users can enter the CONTROL state by touching the pad with their index finger. Once in the CONTROL state, moving the index finger on the surface triggers Drag events. Putting it in contact with the surface for a brief time interval (C_{time}: $\Delta t < 1s$) triggers a Click (the event gets fired when the index leaves the pad).

Indirect Input: in this mode TriPad functions as a trackpad rather than a tactile screen. Click and Drag events get fired in the same conditions as in Direct Input mode. But an additional state is necessary, FOCUS, for users to Move the cursor without actually firing Drag events. To enter the FOCUS state, users touch the surface with both their index and middle fingers. The cursor's position is then controlled by the middle finger's movements on the pad, so as to keep the cursor stable when clicking with the index. Users have the possibility to trigger Click events from the FOCUS state as well, by taking their index finger away from the surface for a short time interval (C_{time}: $\Delta t < 1s$). If the index finger does not come back into contact with the surface within that interval, the state machine transitions back to either IDLE or FOCUS depending on whether the middle finger is still in contact with the surface or not.

The ability to click in FOCUS state, called Contact-Click, was introduced to provide users with an alternative to the regular Click. When in FOCUS, performing a regular Click is tedious. It entails lifting the hand from the TriPad, adapting the hand posture – bending the middle finger – and tapping with the index finger before

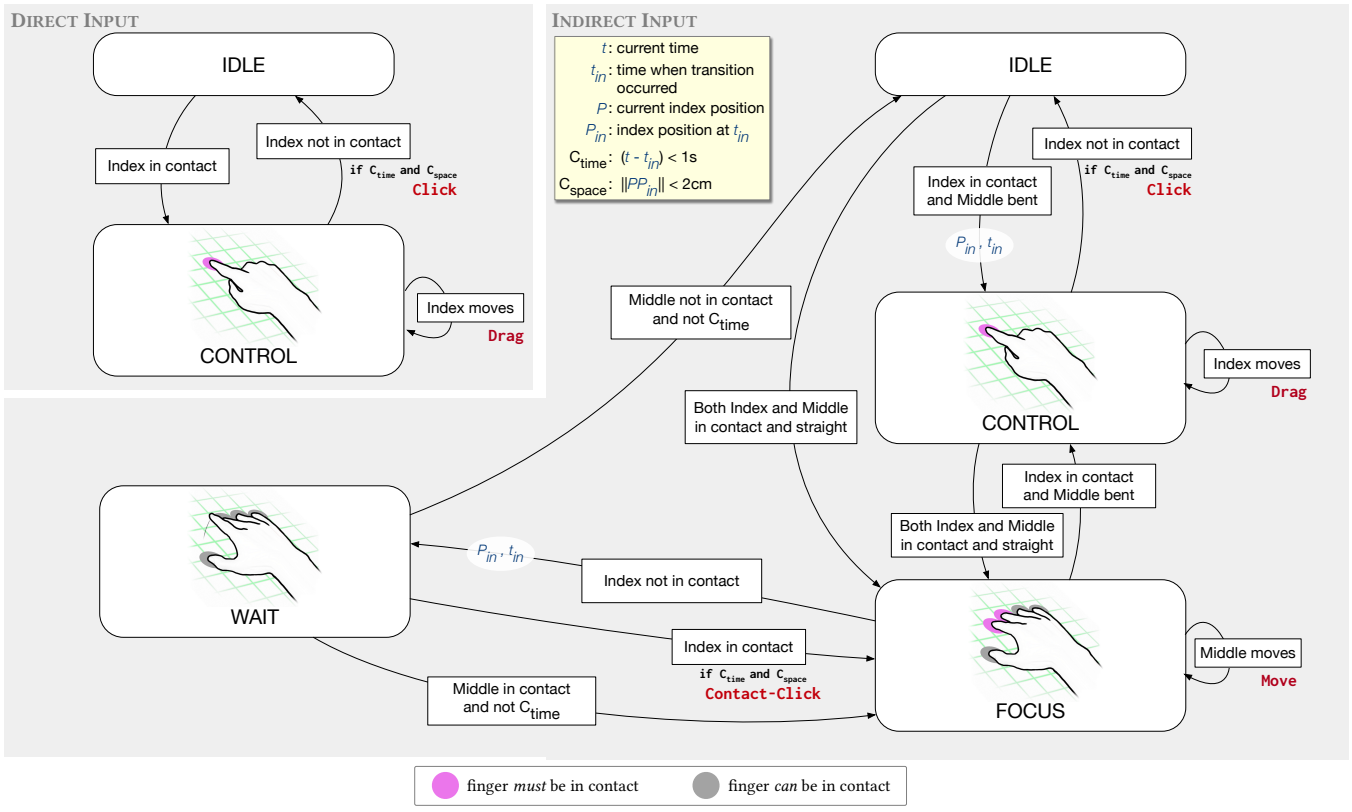


Figure 3: State machines for triggering input events on an active TriPad, based on index and middle finger movements relative to the virtual plane. Input events fired by the state machine (`Move`, `Drag`, `Click`, `Contact-Click`) are colored red. Only one state machine can be active at a given time, either DIRECT INPUT or INDIRECT INPUT.

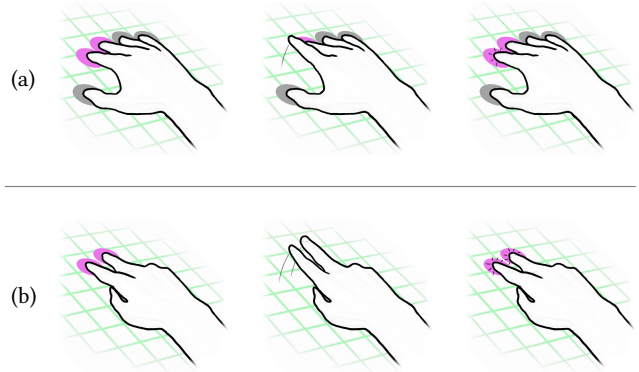


Figure 4: Two possible ways to trigger a Contact-Click.

going back to FOCUS with both fingers in contact with the TriPad. On the contrary, the Contact-Click gets triggered directly from the FOCUS state, without adapting the hand posture – keeping the middle finger straight,[3] which is much more effective and comfortable. Figure 4 illustrates two ways to perform a Contact-Click. An interesting side effect is that clicks that occur either in FOCUS or CONTROL are distinct events. They can thus be mapped to different actions depending on the application needs, in the spirit of the left and right mouse buttons.

3.3 Implementation

The TriPad state machine has been implemented in C# using MRTK 2 and Unity, and tested with two different headsets: a Microsoft Hololens 2 and a Meta Quest Pro (see apparatus descriptions in Sections 4 and 5). TriPad relies essentially on two key features built in recent headsets: 1) the hand tracking API, which provides finger-joint data that triggers transitions in TriPad's state machine; and 2) the 6DoF inside-out positional tracker, which keeps TriPads properly positioned in physical space when users move around – as all other world-anchored virtual objects.

A fingertip is considered in contact with a TriPad as soon as it enters the associated 3D volume, instantiated as a Unity box collider. In our implementation, a TriPad covers a rectangular surface of 40 cm × 40 cm, with a thickness of 5.5 cm after extrusion distributed as follows: 0.5 cm above the surface plane defined by the three fingertip points; and 5 cm below that plane.

[3]The implementation of TriPad used in Experiment 1 (Section 4) required users to keep their middle finger in contact with the TriPad while lifting their index finger. We relaxed this constraint based on feedback from participants. The middle finger – and thus the whole hand – can be lifted up to perform a Contact-Click, as illustrated in Figure 4.

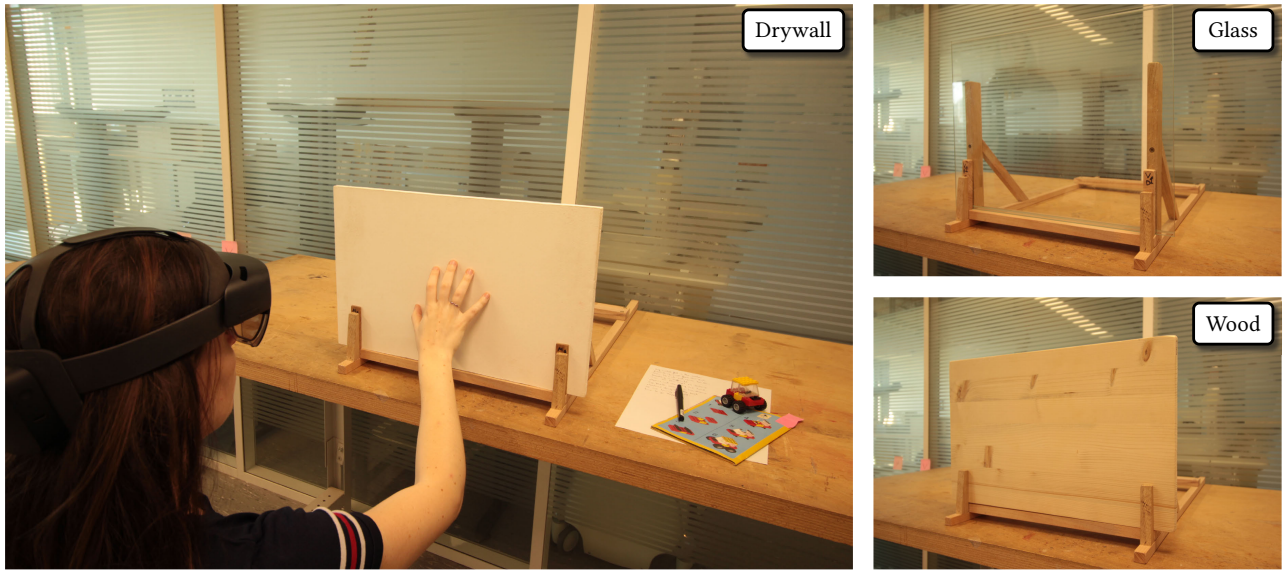


Figure 5: Experiment 1 setup (left). Participants interact with three different types of vertical surfaces, using a Hololens 2: a piece of painted *Drywall* (left), *Glass* (top right), and *Wood* (bottom right). The lego model used in Phase 1 is visible next to the piece of drywall.

These thresholds have been set empirically, accounting for the limits of hand tracking precision with the tested hardware. Inaccuracies in hand tracking can sometimes cause a small offset between the user's physical hand and its virtual counterpart, most particularly in the depth dimension. While this is not an issue for many mid-air freehand interactions, it can be a concern when interacting with physical objects and surfaces. TriPad limits the adverse impact of this offset by design. Any offset incurred by the virtual hand will have been propagated to the TriPad's virtual surface in the first place, since that surface was positioned and oriented according to that same virtual hand. TriPad only relies on the position of virtual finger joints *relative* to the virtual plane, and the offset is thus cancelled out.

As the TriPad is not a 2D plane but rather a 3D volume, the position of fingertips on the surface is computed through projection. However, we have observed that in **Indirect Input** mode the cursor position may slightly shift during finger in-and-out movement made when clicking. To avoid the resulting loss of accuracy, Click events are broadcast with coordinates that correspond to the finger position at the time of initial contact with the surface. Any intermediate drag or move event that may have been triggered in-between is discarded. During the design phase of TriPad, we explored the option of controlling the cursor position using relative movements of the palm rather than the finger, as the former tends to remain more stable during click interactions. This approach was eventually discarded, as it resulted in lower precision due to the lever effect, where palm movements have a lower amplitude compared to the resulting finger movements.

Finally, the current implementation can also compensate for fingertip tracking imprecision at TriPad creation time by auto-correcting the plane's orientation. TriPads which are less than 10° from a vertical (resp. horizontal) orientation can have their orientation slightly adjusted, based on the assumption that many surfaces in our physical environment are vertical or horizontal. This auto-correction feature can of course be deactivated.

4 EXPERIMENT 1: TECHNICAL EVALUATION

Before assessing TriPad's usability and performance, we first conducted an experiment to validate its technical feasibility. Our goal was twofold: 1) determine if users sometimes inadvertently create TriPads while engaged in unrelated activities, and if so how frequently *false positives* occur; and 2) evaluate how accurate TriPad-related gesture recognition is on various surface materials with off-the-shelf AR eyewear, thereby quantifying the reliable detection of *true positives*.

The experiment was split in two phases: 1) participants were first instructed to perform a couple of casual activities while the TriPad recognition engine was active; 2) they then had to create TriPads on three surfaces made of different material: glass, wood and painted drywall (see Figure 5) and interact with them – they had to perform all gestures found in the TriPad state machine.

4.1 Participants

Twelve participants (8 men, 4 women) volunteered for the experiment. All were right-handed. The experiment started with a question about their prior experience with Mixed Reality (MR) headsets. Six participants reported never or rarely using MR headsets; two using them monthly; one weekly and the other three daily.

4.2 Apparatus

The experiment ran on a Microsoft HoloLens 2. As illustrated in Figure 5, participants were comfortably seated, directly facing the physical surface on which they had to create and interact with

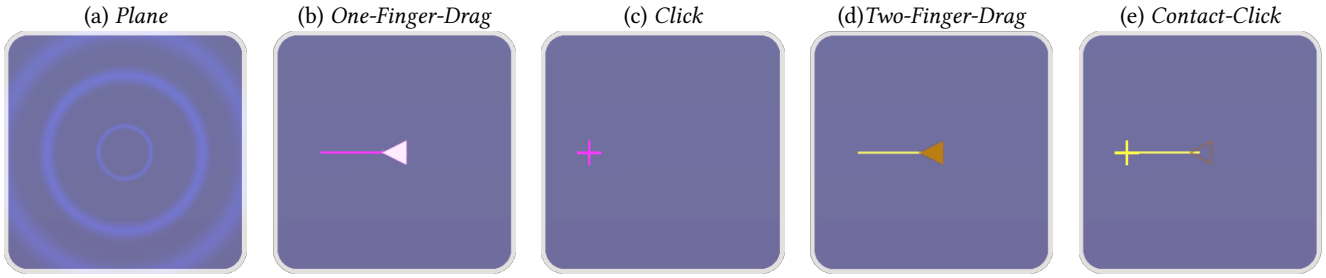


Figure 6: The five types of events participants had to trigger through their actions in Experiment 1.

TriPads. To ensure stability and proper orientation, we designed a custom-made stand that maintained the surface in a secure upright position, positioned within arm's reach.

4.3 Tasks

4.3.1 Phase 1: False positives. The first phase involved two tasks. Participants first had to build a 17-brick Lego Model following a 6-step instruction sheet visible in Figure 5. The operator then distributed a printed sheet with the first ten sentences from a test phrase set [34]. Participants were asked to copy those sentences on a notebook. Participants were wearing the AR headset with the TriPad recognition engine turned on throughout this phase, including when they had to interact with virtual widgets to proceed to the next task and start/stop trials. Any accidental TriPad creation was recorded along with a screenshot of the participant's perspective through the headset in order to understand the context within which the false positive occurred.

4.3.2 Phase 2: True positives. In the second phase, participants had to perform multiple series of five actions in a row to trigger the events of the TriPad state machine. The first action of a series was always to create a TriPad. The system then asked participants if they were satisfied with the plane's orientation. If not, they could recreate TriPads at will, the system logging how many instances were created. Participants then had to perform the same action four times, in order to trigger either a *One-Finger-Drag*, a *Click*, a *Two-Finger-Drag* or a *Contact-Click*. To increase the ecological validity of our observations, the four clicks or drags varied in either location (clicks) or direction (drags). Participants had to perform clicks in four different locations: Left, Right, Up, Bottom. Similarly, they had to perform drags along four directions: Left, Right, Up, Bottom. Figure 6 illustrates click events in the Left area of the pad, and drags towards the Left of the pad. TriPad's recognition engine was constantly running, logging all events recognized.

Figure 6 illustrates the stimuli participants received for each type of event. For click events, a cross indicated where participants had to click (Figure 6-c & 6-e). The experiment software logged the distance between the position of the click and the cross' center. For drag events, the drag action was indicated by a line with an arrowhead (Figure 6-b & 6-d). Participants had to initiate their drag gesture on the arrowhead and then drag it towards the other end of the line. Participants had to drag at least 80% of that distance. If they lifted their finger from the TriPad before reaching that point, the arrowhead jumped back to its initial location and they had to perform the action again.

4.4 Design and Procedure

The experimental procedure started with reading and signing the participation consent form. Participants then proceeded to the first phase (*false positives*) where they had to perform the Lego construction and the text transcription tasks. This phase lasted 10 minutes maximum.

We considered two main factors in the second phase: surface material and event type. Participants had to create a TriPad and interact with it on different Surface materials: a piece of *Drywall*, *Glass*, or *Wood* (Figure 5). On each type of surface, participants had to perform actions that triggered five different Events: create TriPads (*Plane*), perform single finger drags (*One-Finger-Drag*), clicks (*Click*), two-finger drags (*Two-Finger-Drag*) or contact clicks (*Contact-Click*).

Trials were grouped by Surface condition whose presentation order was counterbalanced with a Latin Square across participants. Within the five possible values for Event, the four *action events* (*One-Finger-Drag*, *Click*, *Two-Finger-Drag*, *Contact-Click*) required performing a TriPad *creation event* (*Plane*) beforehand. To ensure an equal distribution of observations for each of the five Event values, we devised the following presentation strategy.

Each Surface condition consisted of four *action event* blocks, each featuring both TriPad *creation events* and one type of *action event*. Each block was divided into 3 identical sub-blocks, with the first one serving as a training phase. Each sub-block started with the creation of a TriPad, followed by four repetitions of the same *action event*, with variations in location or direction. Consequently, excluding training, each of the four *action event* blocks encompassed 2 (2 sub-blocks $\times$ 1) repetitions of the specific *Plane creation event* and 8 (2 sub-blocks $\times$ 4) repetitions of a given *action event*. This approach ensured a balanced collection of 8 repetitions for each of the 5 events (*Plane*, *One-Finger-Drag*, *Click*, *Two-Finger-Drag*, or *Contact-Click*) for each participant.

This first experiment was primarily about evaluating TriPad's viability from a *technical* perspective, under the assumption that participants always performed the expected interactions. To ensure that this was indeed the case, the operator was monitoring participants' gestures to differentiate an actual system misrecognition from a participant's failure to follow instructions. In the latter case, the operator asked the participant to perform the trial again. To

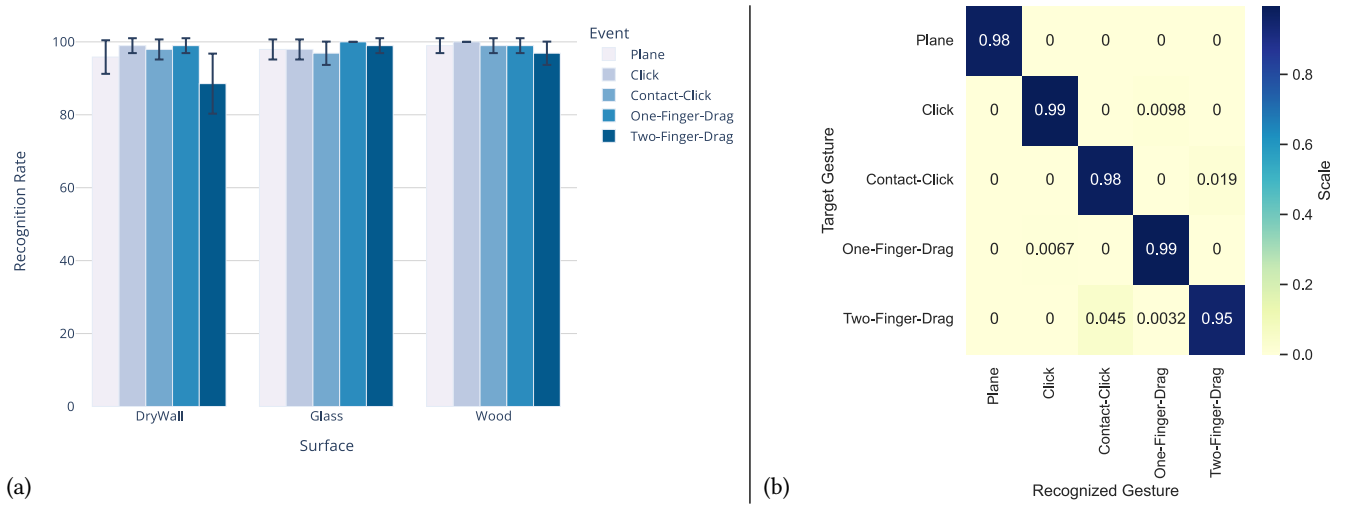


Figure 7: a) Recognition rate per Surface and Event: percentage of trials where the first event recognized by the system matched the task stimulus. Error bars represent the 95% confidence interval relative to all the data points collected in the corresponding condition. b) Confusion matrix between target gestures and recognized gestures.

further mitigate errors due to instruction misinterpretation – for instance, performing another gesture than the one expected – we presented the four *action event* blocks always in the same order (increasing difficulty): *One-Finger-Drag*, *Click*, *Two-Finger-Drag*, and finally *Contact-Click*. Each sub-block started with a video displayed in AR that demonstrated the gesture to perform. Finally, each of the three Surface conditions ended with a questionnaire asking participants to rate the different events along the following aspects with 5-point Likert scales: comfort, ease of use, efficiency and tiredness. The second phase lasted 30-to-40 minutes.

In total, we collected 12 participants × 3 Surface × 5 Event × 8 repetitions = 1440 events for analysis.

4.5 Results

4.5.1 False positives. During Phase 1, only four dwell events (three fingers remaining stationary for 500ms) were detected overall: three over the cumulated time of Lego-assembly and writing tasks (51 minutes across participants); and one while a participant was interacting with virtual widgets. Such dwell events cause the recognition engine to enter a state where it waits for a subsequent hand tap, that will actually create a TriPad if performed. A hand tap was detected only once. This single false positive indicates that while simple, the sequence of actions that must be performed to actually create a TriPad is fairly robust against accidental activations.

4.5.2 True positives. Recognition rate is the main measure in Phase 2. To complete a task, participants had to perform a given target gesture. Whenever the system detected an event other than the target one, the experiment software logged an error along with the recognized event.

Task completion was considered successful when the initial attempt was recognized as the target gesture. Figure 7-a) reports the percentage of successful trials per Surface × Event condition. The recognition rate (percentage of success) is higher than 95% in all conditions but *Drywall* × *Two-Finger-Drag* (88.5%). However, a Cochran test of Surface or Event did not reveal any significant effect on *Success*. Further analyses through pairwise tests that encompassed both factors revealed three pairs of conditions that were significantly different. However, it is important to note that these differences only held statistical significance without applying any correction. When a Holm correction was applied, those pairs did not significantly differ. Specifically, the pairs <*Click*,*Two-Finger-Drag*> (p<0.05, *p-corr*=0.4) and <*One-Finger-Drag*,*Two-Finger-Drag*> (p<0.05, *p-corr*=0.1) were significantly different across Surface conditions. Additionally, the <*Glass*,*Drywall*> pair was significantly different but only for *Two-Finger-Drag* events (p<0.05, *p-corr*=0.5).

We looked precisely at the instances where confusion occurred between different gestures by analyzing all gesture attempts (n = 1535) and how they were classified by the recognition engine. This analysis is presented as a confusion matrix in Figure 7-b. We observed that in approximately 5% of cases where participants intended to execute a *Two-Finger-Drag* gesture, they actually triggered a *Contact-Click* event. This likely occurred because of noise in hand tracking that made the index finger leave the Tripad's volume. Below we describe how we took into account both this observation and qualitative feedback about gestures to execute a *Contact-Click* when we iterated on the design and implementation of gestures to trigger a *Contact-Click* (see Section 4.6).

Click precision was good overall, 6 mm from the target on average (cross center). No significant difference was observed between surfaces. Participants asked to recreate a TriPad because they were not satisfied with its orientation only 7 times in total (2.4% of the plane creation trials). We also logged the adjustment value of orientation auto-correction, which was applied automatically when the TriPad was less than 10° from a vertical orientation (see Section 3.3). Average adjustment was respectively 3.5° for *Glass*, 2.2° for *Drywall* and 2.2° for *Wood*, with no significant difference between Surface conditions. Auto-correction is a viable option for both horizontal

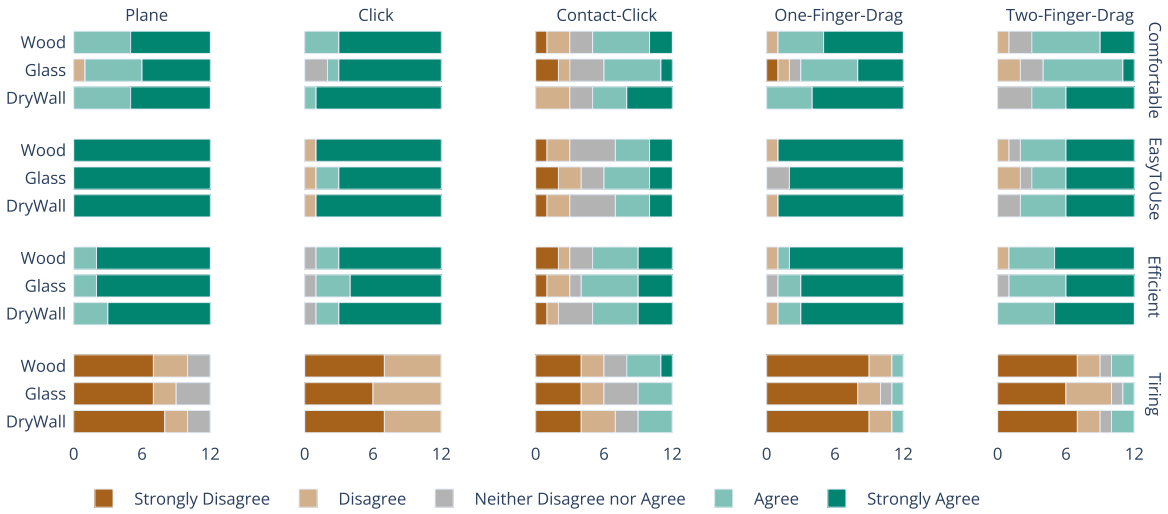


Figure 8: Distribution of participants' 5-pt ratings per SURFACE and EVENT for statements *"The technique was {comfortable, easy to use, efficient, tiring}".*

and vertical orientations as it is reasonable to assume that many surfaces will align with one of these two orientations. When surface orientation is close to neither vertical nor horizontal, no assumption can be made to apply auto-correction. But we can quantify the impact of tilt error in such situations based on the above values. As the TriPad volume extends 0.5 cm above the actual plane, a tilt error of 3.5° (resp. 2.2°) means that in one direction the pad will fall below the physical surface 8.2 cm (resp. 13.2 cm) away from its center. For a 40 cm square TriPad, this corresponds to an offset of 1.2 cm (resp. 0.8 cm) between the pad's edge and the actual surface.

Finally, we analyzed participants' answers to the questionnaire, summarized in Figure 8. Friedman tests revealed that SURFACE has a significant effect on perceived *Comfort* (p<0.001) and *Ease of Use* (p<0.05). Post hoc pairwise tests specifically emphasized the fact that a *Glass* surface is less comfortable than a *Drywall* surface (p<0.05), with some participants mentioning that it causes too much friction. We also observed a significant effect of EVENT on all four scales: *Comfortable*, *Easy to Use*, *Efficient* and *Tiring* (all p's<0.001). Based on post hoc pairwise tests, the most notable observation is that participants perceived the *Contact-Click* event as significantly less comfortable, more difficult to use, and less efficient than the other four events (all p's < 0.05). It was also perceived as significantly more tiring than *Click* and *One-Finger-Drag* (p's < 0.05).

4.6 Summary and Design Iteration

The above findings demonstrate the viability of our approach. We observed a high recognition accuracy overall and received positive feedback from participants in the questionnaire. Nevertheless, we also identified specific areas for improvements regarding *Contact-Click* and *Two-Finger-Drag* events.

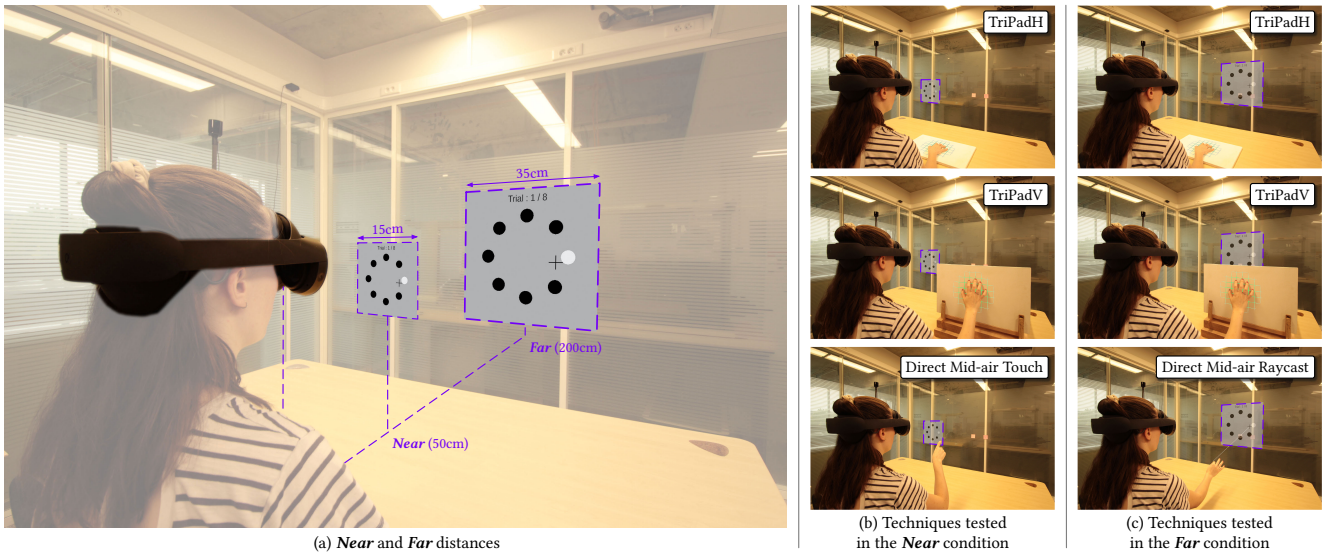
Addressing unintentional Contact-Click events. In the initial TriPad implementation evaluated in this experiment, a *Contact-Click* event required the index finger to only briefly leave and re-enter the pad. We observed that during the execution of *Two-Finger-Drag* events, the index finger might unintentionally leave the surface due to tracking inaccuracies and surface friction, resulting in unintended *Contact-Click* events. To address this issue, we refined the *Contact-Click* recognition criterion. Instead of relying on the index finger leaving and re-entering the pad, we now rely on the relative up-and-down movements of the index finger. This modification not only enhances robustness against accidental *Contact-Click* detection, but also improves the overall user experience. Since the index finger no longer needs to exit the pad's volume, users can perform low-amplitude index movements (8 mm in our implementation), making this interaction more comfortable.

Improving comfort of Contact-Click events. The initial implementation of *Contact-Click* in TriPad required maintaining the middle in contact with the pad, which some participants found uncomfortable. We thus decided to adjust the criteria regarding possible hand postures in the WAIT state (Figures 3 & 4). More specifically, we no longer require the middle finger to maintain contact with the pad, similar to what we already did for the other fingers, as long as the index finger is raised. Users can thus lift their middle finger as well (if more comfortable) when triggering a *Contact-Click* event.

4.7 Limitations

As in most experiments, there are limitations to consider. One first limitation is that we had to restrict what conditions to test so as to keep the experiment's duration per participant within reasonable limits. We focused on vertical TriPads partly for this reason. But this choice was also made to ensure that participants remain comfortable: testing horizontal TriPads would have caused significant neck fatigue [36], as the HoloLens 2 requires that the user's hands be in their field of view while interacting.

Another limitation is that we had to restrict the interactions performed in Phase 1 when evaluating accidental TriPad creations.



(a) ***Near*** and ***Far*** distances

(b) Techniques tested in the ***Near*** condition

(c) Techniques tested in the ***Far*** condition

Figure 9: Experiment 2 setup: a) the two DISTANCE values illustrated with a *Pointing* task ; b) the three TECHNIQUES when DISTANCE = *Near* ; c) the three TECHNIQUES when DISTANCE = *Far*.

Conducting a longitudinal study in which participants would wear the headset for prolonged periods of time would provide a more comprehensive dataset with a broader range of hand movements and object manipulations, both physical and virtual. In particular, Phase 1 of our experiment involved only a few interactions with virtual widgets. However, based on our personal experience developing and evaluating the technique, we could observe that typical interactions with mid-air virtual widgets using touch or raycast are easily differentiated from TriPad creation interactions. Touch interactions are typically performed with the index finger straight and all other fingers bent – a hand posture that does not match the TriPad creation criteria. Raycast interactions typically involve pinching the thumb and index finger. Such pinch gestures can be performed with all fingers straight. That posture is compatible with a TriPad creation interaction and can thus lead to an accidental trigger if the user performs a rapid back-and-forth movement resembling our confirmation step. But this can easily be avoided by disabling TriPad's recognition engine whenever a pinch gesture is detected. Nevertheless, more empirical data about false positive detection when interacting with virtual widgets in such conditions remains to be gathered to confirm that this is sufficient.

5 EXPERIMENT 2: PERFORMANCE EVALUATION

The goal of this second experiment was to assess the performance of TriPad's indirect input compared to direct AR input, for both discrete and continuous actions. Direct AR input techniques depend on the distance between the user and the virtual content. Usually, interaction with *Near* objects (*i.e.*, that are within physical reach) relies on direct *Touch*, while interaction with objects that are *Far* relies on *Raycast* point and commit. In order to increase the ecological validity of our observations, we considered both vertical and horizontal surfaces [3] for TriPad interactions.

5.1 Hypotheses

As discussed earlier, mid-air input has been observed to cause physical fatigue and lack precision mostly because of the absence of tactile feedback [7, 21, 23, 36, 37]. Furthermore, in Medeiros *et al.*'s recent study [36], which specifically studies the benefits of haptic feedback and remapping techniques, participants performed better when the input surface was horizontal than when it was vertical. Consequently, we formulate our initial hypotheses as follows:

- H_1: *TriPad indirect techniques are less tiring and more precise than direct touch (resp. raycast) techniques when interacting with virtual content that is near (resp. far).*
- H_2: *a horizontal TriPad performs better than a vertical TriPad.*

5.2 Participants

Twelve participants (10 men, 2 women) volunteered for the experiment. Eleven of them had also participated to Experiment 1 or to its pilot phase. All of them were right-handed. Our experiment started with a question about their prior experience with Mixed Reality (MR) headsets. Self-reported experience with MR headsets was the same as in Experiment 1 (Section 4.1) except that one participant declared using them monthly and two participants weekly.

5.3 Apparatus

Testing TriPad interactions with both vertical and horizontal surfaces requires covering a larger hand tracking volume than the Microsoft Hololens 2 can. We first tried to use an Ultraleap Stereo IR 170 evaluation kit mounted on top of the headset and oriented so as to track hands in both the vertical and horizontal configurations illustrated in Figure 9. The quality of the hand tracking did not match that of the Hololens 2's and we looked for an alternative. Eventually we opted for the Meta Quest Pro Mixed Reality headset with video passthrough activated. The embedded hand tracking

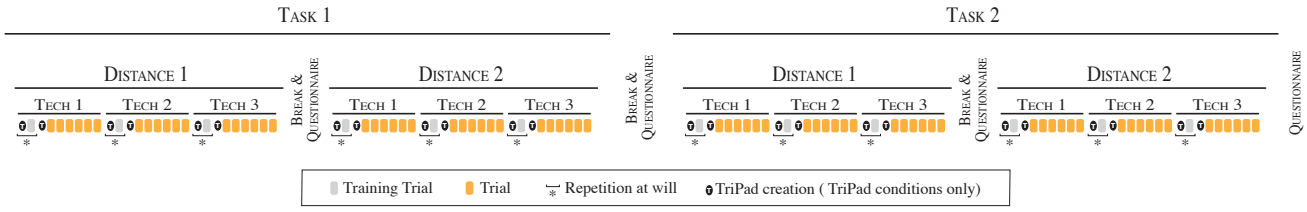


Figure 10: Organization of Experiment 2, counterbalancing Task, Distance and Technique.

system is very reliable and covers a volume appropriate for our purposes.

Participants were comfortably seated at a desk (Figure 9), thereby minimizing the potential for any sensation of cybersickness. TriPads were created on the same piece of drywall as in Experiment 1. The surface was either laid horizontally on the desk (*TriPadH*) or put vertically (*TriPadV*) using the same custom-made stand as in Experiment 1. The vertical position of the drywall could be adjusted upon participants' request by piling wooden planks between the desk and the stand.

Direct Touch and Raycast conditions were as follows:

- Near-field input (mid-air touch, Figure 9-b): users click and drag directly with their finger on virtual content within arm's reach. Clicking on a target in the *Pointing* task requires that the index fingertip be within the bounds of the object when entering and leaving it.
- Far-field input (raycast, Figure 9-c): users control a cursor by raycasting. Performing a quick pinch and release gesture with the index and thumb triggers a click. Maintaining a pinch posture while moving the hand triggers drag events.

5.4 Tasks

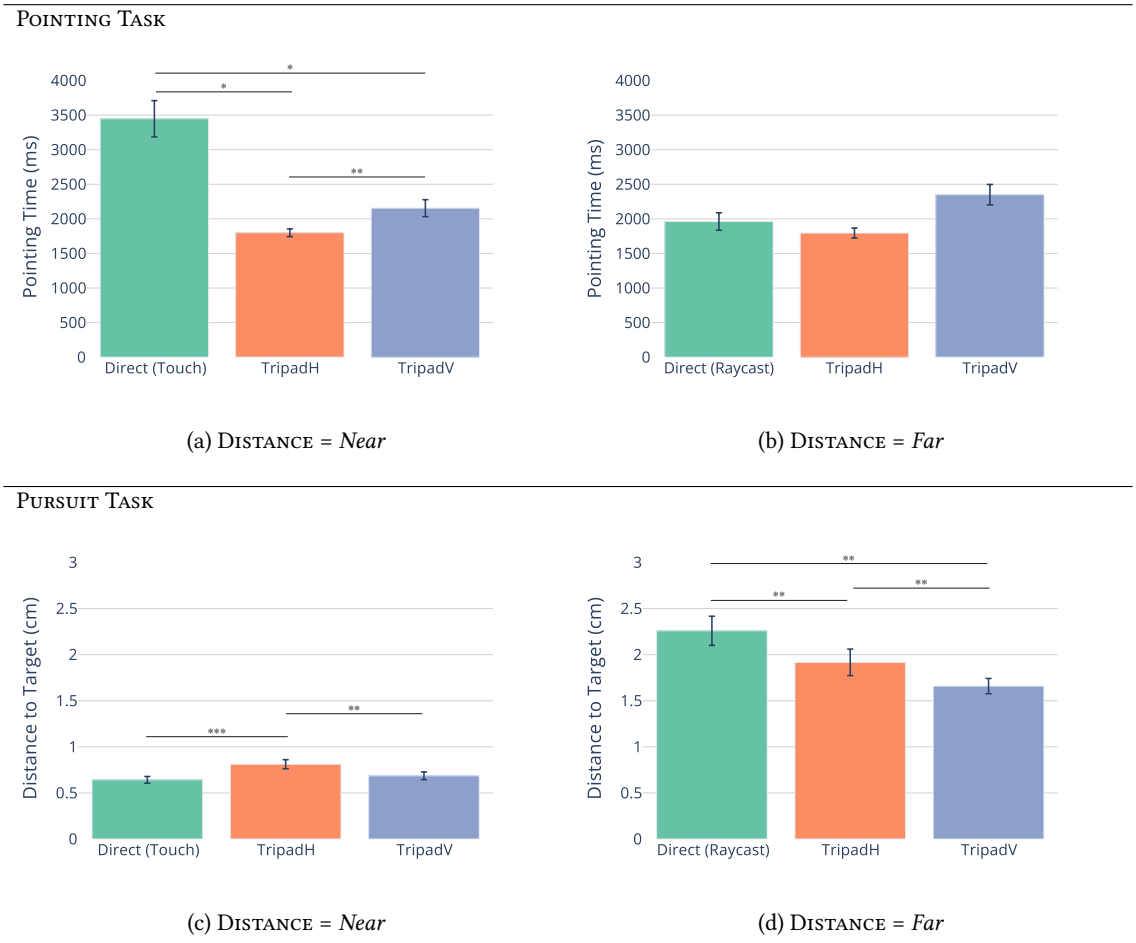
Building upon recent studies about AR input [6, 10], we consider both discrete input with a pointing task and continuous input with a pursuit task. In both cases participants interact within the bounds of a square canvas. This virtual canvas is located either 50 cm (condition *Near*) or 200 cm (condition *Far*) in front of the participant, and the length of its sides is either 15 cm or 35 cm (Figure 9-a). Control-display gain for indirect input techniques (TriPad conditions) is set so that a 1 cm finger movement translates to a cursor movement that corresponds to $1/8^{th}$ of the canvas' side length.

5.4.1 Pointing Task. Participants have to complete 8 pointing tasks in different directions by acquiring a series of targets arranged in a circular manner (ISO 9241-9 standard [24]). The next target to acquire is colored white. It turns green as soon as the cursor enters it. Participants then have to click to acquire the target, causing the subsequent target to turn white. All targets in a series have the same size: 1.5 cm (*Near*) or 3.5 cm (*Far*) in diameter; and are laid out on a circle 9 cm (*Near*) or 21 cm (*Far*) in diameter. We discard the first pointing task in each series since the cursor is located in the center of the canvas. The cursor is represented as a black cross. The sequence of events is a point-and-click (Move and Click events in Figure 3 for TriPad conditions).

5.4.2 Pursuit Task. This task operationalizes continuous interactions that typically occur during the direct manipulation of objects. We adopt a design analogous to that of recent experiments [6, 10], based on the pursuit tracking task initially introduced by Poulton [45]. Participants have to follow a circular target (size: 1.5 cm (*Near*) or 3.5 cm (*Far*) in diameter) that moves in a quasi-random manner inside the canvas for 15 seconds. Participants are instructed to move the cursor so as to minimize its distance to the target object. While the target's trajectory appears random to participants, it is actually pre-computed to ensure consistent levels of difficulty across participants and conditions. In these pointing tasks, the controlled cursor is represented as a disc (size: 0.75 cm (*Near*) or 1.75 cm (*Far*) in diameter). Its color varies along a gradient from black to red based on its distance to the moving target. The sequence of events is a series of drags (Drag events in Figure 3 for TriPad conditions).

5.5 Design and Procedure

The experimental procedure lasted 60-to-90 minutes per participant. It started with reading and signing the participation consent form. As illustrated in Figure 10, participants then went through four different phases where they experienced 2 Task { *Pointing*, *Pursuit* } × 2 Distance { *Near*, *Far* } conditions. Distance conditions were blocked by Task. Presentation order of Task blocks was counterbalanced across participants. Each Task block was divided into two Distance sub-blocks, whose presentation order was counterbalanced across Task conditions and participants. Within each Task × Distance condition, participants performed trials with each of the three techniques (Technique ∈ { *Direct*, *TriPadH*, *TriPadV* }). Trials were blocked per Technique. The presentation order of Technique blocks across Task × Distance conditions and participants was counterbalanced with a Latin Square. Within each Technique block, participants performed a series of 7 trials. The first trial served for training. Participants could repeat this first training trial at will until they felt comfortable with the technique. A *Pursuit* trial lasted 15 seconds. A *Pointing* trial consisted of 8 target acquisitions. The pointing distance to the first target was smaller than for the 7 subsequent ones in a trial, as the cursor was initially positioned at the center of the scene. In our analyses we thus ignored the first target acquisition of all *Pointing* trials. In TriPad conditions, participants first had to create a TriPad to initiate a training trial or the actual sequence of the six measured trials. If the TriPad was created with an orientation that neither matched a vertical nor a horizontal orientation (with a 10° tolerance), participants were notified and asked to create a TriPad again (this happened only 5 times in total: 4 times with one participant, and once with another participant).



(a) Distance = *Near* (b) Distance = *Far*

(c) Distance = *Near* (d) Distance = *Far*

Figure 11: Participants' average performance in the four phases of Experiment 2: (a-b) pointing time (in ms) per Technique; (c-d) average distance to the moving target (in cm) per Technique. Error bars represent the 95% confidence interval relative to all the data points collected in the corresponding condition. * indicates statistical significance (* : $p < 0.001$, **: $p < 0.01$, *: $p < 0.05$).**

In the *Pursuit* condition, we collected 12 participants × 2 Distance × 3 Technique × 6 Repetition = 432 trials for analysis. In the *Pointing* condition, we collected 12 participants × 2 Distance × 3 Technique × 6 Repetition × 7 targets = 3024 trials for analysis.

At the end of each Task × Distance block, participants filled a questionnaire where they had to rate the three techniques in terms of comfort, ease of use, efficiency and tiredness using 5-point Likert scales. They also had to rank the techniques.

5.6 Results

We analyze participants' performance measures (*Pointing* and *Pursuit* tasks) and their answers to the questionnaire. As the *Direct* technique differs between tasks performed at *Near* and *Far* distances, we do not consider Distance as a factor to be crossed with Technique, but rather study the effect of factor Technique under the four Task × Distance phases of the experiment.

5.6.1 Performance. Figure 11 illustrates participants' task performance in the four phases of Experiment 2.

Pointing Task. The main measure for pointing tasks is the target acquisition time, *i.e.*, the time interval between two successful clicks on two consecutive targets.

When Distance=*Near* (Figure 11-a), Technique has a significant effect on average acquisition time ($F_{2,22} = 8.74, p < 0.001, \eta^2_G = 0.010.32$), with all pairwise comparisons yielding significant differences. *TriPadH* (1799 ms) is the fastest, followed by *TriPadV* (2155 ms) and then *Direct* Touch (3447 ms). We attribute the inferior performance of *Direct* Touch to the fairly high pointing difficulty. We deliberately selected small targets for our experiments to assess the performance of the tested techniques in precision tasks (Hypothesis H_1). These targets were actually smaller than industry recommendations [38, 39]. Although hand tracking technology is generally accurate, particularly in tracking hand postures and movements, there tends to be a small discrepancy in estimating the absolute positions of individual joints. This slight inaccuracy can make precise direct acquisition with the index fingertip challenging. In the experiment, participants had difficulty maintaining their index fingertip within the boundaries of the target, both when entering and

exiting the small target. In contrast, TriPad, which relies on a cursor controlled indirectly through relative finger movements, was less affected and thus better suited to such precision tasks. An analysis of the number of clicks outside the target's bounds (wrong clicks) supports this as we observe a significant effect of Technique on the number of wrong clicks ($F_{2,22} = 15.4$, $p < 0.001$, $\eta^2_G = 0.0010.3$) with *Direct* Touch causing significantly more wrong clicks than both TriPad techniques (p-corr = 0.01 for *TriPadH* and p-corr < 0.001 for *TriPadV*). In contrast, TriPad's indirect input method facilitated precise cursor position adjustments with relative movements, enabling participants to reach a good speed-accuracy trade-off. The superior performance of *TriPadH* over *TriPadV* may be explained by users' familiarity with trackpads and mice. Ten participants out of twelve spontaneously stated that *TriPadH* was like a mouse, whereas only one participant made this analogy for *TriPadV*.

When Distance=*Far* (Figure 11-b), Technique does not have a significant effect on average acquisition time (p=0.11). In this condition, *Direct* Raycast is very efficient to cover large distances as a small adjustment to the ray's orientation results in a significant displacement at the endpoint of the ray. A typical limitation of raycast techniques is their relatively low precision, however. To compensate for that, the headset input system implements a filtering mechanism that effectively reduces jitter. This enhancement enables participants to point and click with good efficiency. Overall, *Direct* Raycast and the TriPad techniques perform on par in this context. Looking at the two TriPad techniques in particular, it is worth noting that they significantly differ when Distance = *Near* (p-corr = 0.003), while when Distance = *Far* the effect of Technique is not significant. Considering that both TriPads are indirect input techniques that have the same control-display gain, it is not immediately clear why their relative performance would be influenced by the distance to the scene. Tracking accuracy appeared to be less robust in vertical conditions overall, leading participants to test and adopt different hand postures. We observed seven distinct hand postures, with variations in which finger makes contact and remains straight when moving, as well as which fingers are lifted when clicking. By contrast, in horizontal conditions, the variability was significantly lower, with 11 participants consistently adopting the same hand posture – keeping their hands roughly flat and lifting only the index finger to perform clicks. This led to more variability overall with *TriPadV*. A two-way anova test on observations with TriPad conditions for pointing tasks only revealed that the interaction effect Distance × Technique is not significant (p=0.24).

Pursuit Task. The main measure for pursuit tasks is the average distance between the moving target object and the cursor over 15 seconds.

When Distance=*Near* (Figure 11-c), Technique has a significant effect on mean average distance to target ($F_{2,22} = 16.75$, $p < 0.001$, $\eta^2_G = 0.0010.3$). Notably, *TriPadH* (0.81 cm) performs significantly worse than both Touch *Direct* (0.64 cm) and *TriPadV* (0.69 cm). However, it is important to note that there is no significant difference between *TriPadV* and *Direct* Touch (p=0.13). In a pursuit task, what matters most is participants' ability to execute smooth and continuous movements. It is therefore not surprising that a direct input technique such as Touch outperforms an indirect input technique such as *TriPadH*; the same as how poorly a trackpad supports freeform input when compared with touch input on a tactile screen. Interestingly, the performance of *Direct* Touch and *TriPadV* seem quite similar, echoing observations made in the ARPads project [6], where participants performed better when the mid-air input pad and the graphical scene had the same orientation. One of our participants mentioned that it was easier to map the visual movement to the physical movement when the pad was vertical.

When Distance=*Far* (Figure 11-d), Technique has a significant effect on mean average distance to target ($F_{2,22} = 16.12$, $p < 0.001$, $\eta^2_G = 0.0010.27$), all pairs being significantly different. *TriPadV* is the most precise (1.65 cm), followed by *TriPadH* (1.92 cm) and then *Direct* Raycast (2.26 cm). In this condition, far-field raycast input proved fairly imprecise. It can be surprising as, on the opposite, it achieved good precision in pointing tasks. However, in this specific pursuit context, the filtering mechanism to reduce jitter, which is especially strong when users maintain a pinch posture, made it challenging for participants to execute fluid and precise continuous movements. While this filtering mechanism is effective in enhancing clicking precision, it gives a sensation of a "sticky" ray with some inertia when users perform drag interactions by maintaining a pinch gesture. Comparing TriPad conditions, findings are similar to those of the *Near* condition, with participants performing better with *TriPadV* than with *TriPadH*. This aligns with our previous observations regarding the reduced difficulty of tracking continuous movements with indirect input when the control and visual planes share the same orientation. It is worth noting that the larger average distances observed for all techniques in the *Far* condition, compared to the *Near* condition, are an expected outcome of our experimental design. In the *Far* condition, the scene, and the target in particular, are larger (Figure 9). Consequently, the distance to the target's center tends to be greater in this scenario.

5.6.2 Hand postures. In this experiment we observed participants adopting various hand postures while operating TriPads, especially when performing Move and Contact-Click actions in *Pointing* tasks. We recorded six distinct postures with *TriPadV* and three with *TriPadH*. These postures differ in the number of fingers in contact with the surface, and in the extent to which fingers are curved. Some participants preferred keeping only their index and middle fingers straight and in contact, while others kept their thumb in contact as well, or even all five fingers. Some participants explicitly mentioned that using all five fingers mimics the posture they adopt when interacting with a physical mouse. As the recognition of events only involves tracking the thumb and middle finger (see Section 4.6), users can choose and adopt the posture that best suits them in terms of preference and comfort.

5.6.3 Questionnaire. Figure 12 gives an overview of participants' answers to the questionnaire after each Distance × Task phase of the experiment. We ran a series of Friedman tests to analyze the effect of Technique on the different scales for each of these phases in order to report all statistically significant differences (p's < 0.05).

In terms of *Comfort*, participants consistently favored *TriPadH* over the other techniques. In particular, *TriPadH* was significantly more comfortable for pointing tasks. For pursuits at a distance, both

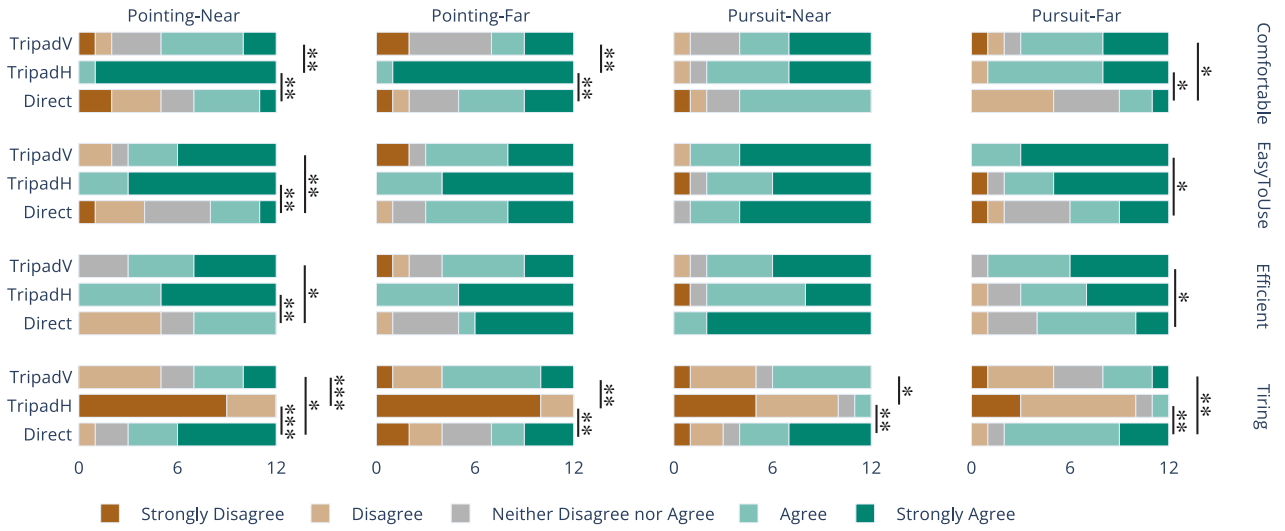


Figure 12: Distribution of participants' 5-pt ratings per Task and Technique for statements *"The technique was {comfortable, easy to use, efficient, tiring}"*. * indicates statistical significance (* : $p < 0.001$, **: $p < 0.01$, *: $p < 0.05$).**

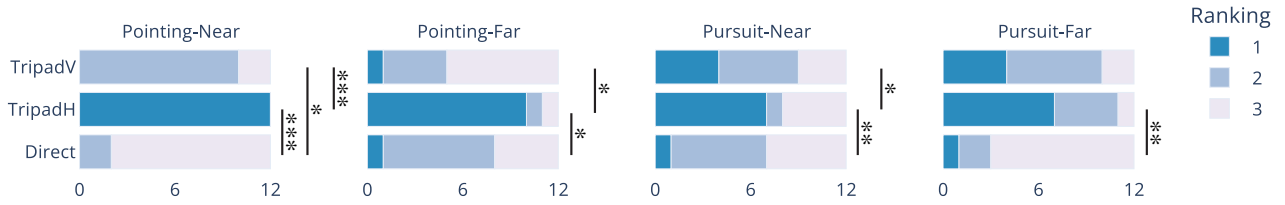


Figure 13: Distribution of participants' preferences per Task and Technique (1 is the most preferred, 3 is the least preferred). * indicates statistical significance (* : $p < 0.001$, **: $p < 0.01$, *: $p < 0.05$).**

TriPadH and *TriPadV* were deemed more comfortable than *Direct* Raycast.

Participants' evaluation of *Ease-of-Use* and *Efficiency* were largely aligned, with *TriPadV* being perceived as significantly better than *Direct* Raycast for pointing at a distance. Additionally, both TriPads outperformed *Direct* Touch for pointing tasks within arm's reach.

Regarding *Fatigue*, participants consistently rated *TriPadH* as less tiring than the other techniques. This trend persisted across all four phases. Except for pursuits at a distance, *TriPadH* also proved significantly less tiring than *TriPadV* which, in turn, was rated as significantly less tiring than the direct techniques for pursuit at a distance and pointing within arm's reach.

Overall, participants found significant advantages to TriPads in terms of comfort and fatigue, and with *TriPadH* in particular, without perceiving any significant drop in terms of *Ease-of-Use* and *Efficiency*. This is consistent with performance measures reported earlier. This resulted in participants generally preferring *TriPadH* over the other techniques (Figure 13). For pointing tasks, *TriPadH* was ranked significantly higher than the other two techniques. *TriPadH* was also ranked significantly higher than *Direct* Raycast for pursuit tasks. Finally, *TriPadV* was significantly preferred to *Direct* Touch when pointing within arm's reach specifically.

5.7 Summary

Our observations support hypothesis H_1 (*TriPad indirect techniques cause less fatigue and are more accurate compared to direct techniques*). TriPad techniques are less tiring than direct conditions overall. More specifically, *TriPadH* is the least tiring technique across all conditions. In terms of performance, TriPad techniques are fast and precise. For point-and-click tasks (*Pointing*), TriPad techniques, and *TriPadH* in particular, achieve a very good speed-accuracy trade-off, yielding target acquisition times that are either as good as (*Far*) or better than (*Near*) direct AR input techniques. For dragging tasks (*Pursuit*), TriPad techniques demonstrate similar or even greater precision compared to direct AR input techniques, with the exception of *TriPadH* which had lower precision compared to direct touch input.

Our findings are more nuanced regarding hypothesis H_2 (*TriPadH outperforms TripadV*). *TriPadH* outperforms *TriPadV* in pointing tasks, but *TriPadV* performs better than *TriPadH* in dragging tasks. It is worth noting that for the latter task, the results might

have been different, had the virtual canvas been oriented horizontally. However, in many practical scenarios a vertical orientation better fits with the natural gaze direction, making our experimental setup representative of the most common use case.

5.8 Limitations

There are limitations to consider in this experiment as well. Whereas the first experiment primarily aimed at evaluating the technical viability of our approach, this second experiment was rather about evaluating user pointing and pursuit task performance. The input technique was thus the primary factor of interest. To capture a representative range of interactions, we considered two high-level aspects: type of task (discrete and continuous) and distance (near and far) and had to keep other parameters constant to keep the experiment tractable. As a next step, lower-level aspects could be investigated in more detail, including: different task difficulties (pointing and pursuit target parameters); as well as different implementations of cursor control. As one of our key design objectives was to address the limitations associated with direct input methods, we deliberately tested challenging tasks, including targets with smaller sizes than recommended in industry guidelines as those account for direct input's limited precision. This has enabled us to observe that participants can better handle difficult tasks with TriPad indirect input than they can with direct techniques, opening the door for alternative, more compact widget layouts in AR and MR user interface design. Furthermore, like any other technique that supports indirect input, TriPad could be further improved by optimizing Control-Display gain [13] with fine-tuned transfer functions [17]. We chose a CD-gain value that we deemed reasonable based on informal tests. Another possible next step would be to conduct a more comprehensive and formal exploration of possible values to fine-tune TriPad's implementation, which could enhance usability even more.

6 CONCLUSION AND FUTURE WORK

The above empirical results indicate, beyond comparative task performance measures, that participants are quickly able to create TriPads and interact with them. These results also indicate that interacting with a TriPad – especially on a horizontal surface – is often preferred to the mid-air alternative, in line with findings from previous studies [7, 32, 36].

This demonstrates the viability of an approach to the opportunistic use of passive surfaces based on a purely *hand-centric* view of the problem. Such a view involves analyzing hand posture and movements only, removing the need for any kind of object recognition or instrumentation. Prior interaction techniques that made use of a hand-centric view include Gripmarks and TouchTokens. GripMarks [62] associates distinct hand grips with different 3D shape primitives to detect which physical object the user is holding. TouchTokens [40] recognizes which passive token users are manipulating on a capacitive touch surface based on the relative spatial configuration of their fingers. TriPad adopts a hand-centric view as well, but to address yet another problem: turn any passive surface into a touch pad.

The fundamental premise of our work was that, with recent improvements in hand tracking precision and robustness [14], it should be possible to create touch pads opportunistically based on hand posture and finger movement data alone. Results from our two experiments indicate that this can be achieved with off-the-shelf hardware. Indirect interaction with a TriPad further requires that the hands be trackable with high precision even if not visible in the user's field of view, but some recent headsets – as well as several upcoming ones – are already embedding this capability.

Given the advantages of freehand interaction [16], future generations of AR eyewear will keep featuring high-quality hand tracking capabilities. Future generations of AR eyewear are also expected to keep miniaturizing. In that respect, power consumption is an issue, and user input, as much as all other embedded functionalities, should aim to conserve energy. Hand tracking is arguably still resource-intensive. But progress is being made on this front (see, *e.g.*, [14]) and at this point the functionality has demonstrated too much value to be discarded. From that perspective, TriPad's key properties is that it does not require any additional sensor and that it adds very little computing cost of its own on top of hand tracking.

Another good property of our hand-centric approach is that it works on a variety of surfaces as it is less sensitive than previous approaches to the material's roughness and infrared reflectivity. But beyond the question of material, adopting a hand-centric approach also means that the surface can feature macro "irregularities", such as, *e.g.*, objects laid on a desk [58], or physical knobs and sliders protruding from a control panel with otherwise free planar areas. The surface can also feature patterns, embossed or carved as in a perforated panel.

Avenues for future work include investigating ways to increase TriPad's expressive power while keeping the approach fully hand-centric. A first option would be to add support for idle finger gestures [31] based on hand tracking only. A second, more ambitious possibility would be to consider both hands for interaction. This could be useful to create larger pads, computing the best fitting plane. Pushing this idea further, information from the two hands could perhaps be used to interpolate simple concave or convex surfaces when the hands are in clearly-distinct planes, though the feasibility of such an approach remains to be determined.

Declaring TriPads with two hands would also provide a means for users to instantiate different types of TriPads based, for instance, on the distance between the two hands, or based on their relative orientation. Users could then specify that they want to create a regular TriPad for 2D tap and drag input, a TriPad for typing, or some custom TriPad for sketching or gesturing commands.

Finally, TriPad requires three fingers to define a plane, but these need not necessarily be the thumb, middle and pinky fingers. Automatically selecting which fingers to consider based on tracking stability metrics could further increase alignment precision and would improve the technique's accessibility as well.

ACKNOWLEDGMENTS

We would like to thank all participants to our studies. Additional credits: Vincent Cavez composed the companion video's soundtrack. This project has been partially supported by the following grants: ANRT (CIFRE 2022/0586); ANR Continuum (ANR-21-ESRE-0030); ANR Interplay (ANR-21-CE33-0022).

REFERENCES

[1] Michelle Annett, Tovi Grossman, Daniel Wigdor, and George Fitzmaurice. 2011. Medusa: A Proximity-Aware Multi-Touch Tabletop. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology* (Santa Barbara, California, USA) *(UIST '11)*. Association for Computing Machinery, New York, NY, USA, 337–346. https://doi.org/10.1145/2047196.2047240

[2] Apple. 2023. Design for Spatial Input. https://developer.apple.com/videos/play/wwdc2023/10073 - last accessed 2024-01-23.

[3] Myroslav Bachynskyi, Gregorio Palmas, Antti Oulasvirta, Jürgen Steimle, and Tino Weinkauf. 2015. Performance and Ergonomics of Touch Surfaces: A Comparative Study Using Biomechanical Simulation. 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, 1817–1826. https://doi.org/10.1145/2702123.2702607

[4] Ananta Narayanan Balaji, Clayton Kimber, David Li, Shengzhi Wu, Ruofei Du, and David Kim. 2023. RetroSphere: Self-Contained Passive 3D Controller Tracking for Augmented Reality. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 6, 4, Article 157 (jan 2023), 36 pages. https://doi.org/10.1145/3569479

[5] Joanna Bergström and Kasper Hornbæk. 2019. Human–Computer Interaction on the Skin. *ACM Comput. Surv.* 52, 4, Article 77 (aug 2019), 14 pages. https://doi.org/10.1145/3332166

[6] Eugenie Brasier, Olivier Chapuis, Nicolas Ferey, Jeanne Vezien, and Caroline Appert. 2020. ARPads: Mid-air Indirect Input for Augmented Reality. In *2020 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, 332–343. https://doi.org/10.1109/ISMAR50242.2020.00060

[7] Yi Fei Cheng, Tiffany Luong, Andreas Rene Fender, Paul Streli, and Christian Holz. 2022. ComforTable User Interfaces: Surfaces Reduce Input Error, Time, and Exertion for Tabletop and Mid-air User Interfaces. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, 150–159. https://doi.org/10.1109/ISMAR55827.2022.00029

[8] Andrea Colaço, Ahmed Kirmani, Hye Soo Yang, Nan-Wei Gong, Chris Schmandt, and Vivek K. Goyal. 2013. Mime: Compact, Low Power 3D Gesture Sensing for Interaction with Head Mounted Displays. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology* (St. Andrews, Scotland, United Kingdom) *(UIST '13)*. Association for Computing Machinery, New York, NY, USA, 227–236. https://doi.org/10.1145/2501988.2502042

[9] Neil Xu Fan and Robert Xiao. 2022. Reducing the Latency of Touch Tracking on Ad-Hoc Surfaces. *Proc. ACM Hum.-Comput. Interact.* 6, ISS, Article 577 (nov 2022), 11 pages. https://doi.org/10.1145/3567730

[10] Tiare Feuchtner and Jörg Müller. 2018. Ownershift: Facilitating Overhead Interaction in Virtual Reality with an Ownership-Preserving Hand Space Shift. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology* (Berlin, Germany) *(UIST '18)*. Association for Computing Machinery, New York, NY, USA, 31–43. https://doi.org/10.1145/3242587.3242594

[11] Jun Gong, Aakar Gupta, and Hrvoje Benko. 2020. Acustico: Surface Tap Detection and Localization Using Wrist-Based Acoustic TDOA Sensing. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '20)*. Association for Computing Machinery, New York, NY, USA, 406–419. https://doi.org/10.1145/3379337.3415901

[12] Yizheng Gu, Chun Yu, Zhipeng Li, Weiqi Li, Shuchang Xu, Xiaoying Wei, and Yuanchun Shi. 2019. Accurate and Low-Latency Sensing of Touch Contact on Any Surface with Finger-Worn IMU Sensor. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology* (New Orleans, LA, USA) *(UIST '19)*. Association for Computing Machinery, New York, NY, USA, 1059–1070. https://doi.org/10.1145/3332165.3347947

[13] Ravin Balakrishnan Géry Casiez, Daniel Vogel and Andy Cockburn. 2008. The Impact of Control-Display Gain on User Performance in Pointing Tasks. *Human–Computer Interaction* 23, 3 (2008), 215–250. https://doi.org/10.1080/07370020802278163

[14] Shangchen Han, Beibei Liu, Randi Cabezas, Christopher D. Twigg, Peizhao Zhang, Jeff Petkau, Tsz-Ho Yu, Chun-Jung Tai, Muzaffer Akbay, Zheng Wang, Asaf Nitzan, Gang Dong, Yuting Ye, Lingling Tao, Chengde Wan, and Robert Wang. 2020. MEgATrack: Monochrome Egocentric Articulated Hand-Tracking for Virtual Reality. *ACM Trans. Graph.* 39, 4, Article 87 (aug 2020), 13 pages. https://doi.org/10.1145/3386569.3392452

[15] Shangchen Han, Beibei Liu, Robert Wang, Yuting Ye, Christopher D. Twigg, and Kenrick Kin. 2018. Online Optical Marker-Based Hand Tracking with Deep Labels. *ACM Trans. Graph.* 37, 4, Article 166 (jul 2018), 10 pages. https://doi.org/10.1145/3197517.3201399

[16] Shangchen Han, Po-Chen Wu, Yubo Zhang, Beibei Liu, Linguang Zhang, Zheng Wang, Weiguang Si, Peizhao Zhang, Yujun Cai, Tomas Hodan, Randi Cabezas, Luan Tran, Muzaffer Akbay, Tsz-Ho Yu, Cem Keskin, and Robert Wang. 2022. UmeTrack: Unified Multi-View End-to-End Hand Tracking for VR. In *SIGGRAPH Asia 2022 Conference Papers* (Daegu, Republic of Korea) *(SA '22)*. Association for Computing Machinery, New York, NY, USA, Article 50, 9 pages. https://doi.org/10.1145/3550469.3555378

[17] Raiza Hanada, Damien Masson, Géry Casiez, Mathieu Nancel, and Sylvain Malacria. 2021. Relevance and Applicability of Hardware-Independent Pointing Transfer Functions. In *The 34th Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '21)*. Association for Computing Machinery, New York, NY, USA, 524–537. https://doi.org/10.1145/3472749.3474767

[18] Chris Harrison, Hrvoje Benko, and Andrew D. Wilson. 2011. OmniTouch: Wearable Multitouch Interaction Everywhere. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology* (Santa Barbara, California, USA) *(UIST '11)*. Association for Computing Machinery, New York, NY, USA, 441–450. https://doi.org/10.1145/2047196.2047255

[19] Fengming He, Xiyun Hu, Jingyu Shi, Xun Qian, Tianyi Wang, and Karthik Ramani. 2023. Ubi Edge: Authoring Edge-Based Opportunistic Tangible User Interfaces in Augmented Reality. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 461, 14 pages. https://doi.org/10.1145/3544548.3580704

[20] Steven J. Henderson and Steven Feiner. 2008. Opportunistic Controls: Leveraging Natural Affordances as Tangible User Interfaces for Augmented Reality. In *Proceedings of the 2008 ACM Symposium on Virtual Reality Software and Technology* (Bordeaux, France) *(VRST '08)*. Association for Computing Machinery, New York, NY, USA, 211–218. https://doi.org/10.1145/1450579.1450625

[21] Juan David Hincapié-Ramos, Xiang Guo, Paymahn Moghadasian, and Pourang Irani. 2014. Consumed Endurance: A Metric to Quantify Arm Fatigue of Mid-Air Interactions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Toronto, Ontario, Canada) *(CHI '14)*. Association for Computing Machinery, New York, NY, USA, 1063–1072. https://doi.org/10.1145/2556288.2557130

[22] Ken Hinckley, Seongkook Heo, Michel Pahud, Christian Holz, Hrvoje Benko, Abigail Sellen, Richard Banks, Kenton O'Hara, Gavin Smyth, and William Buxton. 2016. Pre-Touch Sensing for Mobile Interaction. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI '16)*. Association for Computing Machinery, New York, NY, USA, 2869–2881. https://doi.org/10.1145/2858036.2858095

[23] Yi-Ta Hsieh, Antti Jylhä, Valeria Orso, Luciano Gamberini, and Giulio Jacucci. 2016. Designing a Willing-to-Use-in-Public Hand Gestural Interaction Technique for Smart Glasses. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI '16)*. Association for Computing Machinery, New York, NY, USA, 4203–4215. https://doi.org/10.1145/2858036.2858436

[24] ISO. 2000. 9241-9 Ergonomic requirements for office work with visual display terminals (VDTs)-Part 9: Requirements for non-keyboard input devices. *International Organization for Standardization* (2000), 47 pages.

[25] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. 2011. KinectFusion: Real-Time 3D Reconstruction and Interaction Using a Moving Depth Camera. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology* (Santa Barbara, California, USA) *(UIST '11)*. Association for Computing Machinery, New York, NY, USA, 559–568. https://doi.org/10.1145/2047196.2047270

[26] Wolf Kienzle and Ken Hinckley. 2014. LightRing: Always-Available 2D Input on Any Surface. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology* (Honolulu, Hawaii, USA) *(UIST '14)*. Association for Computing Machinery, New York, NY, USA, 157–160. https://doi.org/10.1145/2642918.2647376

[27] Hideki Koike, Yoichi Sato, and Yoshinori Kobayashi. 2001. Integrating Paper and Digital Information on EnhancedDesk: A Method for Realtime Finger Tracking on an Augmented Desk System. *ACM Trans. Comput.-Hum. Interact.* 8, 4 (dec 2001), 307–322. https://doi.org/10.1145/504704.504706

[28] Eric Larson, Gabe Cohn, Sidhant Gupta, Xiaofeng Ren, Beverly Harrison, Dieter Fox, and Shwetak Patel. 2011. HeatWave: Thermal Imaging for Surface User Interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vancouver, BC, Canada) *(CHI '11)*. Association for Computing Machinery, New York, NY, USA, 2565–2574. https://doi.org/10.1145/1978942.1979317

[29] Lik-Hang Lee and Pan Hui. 2018. Interaction Methods for Smart Glasses: A Survey. *IEEE Access* 6 (2018), 28712–28732. https://doi.org/10.1109/ACCESS.2018.2831081

[30] Julien Letessier and François Bérard. 2004. Visual Tracking of Bare Fingers for Interactive Surfaces. In *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology* (Santa Fe, NM, USA) *(UIST '04)*. Association for Computing Machinery, New York, NY, USA, 119–122. https://doi.org/10.1145/1029632.1029652

[31] Hyunchul Lim, Jungmin Chung, Changhoon Oh, SoHyun Park, Joonhwan Lee, and Bongwon Suh. 2018. Touch+Finger: Extending Touch-Based User Interface Capabilities with "Idle" Finger Gestures in the Air. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology* (Berlin, Germany) *(UIST '18)*. Association for Computing Machinery, New York, NY, USA, 335–346. https://doi.org/10.1145/3242587.3242651

[32] Robert W. Lindeman, John L. Sibert, and James K. Hahn. 1999. Towards Usable VR: An Empirical Study of User Interfaces for Immersive Virtual Environments.

In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Pittsburgh, Pennsylvania, USA) *(CHI '99)*. Association for Computing Machinery, New York, NY, USA, 64–71. https://doi.org/10.1145/302979.302995

[33] Feiyu Lu and Yan Xu. 2022. Exploring Spatial UI Transition Mechanisms with Head-Worn Augmented Reality. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI '22)*. Association for Computing Machinery, New York, NY, USA, Article 550, 16 pages. https://doi.org/10.1145/3491102.3517723

[34] I. Scott MacKenzie and R. William Soukoreff. 2003. Phrase Sets for Evaluating Text Entry Techniques. In *CHI '03 Extended Abstracts on Human Factors in Computing Systems* (Ft. Lauderdale, Florida, USA) *(CHI EA '03)*. Association for Computing Machinery, New York, NY, USA, 754–755. https://doi.org/10.1145/765891.765971

[35] Shahzad Malik and Joe Laszlo. 2004. Visual Touchpad: A Two-Handed Gestural Input Device. In *Proceedings of the 6th International Conference on Multimodal Interfaces* (State College, PA, USA) *(ICMI '04)*. Association for Computing Machinery, New York, NY, USA, 289–296. https://doi.org/10.1145/1027933.1027980

[36] Daniel Medeiros, Graham Wilson, Mark Mcgill, and Stephen Anthony Brewster. 2023. The Benefits of Passive Haptics and Perceptual Manipulation for Extended Reality Interactions in Constrained Passenger Spaces. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 232, 19 pages. https://doi.org/10.1145/3544548.3581079

[37] Manuel Meier, Paul Streli, Andreas Fender, and Christian Holz. 2021. TapID: Rapid Touch Interaction in Virtual Reality using Wearable Sensing. In *IEEE Virtual Reality and 3D User Interfaces (VR)*. IEEE, 519–528. https://doi.org/10.1109/VR50410.2021.00076

[38] Meta. 2023. Designing for Hands - Best Practices. https://developer.oculus.com/resources/hands-design-bp - last accessed 2024-01-23.

[39] Microsoft. 2022. Mixed Reality Design Interaction. https://learn.microsoft.com/en-us/windows/mixed-reality/design/direct-manipulation - last accessed 2024-01-23.

[40] Rafael Morales González, Caroline Appert, Gilles Bailly, and Emmanuel Pietriga. 2016. TouchTokens: Guiding Touch Patterns with Passive Tokens. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI '16)*. Association for Computing Machinery, New York, NY, USA, 4189–4202. https://doi.org/10.1145/2858036.2858041

[41] Sundar Murugappan, Vinayak, Niklas Elmqvist, and Karthik Ramani. 2012. Extended Multitouch: Recovering Touch Posture and Differentiating Users Using a Depth Camera. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology* (Cambridge, Massachusetts, USA) *(UIST '12)*. Association for Computing Machinery, New York, NY, USA, 487–496. https://doi.org/10.1145/2380116.2380177

[42] Takehiro Niikura, Yoshihiro Watanabe, and Masatoshi Ishikawa. 2014. Anywhere Surface Touch: Utilizing Any Surface as an Input Area. In *Proceedings of the 5th Augmented Human International Conference* (Kobe, Japan) *(AH '14)*. Association for Computing Machinery, New York, NY, USA, Article 39, 8 pages. https://doi.org/10.1145/2582051.2582090

[43] Ju Young Oh, Ji-Hyung Park, and Jung-Min Park. 2020. FingerTouch: Touch Interaction Using a Fingernail-Mounted Sensor on a Head-Mounted Display for Augmented Reality. *IEEE Access* 8 (2020), 101192–101208. https://doi.org/10.1109/ACCESS.2020.2997972

[44] Brice Parilusyan, Marc Teyssier, Valentin Martinez-Missir, Clément Duhart, and Marcos Serrano. 2022. Sensurfaces: A Novel Approach for Embedded Touch Sensing on Everyday Surfaces. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 6, 2, Article 67 (jul 2022), 19 pages. https://doi.org/10.1145/3534616

[45] Eustace Christopher Poulton. 1974. *Tracking skill and manual control*. Academic press.

[46] Yilei Shi, Haimo Zhang, Jiashuo Cao, and Suranga Nanayakkara. 2020. VersaTouch: A Versatile Plug-and-Play System That Enables Touch Interactions on Everyday Passive Surfaces. In *Proceedings of the Augmented Humans International Conference* (Kaiserslautern, Germany) *(AHs '20)*. Association for Computing Machinery, New York, NY, USA, Article 26, 12 pages. https://doi.org/10.1145/3384657.3384778

[47] Yilei Shi, Haimo Zhang, Kaixing Zhao, Jiashuo Cao, Mengmeng Sun, and Suranga Nanayakkara. 2020. Ready, Steady, Touch! Sensing Physical Contact with a Finger-Mounted IMU. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 4, 2, Article 59 (jun 2020), 25 pages. https://doi.org/10.1145/3397309

[48] Tomoya Tada and Shigeyuki Hirai. 2020. Transmissive LED Touch Display for Engineered Marble. In *Adjunct Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '20 Adjunct)*. Association for Computing Machinery, New York, NY, USA, 145–147. https://doi.org/10.1145/3379350.3416162

[49] UltraLeap. 2023. XR Design Guidelines. https://docs.ultraleap.com/xr-guidelines - last accessed 2024-01-23.

[50] Daniel Vogel and Patrick Baudisch. 2007. Shift: A Technique for Operating Pen-Based Interfaces Using Touch. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI '07)*. Association for Computing Machinery, New York, NY, USA, 657–666. https://doi.org/10.1145/1240624.1240727

[51] Daniel Wigdor, Sarah Williams, Michael Cronin, Robert Levy, Katie White, Maxim Mazeev, and Hrvoje Benko. 2009. Ripples: Utilizing per-Contact Visualizations to Improve User Interaction with Touch Displays. In *Proceedings of the 22nd Annual ACM Symposium on User Interface Software and Technology* (Victoria, BC, Canada) *(UIST '09)*. Association for Computing Machinery, New York, NY, USA, 3–12. https://doi.org/10.1145/1622176.1622180

[52] Andrew D. Wilson. 2004. TouchLight: An Imaging Touch Screen and Display for Gesture-Based Interaction. In *Proceedings of the 6th International Conference on Multimodal Interfaces* (State College, PA, USA) *(ICMI '04)*. Association for Computing Machinery, New York, NY, USA, 69–76. https://doi.org/10.1145/1027933.1027946

[53] Andrew D. Wilson. 2010. Using a Depth Camera as a Touch Sensor. In *ACM International Conference on Interactive Tabletops and Surfaces* (Saarbrücken, Germany) *(ITS '10)*. Association for Computing Machinery, New York, NY, USA, 69–72. https://doi.org/10.1145/1936652.1936665

[54] Andrew D. Wilson and Hrvoje Benko. 2010. Combining Multiple Depth Cameras and Projectors for Interactions on, above and between Surfaces. In *Proceedings of the 23nd Annual ACM Symposium on User Interface Software and Technology* (New York, New York, USA) *(UIST '10)*. Association for Computing Machinery, New York, NY, USA, 273–282. https://doi.org/10.1145/1866029.1866073

[55] Jacob O. Wobbrock, Meredith Ringel Morris, and Andrew D. Wilson. 2009. User-Defined Gestures for Surface Computing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Boston, MA, USA) *(CHI '09)*. Association for Computing Machinery, New York, NY, USA, 1083–1092. https://doi.org/10.1145/1518701.1518866

[56] Robert Xiao, Chris Harrison, and Scott E. Hudson. 2013. WorldKit: Rapid and Easy Creation of Ad-Hoc Interactive Applications on Everyday Surfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Paris, France) *(CHI '13)*. Association for Computing Machinery, New York, NY, USA, 879–888. https://doi.org/10.1145/2470654.2466113

[57] Robert Xiao, Scott Hudson, and Chris Harrison. 2016. DIRECT: Making Touch Tracking on Ordinary Surfaces Practical with Hybrid Depth-Infrared Sensing. In *Proceedings of the 2016 ACM International Conference on Interactive Surfaces and Spaces* (Niagara Falls, Ontario, Canada) *(ISS '16)*. Association for Computing Machinery, New York, NY, USA, 85–94. https://doi.org/10.1145/2992154.2992173

[58] Robert Xiao, Scott Hudson, and Chris Harrison. 2017. Supporting Responsive Cohabitation Between Virtual Interfaces and Physical Objects on Everyday Surfaces. *Proc. ACM Hum.-Comput. Interact.* 1, EICS, Article 12 (jun 2017), 17 pages. https://doi.org/10.1145/3095814

[59] Robert Xiao, Julia Schwarz, Nick Throm, Andrew D. Wilson, and Hrvoje Benko. 2018. MRTouch: Adding Touch Input to Head-Mounted Mixed Reality. *IEEE Transactions on Visualization and Computer Graphics* 24, 4 (2018), 1653–1660. https://doi.org/10.1109/TVCG.2018.2794222

[60] Yang Zhang, Gierad Laput, and Chris Harrison. 2017. Electrick: Low-Cost Touch Sensing Using Electric Field Tomography. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/3025453.3025842

[61] Yang Zhang, Chouchang (Jack) Yang, Scott E. Hudson, Chris Harrison, and Alanson Sample. 2018. Wall++: Room-Scale Interactive and Context-Aware Sensing. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–15. https://doi.org/10.1145/3173574.3173847

[62] Qian Zhou, Sarah Sykes, Sidney Fels, and Kenrick Kin. 2020. Gripmarks: Using Hand Grips to Transform In-Hand Objects into Mixed Reality Input. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–11. https://doi.org/10.1145/3313831.3376313