Demo: Using Smartphone and Optical Mouse to Enhance Web Interaction

Zhiwei Wang*†, Yihui Yan*†, Qianyi Huang‡, Haitian Ren†, Yizhou Zhang†, Zhice Yang†
†*School of Information Science and Technology, ShanghaiTech University*
‡*Southern University of Science and Technology, and Peng Cheng Laboratory*
†{wangzhw, yanyh, renht, zhangyzh, yangzhc}@shanghaitech.edu.cn, ‡huangqy@sustech.edu.cn
*Co-primary

***Abstract*—This demo shows example applications of the MousePath system–a novel approach to leverage optical mouse to enhance the web interaction on general PCs. MousePath works as a lightweight and ubiquitous bridge to convey information between the co-located smartphone and desktop web pages. MousePath consists of two major entities, the *MousePath transmitter* and the *MousePath receiver*. The *MousePath transmitter* is an app running on the user's smartphone. The *MousePath receiver* is a web script offered by the webserver and running in the web browser on the PC. MousePath works by putting the optical mouse on top of the smartphone's screen, and its transmission is then started and instantly finished without association and login frictions. Its core mechanism is encoding data into the movement of the smartphone's display content and using the optical mouse of the computer to sense the movement for decoding. This demo illustrates MousePath in two practical scenarios: general message sharing, *e.g.*, sharing a URL/password from smartphone to PC's web, and hardware interface sharing, *e.g.*, making smartphone sensor accessible for PC's web.**

***Index Terms*—Cross-device Information Sharing; Optical Mouse; Visible Light Communication (VLC)**

I. Introduction

With the widespread and adoption of mobile computers, such as smartpads and smartphones, leveraging their computing functionalities and sensing abilities to enhance the interactions of co-located desktops is an appealing problem that has attracted extensive research [1]. According to our investigation, despite the user overhead in setting up the connection, *i.e.*, turning on the radio, searching the correct ID, pairing, and probably inputting the password, existing approaches all require deep modifications on the existing web framework, and also, a local application or a browser extension is still needed on the PC host to hand the information to the web page, which suppresses their adoptions in practice.

In [2], we propose MousePath, which provides a more ubiquitous and convenient approach of cross-device data transmission between smartphones and web on PCs. MousePath utilizes the motion sensing ability of optical mice. Apart from the reflected backlight, the image sensor of the optical mouse can also capture the light signals from underlying display content. Utilizing this property, MousePath converts the shifts of the display content on the smartphone's screen to the system's mouse trajectories, through which the data can be transmitted. The key benefit of MousePath is that it directly conveys smartphone's data to the webserver, without cumbersome paring or doing large framework changes. As a result, its deployment incurs minimum overhead and modifications. To hand the data to the web page, the receiver application only needs to run a piece of JavaScript code and no special hardware is required. And then the web features depending on certain hardware could be augmented via MousePath, since multifarious information can be obtained from smartphones.

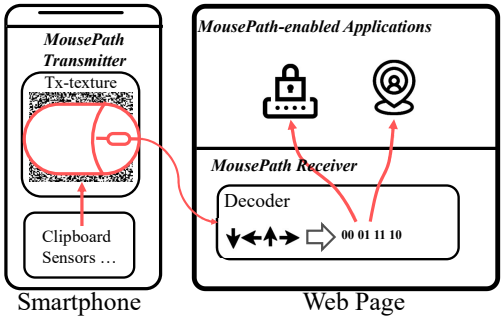

Fig. 1. Overview of MousePath. MousePath consists of two major entities. The *MousePath transmitter* app in the smartphone and the *MousePath receiver* script embedded in the web page. The *MousePath transmitter* gathers information in the smartphone, like sensor data and account credentials, and transfer them to the *MousePath receiver* to enable new web functionalities, *e.g.*, enabling Precise Location-based Service, and facilitating Password Managers.

As Figure 1 shows, the MousePath system consists of two major entities, the *MousePath transmitter* app, and the *MousePath receiver* web script. The *transmitter* modulates useful information into the shifts of the displayed texture, called *tx-texture*. Then the optical mouse, put on the smartphone's screen, obtains the movement vectors from the *tx-texture* and reports them to the web script. Finally, the *receiver* script in the web page can infer the movements of the display content from the system's mouse trajectories, through which the data can be decoded. The data, *e.g.*, the more precise location information, can be used to enhance the user experience.

To present the full MousePath system, we implement a transmitter app on a Android smartphone and a web receiver application. The receiver application is written in JavaScript, which can be embedded in webpages and run in an unmodified web browser. In this demo, we describe two application scenarios for demonstrating the potential and features of our MousePath system. One is the case of sensing sharing and the other one presents message sharing.

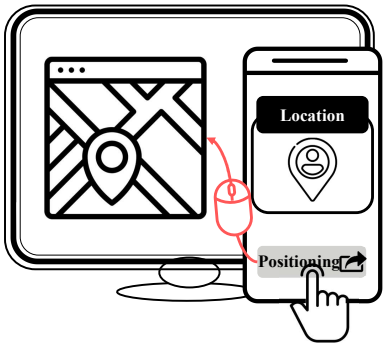

Fig. 2. Precise Location-based Service. PCs are lack precise localization hardware, but the location of a co-located smartphone can be shared to the PC to enable location-based web services.

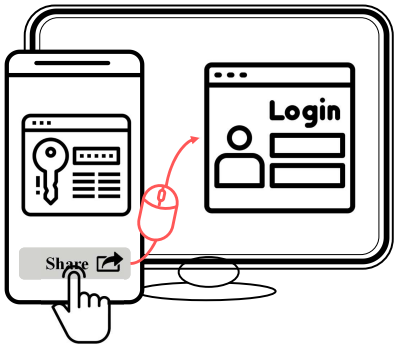

Fig. 3. Password Manager Companion. A message, *e.g.*, a password, can be shared from the smartphone to the webserver conveniently.

II. Demo Detail

We implement our prototype of MousePath running on a Xiaomi Mi Mix 2S as the transmitter and a desktop equipped with Logitech M100 optical mouse as the receiver.

A. Implementation

On the smartphone, we run our MousePath transmitter app (Figure 4b) for data transmission. MousePath transmitter app calls C++ Reed-Solomon library through Java Native Interface (JNI) for data encoding. The Galois field $GF(2^6)$ is used in Reed-Solomon coding. To modulate the movement vectors, the MousePath transmitter app displays a 500×500 pixel *tx-texture* on the screen. The *tx-texture* is updated 60 times per second. To avoid synchronous decoding errors from Jank [3], the MousePath transmitter app uses GPU rendering instead of CPU rendering and binds the *tx-texture* to the GPU texture cache, which ensures that the rendering latency is within 16 ms.

The MousePath receiver hardware includes an optical mouse and a PC. We implement the MousePath receiver as a Javascript application (Figure 4(c)) so that it can be embedded in web pages. In order to capture the mouse movement events, the MousePath receiver uses the `Mouse Lock API` [4] to constrain the target mouse cursor within an HTML widget to prevent the cursor from going past the boundary of the window. the MousePath receiver captures and demodulates movement vectors from `mousemove` events. Then the MousePath receiver calls the JavaScript Reed-Solomon library for decoding and error correction.

B. Setup

MousePath takes the real use cases of the web on PCs into account and proposes a generic and convenient data transfer scheme from the smartphone to the web. Specifically, we demonstrate how MousePath enhancing the web's capability in two scenarios. In each scenario, we would briefly describe interactions of using the MousePath system to input in a web server. We believe MousePath does not introduce too much burden for users.

Scenario I: Precise Location-based Service (Figure 2). When using location-based web services, such as web maps, online shopping systems, *etc.*, precise location information could greatly improve user experience, *e.g.*, automatically filling the shipping address. Smartphone's localization service is based on fusing location information from several wireless technologies, like GPS, iBeacons, Wi-Fi and cellular IP address, *etc.* Without the aid of such hardware, it is essentially not possible to fundamentally increase the precision of the location of a desktop or laptop, which is primarily inferred from its IP address. We enable the PCs to have precise localization ability by sharing the location information from a co-located smartphone through the MousePath channel, which reduces localization error by nearly 94%. Figure 4 shows the example of using MousePath to share user location through location enhancing app and MousePath-enabled web page.

The user interacts with two interfaces. One is the *MousePath input app* in her smartphone. The main action the user takes is to put the PC's optical mouse on top of the smartphone's screen. The second interface is the web browser of the PC. A piece of JavaScript code is embedded in the web page as the *MousePath receiver*. The receiver script is mainly responsible for gathering mouse events and then decoding them. The screenshots of the two interfaces and the action steps (①-④) are shown in Figure 4.

Step ①. The user opens the *MousePath input app* in her smartphone. The app contains a coding area for displaying tx-texture, a data field to prompt the user of the message needed to be sent, *e.g.*, user location in this case, and a start button (Figure 4(a)). Then, the user places the phone flat on the desktop.

Step ②. The user opens the target *MousePath-enabled web page* in the PC, containing the input area, *e.g.*, a form of user location, and a mouse capture area. To start getting the location from the user smartphone, the user moves the cursor to click the mouse capture area to lock the pointer via the `Mouse Lock API` (Figure 4(c)). Then, the receiver script will capture every mouse event via mousemove event listener [5], involving mouse movement vectors and corresponding timestamps.

Step ③. To ensure the location information being directly delivered to the web server via MousePath, the user puts the optical mouse of the computer on the coding area of the smartphone, which is a the QR code-like area in the

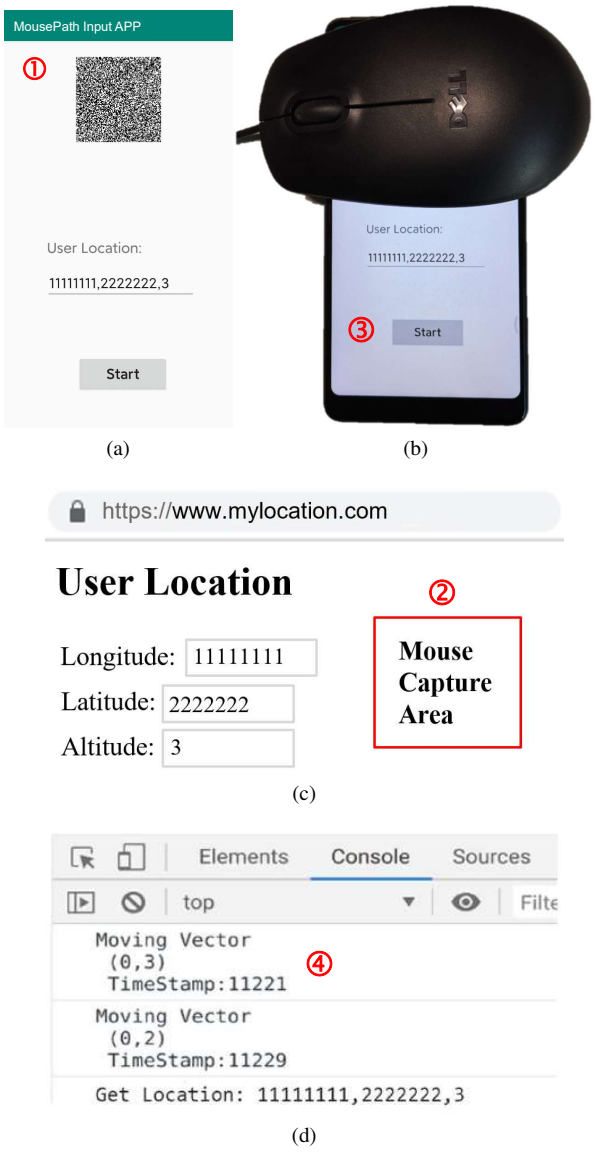

Fig. 4. Example of Scenario II. (a) Screenshot of transmitter UI for location sharing. (b) User puts the mouse on the screen and click 'Start' to transmit the precise address to the PC.(c) The layout of MousePath-enabled web page, involving data field and mouse capture area. (d) Output from the browser's console, including the moving vector values and corresponding timestamps. After successful decoding, the receiver got user's location from smartphone.

MousePath input app. Then, she presses the `Start` button to start the localization service sharing (Figure 4(b)). Utilizing the `Location Manager API` [6] in android, the *MousePath input app* obtains the user's real-time location, including a valid latitude, a longitude, and an altitude, which are displayed in the data field. Then, it encodes the input message into the smartphone's display content and leverages the optical mouse of the PC to sense.

Step ④. If everything goes right, the *MousePath receiver* could infer the movements of the display content from captured mouse events. The moving vector and timestamp of each mouse event are shown in the browser's console (Figure 4(d)). After several seconds, all data demodulated and decoded successfully. The *MousePath receiver* gets user location information and fills it in to the corresponding forms. Now, the web page gets the more precise localization service from the user's smartphone.

Scenario II: Password Manager Companion (Figure 3). Currently, transferring the password to the PC through network protocols, such as Bluetooth, or messaging apps, requires cross-application data copy. Since the current clipboard allows access from any applications, thus is vulnerable to eavesdropping attacks [7]. We augment the password manager by providing an option for directly transferring the password to the web server through MousePath. As MousePath is based on detecting mouse movement, which is only allowed by the focused foreground application, whose attack surface is much smaller.

In Android, we added the MousePath Input App to the contextual action bar (CAB). The user selects the password to be shared and chooses MousePath in the pop-up CAB. The chosen password will be automatically filled into the data field. Then, similar to what was introduced in scenario II, the user puts the optical mouse of the computer on the coding area of the app and presses the `Start` button to transfer the password to the *MousePath receiver* via the screen-to-mouse channel. Note that, the password is directly conveyed into the web page. Thus, MousePath performs better at both security and convenience in comparison with manually copying from clipboard.

Acknowledgements

We sincerely thank anonymous PerCom reviewers for their valuable comments. This work is supported by the Key-Area Research and Development Program of Guangdong Province 2020B0101390001, the ShanghaiTech Startup Fund, the Shanghai Sailing Program 18YF1416700, the "Chen Guang" Program 17CG66 supported by Shanghai Education Development Foundation and Shanghai Municipal Education Commission, NSFC 62002224 and 62002150, and the Project of "FANet: PCL Future Greater-Bay Area Network Facilities for Large-scale Experiments and Applications" LZC0019.

References

[1] H. Xia, J. Zhang, Y. Zhu, C. Yu, and Y. Shi, "Mobile assistant: enhancing desktop interaction using mobile phone," in *Proceedings of the 2012 ACM international conference on Interactive tabletops and surfaces*, 2012, pp. 379–382.

[2] Z. Wang, Q. Huang, Y. Yan, Z. Yang, H. Ren, and Y. Zhang, "MousePath: enhancing web interaction on PCs through smartphone and optical mouse," in *2021 IEEE International Conference on Pervasive Computing and Communications (PerCom) (PerCom 2021)*, Kassel, Germany, Mar. 2021.

[3] Y.-D. Lin, E. T.-H. Chu, E. Chang, and Y.-C. Lai, "Smoothed graphic user interaction on smartphones with motion prediction," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2017.

[4] "Mouse Lock API - Web APIs," https://developer.mozilla.org/en-US/docs/Web/API/Pointer_Lock_API, 2020.

[5] "mousemove - Web API," https://developer.mozilla.org/zh-CN/docs/Web/API/Element/mousemove_event, 2020.

[6] "Location — Android Developers," https://developer.android.com/reference/android/location/Location, 2020.

[7] "clipboard attacks," https://security.stackexchange.com/questions/33428/is-a-password-in-the-clipboard-vulnerable-to-attacks, 2020.