Mobile Touch-Free Interaction for Global Health

Nicola Dell, Krittika D'Silva, Gaetano Borriello
Department of Computer Science & Engineering
University of Washington
{nixdell, kdsilva, gaetano}@cs.washington.edu

ABSTRACT

Health workers in remote settings are increasingly using mobile devices to assist with a range of medical tasks that may require them to handle potentially infectious biological material, and touching their mobile device in these scenarios is undesirable or potentially harmful. To overcome this challenge, we present Maestro, a software-only gesture detection system that enables touch-free interaction on commodity mobile devices. Maestro uses the built-in, forward-facing camera on the device and computer vision to recognize users' in-air gestures. Our key design criteria are high gesture recognition rates and low power consumption. We describe Maestro's design and implementation and show that the system is able to detect and respond to users' gestures in real-time with acceptable energy consumption and memory overheads. We also evaluate Maestro through a controlled user study that provides insight into the performance of touch-free interaction, finding that participants were able to make gestures quickly and accurately enough to be useful for a variety of motivating global health applications. Finally, we describe the programming effort required to integrate touch-free interaction into several open-source mobile applications so that it can be used on commodity devices without requiring changes to the operating system. Taken together, our findings suggest that Maestro is a simple and practical tool that could allow health workers in remote settings to interact with their devices touch-free in demanding settings.

Keywords

Touch-free interaction; mobile device; smartphone; situational impairment; global health; mHealth.

1. INTRODUCTION

The field of mobile health is an emerging area of research that encompasses the use of mobile devices, such as smartphones or tablet computers, to deliver health services and information. The rapid increase in mobile device penetration throughout the world, and particularly in developing countries, has resulted in the creation of a large number of mobile health applications designed to increase access to healthcare and health-related information, improve disease diagnosis and tracking, and provide health workers with ongoing medical education and training. However, many of the medical tasks that these mobile applications are designed to assist with also require health workers to handle potentially infectious biological material. For example, health workers using a mobile device to process rapid diagnostic tests for infectious diseases must handle blood samples [4]. Although health workers typically wear latex gloves when handling this potentially harmful material, touching a mobile device in these scenarios risks contamination of both the device from the gloves, or the gloves from the device. In these situations, it would be beneficial if health workers were able to interact with the device without touching it at all.

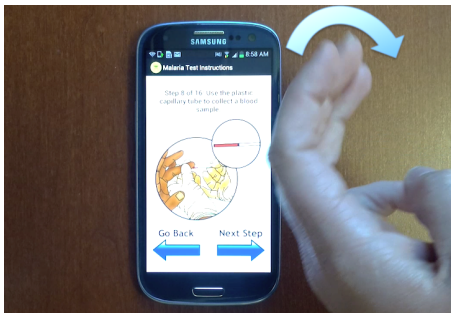


Figure 1: Maestro uses the forward-facing camera on mobile devices to detect users' in-air gestures.

Fortunately, modern mobile devices come equipped with a range of sensors, including cameras, that can be used to create alternate methods of interaction with devices. A device's programming interface gives developers access to raw data from these sensors, but significant programming effort from the developer, unrelated to the application's core functionality, is needed to facilitate a touch-free experience. To overcome this barrier, we created Maestro, a software library for Android that uses computer vision to detect in-air gestures (see Figure 1). The Maestro API provides application developers with easy, high-level access to a variety of touch-free gestures that can be detected using any commodity smartphone equipped with a basic forward-facing camera.

Many of the global health scenarios that we target lack reliable Internet connectivity. As a result, we focus on recognizing a minimal set of touch-free gestures using image

HotMobile '15, February 12 - 13 2015, Santa Fe, NM, USA
ACM 978-1-4503-3391-7/15/02 ... $15.00
http://dx.doi.org/10.1145/2699343.2699355

processing algorithms that run entirely on the device. This simple and intuitive gesture set is appropriate for a variety of global health applications, including mobile job aids for clinical procedures and applications that assist with disease diagnosis [4], and a range of diverse users can achieve high enough gesture recognition rates to be practical in demanding public health settings.

Key results from our work demonstrate that Maestro can successfully enable touch-free interaction for a range of mobile applications. In addition, the energy consumption and memory usage overheads imposed by the system are acceptable for practical use. The system is also capable of processing image data obtained from the device's camera fast enough to detect and respond to users' gestures in real-time, with the overall speed of interaction limited by how fast users move their hands, rather than by the speed of computation. Furthermore, findings from a user evaluation with 18 participants show that, within minutes of first being introduced to Maestro's gestures, many participants were able to sustain speeds of over 40 correct touch-free gestures per minute. Finally, participants were also able to use Maestro's touch-free gestures to successfully perform a range of common interactions (*e.g.*, scrolling, swiping and selecting targets) and navigate our target global health applications touch free.

This paper makes the following contributions: (1) the identification of key design principles required to make a touch-free system appropriate for global health scenarios in low-resource settings; (2) the design of Maestro, a software-only gesture detection system that uses the forward-facing camera on commodity mobile devices to detect and respond to users' in-air gestures; (3) an inspection-based evaluation that details the programming effort required to add touch-free interaction to several open-source mobile applications; (4) a performance evaluation that quantifies the responsiveness of the system and the energy and memory usage overheads incurred by Maestro's algorithms; and (5) a user evaluation that shows Maestro's gestures are easy-to-learn and that people are capable of using them to successfully navigate a variety of user interfaces and realistic applications. Our findings suggest that Maestro is a simple and practical tool that allows users in a range of global health scenarios to interact with commodity mobile devices touch free.

2. RELATED WORK

Gestural input for natural human-computer interaction has for decades inspired research that explores potential future scenarios [1]. We wanted to create a practical working solution that runs on commodity mobile devices and that is viable for immediate use in the field.

A large number of existing desktop-based vision systems focus on detecting and segmenting the hand from the background using shape or skin color [11]. Systems have been designed to detect both static hand poses and dynamic gestures [18] and many of these systems have been useful for sign-language recognition [16]. The reliability of hand-tracking systems depends heavily on the specific features used for tracking and segmentation and on powerful machine-learning algorithms for recognition [11]. The sophistication of these algorithms enables many of these systems to recognize between 5 and 50 gestures. However, developing a vision-based touch-free system for use by health workers in low-resource settings requires a different approach for several reasons. First, most prior solutions run on computationally powerful desktop computers. Our work focuses on mobile devices with limited computational capabilities, which makes it more difficult to meet real-time requirements using such intensive approaches. In addition, these systems generally expect users to dedicate both hands (and their full attention) to interacting with the system. However, in global health scenarios, users' hands may be occupied with other tasks, such as holding a biological sample, and their attention should be primarily on patients and not on the system. These constraints suggest that, rather than focusing on a large and complex gesture set, a system targeting global health scenarios should instead focus on recognizing simple, intuitive and easy-to-remember gestures that can be performed while the health worker's hands are occupied with other tasks and the device is on a surface.

A variety of other existing systems use additional hardware to perform gesture detection. Glove-based systems, like ImageGlove [10], require users to wear a special glove and map motion changes in the glove to gestures. Another class of systems use specialized sensing devices designed to be worn on the body, like Abracadabra [5]. There are also systems that augment commodity devices with additional sensors, such as Hoverflow [7] and SideSight [3]. PalmSpace [8] uses a depth camera to track 3D hand positions, while Pouke et al. [14] use a Bluetooth-connected sensor attached to the hand. Although the use of additional hardware can lead to powerful solutions, users must purchase and set up the additional components. This may be particularly challenging for non-technical users in developing countries and the additional components only increase the likelihood that something may get lost, broken, or exhaust its own batteries. By using a self-contained commodity device, Maestro greatly simplifies deployments and training.

Finally, a number of specialized consumer devices, like Leap Motion's Leap [9] enable touch-free interaction with large displays or televisions. In addition, several specific mobile device models, like the Samsung Galaxy S4 [15], also offer some built-in touch-free capabilities. However, the algorithms and APIs that provide this functionality have not been documented or released, and no rigorous system or user evaluations have been presented that describe the usability of these systems. We wanted to create a solution that works on a wide range of commodity devices that are affordable and available in developing countries, rather than force users to purchase a specific device model.

3. SCENARIO AND DESIGN PRINCIPLES

We focus on the following usage scenario for Maestro: a health worker in a rural clinic in Africa has been issued a mobile device to assist with medical duties. The health worker enters a patient exam room to perform a medical procedure, such as administering a rapid diagnostic test for HIV [4]. She removes the mobile device from a pocket and places it flat on the table. She also collects and prepares any necessary medical equipment, like syringes, gauzes and protective gloves. When ready, she starts the appropriate mobile application on the device and activates touch-free interaction. Then, she puts on a pair of protective latex gloves

and moves through the steps of the procedure, interacting with the patient and with the device touch-free by passing a hand over it as if "swiping in mid-air." When instructed to by the application, she collects the necessary blood sample from the patient and places it on the test. After completing the procedure, she disposes of any infectious material, removes her gloves, and can then touch the device again.

Our key design principles are informed by current literature and the constraints presented by global health scenarios.

Camera-Based Input: We considered a variety of input modalities, including sound and voice, before settling on camera-based gesture detection. Clinics in developing countries are usually noisy, busy and crowded, which could decrease the accuracy of voice recognition. In addition, collecting blood or biological samples from patients (especially children) may result in crying or other noises that could interfere with a voice or sound-based system. Moreover, we wanted to build a system that works for diverse populations who speak different languages. This made it undesirable to require users to speak voice commands in a potentially unfamiliar language. We also wanted to ensure that health workers are free to speak clearly to patients, and issuing voice commands to the system could interfere with health worker-patient conversations. Finally, camera-based input is already used in clinical settings for a range of tasks, including data collection [6], microscopy [2], and disease diagnosis [4].

Flexible and Easy-to-Learn: Users need to be able to interact with a device while their hands are occupied, such as holding a biological sample. Thus, Maestro does not rely on users being able to make specific shapes with their fingers or hands and instead targets a simple and flexible, although limited, gesture set. Furthermore, to prevent contamination users need not touch other objects, such as a stylus or tapping a tabletop. Touch-free interaction should be intuitive and easy to learn. We specifically target a simple gesture set that people would quickly understand and easily remember but provide basic navigation and selection primitives.

No Additional Hardware and Device-Independent: Requiring that people purchase and setup additional hardware is a significant barrier for many potential users in developing countries and increases the likelihood that hardware will get lost or broken or not be properly connected or powered. Maestro only uses hardware that is integral to a range of commodity mobile devices readily available around the world. Health programs in developing countries have diverse needs and budgets, thus, we do not want to constrain users to specific models (*e.g.*, the Samsung Galaxy S4). Instead, Maestro is a software-only solution that enables touch-free interaction on a wide variety of different device models.

Local Computation and Calibration-Free: Many remote clinics in developing countries do not have reliable Internet access. Thus, all of the computation required to detect and respond to touch-free gestures must be performed locally on the device. Maestro targets global health applications that will be used by a diverse range of people in a variety of scenarios. In addition, in many developing countries, multiple users often share a single device. Thus, the system must work "out of the box" for a wide range of people (including different skin tones) and not require per user calibration.

Application-Level and Developer-Friendly: Enabling touch-free interaction should not require users to make changes to a device's underlying operating system because many commercially available mobile devices are locked by manufacturers and system-level changes may void manufacturer warranties. Maestro exposes a robust and usable API that developers can use to enable touch-free interaction on existing applications with only minimal modifications.

4. MAESTRO

Touch-free interaction will be useful for a range of specific applications in which it is undesirable or potentially harmful for users to touch a device. We are not suggesting that touch-free interaction will replace touch, nor have we tried to fully replicate multi-touch gestures provided by many touch-screen devices. Instead, Maestro provides much of the same functionality as a desktop mouse, allowing users to move a cursor around the screen, scroll, and select user interface targets. We expect that this limited gesture set will achieve simplicity and ease-of-use while also providing sufficient expressivity and flexibility for our target applications and many others. At a high-level, Maestro monitors image data from a device's front-facing camera in real-time and detects five basic gestures: up, down, left, right and dwell. The directional gestures are triggered by moving any object, including a finger or hand, across the camera's field of view in the desired direction. The speed of the gesture is recorded and can be used to add further expressivity to the motion commands. The dwell gesture is triggered when the user covers the camera's field of view for a short period of time. This gesture is used to indicate a click or tap.

4.1 Algorithm

In keeping with our design principles, Maestro's gesture recognition algorithm is relatively simple, both conceptually and computationally. The first stage of the algorithm works by comparing pairs of consecutive grayscale video frames captured by the forward-facing camera. For each pair of frames, we calculate the absolute value of the per-pixel difference between the images. This gives us the number of pixels at which movement occurred between the two frames. We then find the moment that the number of moving pixels exceeds a dynamically adjusted threshold, and consider that point in time to be the start of a gesture. Once the number of moving pixels drops below the threshold, the gesture is considered complete. To calculate an appropriate threshold, we keep a running average of the amount of motion between consecutive pairs of frames for the last few hundred frames and pick a value that is substantially larger (*e.g.*, 15% higher) than the average current motion. We also calculate the duration of the gesture by recording the time at which the motion started and ended. Gestures that are classified as being either too short or too long are filtered as noise. In addition, we enforce a minimum between-gesture time to minimize the chances that backswing, secondary, or recovery motions are falsely detected as gestures. After detecting a gesture, the next step is to determine whether the gesture is directional, or whether the user paused over the camera for a dwell gesture. To differentiate between gesture types, we analyze the average grayscale intensity of each frame that makes up the gesture. In a directional gesture, the intensities of the frames get darker as the user moves towards the camera, and then lighter again as the user moves away. By

contrast, with a dwell gesture, the intensities of the frames get darker as the user covers the camera and stay dark until the motion has stopped. To detect this difference, the system again keeps a running average of the grayscale intensity for the last few hundred frames. This average constitutes a background intensity value. If the intensity of the final gesture frame is sufficiently lower (*e.g.*, 15% lower) than the current background intensity, it is classified as a dwell gesture. If a dwell gesture is not detected, the gesture is deemed to be directional and to determine its direction, we compute the average geometric coordinates of all the pixels at which motion was detected at the start of the gesture. This computation results in a single point at which movement was centered when the gesture began. We then compute the corresponding center of motion at the end of the gesture. If the distance covered along one of the axes is sufficiently large (*e.g.*, 20% of the screen's width), the gesture is classified as being in that direction.

4.2 Mapping Gestures to Interactions

UI targets include buttons, checkboxes, and lists. Typically, these have listeners that trigger an action when the element is selected. Although users can touch anywhere on the screen, it often only makes sense to touch a target. Target-awareness has been exploited in prior work, including using the tab key to navigate the focus of an application from one target to the next. Maestro also exploits the focus property of UI targets. Directional gestures can move the application's focus to the next target in the specified direction and the dwell gesture used to select the target currently in focus. In addition to navigating UI elements, touch-free gestures can be directly mapped to horizontal and vertical scrolling. The speed of the gesture can control the speed of scrolling. A dwell gesture can be used to activate scrolling, and another dwell or timeout used for deactivation. Finger-based touch gestures, such as swipes and flicks, have become popular in touchscreen applications and provide a natural style of user interaction. Maestro's directional gestures provide a similar interaction style and developers can choose to map Maestro's gestures (*e.g.*, using their speed) to these finger-based touch gestures to achieve similar effects.

4.3 Implementation

We implemented Maestro as an application-level software library on the Android platform. The image processing components of the library were implemented in native code using OpenCV [13], an open-source computer vision library. Gesture detection and classification were implemented using Android's Java framework and use the JNI to communicate with OpenCV's native image processing algorithms. All of the computation is performed locally on the device.

5. ADDING TOUCH-FREE INTERACTION TO EXISTING MOBILE APPLICATIONS

Maestro exposes an API that developers can use to map touch-free gestures to UI interactions. Any application module or activity that will respond to touch-free gestures should contain a small amount of code to initialize the camera and the UI, and five methods that specify the actions to be taken when each of the five touch-free gesture types is detected. This section describes the programming effort required to integrate Maestro into several popular open-source applications: Open Data Kit (ODK) Collect [6], the Vanilla media player [17] and NewsBlur [12] (see Figure 2).

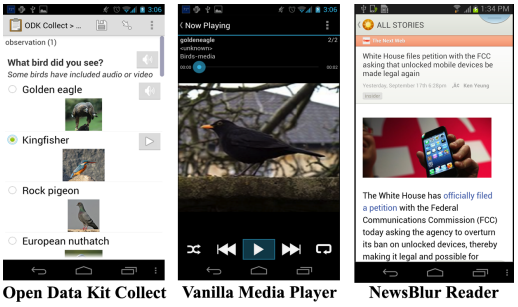


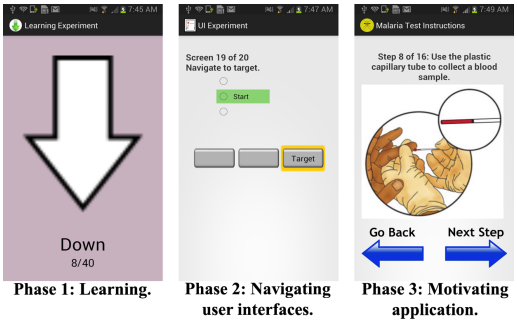
Figure 2: Screenshots from existing applications that we modified to enable touch-free interaction.

ODK Collect has thousands of users and is the preferred mobile data collection tool for many global health applications [6]. Users navigate through screens of the application using swipe gestures, enter data, and read text. Vanilla [17] is a popular media player that provides lists of albums, artists and songs and an interface for playing tracks. NewsBlur [12] is a personal newsreader that allows users to subscribe to news feeds and to read, tag and share stories. We did not attempt to make every module or activity in each application touch free. Instead, we focused on activities in which touch-free interaction would be most useful. For ODK Collect, we added touch-free interaction to the activities for navigating menus and lists, swiping through screens, and entering, editing and saving data. For Vanilla, we added touch-free interaction to the activities for browsing and selecting artists, albums and songs, and for playing and navigating tracks. For NewsBlur, we added touch-free interaction to the activities for browsing, reading and navigating articles.

The code to integrate Maestro into these applications was written by a second-year undergraduate who was familiar with Maestro but not with the target applications. Enabling touch-free interaction required a small amount of code to be added to each module or activity that will respond to touch-free gestures. This code can be divided into four tasks that were common to all activities across all applications: (1) system initialization of the camera (30 lines of code identical across all apps); (2) initialization to make UI elements "focusable" (2-20 lines per activity); (3) focus navigation required four routines for each of the directional gestures (14-20 lines of code); and, (4) target selection to handle dwell gestures (4 lines - identical across all activities). Although we realize that the amount of code added to each application does not necessarily translate to ease of integration, we wanted to include these measurements to provide readers with a rough indication of the effort required (*i.e.*, enabling touch-free interaction required tens of lines of code rather than hundreds or thousands of lines of code).

In summary, we added touch-free interaction to five activities in ODK Collect. These activities averaged 793 lines of code each without Maestro. We added an average of 66 lines of code to each activity, 34 of which were identical to all five activities. In addition, two sets of two activities shared the exact same code. We added touch-free interaction to two activities in Vanilla that averaged 1008 lines of code each

without Maestro. We added an average of 59 lines to each activity, 34 of which were common to both activities. Furthermore, the code added to one of the Vanilla activities was exactly the same as that added to two of the ODK Collect activities. Finally, we added touch-free interaction to three activities in NewsBlur that averaged 325 lines of code each without Maestro. We added an average of 50 lines of code to each activity, 34 of which were common to all activities. In addition, two NewsBlur activities shared identical code.



Phase 1: Learning. | Phase 2: Navigating user interfaces. | Phase 3: Motivating application.

Figure 3: Screenshots from our usability experiments: learning drills, UI navigation, and app use.

6. EVALUATION

We performed experiments to evaluate the responsiveness of the system as well as the energy and memory overhead added by Maestro. Our experiments were conducted using a Samsung Galaxy S3 device running Android v.4.1. All of the computation was performed locally on the device using a 1.4 GHz processor, 1 GB RAM and a 4.8 inch capacitive touchscreen. Touch-free gestures were detected using the built-in 1.9 mega-pixel forward-facing camera.

Responsiveness: We performed several experiments to quantify the responsiveness of the system to the user. The data for these experiments was collected by running Maestro on the device and recording all of the system timing data for five sets of 100 random touch-free gestures. Our data set thus consisted of the timing data for a total of 15,199 captured image frames and 500 touch-free gestures. The system is able to operate at 29 frames/second. Maestro's per-frame computation is 6.5ms including all image/motion processing to determine a gesture's start. An average of 184.5ms is spent detecting the end of a gesture but this is mostly due to human motion rather than computation. Only 0.2ms are needed to classify the gesture and trigger the application callback. Taken together, these findings indicate that the system is able to perform all of the computation required for gesture detection and classification in a fraction of the time that it takes a user to move a hand over the camera.

Energy: Since mobile devices are battery powered, energy consumption is a critical factor in assessing the viability of a system for practical, daily use. To compute Maestro's energy overhead, we measured the battery consumption of a system running Maestro and continuously processing image data for a period of four hours. We then measured the battery consumption of the same system without Maestro. In both cases, the device was fully charged prior to the start of the experiment. In addition, the screen was kept on for the duration of the experiment and no other applications were running. At the end of four hours of continuous processing, the system running Maestro had a remaining battery level of 28%, while the system without Maestro had a remaining battery level of 53%. It is important to note that these percentages represent the worst case energy consumption of Maestro. In most cases, the device would also be running other applications that would be consuming energy, and this would lower the percentage of battery usage due to Maestro. In addition, this experiment shows energy consumption resulting from continuously running Maestro for a long period of time. In reality, we expect that Maestro would be activated only when touch-free interaction is necessary, used for a short period of time, and then deactivated when it is safe for the user to touch the device again.

Memory Overhead: We ran each version of the application for four hours and computed the maximum resident set size in each case. Our findings show that Maestro imposes an 8% memory overhead when compared to the same application not using Maestro. On a mobile device with 1GB of RAM, we do not consider this to be of significance.

User Evaluation: Although we anticipate that touch-free interaction will primarily be useful in scenarios where it is undesirable or potentially harmful to touch the device, we wanted to provide readers with a way to understand the performance of touch-free interaction in comparison to a known point of reference: touch interaction. We expected that touch-free interaction would be slower than touch, but we wanted to quantify the difference to provide readers with context and insight into the usability of touch-free interaction. In addition, we were unable to identify a viable alternative touch-free system with which to compare Maestro. We conducted a controlled laboratory study with 18 participants to: (1) evaluate the speed and accuracy with which users learn to interact touch free; (2) evaluate how well people can use touch-free gestures to interact with user interfaces (UIs); and (3) understand whether people are able to navigate a realistic clinical application - processing a rapid diagnostic test for HIV - without touching the device. Participants completed the experiments while seated with the device flat on a table in front of them. They first watched a two-minute video introducing the gesture techniques and were given one minute to practice making gestures. They then completed three study phases: learning, UI navigation, and completing a clinical application touch-free (see Fig. 3).

Learning: As expected, participants made touch gestures significantly faster than touch-free gestures, 0.66 seconds for touch vs. 1.63 seconds for touch-free. Their accuracy as they learned the touch-free gestures increased leveling out at only 7.5% errors vs. 1% for touch. The learning task encouraged participants to make gestures as fast as possible and it was encouraging that, within minutes of learning the touch-free gestures, many participants were able to sustain speeds of over 40 touch-free gestures per minute. This finding suggests that touch-free gestures were intuitive and easy to learn.

Navigating: Participants took longer to navigate the UI screens using touch-free gestures than touch gestures, spending an average of 1.70 seconds per screen with touch and 6.67 seconds with touch-free. The magnitude of the difference (approximately 4x) between touch and touch-free gestures was expected and can be explained by two factors. First, successfully completing the tasks required participants to

make twice as many touch-free gestures as touch gestures. Second, as we saw during the learning phase each touch-free gesture takes roughly 2.5x as long as each touch gesture.

Using the App: For navigating the clinical application, we expected to see a smaller time difference between touch and touch-free interaction than in previous experiments for two reasons. First, participants needed to make only one gesture per screen for both methods. Second, rather than making gestures as fast as possible, the experiment required participants to read a sentence of text on each screen, which increased the task time for both methods. There turned out to be no detectable difference in the times taken to navigate the app using touch and touch-free gestures as the time was dominated by reading (doing the work) rather than by the speed of interaction.

Participant Feedback: A common theme voiced by many participants was *"I was surprised how well [touch-free gestures] worked. For how much this type of technology has been in sci-fi for years, I've never seen it actually used. I always suspected that's because it's not reliable, but it was."* Although this positive feedback is highly encouraging, further research is needed to assess the performance of the system in real global health settings. Integrating Maestro into health workers' clinical routines will likely present a range of additional challenges that will need to be addressed. Moreover, care will need to be taken to ensure that patients are comfortable with health workers using the system during clinical procedures and that patient privacy is properly protected.

7. CONCLUSION

It is important to note that there is no requirement for an application to choose between touch and touch-free interaction. Instead, applications can incorporate both gesture types and switch between them as circumstances warrant. Maestro's simplicity offers a number of advantages. Since the algorithm looks for movement alone (not color or shape), it can recognize any object, including a hand regardless of skintone or covering. Maestro's gesture set also allows novice users to quickly interact touch-free with minimal training. Finally, the computational simplicity of the algorithm makes the system highly responsive. However, the current implementation also has several limitations. For example, the gesture set is limited and does not provide an easy way to simulate more advanced multi-touch gestures like pinch. In addition, our current design is not well suited to text entry. Instead, touch-free interaction is better suited to tasks like scrolling and swiping, while text entry is better suited to touch interaction. Another limitation is that adding touch-free interaction to an application requires that the Maestro library be incorporated into the application's code. We are unable to interact touch-free with closed-source applications already installed on the device. Redirecting gesture input to these applications would require system-level changes that we have specifically avoided because of usage constraints for public health scenarios in the developing world.

Health workers in global health scenarios are increasingly using mobile devices to assist with a range of medical tasks. However, touching the device when performing these tasks may be undesirable or potentially harmful. We identified key design principles for a mobile touch-free system that targets these scenarios. We then presented Maestro, a vision-based system that recognizes in-air gestures using the built-in forward-facing camera on commodity mobile devices. We described the programming effort required to integrate touch-free functionality into three widely-used mobile applications. We also show that the system is capable of responding to users' gestures in real-time while imposing acceptable energy and memory overheads. Finally, we presented a user evaluation that shows people were able to quickly learn touch-free gestures and use them to complete a variety of tasks without touching the device. We conclude that Maestro is a practical tool that could allow users in global health scenarios to interact with commodity mobile devices touch free.

8. ACKNOWLEDGEMENTS

This work was funded by NSF grant 1111433. We also thank Leeran Raphaely and Jacob Wobbrock.

9. REFERENCES

[1] R. A. Bolt. "Put-that-there": Voice and Gesture at the Graphics Interface. In *Computer Graphics and Interactive Techniques*, pages 262–270, 1980.

[2] D. Breslauer, R. Maamari, N. Switz, W. Lam, and D. Fletcher. Mobile phone based clinical microscopy for global health applications. *Plos One*, 4(7), 2009.

[3] A. Butler, S. Izadi, and S. Hodges. Sidesight: Multi-"touch" interaction around small devices. In *User Interface Software and Technology*, UIST '08, pages 201–204, 2008.

[4] N. Dell, I. Francis, H. Sheppard, R. Simbi, and G. Borriello. Field Evaluation of a Camera-Based Mobile Health System in Low-Resource Settings. In *Human-Computer Interaction with Mobile Devices and Services*, 2014.

[5] C. Harrison and S. E. Hudson. Abracadabra: Wireless, high-precision, and unpowered finger input for very small devices. In *User Interface Software and Technology*, 2009.

[6] C. Hartung, Y. Anokwa, W. Brunette, A. Lerer, C. Tseng, and G. Borriello. Open Data Kit: Building Information Services for Developing Regions. In *Information and Communication Technologies and Development*, 2010.

[7] S. Kratz and M. Rohs. Hoverflow: Exploring around-device interaction with ir distance sensors. In *Human-Computer Interaction with Mobile Devices and Services*, 2009.

[8] S. Kratz, M. Rohs, D. Guse, J. Müller, G. Bailly, and M. Nischt. Palmspace: Continuous around-device gestures vs. multitouch for 3d rotation tasks on mobile devices. In *Working Conference on Advanced Visual Interfaces*, 2012.

[9] Leap Motion. https://www.leapmotion.com.

[10] C. Maggioni. A novel gestural input device for virtual reality. In *Virtual Reality Annual Symposium*, 1993.

[11] G. Murthy and R. Jadon. A Review of Vision-Based Hand Gestures Recognition. *Information Technology and Knowledge Management*, 2(2):405–410, 2009.

[12] NewsBlur. http://www.newsblur.com/.

[13] OpenCV. http://opencv.willowgarage.com/wiki/.

[14] M. Pouke, A. Karhu, S. Hickey, and L. Arhippainen. Gaze tracking and non-touch gesture based interaction method for mobile 3d virtual spaces. In *Australian Computer-Human Interaction Conference*, 2012.

[15] Samsung Galaxy S4. http://www.samsung.com/GalaxyS4.

[16] T. Starner, A. Pentland, and J. Weaver. Real-time american sign language recognition using desk and wearable computer based video. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(12):1371–1375, 1998.

[17] Vanilla Media Player. https://play.google.com/store/apps/details?id=ch.blinkenlights.android.vanilla.

[18] Z. Yang, Y. Li, Y. Zheng, W. Chen, and X. Zheng. An interaction system using mixed hand gestures. In *Asia Pacific Conference on Computer Human Interaction*, 2012.