MousePath: Enhancing PC Web Pages through Smartphone and Optical Mouse

Zhiwei Wang¹, Qianyi Huang², Yihui Yan¹, Haitian Ren¹, Yizhou Zhang¹, Zhice Yang¹

¹SIST, ShanghaiTech University
²Southern University of Science and Technology, and Peng Cheng Laboratory
PC Web Pages Can be Enhanced by Client-side Info
Can PC Web Pages be further enhanced by taking information from the co-located smartphone?
Smartphone-to-Web Communication

• Potential Application 1: Password Manager Companion
  • Retrieving the account and password stored in the smartphone for web login.
Smartphone-to-Web Communication

• Potential Application 2: Precise Location-based Service
  • Enabling precise location-based service on PC web pages.
  
  Location obtained via sensor-fusion

  Wrong location from IP

  Precise location from smartphone

  Real Location

  Precise Location
Smartphone-to-Web Communication: Problem

• Potential Approaches
  • Cloud Services
    • e.g., Your Phone App in Windows 10
  • Ad-hoc wireless protocols
    • e.g., Bluetooth

• Limitations
  • Setup overhead
    • e.g., link establishment, account sync
  • Not directly transferring to the web pages!
    • Additional effort is needed, i.e., installing a browser plugin
    • Phone-to-PC Path != Phone-to-Web Path
Our Idea: a Channel between Mouse and Smartphone

- Putting an **optical mouse** on top of the smartphone screen, then information from the phone is **directly** transferred to the **web** page.
How Do Optical Mice Work?

• **Mechanism:** optical mouse uses image sensor to detect movement.
Key Observation

- The optical mouse can sense the movements of the display content on the smartphone screen.

A texture moves on the phone screen.

The PC cursor moves accordingly.
Design: Modulating Movement Vectors

• Direction Shift Keying

Shifting the texture in the transmitter

<table>
<thead>
<tr>
<th>Bits</th>
<th>00</th>
<th>01</th>
<th>10</th>
<th>11</th>
<th>null</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symbol</td>
<td>↓</td>
<td>←</td>
<td>→</td>
<td>↑</td>
<td>•</td>
</tr>
</tbody>
</table>

Modulating the shift direction of the texture.
Design: Packet Structure

- Reed-Solomon (RS) coding for error correction
- Preamble for identifying the beginning of a packet
Challenge: Fine Synchronization

- Constraints:
  - Movement Vectors are Noisy
  - Preamble Sequence is Short

How to find the optimal sample of each symbol?
Inferring the Screen Refresh Time from Statistics

- The amplitude (one dimension) of reported movement vectors

![Graph showing sampling time vs. amplitude](image)

*Observing the statistics by folding the samples into one frame cycle \([0, T_{frame}]\)*
Inferring the Screen Refresh Time from Statistics

- Folding samples into one frame cycle \([0, T_{\text{frame}}]\)
Inferring the Screen Refresh Time from Statistics

• Observation:
  • High-amplitude samples are concentrated in one mouse cycle $[0,T_{mouse}]$, i.e., the blue window.

Why?

The blue window reflects the time slots of the mouse events just after screen refreshes:
• high intensity of movement stimulation.
Design: Re-sampling

• The Reference Optimal Sampling Time ($t_{opt}$):
  • The centers of the blue window in each frame.

Thus, we can re-sample each frame by only selecting the mouse events in the blue window.
Design: Re-sampling and Demodulation

- Choosing the Sample Close to $t_{opt}$ in Each Frame

Re-sampling movement vectors

Mapping the movement vectors into bits.

<table>
<thead>
<tr>
<th>Movement Vector $\vec{M} = [M_x, M_y]$</th>
<th>Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_y &gt; 0 &amp;&amp;</td>
<td>M_y</td>
</tr>
<tr>
<td>$M_x &gt; 0 &amp;&amp;</td>
<td>M_x</td>
</tr>
<tr>
<td>$M_x &lt; 0 &amp;&amp;</td>
<td>M_x</td>
</tr>
<tr>
<td>$M_y &lt; 0 &amp;&amp;</td>
<td>M_y</td>
</tr>
</tbody>
</table>
Implementation

• Transmitter App in the Smartphone

① Movement texture
② Click ‘Start’ to transmit the precise location to the web application

Transmitter UI for location sharing

Putting the mouse on the screen
Implementation

• Receiver JavaScript in the Web Page

③ Use Mouse Lock API to avoid cursor random walk

④ Mouse move events from mousemove listener

User Location

Longitude: 11111111
Latitude: 2222222
Altitude: 3

The layout of MousePath-enabled web page

Output from the browser’s console
Evaluation on Device-related Factors

• Error Rate V.S. Different Hardware:

(a) Xiaomi Mix 2S with **LCD Screen**  
(b) Xiaomi 8 with **OLED Screen**
Evaluation on Texture-related Factors

- Angle Error:
Q&A
Thank you!