Demo: Telekinetic Thumb Summons Out-of-reach Touch Interface Beneath Your Thumbtip

Inseok Hwang
IBM
Austin, TX, USA
ihwang@us.ibm.com

Eric Rozner
University of Colorado Boulder
Boulder, CO, USA
eric.rozner@colorado.edu

Chungkuk Yoo
IBM
Austin, TX, USA
ckyoo@ibm.com

CCS CONCEPTS

• **Human-centered computing → Ubiquitous and mobile computing systems and tools**; *Touch screens*; *Gestural input*; *User interface programming*.

ACM Reference Format:
Inseok Hwang, Eric Rozner, and Chungkuk Yoo. 2019. Demo: Telekinetic Thumb Summons Out-of-reach Touch Interface Beneath Your Thumbtip. In *The 17th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '19), June 17–21, 2019, Seoul, Republic of Korea.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3307334.3328571

1 INTRODUCTION

As personal interactive devices become more ingrained into our daily lives, it becomes more important to understand how seamless interaction with those devices can be fostered. A typical mechanism to interface with a personal device is via a touch screen, in which users use their fingertip or stylus to scroll, type, select, or otherwise control device usage. Touch-based techniques, however, can become restrictive or inconvenient under a variety of scenarios. For example, personal devices such as phones or tablets are continuously increasing in size, making one-handed interaction difficult because one cannot easily hold the phone and touch the screen (with the thumb) at the same time with one hand. Therefore, in this demo, we present a new technique to interact with personal devices in which the screen and touch screen interactions can *adapt* to a user's grip or current touch constraints.

Specifically, we design a technique called Telekinetic Thumb that allows a user to modify the location or content of the screen in a seamless manner. In a Telekinetic Thumb-enabled application, Telekinetic Thumb monitors the user's above-screen finger gestures in the background while the user is using the application as usual. Upon detecting a pre-determined *telekinetic* gesture, Telekinetic Thumb moves the whole application screen underneath the user's finger, preferably in the way that the UI element the user intended to touch is relocated precisely beneath the finger. Moving the out-of-reach UI element below the finger allows for seamless one-handed control over devices of arbitrary screen size without re-gripping the device. Telekinetic Thumb takes advantage of *pretouch* technology, in which a user's hovering finger can be detected above the screen (as found in Samsung Airview or prior work [5]). By utilizing pretouch to define and detect the gesture instead of on-screen interaction, our technique is unlikely to interfere with preexisting application or OS behavior.

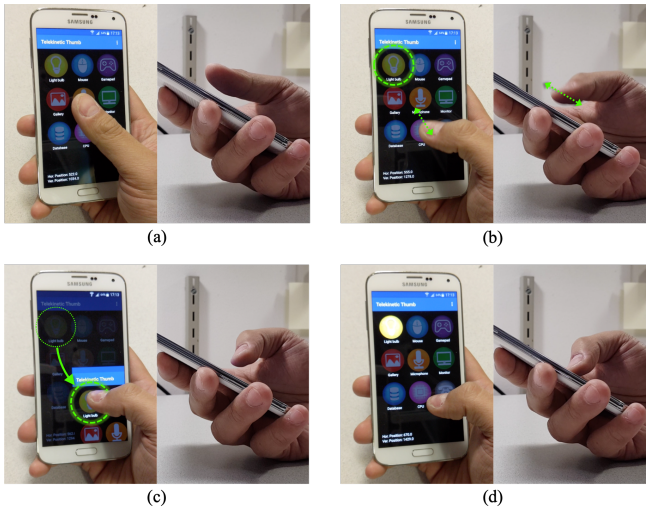

Figure 1: (a) Some icons are out of the thumb's reach. (b) User performs a pulling gesture above the screen. (c) The entire application screen relocates closer to the thumb location. (d) Clicking an icon on the relocated screen is applied to the original application.

Figure 1 demonstrates a user interacting with a sample application with Telekinetic Thumb enabled. The user is holding the phone in a single-handed grip, which would be a very typical way to hold the phone while the user is moving. Figure 1(a) shows that the user is trying to click an icon at the far top-left corner, but it is out of the thumb's reach. Instead of changing the grip or using another hand, in Figure 1(b), the user performs a quick "pulling" gesture–a wiggle in the air above the screen in this case. At this moment, Telekinetic Thumb intelligently relocates the entire application screen based on the magnitude and direction of the gesture vector [2–4] as shown in Figure 1(c). Now the target icon that the user intended to reach is right beneath the user's thumb. Figure 1(d) shows that, upon clicking this relocated icon, the input has been applied to the original application.

We highlight that Telekinetic Thumb is not just a new form of mobile interactivity, but backed by a thoughtful system that makes Telekinetic Thumb highly generalizable to most mobile applications and friendly to the application developers. Below we detail our

MobiSys '19, June 17–21, 2019, Seoul, Republic of Korea

ACM ISBN 978-1-4503-6661-8/19/06.
https://doi.org/10.1145/3307334.3328571

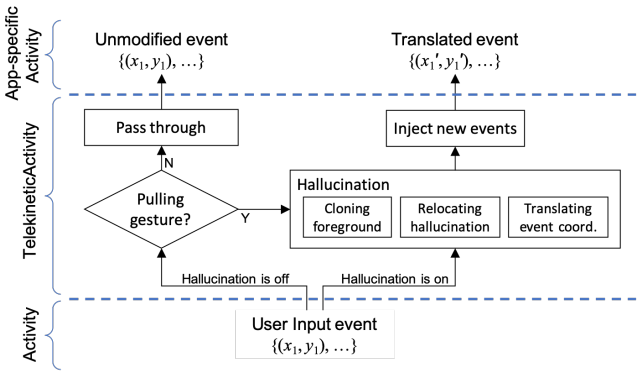

Figure 2: Input event flow inside TelekineticActivity.

implementation strategies, followed by a demonstration plan and a discussion on future works.

2 SYSTEM DETAILS

We implemented a prototype of Telekinetic Thumb as a library on Android 6.0.1 platform and a Samsung Galaxy S5 smartphone.

Our key philosophy in implementing Telekinetic Thumb was to maximize the generalizability and friendliness to the application developers. Recognizing finger gestures and altering the foreground user interface could be implemented in multiple different ways, but it would be of diminishing value if it requires significant changes to an existing application's code base to incorporate Telekinetic Thumb feature in their applications.

In this light, we devised an implementation strategy asking the application developers to change only a single line of code per activity of their applications. Specifically, the key features of Telekinetic Thumb are encapsulated within a class named `TelekineticActivity` that extends the `Activity` class of Android SDK. Every Android application with foreground user interfaces consists of activities, which are essentially application-specific derivations of the `Activity` class. Thus, incorporating Telekinetic Thumb features into an existing application's code base is as simple as substituting the base class of `Activity` with `TelekineticActivity` that an application's custom activity inherited from. This is no more than adding one more layer of inheritance between the base `Activity` class and an application's custom activity class.

The detailed operations of `TelekineticActivity` are shown in Figure 2. Upon initialization, `TelekineticActivity` intercepts all incoming user input events by the `Activity.dispatchTouchEvent()` method and has a classifier that determines whether these events are a pulling gesture or not. If not a pulling gesture, these events are dispatched back to the application and consumed along the normal view hierarchy. If a pulling gesture, `TelekineticActivity` creates a `Hallucination`, which is a new `View` floating on top of the application showing a screenshot of the application's current content view. Then, subsequent user input events are dispatched exclusively to the `Hallucination`, so that the user may drag the `Hallucination` to a more convenient location, and/or touch a fake UI element shown on the `Hallucination`. To deliver this touch event to the right location on the application's activity, `TelekineticActivity` creates a clone of this event, reconfigures the event coordinates by reverting the relocation amounts, removes the floating `Hallucination`, and injects it to the screen via Java Reflection. Now the application's activity takes over the event and the application-specific processing continues.

3 DEMONSTRATION PLAN

We demonstrate Telekinetic Thumb on a commodity smartphone with an exemplary application (see Figure 1). While running the application, multiple buttons are evenly located on the screen. A user is requested to touch one of them that are located outside of the thumb's coverage of his/her hand which holds the phone. Without Telekinetic Thumb, the user needs another hand to touch the target or changes the grip. However, Telekinetic Thumb enables the user to seamlessly touch the button without using another hand or changing the grip. It recognizes the user's intention to touch the target using the pretouch gesture to reach the target and then brings the target button underneath the thumb. In our demonstration, we use the commodity smartphone's pretouch feature, AirView, for pretouch gesture sensing. Future designs may use other pretouch features enabled by self-capacitance screen support [1] or enable pretouch through prior art like SymmetriSense [5] that enables the pretouch feature on smartphones without additional hardware support.

4 CONCLUSION

Telekinetic Thumb provides a seamless way for a device screen to adapt to a user's grip or finger without interfering with touch screen behavior. We believe this is a powerful primitive that will enable much future work. For example, we are actively researching techniques that learn user intents in order to relocate portions of the screen under a finger in a more accurate and effective manner. Intelligent pretouch techniques can be used to increase accessibility by tracking the trajectory of a finger before a touch or by allowing for more flexible usage patterns. Additionally, intelligent prefetch techniques can optimize system performance by preloading application data or behavior associated with an imminent touch event.

REFERENCES

[1] K. Hinckley, S. Heo, M. Pahud, C. Holz, H. Benko, A. Sellen, R. Banks, K. O'Hara, G. Smyth, and W. Buxton. Pre-touch sensing for mobile interaction. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 2869–2881. ACM, 2016.

[2] I. Hwang, J. Mukundan, E. J. Rozner, and C. Yoo. Displaying virtual target window on mobile device based on directional gesture, Aug. 7 2018. US Patent 10042550.

[3] I. Hwang, J. Mukundan, E. J. Rozner, and C. Yoo. Displaying virtual target window on mobile device based on user intent, Oct. 2 2018. US Patent 10091344.

[4] H. Xia, R. Jota, B. McCanny, Z. Yu, C. Forlines, K. Singh, and D. Wigdor. Zero-latency tapping: Using hover information to predict touch locations and eliminate touchdown latency. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, UIST '14, pages 205–214, New York, NY, USA, 2014. ACM.

[5] C. Yoo, I. Hwang, E. Rozner, Y. Gu, and R. F. Dickerson. Symmetrisense: Enabling near-surface interactivity on glossy surfaces using a single commodity smartphone. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 5126–5137. ACM, 2016.