Enabling Tangible Interaction on Non-touch Displays with Optical Mouse Sensor and Visible Light Communication

Yihui Yan*†‡
School of Information Science and Technology
ShanghaiTech University
Shanghai, China
yanyh@shanghaitech.edu.cn

Zezhe Huang*
School of Information Science and Technology
ShanghaiTech University
Shanghai, China
huangzzh@shanghaitech.edu.cn

Feiyang Xudu*
School of Information Science and Technology
ShanghaiTech University
Shanghai, China
xudfy@shanghaitech.edu.cn

Zhice Yang§
School of Information Science and Technology
ShanghaiTech University
Shanghai, China
yangzhc@shanghaitech.edu.cn

Figure 1: Centaur Consists of an Ordinary Screen Panel ❶ and MouseToken Tangibles ❷❸. MouseTokens are commercial optical mice ❸ or wireless tokens ❷ with optical mouse sensors inside (the decomposition is shown in ❹). The tokens' physical attributes are trackable once they are put on the panel. Users can dock, move, and rotate them to manipulate the digital content.

ABSTRACT

This paper presents Centaur, an input system that enables tangible interaction on displays, *e.g.*, untouchable computer monitors. Centaur's tangibles are built from low-cost optical mouse sensors, or can alternatively be emulated by commercial optical mice already available. They are trackable when put on the display, rendering a real-time and high-precision tangible interface. Even for ordinary personal computers, enabling Centaur requires no new hardware and installation burden. Centaur's cost-effectiveness and wide availability open up new opportunities for tangible user interface (TUI) users and practitioners. Centaur's key innovation lies in its tracking method. It embeds high-frequency light signals into different portions of the display content as location beacons. When the tangibles are put on the screen, they are able to sense the light signals with their optical mouse sensors, and thus determine the locations accordingly. We develop four applications to showcase the potential usage of Centaur.

CCS CONCEPTS

• **Human-centered computing** → **Interaction devices**; • **Hardware** → **Tactile and hand-based interfaces**; *Sensor devices and platforms.*

KEYWORDS

Tangible, Tabletop, Optical Mouse, Visible Light Communication

ACM Reference Format:
Yihui Yan, Zezhe Huang, Feiyang Xudu, and Zhice Yang. 2022. Enabling Tangible Interaction on Non-touch Displays with Optical Mouse Sensor

*Authors contributed equally to this research.
†Also with Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Science.
‡Also with University of Chinese Academy of Sciences.
§Corresponding author

CHI '22, April 29-May 5, 2022, New Orleans, LA, USA

ACM ISBN 978-1-4503-9157-3/22/04.
https://doi.org/10.1145/3491102.3517666

and Visible Light Communication. In *CHI Conference on Human Factors in Computing Systems (CHI '22), April 29-May 5, 2022, New Orleans, LA, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3491102.3517666

1 INTRODUCTION

Tangible User Interfaces (TUIs) have been discussed at length over the past decades [35]. They allow users to interact with digital objects through manipulating physical objects, *i.e.*, the tangibles. TUIs have benefits over mouse or tactile interactions in certain tasks including bimanual interaction [58, 59], eye-free control [66], 3D-manipulation [18], *etc.*

In essence, TUI systems must be able to precisely track the physical attributes (*e.g.*, position, orientation, *etc.*) of the tangibles in order to mirror the input actions properly in the digital content. This is the *tangible tracking* problem, which has attracted many discussions. One approach is to utilize dedicated positioning systems, such as radio sensors [61], modulated light projector [26], and depth cameras [18]. All of them, however, bring in overheads in both device costs and installation setup, limiting their applicability to very specific scenarios, such as education [10] and gaming [7].

Tangible in combination with digital display, *e.g.*, tabletop, is another popular TUI form [60]. The screen graphically renders dynamic virtual objects while the tangibles tethered to the screen panel serve for the physical presentation (see, for example, reacTable [37]). Meanwhile, the tracking problem can be relatively smoothly handled by taking advantage of the built-in capabilities of interactive screens. Early work used a diffused illumination tabletop to identify and locate the tangibles on the panel [37, 67]. Recent work is mainly based on capacitive touch panels. They are leveraged to sense and track tangibles by capacitive footprint [45, 50]. While a widely-discussed solution, tracking on capacitive panels has grounding issues where tangibles lose tracking when not touched by the user [62, 63]. This results in potential interaction friction and design complexities.

This paper introduces Centaur, a novel TUI system that tries to get rid of the limitations of existing tangible tracking methods. It enables TUIs for display systems whilst keeping the device costs and setup burden to a minimum.

As shown in Figure 1, the Centaur input system uses a horizontal screen panel as the interactive surface and multiple MouseTokens as the tangibles. The screen can either be touchable or non-touchable. The user can simply rotate and re-use the available desktop display as the interactive surface through an adjustable monitor stand [6]. There are two types of MouseTokens. The smaller one is a low-cost embedded system with an optical mouse sensor inside. It mimics an optical mouse and can be flexibly customized. The larger ones are just ordinary optical mice. They allow Centaur to be deployed even without purchasing or crafting dedicated MouseTokens. The larger form factor might also be preferred by young children, for instance, whose fine motor skills are not yet fully developed. When MouseTokens are put on the screen, Centaur is able to track their locations and orientations in real-time. Based on the interface, users can interact with the virtual content shown on the screen by manipulating the tokens.

The core mechanism that Centaur leverages to solve the tangible tracking problem is visible light communication (VLC) [51]. We notice that an optical mouse is actually an imaging device similar to a camera, hence MouseTokens are able to capture the light signals from the screen beneath. The display content of Centaur's screen is divided into tiny regions, each of which is assigned with a unique location ID. The screen continuously broadcasts IDs as beacons for the tokens via VLC signals. The tokens are positioned by decoding the captured VLC signals to obtain the location ID (see Figure 2). To avoid flickers in the broadcast, Centaur uses a high-refresh-rate screen, *e.g.*, 240 Hz, to display VLC signals in the high-frequency domain. Centaur further makes use of the fine-grained spatial features of the sub-pixel structure of the screen panel to infer the token's orientation[1].

Despite the hardware requirement, the above tracking mechanism can also be implemented in software with low system overheads. Centaur accesses the raw data of optical mouse sensors through the lightweight serial interface over Bluetooth or USB. The VLC signals are preloaded as special static textures in GPU, which can be rendered as an overlay very efficiently. The source code and examples are available at [14].

Centaur contributes to the area of interactive devices in the following two aspects.

First, we introduce a precise and unobtrusive tracking method for on-screen objects. Visual-pattern-based tracking is popular in the HCI community for tracking devices on screens [68, 71]. A common practice is to display a visible visual pattern to allow the device to determine the location through the light/image sensors. The pattern shrinks and moves along with the device to minimize visual interference. We think this approach is a good compromise but not an ideal solution. The visual pattern can still affect the aesthetics of the original display content. When the device is out of tracking, the pattern must enlarge and flash for re-tracking [71], making it obtrusive and unsuitable for people with photosensitive epilepsy. With VLC and high-refresh-rate display techniques, Centaur hides the visual patterns into the time domain, making them much less obtrusive. We believe it renders an alternative and interesting approach for on-screen interactive designs.

Second, we design and implement a cost-effective TUI system. Its hardware requirements might already have been met in many personal computers, meaning that it offers a potentially affordable way to deploy and study TUIs in practice at scale. Behind this ambition, our key design choice is to adopt an optical mouse sensor as the signal sensing device to make use of the VLC tracking method. The mouse sensor has two niche features for this scenario. As a well-established technology, not only has it been used in nearly every commercial mouse, but it also reduces the manufacturing costs of customized MouseTokens (the prototyping costs of a token is about $10). Additionally, compared with low-cost cameras [32] and raw optical sensors [26], the mouse sensor is able to report accurate relative displacements frequently. This built-in feature is complementary to the VLC absolute tracking and improves the interaction smoothness (this point will soon be clear in §4.3 and the Hybrid Gaming Board application in §7).

[1]We note that the way that Centaur uses mouse sensors is completely different from the traditional mouse interface. In the latter, the mouse sensor is only used to measure the relative displacement. The digital object and the mouse input are indirectly associated through GUI and human cognitive mechanisms [40]. Centaur uses a mouse sensor as a camera to directly associate the sensor's physical locations and orientations to the TUI.

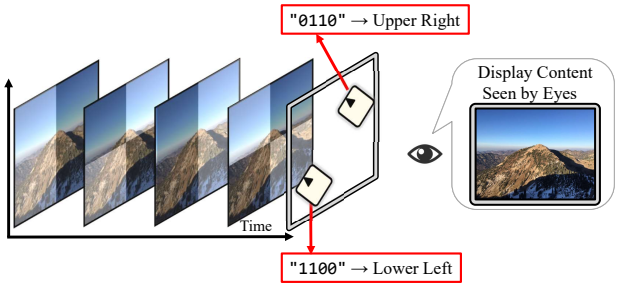


Figure 2: Idea of Tracking MouseToken Location. The screen varies the light intensity of its display frames differently at different locations. The optical mouse sensor inside the token captures the intensity changes for positioning. The intensity variation of frames is of high frequency, so it will not affect the normal viewing experience.

2 RELATED WORK

This section reviews tangible tracking designs, special usages of optical mouse sensors, and screen-to-camera communication.

Tracking Tangibles. Tangible User Interfaces (TUIs) allow users to interact with virtual objects by manipulating the binding physical objects. Studies have shown that the benefits of TUIs over other interfaces lie in several aspects including spatial multiplexing [25, 39], embodiment [31], controlling degrees-of-freedom (DoF) [18, 36], *etc.* Interface designers have proposed various TUIs to facilitate education [76], entertainment [61], productivity [36], and collaboration [20]. Tracking tangible objects is the key technical challenge in TUI systems. In the following, we review the closely related 2D tracking methods.

Tangibles on a diffusion illumination desk can be differentiated through optical ways such as IR cameras [27, 28, 37, 66]. Based on this technique, Hartmann *et. al.* [28] discussed the potential interactive benefits of using traditional mice and tabletops together. Unlike us, they did not use mice or mouse sensors for tangible interactions.

Early instruments are bulky and need sophisticated installation. Diffusion illumination tabletops are rarely seen nowadays, and the focus is turned to touch screens. Many novel TUIs are proposed based on the tracking capability of touch panels [23, 27, 38, 45, 73, 74]. However, the touch panels are designed to track the human body, not devices. The workaround is to conduct the human body or the ground to the tangible tags to make them visible to capacitive sensing units. A subtle problem occurs when the tangibles are not grounded/being touched, they become invisible and lose track. Voelker *et. al.* [62, 63] proposed to generate mutual capacitance difference on the tangible to make it temporally visible, but their method cannot completely eliminate the interaction frictions. Tracking errors can still occur, for example, when the tangible is not touched by the user but is occasionally moved by a neighboring tangible. Another practical limitation of capacitive tracking is scalability. The touch screen differentiates tangibles through their capacitive footprints, but since the number of touch points is usually limited, the number of supported concurrent tangibles is restricted too [73]. Centaur relies on a VLC tracking mechanism compatible with both touch and non-touch screens, and thus is free of the above limitations.

RFID is another commonly-used method for object tracking. Existing work attaches RFID tags to tangible objects and uses radio sensing devices to localize them [33, 49, 61]. Magnetic signals can be used similarly as well [19]. Medusa [16] uses proximity sensor arrays to track not only the on-surface objects but also the surrounding users.

Centaur is closely related to the tracking approaches exploiting visible light signals. A popular choice in TUI research is to use the Digital Light Processing (DLP) projector to broadcast location beacons in high-frequency projected light signals and use the light sensors on the tangibles to receive and track the location [26, 30, 42]. Centaur is similar to this approach in the way that it distributes and hides location beacons. However, Centaur does not require the purchasing and installing of the DLP projector. Instead, its locations are broadcast through an ordinary screen.

Another closely related approach is to use light sensors [48, 56, 71] or cameras [68] to sense the display patterns on the screen for tracking. In particular, WATouCH [71] is a recent tangible design that uses the smartwatch as a proxy to simulate touch actions on non-touch screens. Its location tracking is done by making use of the heart rate sensor (a special kind of light sensor) of the smartwatch to sense the structured pattern on the screen. Instead of showing visible patterns for tracking, Centaur emphasizes invisibility and keeps the visual obtrusiveness minimal. Additionally, it uses mice as receivers, which are more prevalent and reasonable in general computer interaction scenarios.

Using optical mouse sensors to track the absolute location and orientation of tangibles has been explored by Dothraki [53]. Its tracking is aided by the mouse pad with printed structured patterns. Similar to Centaur, it uses the mouse sensor to capture the patterns and makes use of the pattern differences at different positions for positioning. However, unlike Centaur, which also uses temporal patterns, Dothraki cannot work with arbitrary orientations.

Ad-Hoc Usage of Optical Mouse Sensor. Since mice are able to track relative displacement, they have been used as motion tracking tools [46, 54]. These approaches rely on other hardware to correct the orientation and determine the initial location, whereas Centaur only requires optical mouse sensors. WristLens [72] is another example. It equips a mouse sensor on the wrist band to track arm movement as an additional input interface. It only tracks the relative movement like normal mice. The optical mouse sensor is known to be capable of optical sensing and imaging [3, 9, 55], but none of them has explored this feature to track tangibles. The mouse sensor and display system are explored in [21] to determine the latency of the display system, which is different from our goal.

Screen-to-camera Communication. Screen-to-camera communication is an emerging way to allow devices to receive information from screens. It is a special kind of Visible Light Communication (VLC). Its basic approach is to display QR-code-like image streams on the screen and use a camera to capture and decode. Recent research improved it in various aspects [34, 51]. As the screen is primarily used for illustration, it is important to reduce the visual obtrusiveness of the coded area [15, 65, 75]. The approach is to hide the code in the high-frequency temporal and/or spatial domains to fool the human eye. Centaur's design is inspired by existing VLC

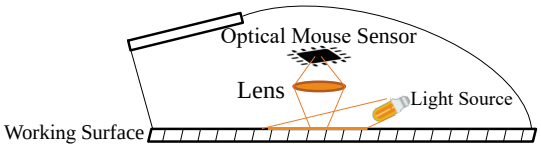


Figure 3: Architecture of an Optical Mouse. The optical mouse uses its sensor to detect movement. The sensor works like a camera and continuously takes photos of the working surface.

work, but as it uses an optical mouse sensor rather than a camera as the receiver, it contains unique design challenges. Performing VLC with an optical mouse was explored by [22], but they use it to receive VLC signals of an LED panel several centimeters away, which did not imply meaningful applications. On the contrary, Centaur uses a mouse to receive signals from the LCD panel for tracking and interaction.

Applying VLC for positioning has been studied by existing work but normally under the context of open space [41, 69] rather than screen surface. Centaur uses VLC to determine the location and makes use of the sub-pixel structure of the screen panel to determine the orientation, which is novel and unique.

3 OVERVIEW

This section first introduces the background of the optical mouse sensor, followed by the idea of Centaur, which makes extensive use of the light-sensing ability of the optical mouse sensor.

3.1 Optical Mouse is a Camera

Computer mice were invented half a century ago, since then, they have been a major way for people to control a graphic user interface (GUI), *e.g.*, to move the cursor. A mouse works by continuously tracking its relative movement on the working surface. The GUI system scales the movement values and updates the display content, *e.g.*, the cursor location.

Optical mice (Figure 3) rely on the optical imaging system to track the movement. A light source is used to illuminate the working surface so that the surface can be monitored by the optical mouse sensor. The optical mouse sensor is a type of image sensor with a high sampling rate. It continuously samples the image of the working surface. When movement happens, the relative displacement can be inferred through the difference between consecutive images. The movement detection process is finished in the computing unit of the sensor. The final displacement values, *i.e.*, the motion vectors, are reported to the host.

The optical mouse sensor provides a register read/write interface to access its raw sensing data beyond the displacement values. This is a typical feature reserved for development purposes [57]. This interface provides statistical descriptions of the current imaging area, such as the average light intensity, the maximum light intensity, *etc.*, and even the raw image captured by the sensor. According to our experience of using different models of mouse sensors (§6), they have the following properties in common (to avoid possible confusion, we use "pixel^M" to indicate the pixel of the image of the mouse sensor, and normal "pixel" to indicate the pixel of the screen panel):

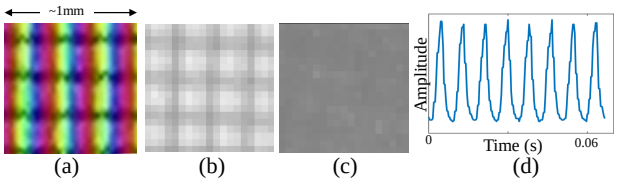


Figure 4: Imaging the Screen Panel with an Optical Mouse. (a) is a photo taken by a macro camera, showing the microstructure of the screen panel. (b) and (c) are raw images obtained from the optical mouse sensor, when the mouse is put on the screen and the screen shows white and black, respectively. (d) plots values of the first light sensing unit, *i.e.*, the first pixel^M of (b)(c), when the screen is periodically showing black and white.

- **Low Resolution**. The sensor has a square array of light-sensing units, with each line consisting of tens of units. The output of the sensor can be arranged into a grayscale image, with each pixel^M value being equal to the light intensity captured by the corresponding light sensing unit (see examples in Figure 4 (b)(c), which are captured by an optical mouse sensor having 19×19 sensing units).
- **Narrow Coverage**. The sensor is only a few millimeters above the surface, hence its imaging area is very narrow, typically around 1×1 mm^2 [17].
- **Light Sensitivity**. The sensing unit is able to capture moderate light intensity variations. We have more a detailed discussion in §4.1.1.
- **Temporal-Spatial Uncertainty**. Registers of the optical mouse sensor can be accessed at a rate of more than 1 kHz [57]. However, as each pixel^M of the raw image takes one register read action, the reading rate of a full sensor image is around 3 to 4 Hz. Further, the raw image pixel^{M}s must be accessed in a linked manner, *i.e.*, from left to right and from top to down. Indexed or random access is not provided. As a result, if we want to increase the resolution in the time domain, we must sacrifice the resolution in the spatial domain, *e.g.*, one can either read the first pixel^M at 1 kHz or read the full image of 19×19 pixel^{M}s at 2.5 Hz.

3.2 Idea of Using Optical Mouse Sensor for Tangible Tracking

TUI systems must solve the tangible tracking problem. The idea of Centaur is based on the observation that when a mouse is put on the screen, its optical sensor is able to take "photos" of the screen panel (see Figure 4). This observation has two important implications.

- First, the optical mouse sensor is able to capture the fine structure of the screen panel. This is because the sensor is very close to the panel surface and is focused to the surface by the lens. When the optical mouse is put on the screen panel showing the white color, the captured image[2] is shown in Figure 4(b), which contains grid-like dark gaps between bright areas. These gaps are caused

[2]Grayscale values of the images captured by the optical mouse are added by 128 for better illustration.

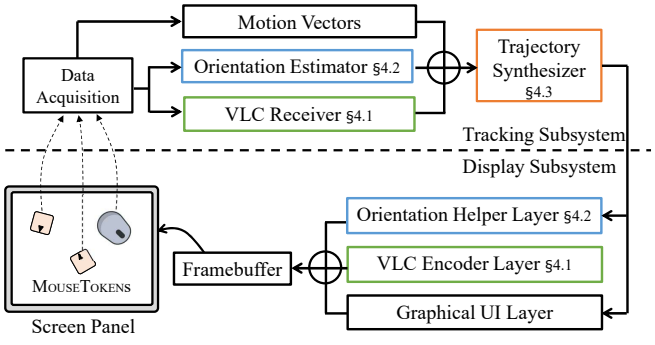


Figure 5: Centaur System Architecture.

by the sub-pixel structure of the screen panel, and match with Figure 4(a), a photo of the same area of the panel.

- Second, the optical mouse sensor is sensitive enough to the light intensity changes of the screen panel, and it can sample the changes at a relatively high rate. Figure 4(c) is taken when the screen is showing black. Note Figure 4(b) is brighter than Figure 4(c), this is because, in order to display the white color, all RGB pixels of the panel are lit, resulting in higher light intensity. If we only request the value of the first pixel^M of the mouse sensor, the sampling rate can reach more than 2 kHz, which is enough to capture the light intensity changes when the screen varies its intensity, as shown in Figure 4(d).

The first implication suggests that a naive way is to fingerprint the screen panel with the mouse sensor and then position it according to the fingerprint map. However, as the resolution of the optical mouse sensor is very low (*e.g.*, 19×19 pixel^{M}s for our low-end sensor) and its coverage is very narrow (1 mm^2 only covers 3×3 pixels of a 23.5 inch 1080p screen), it is hard to identify accurate locations merely based on the spatial information. According to the second implication, VLC designs from the screen-to-camera channel [44, 47, 65] can be applied to this situation. As in these designs, the intensity of display frames is encoded to vary in the time domain and the receiver decodes information by sampling the light intensity changes. This is also possible by using the optical mouse sensor.

Based on the above analysis, the intuition of Centaur is to use an optical mouse sensor as the VLC receiver to build trackable tangibles on the screen surface. Since the coverage of the optical mouse sensor is narrow, if different areas of the screen show different VLC signals, the location of the mouse sensor can be uniquely determined according to the received VLC signals. Figure 2 illustrates this idea. We term this type of tangibles as MouseToken. It can either be commercial optical mice or customized tokens (see Figure 1 ❸ and ❷, respectively). They both use the same tracking method based on optical mouse sensors.

4 DESIGN

Figure 5 shows the system architecture of Centaur. It consists of the tracking and display subsystems. They work closely together to support tangible interaction. The screen renders the graphical content while transmitting VLC signals for tracking MouseTokens. When the tokens are put on the screen, they continuously capture the VLC signals and transmit them to the host system. Then, the host processes the data to get the token locations and reflects the token information in the graphical content. Meanwhile, in order to determine the orientation of the token, Centaur incorporates another two modules to aid the measurement. Finally, as receiving a complete VLC location beacon takes time and results in tracking delay, Centaur synthesizes the real-time relative movement measurements of the mouse sensor with the VLC tracking to provide a smooth interaction experience.

4.1 Tracking MouseToken Location

Centaur uses the screen to broadcast VLC location beacons. The challenge is how to reliably receive VLC signals with the optical mouse sensor while avoiding flicking of the VLC signals.

4.1.1 VLC Design. Centaur's VLC is performed between its screen and MouseTokens. Figure 6 shows the VLC frame structure, which is inspired by Inframe++ [65] and optimized for optical mouse sensors. For simplicity, we use grayscale to describe the VLC frame design. Let $I_{i,j}$ denote the value of the pixel at i-th line and j-th column of the screen panel. $I_{i,j}(t)$ is the value of the $I_{i,j}$ at time t. $I_{i,j}(t)$ is determined by the display content. The VLC encoding scheme generates $\delta_{i,j}(t)$ to modify the display content to

$$I^{\text{vlc}}_{i,j}(t) = I_{i,j}(t) + k \cdot \delta_{i,j}(t). \tag{1}$$

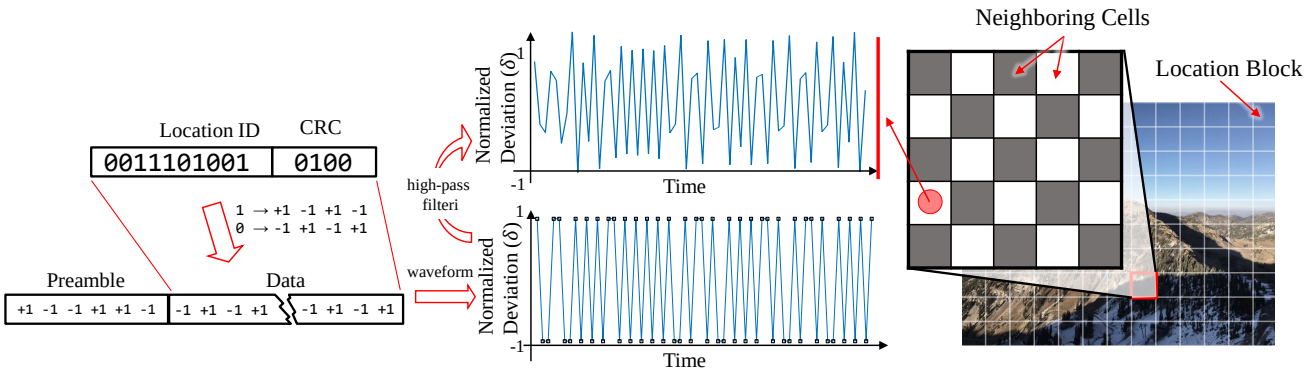
$\delta_{i,j}(t)$ denotes the VLC signal. It uses spatial structures and temporal variations to convey information, and at the same time to fool people's eyes to ensure an unobtrusive viewing experience. k is the scaling factor controlling the intensity of the VLC signal.

Spatial Structure. $\delta_{i,j}(t)$ has a hierarchical spatial architecture as shown in Figure 6(c). The display content is divided into location blocks. Temporal VLC signals within one block transmit the location ID of the block. One location block consists of multiple cells. Cells within the same location block transmit the same location ID, but neighboring cells vary conversely, *i.e.*, $\delta_{i,j}(t)$ equals to $-\delta_{m,l}(t)$, if (i, j) and (m, l) are in the same location block and neighboring cells. Each cell consists of several pixels, which vary with the same temporal pattern, *i.e.*, $\delta_{i,j}(t)$ equals to $\delta_{m,l}(t)$ if (i, j) and (m, l) belong to the same cell.

Temporal Variation. $\delta_{i,j}(t)$ varies in the time domain with designated patterns to transmit VLC location beacons. As shown in Figure 6(a), The beacon packet contains two fields. The header field is a unique symbol sequence used to synchronize the start of the packet; the data field contains the ID of the location block. l-bit data field is able to represent 2^l different location blocks, providing positioning resolution in the unit of $\frac{\texttt{Area}_{\texttt{Screen}}}{2^l}$, where $\texttt{Area}_{\texttt{Screen}}$ is the area of the screen panel. In practice, the length of the data field is chosen to balance the positioning resolution and latency. In order to remove the DC in VLC signals, bits are mapped to symbols with a line code scheme similar to the Manchester code:

$$\text{Bit } 1 \rightarrow +1, -1, +1, -1; \qquad \text{Bit } 0 \rightarrow -1, +1, -1, +1$$

Note the scheme deliberately reduces the data rate of Manchester code by half to reduce the low-frequency components in VLC signals to reduce the visual flicking. For the same reason, as shown in Figure 6(b), we further pass the symbol waveform to a high-pass



(a) Location Beacon Example **(b) Temporal Variation** **(c) Spatial Structure**

Figure 6: Temporal and Spatial VLC Frames. (a) A location beacon packet is mapped to the intensity variation in (b), which is used to manipulate the corresponding content of the display frame in (c).

filter to obtain $\delta_{i,j}(t)$ before the VLC transmission. An example of $\delta_{i,j}(t)$ is shown in the upper part of Figure 6(b).

VLC Receiver. A MouseToken relies on its optical mouse sensor to receive VLC signals. Due to the temporal-spatial uncertainty (§3.1), the full-image sampling rate is low (3 to 4 fps), hence it is not possible to decode the VLC information from the low-rate full images. Our solution is based on the fact that only one light sensing unit is enough to capture the temporal VLC signals. Recall the second implication in §3.2, we use the first sensing unit of the sensor as the VLC receiver, corresponding to the top-left pixel^M of the captured image. Since the sampling frequency of the single sensing unit can reach up to more than 2 kHz, we are able to capture and decode the temporal VLC signals.

4.1.2 Combating Spatial Interference. In practice, the VLC receiver faces two kinds of spatial interference. The first is the neighboring interference. Due to the spatial structure, neighboring blocks and cells emit different VLC signals. In some locations, especially at the boundaries of cells and blocks, multiple different VLC signals will be received by the optical mouse sensor and cause interference. However, the impact of the neighboring interference is not severe in the single sensing unit design. The reason is that the optical mouse is a lens-based imaging system. Because of the focus effect of the lens, the light-sensing unit can only receive light signals from a very narrow area having a similar incident angle [43]. Therefore, it is unlikely that a single sensing unit receives VLC signals from multiple neighboring blocks at the same time.

However, the single sensing unit suffers from the second kind of interference called *gap blockage*. From Figure 4(a)(b), we know that the sub-pixel structure causes unlighted areas on the surface of the panel. The gap blockage problem refers to the situation where the field of view of the single sensing unit falls into the unlighted areas. In such a situation, the received VLC signals will be partially or totally blocked, resulting in low signal strength and decoding failure. In practice, the occurrence of gap blockage is not rare (about 30% cases), as the unlighted structure is everywhere on the screen panel.

We solve the problem by making full use of the sensing abilities of the mouse sensor. In addition to the raw image values, the sensor also provides statistics about the current sensing area (§3.1). Thus, we use the average intensity register to assist the receiving. Unlike the single sensing unit value, the average intensity is calculated from the whole sensing area, which covers multiple pixels of the screen and will not be affected by the gaps. However, the average intensity cannot be used alone, because it is more fragile to the neighboring interference. Therefore, MouseToken employs a hybrid sensing scheme. It simultaneously requests both the single sensing unit register and the average intensity register to capture the VLC signals. Each register's sampling rate occupies half of the sensor's data bandwidth, *i.e.*, half of 2 kHz, which is still enough to sample the VLC signals. The host decodes the two streams separately, then selects and outputs the one passing the CRC check.

After correct decoding, the data of the VLC beacon packet is obtained, which is the location ID of the VLC block right below the mouse sensor. Therefore, the absolute physical location of the MouseToken on the screen is determined.

4.2 Determining MouseToken Orientation

Rotation is a common type of tangible interaction. Existing TUIs usually use multiple positioning sensors to turn location measurement to angle measurement [66] or rely on Inertial Measurement Unit (IMU) sensors [71]. These approaches can be applied to customized MouseTokens but not commercial mice. In Centaur, the mouse sensor utilized for positioning is also used for orientation measurement. The novel mechanism is shown in Figure 7. Centaur makes use of the unlighted structures of the screen panel, *i.e.*, the gaps (first mentioned in §4.1.2), to determine the orientation of the MouseToken.

These unlighted structures are physical grids used to split adjacent pixels. The grid lines are parallel to the X_s or the Y_s axis of the screen (see the coordinate system in Figure 9). Therefore, the direction of the gap lines[3] in the captured image of the mouse sensor indicates the bearing angle of the MouseToken and the screen, *i.e.*, θ in Figure 9.

[3] Obtained by Hough transform [24].

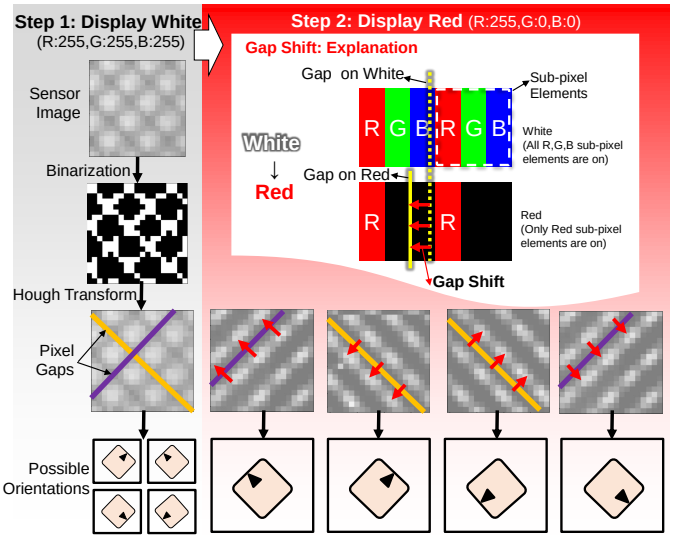


Figure 7: Determining MouseToken Orientation. Step 1: The dark gaps on the screen panel are used to find possible orientations of the MouseToken. Step 2: Since the sub-pixel elements are lit by the display content, the dark gaps shift according to the displayed color. The true orientation can be determined by considering the gap shift direction in the captured images.

However, the simple calculation has four ambiguities. Centaur finds the correct one by considering the dynamic feature of the sub-pixel structures. As shown in Step 2 of Figure 7, when the display color is white, the three RGB sub-pixel elements are lit and the center of the gap is close to the boundary of the pixel. When the display color is changed, *e.g.*, to red. The green and blue elements are turned off. As a result, the corresponding parts of the captured image become dark. Since the sub-pixel RGB elements are arranged from left to right ($-X_s$ to $+X_s$) on the panel [4], turning off green and blue elements increases the width of the dark gaps aligned along the Y_s axis and shifts the center of the gaps towards the $-X_s$ direction. According to this observation, the orientation ambiguities can be eliminated.

Centaur determines MouseTokens' orientations after obtaining their locations from VLC location beacons. Specifically, the screen displays two solid color blocks sequentially to help the orientation estimation. The color blocks are only displayed right below the MouseTokens, so they are covered by the token and will not be visible to the user. The state machine of the orientation estimation process is shown in Figure 8. As a directional token, we label a black arrow on the customized MouseToken to indicate its front direction (see Figure 1 ❷❹). The tail of the commercial mouse is defined as the front direction for the same purpose.

[4]For panels with different sub-pixel structures, the algorithm can be adjusted accordingly. When the pixel density exceeds the mouse sensor's resolution, the sub-pixel structure cannot be revealed, but one can still show dedicated gap textures to generate the gap shift.

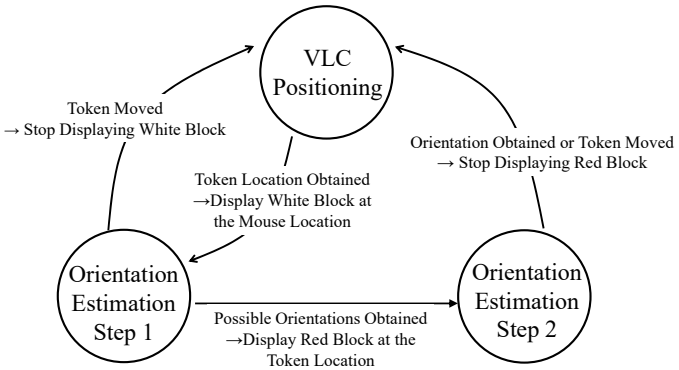


Figure 8: State Machine for Determining Mouse Orientation. The first line on the transition arrow is the transition condition. The arrowed line on the transition arrow is the transition action.

4.3 Synthesized Location Tracking

VLC positioning alone is not sufficient for smooth location tracking. This is because obtaining a location ID needs a complete VLC packet, which takes multiple display frames to transmit. However, the token might have been moved before the transfer is complete. Then, no location update is possible until the token is stopped again. This limitation causes interaction lags.

Our solution is to reuse the mouse sensor's built-in relative tracking to interpolate the trajectory between two absolute VLC locations. The displacement measured by the mouse sensor is represented in its local coordinate system, which cannot be directly mapped to the absolute location on the screen. For example, when the token is moved towards its front ($-Y_m$-axis in Figure 9), its sensor always reports negative y values in the motion vectors. However, the real movement direction could be arbitrary and depends on how the token is placed on the screen, *i.e.*, the orientation of the token with respect to the screen.

Therefore, in order to correctly make use of the motion vectors, Centaur jointly considers the sensor's relative displacement measurement and its absolute orientation. The sensor quantifies its movement into a series of incremental displacement values $[x_m, y_m]^T$ in its local coordinate system, *i.e.*, the motion vectors. The accumulative location $[x, y]^T$, *i.e.*, the latest location, in the token's coordinate system is obtained by integration:

$$[x, y]^T \leftarrow [x, y]^T + [x_m, y_m]^T.$$

The token's location $[x_s, y_s]^T$ in the screen coordinate system can be obtained by

$$[x_s, y_s]^T \leftarrow [S_x, 1; 1, S_y] \cdot M_A \cdot [x, y]^T + [O_x, O_y]^T,$$

where M_A is the coordinate transformation matrix derived from the token orientation. $[O_x, O_y]^T$ is the latest VLC location. S_x and S_y are constant scaling factors used to convert motion vectors to physical displacement. Their values can be obtained through a one-time offline calibration[5]. We note that, due to integration and

[5]We perform this procedure by asking the user to move the token from the left edge of the screen to the right edge five times. The scaling factor is calculated according to the virtual displacement value and the width of the screen.

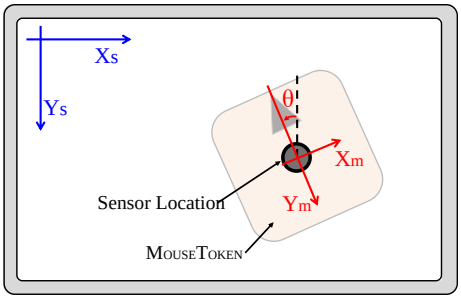


Figure 9: Coordinate Systems of MouseToken and Screen. The direction of the negative Y axis of the token, explicitly shown by the black arrow on it, indicates the "front" of the token.

rotation, the above position tracking method is subject to accumulative errors, hence $[O_x, O_y]^T$ is always updated to the latest VLC location.

4.3.1 Consideration of Tangible Binding. MouseTokens and digital objects support both fixed mappings, *i.e.*, a token is bound to a unique digital object, and dynamic binding, *i.e.*, a token is bound to the digital object having the closest physical distance. In some cases, especially when using commercial mice as tokens, the physical shell of the token might cause occlusion [66]. This issue can be improved by incorporating occlusion-aware interfaces [64] in the display content. Currently, we use the distance to the front area of the token to judge the binding.

5 IMPLEMENTATION

This section describes the implementation of the prototype system.

Host System. We use a commercial desktop computer with Intel Core i5-7500 CPU, 8 GB RAM, and AMD RX550 GPU as the host platform. It is connected to an AOC AG251FZ 240 Hz gaming monitor [2]. This is the screen used for interaction.

Tangibles. The commercial mice we used as MouseToken are Dell MS111, Logitech M100, and Logitech UAG96B. Our customized tokens are designed by referring to MS111. The decomposition is shown in Figure 1 ❹. A token consists of (top to down) a shielding case, a 3.7v 800 mAh Li-ion chargeable battery, a control board with STM32F103C8T6 processor, a CSR BC417143B Bluetooth module, a mouse sensor, and a lens. The mouse sensor (Avago S5085) and lens are identical to that of MS111. We design several PCB boards to host these chips and pack them into a $4.6 \times 4.6 \times 3.7$ cm^3 cubic.

Data Acquisition. The host system polls sensing data from the connected tokens through the register I/O operations of the mouse sensor. The data paths are slightly different for commercial USB mice and customized wireless tokens. For commercial mice, the motion vectors are reported via the USB interrupt transfer and the register values are accessed via the USB control transfer. By default, the mouse device is automatically abstracted as an HID (Human Interface Device) and does not provide access to the USB control transfer [4]. We re-attached the mouse driver as a general USB device to solve this issue. For customized tokens, the control board is connected to the host via Bluetooth and to the mouse sensor via Serial Peripheral Interface (SPI). It does not implement the Bluetooth HID stack and acts as a raw data and control proxy agent between the host and the sensor.

VLC Rendering. In the prototype, the VLC location beacon packet consists of a preamble (+1, −1, −1, +1, +1, −1), and a 14-bit data field with a 10-bit location ID (1024 blocks) and 4-bit CRC. Hence one location beacon consumes 6+14×4=62 frames in total. As the refresh rate of the screen is 240 Hz, the duration of a VLC packet is 0.26 s. The corresponding positioning resolution of 10-bit location IDs is 12×12 mm^2 on a 23.8 inch screen [6]. Finer resolution can be achieved by using more IDs with longer VLC packets. OpenGL [8] (version 3.3) is used as the render backend. The implementation is based on Windows 10 for better GPU driver support. We use a module to overlay VLC frames on top of the original display content. Note that for a specific display pixel, the time-domain variations of its VLC signals are fixed and cyclic, hence the VLC frames can be generated as texture frames and loaded into the GPU in advance to avoid run-time calculation. As a result, the VLC rendering is very lightweight and brings negligible processing overhead.

VLC Positioning. As the register read rate of the mouse sensor is not stable, the received VLC samples are first evenly interpolated in the time domain. Then, correlation is used to detect the preamble. After that, a band-pass filter is used to remove the noise. However, since the correlation is still not accurate enough, an offset of up to five samples is traversed to find the best one for decoding. The bit stream is derived by thresholding on the samples and only the error-free location IDs are reported. To determine the orientation, the VLC render module draws a helper layer showing solid color blocks for each token and then queries full raw images from their mouse sensors for estimating the orientation.

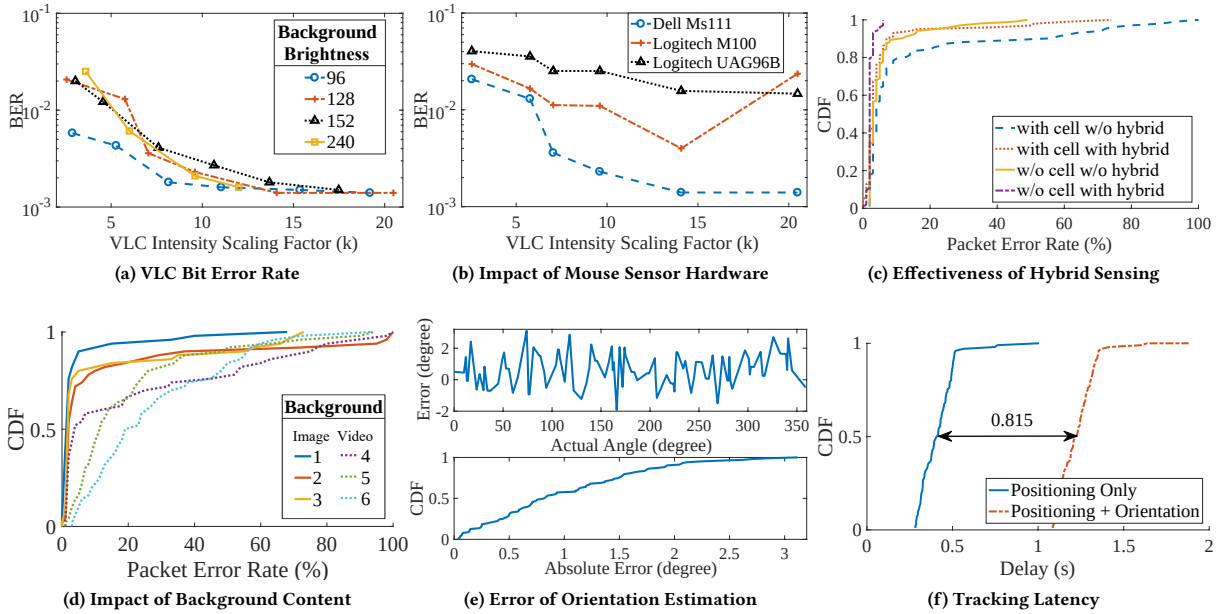
6 EVALUATIONN

This section characterizes the performance of Centaur. It covers functional components, impacting factors, and tracking performance. We use commercial mice in the evaluation to understand the performance in the most restricted situation. The performance of customized tokens is identical to Dell MS111.

VLC Error Rate. The goal of this experiment is to evaluate the VLC performance under different configurations (see equation 1): the scaling factor k governing the intensity of VLC signals, and the brightness of the background signal $I_{i,j}(t)$. In the experiment, Dell MS111 is used as the receiver and is placed in a fixed location on the screen, with its first light sensing unit outside the screen's dark gap areas. The entire screen transmits the same VLC location beacon without blocks and cells, *i.e.*, $\delta_{i,j}(t)$ is the same for all screen pixels. The background is pure gray, *i.e.*, $I_{i,j}(t)$ is the same for all screen pixels, with four levels of intensities from 96 to 240. k is selected from 3 to 21. In each configuration combination, 100 VLC packets with random payload are transmitted. The Bit Error Rate (BER) is calculated from the decoded results.

Figure 10(a) shows the results. In all cases, the mouse sensor is sensitive to the VLC signals generated by the screen. The low BER indicates a reliable positioning service. The BER curves show a consistent decrease trend when k increases, which is consistent with the intuition that the quality of the received signals gets better

[6]The area of 23.8 inch screen is about 300×530 mm^2. As there are 1024 location IDs, the area covered by one ID is $\sqrt{300 \times 530/1024} \approx 12 \times 12$ mm^2.



(a) VLC Bit Error Rate (b) Impact of Mouse Sensor Hardware (c) Effectiveness of Hybrid Sensing

(d) Impact of Background Content (e) Error of Orientation Estimation (f) Tracking Latency

Figure 10: Performance Evaluation.

when the amplitude of the VLC signals increases. However, when k is more than 10, the decreasing trend becomes flat. One reason is that the noise of the optical mouse sensor is close to this level. Another reason is that the screen panel is not perfect in generating high-frequency VLC signals, which also rises up the noise level. In the remaining tests, we conservatively choose k = 20, which works for most test cases and incurs negligible visual flickers [65].

Figure 10(a) also shows the impact of background brightness. The BER curves under different brightness values are close. This is probably because the optical mouse sensor has a sufficient dynamic range to support working on different surfaces and lighting conditions. In the following tests, we simply use the gray background with $I_{i,j}(t)$=128 unless otherwise specified.

Impact of Mouse Sensor Hardware. To our knowledge, Agilent Avago (now manufactured by PixArt) is the dominating supplier of optical mouse sensors. Most optical mice and raw sensors we tested have a similar I/O interface but slightly different imaging quality. To evaluate the performance of different products, we conduct similar experiments as Figure 10(a) but replace the mouse with different models. As shown in Figure 10(b), BER curves are different among mice, but they are all below 0.05, which is usually enough for transferring location IDs. Mice with higher BER have a higher probability of VLC packet error, resulting in longer positioning latency. As our decoding algorithm has not been optimized for Logitech mice, by analyzing their raw data, we believe the space for improvement remains large. From the curve of Logitech M100, we note that the increase of VLC intensity does not necessarily turn into the gain in signal quality. This is because the noise level of this mouse sensor also increases when k increases, which is probably caused by its customized gain control algorithm. In the following experiments, we use Dell MS111.

Images	1. Pure Gray (128,128,128)	2. Wallpaper of Windows 10	3. Lion on the Grassland [5]
Videos	4. Shining Sunlight [13]	5. Animals in Fighting [1]	6. Fast Rotating Color Blocks [12]

Table 1: Background Content Used in Evaluation.

Effectiveness of Hybrid Sensing. In §4.1.2, we propose a hybrid sensing scheme to combine the data from the single sensor unit and the average register to reduce spatial interference. To evaluate its effectiveness, unlike previous experiments, the mouse is put to 100 different random locations on the screen. 100 packets are transmitted in each location to calculate the Packet Error Rate (PER). In addition to that, the spatial VLC patterns (cells and blocks) are used, exactly the same as the real positioning situation.

Figure 10(c) shows the Cumulative Distribution Function (CDF) of PER. The PER with the hybrid sensing scheme (with hybrid) is consistently better than the PER of a single sensing unit (w/o hybrid). This is because the gap blockage interference affects the single sensing unit in some locations. When the spatial pattern is introduced (with cell), the performance decreases. This is because the light signals from neighboring cells pollute the average value. As cells are necessary for blurring the spatial structure to reduce flickers, in the following experiments, both the hybrid sensing and cells are enabled by default.

Impact of Background Content. Using the same settings as the Hybrid Sensing evaluation (with cell, with hybrid), we replace

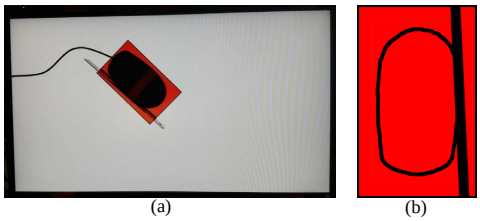


Figure 11: Method of Measuring the Orientation Accuracy.

the gray background with six different backgrounds (Table 1) to evaluate the VLC performance. The videos are 800×400@30 fps `.mp4` files, played in full screen.

Figure 10(d) shows that the best performance is achieved when the background is a solid color. PER curves of the other two static images are close and slightly worse than the solid color. This is because the background brightness differs in different locations. As we measured in Figure 10(a), this results in slightly different BERs. When the background is dynamic, *i.e.*, the video background, the VLC performance is worse than the static ones. This is because when $I_{i,j}(t)$ is not static or is semi-static, its variation will interfere with the VLC signal $\delta_{i,j}(t)$. Video 6 is an extreme case containing drastic changes in brightness and color in both spatial and temporal domains. Its performance is the worst among the six backgrounds. However, its median PER is less than 20%, meaning that the user still has a high probability to get the location ID with one or two VLC packets. Further, the dynamic performance can be improved by using forward error correction or making the decoding algorithm aware of the video content to perform interference cancellation.

Orientation Estimation Error. We evaluate the accuracy of orientation estimation by putting the mouse to 100 different locations and orientations. The method to obtain the ground truth is illustrated in Figure 11(a). In each test, the red pattern in Figure 11(b) is shown on the screen and is rotated for a random angle, which is recorded as the ground truth. Then, a ruler is bound to the mouse shell to help the mouse align to the red pattern with the ruler aligning with the black strip line at the right side of the pattern. Then, the orientation of the mouse is calculated with our algorithm and the error is calculated by comparing it with the ground truth. The results are shown in Figure 10(e), the error is almost uniformly distributed and spans between [-2°, 2°]. This performance is close to tangible tracking approaches based on touchscreens [62], and is better than approaches based on spatial patterns, *e.g.*, VideoMouse [29] (±20°) and Dothraki [53] (only fixed orientations). We also observe that the error is independent of the actual location and orientation of the mouse, this is because the sub-pixel structure that the algorithm is based on is uniform on the screen panel.

Tracking Latency. We measure the tracking delay by first putting the mouse on the screen and then triggering the tracking manually through a keyboard button. The start time is recorded when the button is pressed and the VLC receiver starts to request data from the mouse sensor (it executes automatically in the real deployment). Another two timestamps are obtained. One is taken when the first VLC packet is correctly decoded, which is to measure the VLC positioning delay. The other is taken when the orientation is obtained, which is to measure the delay of orientation estimation.

Figure 10(f) shows the delay profile at 100 different locations and orientations, each measured 100 times. The observed minimum VLC positioning delay is 0.28 s, which is close to the theoretical value of 0.26 s. The delay is 0.41 s on average and 0.76 s for 99% of cases. In addition to the necessary frame duration, two factors contribute to the latency. The first is the decoding error. The second is the timing, as the mouse sensor might start to sample at the middle of a VLC packet and thus need to wait for the next one. The delay of orientation estimation is always around 0.8 s, which is dominated by the latency of transferring two full images from the mouse sensor.

Centaur's VLC positioning delay is close to tracking approaches that also encode information in the time domain [71, 73]. For example, TUIC's [73] frequency tags actively trigger the capacitive screens multiple times to encode more unique tag IDs. But in general, approaches based on touch screens' spatial tracking is much faster. The reporting interval of PERC [62] is about 50 ms. Approaches based on spatial patterns [53, 68] are similar. Because of the waiting time, the VLC positioning should be integrated with the mouse's built-in relative tracking to improve the smoothness. Using VLC alone is more suited for interaction that can tolerate a slight delay. We discuss the potential approaches to reduce the delay in §8.

7 APPLICATION EXAMPLES

Centaur supports basic TUI primitives such as object selecting and dragging. It also allows for fine-grained rotation. It works with commercial mice and is able to support a massive of tokens. We develop four applications to emphasize Centaur's characteristics.

Puzzle Board is an educational application designed to help young kids to learn about letters, shapes, colors, and daily objects. It mimics wooden puzzle boards and offers flexible and interactive content. In the demo shown in Figure 12(a), the child needs to select and drag the letter piece to fit in the hole containing the correct color, shape, and/or object with close pronunciation.

We use a commercial mouse as the MouseToken to interact with the board. It not only lowers the cost of the puzzle board but also retains the physical haptic experience. The large form factor of the mouse allows for easy-grasping, which is easier for young children, whose fine motor skills are not fully developed.

Frog Balance is an educational application designed to facilitate the learning of basic math concepts - counting and comparison. The graphical interface on the screen is a giant frog holding two plates. The plates accept items and the application automatically balances/unbalances the plates in accordance with the "weights" of the items. Only when the "weights" of the two plates are equal, do they level off. The items are baby frogs and physical numbers instanced by MouseTokens. One baby frog weights 1. The weight of the number token is the number on it. The application senses the locations of the tokens to add their weights to corresponding plates.

An example is shown in Figure 12(b). The right plate is preloaded with a mass of "5". The user's goal is to re-balance the plates by adding the correct quantity of frogs to the left plate. The users gain counting skills when practicing this challenge. When the users have become proficient at numbers and counting, they can try to

(a) Puzzle Board (b) Frog Balance (c) Color Widgets (d) Hybrid Game Board

Figure 12: Application Examples. (a) An application guides the user to fill the holes with the correct letters. The user grasps a commercial mouse to interact with the board. (b) An application auto-weights the tokens on the plates. It supports weighting multiple tokens simultaneously. (c) Several knob instances are used to perform eye-free operations such as color tuning. (d) A D&D game board supports hybrid interaction. The user slides the token to select the character action. The above is contained in the video figure.

freely allocate multiple numbers and/or frog tokens to the two plates to learn comparisons and simple arithmetic.

The application needs to sense multiple or even many tokens simultaneously. Since the tokens work independently, the quantity can be increased to meet the demand. One minor implementation issue is the wireless connection. The capacity of one Bluetooth receiver is limited. The user needs to purchase additional receivers to host more tokens. An alternative solution is to replace Bluetooth with Wi-Fi, whose capacity is much larger.

Color Widgets are physical knobs actualized by MouseTokens. Each knob is bound to a color attribute (*e.g.*, contrast, brightness, transparency, *etc.*) and its rotated angles are continuously mapped to the value of that attribute. The example in Figure 12(c) shows that three knobs are mapped to the RGB intensities of the selected virtual objects in the editing window on another screen.

The circle surrounding the knob illustrates the color that it is associated with and the digits report the exact value, providing a professional interface for color editing. When users become familiar with this interface, they might turn to eye-free operations to facilitate editing efficiency. For example, when one hand is selecting/drawing objects with the stylus on the drawing tablet, the other hand can simultaneously tune colors.

Hybrid Gaming Board is an application providing a hybrid experience for strategy games, *e.g.*, D&D games. As shown in Figure 12(d), the screen displays a battle map. The characters in the battle are actualized by MouseTokens and tracked by Centaur. During a turn, the user instructs its character to behave, *e.g.*, move, attack, *etc.* The application calculates and raises events as the results of user actions.

When encountering an event, the user usually needs to make decisions from multiple choices, *e.g.*, attack or defend. If the panel is a touch screen, the user touches the item to select. We also design an interface suitable for non-touch screens. The items to be selected are arranged around the character. To select the item, the user slides the token towards the item and then returns it to the original standing location. We note that this sliding process is fast and cannot be tracked by the VLC positioning. However, it can still be sensed by the mouse sensor with the built-in relative displacement tracking capability.

As active tokens, they support receiving commands from the host application through the wireless connection, primarily to give feedback to the user. We use an LED as the feedback for the character token in Figure 12(d). The LED will light up when the attack action results in a critical hit event. Depending on the implementation, the feedback could be in various forms such as sound, haptic [70], deformation [52], *etc.*

8 LIMITATIONS AND DISCUSSION

This part discusses the limitations of the Centaur system.

Compatibility with General Commercial Mouse. Centaur relies on accessing mouse sensor registers (§3.1). While it is a standard debugging feature of mouse sensors, it is a model-dependent and manufacture-dependent feature of commercial mice. This is because some mice use a micro-controller between the optical mouse sensor and the computer host to support advanced features such as wireless connection, button customization, decoration LEDs, *etc.* In such cases, accessing the raw registers might be blocked by the micro-controller. An upgrade of the controller firmware from the manufacturer is needed to support Centaur.

Screen Refresh Rate. Centaur uses a 240 Hz screen for transmitting VLC location beacons. There are two reasons for doing so. First, high-frequency reduces flicks when transmitting VLC information [65]. Second, high-frequency reduces the transmission delay of a VLC packet, and hence the positioning delay. It is possible to use Centaur with 120 Hz screens, but the system is subject to an increase in the delay by 4 to 8 times while keeping the transmission unobtrusive. Given the quick adoption of high-frequency panels (120 Hz and 144 Hz have already been very common in the market), we believe 240 Hz LCDs and OLEDs with even higher frequencies will be prevalent in the next few years.

Tracking Performance. Centaur's tangible tracking performance can be characterized by positioning accuracy, orientation error, and tracking latency. As we measured in §6, its orientation measurement is very accurate while the position accuracy and delay are similar to some of the existing TUI systems. However, Centaur has no grounding issue and its cost is potentially more affordable. Next, we discuss potential approaches to improve latency and accuracy.

Currently, we adopt low-end optical mouse sensors to understand the cost and performance margin, hence the latency is largely

limited by the register access rate of the mouse sensor (1 kHz). High-end mouse sensors (*e.g.*, PixArt PWM 3360 [11]) can access pixels at a rate of more than 1 MHz, which allows for encoding locations simultaneously in the spatial domain. This might reduce the absolute tracking latency by ×2 to ×4. For further improvement (*i.e.*, by ×10), one might adopt high-resolution (2k or 4k) screens to increase the spatial information density. Similarly, the latency of measuring orientation can be reduced to 30 ms with a high-end mouse sensor [53].

As mentioned in §5, the resolution of absolute VLC positioning depends on the length of location IDs and hence can be increased at the cost of delay. In other words, the reduced latency can be used to increase the precision. Currently, we choose 12×12 mm^2 as the positioning resolution, which bounds the position errors to several mm. This accuracy is close to WATouCH [71], but larger than other tracking approaches based on touch screens [62] (mm level), spatial pattern [53] (sub-mm level), and modulated projection [26] (mm level). Considering the form factor of MouseToken and the integration of the built-in relative tracking, our current setup is sufficient for the application scenarios. Another issue is large screens. Once the ID length is fixed, the precision is inversely proportional to the screen size. *i.e.*, with the same positioning delay, the larger the screen, the lower the physical resolution. In the multi-user collaboration scenario on a large tabletop, dividing the screen surface into multiple regions to multiplex the location IDs could be a workaround.

Positioning Failure. On some very rare locations, it is possible that the first sensing unit of the mouse sensor will point to the unlighted gap, while the center of the mouse sensor is at the block boundaries. Then, the VLC decoding will likely fail, and the failure is hard to recover via re-transmissions. Currently, when the user notices the positioning has taken a longer time than usual, he/she can choose to slightly disturb the mouse to a nearby location. It is then unlikely to encounter the same situation. This works quite well in practice.

9 CONCLUSION

This paper explores a VLC-based approach to track objects on screen surfaces. Compared to existing methods, our approach has several advantages. We show that it allows for low-cost TUI implementation with optical mouse sensors. We also envision new opportunities to apply our designs to benefit other aspects of HCI. It will be interesting to see the proposed techniques interact with existing interfaces such as OLED panels, touch screens, tokens with haptic feedback, and micro-robots.

ACKNOWLEDGMENTS

We thank anonymous CHI reviewers. Their valuable comments help us improve this paper. We thank Shauna Dalton for proofreading. This work is supported by NSFC 62002224 and ShanghaiTech Startup Fund.

REFERENCES

[1] 2022. Animal Video. https://www.youtube.com/watch?v=a5V6gdu5ih8.
[2] 2022. AOC AG251FZ. https://eu.aoc.com/en/gaming/products/ag251fz.
[3] 2022. Convert Optical Mouse into Arduino Web Camera. https://maker.wiznet.io/2014/11/14/convert-optical-mouse-into-arduino-web-camera.
[4] 2022. HID Clients Supported in Windows. https://docs.microsoft.com/en-us/windows-hardware/drivers/hid/hid-architecture.
[5] 2022. Lion. https://www.dropbox.com/s/u86su9hdyuwzagm/lion.pdf?dl=0.
[6] 2022. Monitor Stand. https://cjlegend.en.made-in-china.com/productimage/PjOmkluczWa-2f1j00iTYUhZbWhAkV/China-Black-Mounts-Metal-Folding-Studio-Monitor-Stand.html.
[7] 2022. Nintendo amiibo. https://www.nintendo.com/amiibo/.
[8] 2022. OpenGL - The Industry Standard for High Performance Graphics. https://www.opengl.org/.
[9] 2022. Optical Mouse-Cam. https://spritesmods.com/?art=mouseeye.
[10] 2022. osmo. https://www.playosmo.com/en/.
[11] 2022. Pixart PMW3360. https://github.com/SunjunKim/PMW3360.
[12] 2022. Rotating Color Video. https://www.youtube.com/watch?v=XMmJIMGUe94.
[13] 2022. Shining Sunlight Video. https://www.youtube.com/watch?v=cI9u9R1pRNM&list=PLDTG-mhC5UzPo-QFAg7dXbJUVtRCjUN6J&index=1.
[14] 2022. Source Code. https://github.com/zlab-pub/mousetouch.
[15] Satoshi Abe, Atsuro Arami, Takefumi Hiraki, Shogo Fukushima, and Takeshi Naemura. 2017. Imperceptible Color Vibration for Embedding Pixel-by-Pixel Data into LCD Images. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. 1464–1470.
[16] 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*. 337–346.
[17] AVAGOTECH. 2008. Solid-State Optical Mouse Lens Data Sheet. https://www.sparkfun.com/datasheets/Widgets/AV02-1278EN.pdf.
[18] Lonni Besançon, Paul Issartel, Mehdi Ammi, and Tobias Isenberg. 2017. Mouse, Tactile, and Tangible Input for 3D Manipulation. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 4727–4740.
[19] Andrea Bianchi and Ian Oakley. 2015. MagnID: Tracking Multiple Magnetic Tokens. In *Proceedings of the Ninth International Conference on Tangible, Embedded, and Embodied Interaction*. 61–68.
[20] Scott Brave, Hiroshi Ishii, and Andrew Dahley. 1998. Tangible interfaces for remote collaboration and communication. In *Proceedings of the 1998 ACM conference on Computer supported cooperative work*. 169–178.
[21] Géry Casiez, Stéphane Conversy, Matthieu Falce, Stéphane Huot, and Nicolas Roussel. 2015. Looking through the eye of the mouse: A simple method for measuring end-to-end latency using an optical mouse. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software and Technology*. ACM, 629–636.
[22] Chung-Lin Chan, Jing-Yeu Chen, and Hsin-Mu Tsai. 2014. MouseVLC: Visible Light Communications Using Mouse Sensors. In *2014 IEEE International Conference on Internet of Things(iThings), and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom)*. 328–331.
[23] Liwei Chan, Stefanie Müller, Anne Roudaut, and Patrick Baudisch. 2012. CapStones and ZebraWidgets: sensing stacks of building blocks, dials and sliders on capacitive touch screens. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2189–2192.
[24] Richard O Duda and Peter E Hart. 1972. Use of the Hough transformation to detect lines and curves in pictures. *Communications of The ACM* 15, 1 (1972), 11–15.
[25] George W. Fitzmaurice and William Buxton. 1997. An empirical evaluation of graspable user interfaces: towards specialized, space-multiplexed input. In *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems*. 43–50.
[26] Mathieu Le Goc, Lawrence H. Kim, Ali Parsaei, Jean-Daniel Fekete, Pierre Dragicevic, and Sean Follmer. 2016. Zooids: Building Blocks for Swarm User Interfaces. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. 97–109.
[27] 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*. 4189–4202.
[28] Björn Hartmann, Meredith Ringel Morris, Hrvoje Benko, and Andrew D. Wilson. 2009. Augmenting interactive tables with mice and keyboards. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*. 149–152.
[29] Ken Hinckley, Mike Sinclair, Erik Hanson, Richard Szeliski, and Matt Conway. 1999. The VideoMouse: A Camera-Based Multi-Degree-of-Freedom Input Device. In *Proceedings of the 12th Annual ACM Symposium on User Interface Software and Technology* (Asheville, North Carolina, USA) *(UIST '99)*. Association for Computing Machinery, New York, NY, USA, 103–112. https://doi.org/10.1145/320719.322591
[30] Takefumi Hiraki, Yoshihiro Kawahara, and Takeshi Naemura. 2018. Projection-based Localization and Control Method of Robot Swarms for Swarm User Interfaces. In *Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers*. 584–589.

[31] Eva Hornecker and Jacob Buur. 2006. Getting a grip on tangible interaction: a framework on physical space and social interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 437–446.

[32] Lukas Hostettler, Ayberk Özgür, Séverin Lemaignan, Pierre Dillenbourg, and Francesco Mondada. 2016. Real-time high-accuracy 2d localization with structured patterns. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 4536–4543.

[33] Meng-Ju Hsieh, Rong-Hao Liang, Da-Yuan Huang, Jheng-You Ke, and Bing-Yu Chen. 2018. RFIBricks: Interactive Building Blocks Based on RFID. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 189.

[34] Wenjun Hu, Gu Hao, and Qifan Pu. 2013. LightSync:unsynchronized visual communication over screen-camera links.

[35] Hiroshi Ishii. 2008. The tangible user interface and its evolution. *Communications of The ACM* 51, 6 (2008), 32–36.

[36] Bret Jackson, Tung Yuen Lau, David Schroeder, Kimani C. Toussaint, and Daniel F. Keefe. 2013. A Lightweight Tangible 3D Interface for Interactive Visualization of Thin Fiber Structures. *IEEE Transactions on Visualization and Computer Graphics* 19, 12 (2013), 2802–2809.

[37] Sergi Jordà, Günter Geiger, Marcos Alonso, and Martin Kaltenbrunner. 2007. The reacTable: exploring the synergy between live music performance and tabletop tangible interfaces. In *Proceedings of the 1st international conference on Tangible and embedded interaction*. 139–146.

[38] Kunihiro Kato, Kaori Ikematsu, and Yoshihiro Kawahara. 2020. CAPath: 3D-Printed Interfaces with Conductive Points in Grid Layout to Extend Capacitive Touch Inputs. *Proceedings of the ACM on Human-Computer Interaction* 4, ISS (2020), 1–17.

[39] Hyunyoung Kim, Céline Coutrix, and Anne Roudaut. 2018. KnobSlider: Design of a Shape-Changing UI for Parameter Control. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 339.

[40] Sunjun Kim, Byungjoo Lee, Thomas van Gemert, and Antti Oulasvirta. 2020. Optimal Sensor Position for a Computer Mouse. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–13.

[41] Ye Sheng Kuo, Pat Pannuto, and Prabal Dutta. 2014. Luxapose: indoor positioning with mobile phones and visible light. In *International Conference on Mobile Computing and Networking*.

[42] Johnny C. Lee, Scott E. Hudson, Jay W. Summet, and Paul H. Dietz. 2005. Moveable interactive projected displays using projector based tracking. In *Proceedings of the 18th annual ACM symposium on User interface software and technology*. 63–72.

[43] L. Li, P. Hu, C. Peng, G. Shen, and F. Zhao. 2014. Epsilon: A visible light based positioning system. In *Usenix Conference on Networked Systems Design and Implementation*.

[44] Tianxing Li, Chuankai An, Xinran Xiao, Andrew T Campbell, and Xia Zhou. 2015. Real-time screen-camera communication behind any scene. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*. ACM, 197–211.

[45] Sven Mayer, Xiangyu Xu, and Chris Harrison. 2021. Super-Resolution Capacitive Touchscreens. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–10.

[46] TW Ng. 2003. The optical mouse as a two-dimensional displacement sensor. *Sensors and Actuators A: Physical* 107, 1 (2003), 21–25.

[47] Viet Nguyen, Yaqin Tang, Ashwin Ashok, Marco Gruteser, Kristin Dana, Wenjun Hu, Eric Wengrowski, and Narayan Mandayam. 2016. High-rate flicker-free screen-camera communication with spatially adaptive embedding. In *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*. IEEE, 1–9.

[48] Masa Ogata, Yuta Sugiura, Hirotaka Osawa, and Michita Imai. 2013. FlashTouch: data communication through touchscreens. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2321–2324.

[49] Alex Olwal and Andrew D. Wilson. 2008. SurfaceFusion: unobtrusive tracking of everyday objects in tangible user interfaces. In *GI '08 Proceedings of Graphics Interface 2008*. 235–242.

[50] Esben Warming Pedersen and Kasper Hornbæk. 2011. Tangible bots: interaction with active tangibles in tabletop interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2975–2984.

[51] Samuel David Perli, Nabeel Ahmed, and Dina Katabi. 2010. PixNet: Interference-free wireless links using LCD-camera pairs. In *Proceedings of the sixteenth annual international conference on Mobile computing and networking*. ACM, 137–148.

[52] Majken K Rasmussen, Esben W Pedersen, Marianne G Petersen, and Kasper Hornbæk. 2012. Shape-changing interfaces: a review of the design space and open research questions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 735–744.

[53] Dennis Schüsselbauer, Andreas Schmid, and Raphael Wimmer. 2021. Dothraki: Tracking Tangibles Atop Tabletops Through De-Bruijn Tori. In *Proceedings of the Fifteenth International Conference on Tangible, Embedded, and Embodied Interaction* (Salzburg, Austria) *(TEI '21)*. Association for Computing Machinery, New York, NY, USA, Article 37, 10 pages. https://doi.org/10.1145/3430524.3440656

[54] Daisuke Sekimori and Fumio Miyazaki. 2005. Self-localization for indoor mobile robots based on optical mouse sensor values and simple global camera information. In *2005 IEEE International Conference on Robotics and Biomimetics-ROBIO*. IEEE, 605–610.

[55] Michel Melo Silva, Jose Roberto De Almeida Nozela, Marcio Jose Chaves, Roberto A Braga, and H Rabal. 2011. Optical mouse acting as biospeckle sensor. *Optics Communications* 284, 7 (2011), 1798–1802.

[56] M. Sugimoto, K. Kodama, A. Nakamura, M. Kojima, and M. Inami. 2007. A Display-Based Tracking System: Display-Based Computing for Measurement Systems. In *17th International Conference on Artificial Reality and Telexistence (ICAT 2007)*. 31–38.

[57] Agilent Technologies. 2003. Agilent ADNS-2051 Optical Mouse Sensor Data Sheet. http://bdml.stanford.edu/twiki/pub/Rise/OpticalDisplacementSensor/ADNS2051.pdf.

[58] Lucia Terrenghi, David Kirk, Abigail Sellen, and Shahram Izadi. 2007. Affordances for manipulation of physical versus digital media on interactive surfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 1157–1166.

[59] Philip Tuddenham, David Kirk, and Shahram Izadi. 2010. Graspables revisited: multi-touch vs. tangible input for tabletop displays in acquisition and manipulation tasks. In *Proceedings of the SIGCHI conference on human factors in computing systems*. 2223–2232.

[60] Brygg Ullmer and Hiroshi Ishii. 2000. Emerging frameworks for tangible user interfaces. *IBM systems journal* 39, 3.4 (2000), 915–931.

[61] Nicolas Villar, Daniel Cletheroe, Greg Saul, Christian Holz, Tim Regan, Oscar Salandin, Misha Sra, Hui-Shyong Yeo, William Field, and Haiyan Zhang. 2018. Project Zanzibar: A Portable and Flexible Tangible Interaction Platform. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 515.

[62] Simon Voelker, Christian Cherek, Jan Thar, Thorsten Karrer, Christian Thoresen, Kjell Ivar Øvergård, and Jan Borchers. 2015. PERCs: persistently trackable tangibles on capacitive multi-touch displays. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. 351–356.

[63] Simon Voelker, Kosuke Nakajima, Christian Thoresen, Yuichi Itoh, Kjell Ivar Øvergård, and Jan Borchers. 2013. PUCs: detecting transparent, passive untouched capacitive widgets on unmodified multi-touch displays. In *Proceedings of the adjunct publication of the 26th annual ACM symposium on User interface software and technology*. 101–104.

[64] Daniel Vogel and Ravin Balakrishnan. 2010. Occlusion-aware interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 263–272.

[65] Anran Wang, Zhuoran Li, Chunyi Peng, Guobin Shen, Gan Fang, and Bing Zeng. 2015. Inframe++: Achieve simultaneous screen-human viewing and hidden screen-camera communication. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*. ACM, 181–195.

[66] Malte Weiss, Julie Wagner, Yvonne Jansen, Roger Jennings, Ramsin Khoshabeh, James D. Hollan, and Jan Borchers. 2009. SLAP widgets: bridging the gap between virtual and physical controls on tabletops. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 481–490.

[67] Andrew D. Wilson. 2005. PlayAnywhere: a compact interactive tabletop projection-vision system. In *Proceedings of the 18th annual ACM symposium on User interface software and technology*. 83–92.

[68] Sang won Leigh, Philipp Schoessler, Felix Heibeck, Pattie Maes, and Hiroshi Ishii. 2015. THAW: Tangible Interaction with See-Through Augmentation for Smartphones on Computer Screens. In *Proceedings of the Ninth International Conference on Tangible, Embedded, and Embodied Interaction*. 89–96.

[69] Zhice Yang, Zeyu Wang, Jiansong Zhang, Chenyu Huang, and Qian Zhang. 2015. Wearables can afford: Light-weight indoor positioning with visible light. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*. 317–330.

[70] Kentaro Yasu. 2019. Magnetact: magnetic-sheet-based haptic interfaces for touch devices. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–8.

[71] Hui-Shyong Yeo, Wenxin Feng, and Michael Xuelin Huang. 2020. WATouCH: Enabling Direct Input on Non-touchscreen Using Smartwatch's Photoplethysmogram and IMU Sensor Fusion. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–10.

[72] Hui-Shyong Yeo, Juyoung Lee, Andrea Bianchi, Alejandro Samboy, Hideki Koike, Woontack Woo, and Aaron Quigley. 2020. WristLens: Enabling Single-Handed Surface Gesture Interaction for Wrist-Worn Devices Using Optical Motion Sensor. In *Proceedings of the Augmented Humans International Conference*. 1–8.

[73] Neng-Hao Yu, Li-Wei Chan, Seng Yong Lau, Sung-Sheng Tsai, I-Chun Hsiao, Dian-Je Tsai, Fang-I Hsiao, Lung-Pan Cheng, Mike Chen, Polly Huang, et al. 2011. TUIC: enabling tangible interaction on capacitive multi-touch displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2995–3004.

[74] Neng-Hao Yu, Sung-Sheng Tsai, I-Chun Hsiao, Dian-Je Tsai, Meng-Han Lee, Mike Y. Chen, and Yi-Ping Hung. 2011. Clip-on gadgets: expanding multi-touch interaction area with unpowered tactile controls. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*. 367–372.

[75] Kai Zhang, Chenshu Wu, Chaofan Yang, Yi Zhao, Kehong Huang, Chunyi Peng, Yunhao Liu, and Zheng Yang. 2018. ChromaCode: A Fully Imperceptible Screen-Camera Communication System. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*. ACM, 575–590.

[76] Oren Zuckerman, Saeed Arida, and Mitchel Resnick. 2005. Extending tangible interfaces for education: digital montessori-inspired manipulatives. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 859–868.