GripSense: Using Built-In Sensors to Detect Hand Posture and Pressure on Commodity Mobile Phones

Mayank Goel[1], Jacob O. Wobbrock[2], Shwetak N. Patel[1]

[1]Computer Science & Engineering
DUB Group
University of Washington
Seattle, WA 98195 USA
{mayank, shwetak}@cs.washington.edu

[2]The Information School
DUB Group
University of Washington
Seattle, WA 98195 USA
wobbrock@uw.edu

ABSTRACT

We introduce *GripSense*, a system that leverages mobile device touchscreens and their built-in inertial sensors and vibration motor to infer hand postures including one- or two-handed interaction, use of thumb or index finger, or use on a table. GripSense also senses the amount of pressure a user exerts on the touchscreen despite a lack of direct pressure sensors by observing diminished gyroscope readings when the vibration motor is "pulsed." In a controlled study with 10 participants, GripSense accurately differentiated device usage on a table *vs.* in hand with 99.7% accuracy; when in hand, it inferred hand postures with 84.3% accuracy. In addition, GripSense distinguished three levels of pressure with 95.1% accuracy. A usability analysis of GripSense was conducted in three custom applications and showed that pressure input and hand-posture sensing can be useful in a number of scenarios.

ACM Classification: H5.2 [Information interfaces and presentation]: User Interfaces—*graphical user interfaces*.

General terms: Design, Human Factors, Experimentation.

Keywords: Touchscreen; situational impairments; mobile; inertial sensors; gyroscope; hand posture; posture

INTRODUCTION

A typical computer user is no longer confined to a desk in a relatively consistent and comfortable environment. The world's typical computer user is now holding a mobile device smaller than his or her hand, is perhaps outdoors, perhaps in motion, and perhaps carrying more things than just a mobile device. A host of assumptions about a user's environment and capabilities that were tenable in comfortable desktop environments no longer applies to mobile users. This dynamic state of a user's environment can lead to *situational impairments* [28], which pose a significant challenge to effective interaction because our current mobile devices do not have much awareness of our environments or how those environments affect users' abilities [33].

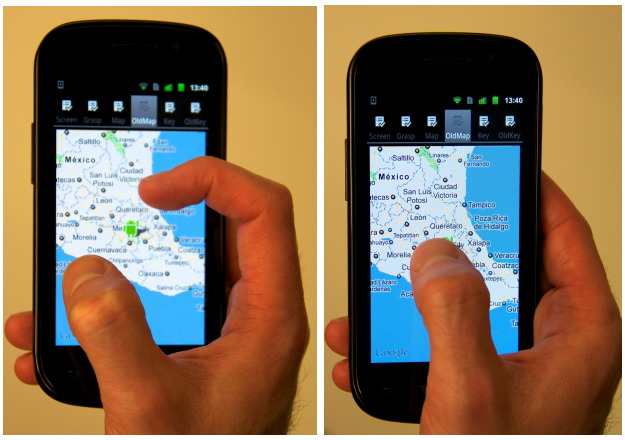

Figure 1. (left) It is difficult for a user to perform interactions like *pinch-to-zoom* with one hand. **(right)** GripSense senses user's hand posture and infers pressure exerted on the screen to facilitate new interactions like zoom-in and zoom-out.

One of the most significant contextual factors affecting mobile device use may be a user's hand posture with which he or she manipulates a mobile device. Research has shown that hand postures including grip, one or two hands, hand pose, the number of fingers used, and so on significantly affect performance and usage of mobile devices [34]. For example, the pointing performance of index fingers is significantly better than thumbs, as is pointing performance when using two hands versus one hand. Similarly, the performance of a user's dominant hand is better than that of his or her non-dominant hand. Research has found distinct touch patterns for different hand postures while typing on on-screen keyboards [1]. And yet our devices, for the most part, have no clue how they are being held or manipulated, and therefore cannot respond appropriately with adapted user interfaces better suited to different hand postures.

Researchers have explored various techniques to accommodate some of these interaction challenges, like the change in device orientation due to hand movement [2,15]. But despite prior explorations, there remains a need to develop new techniques for sensing the hand postures with

UIST '12, October 7–10, 2012, Cambridge, Massachusetts, USA.

which people use mobile devices in order to adapt to postural and grip changes during use.

In this paper, we present *GripSense* (Figure 1), a system that uses a combination of the touchscreen and the built-in inertial sensors (gyroscope, accelerometer) and built-in actuators (vibration motors) already present on most commodity mobile phones to infer hand postures and pressure. GripSense detects hand postures over the course of a small number of interaction steps (*e.g.*, tapping, swiping the screen). It infers postures like the use of an index finger, left thumb, right thumb, which hand is holding the device, or whether the phone is lying on a flat surface. GripSense performs this sensing by measuring a device's rotation, tap sizes, and the arc of swiping motions. GripSense additionally leverages the built-in vibration motors in a new way to help infer the amount of pressure being applied to the screen when interacting with the phone, which can be used to enable alternate interaction techniques with mobile devices that have no additional hardware for pressure sensing. As an example, GripSense allows users to zoom-in and zoom-out of maps using pressure input. In addition, GripSense is able to detect when the phone is being squeezed, which could be used to quickly silence a phone while in a pocket. Previous work on hand-posture detection has leveraged accelerometers for detecting whether a device is used in a stationary environment, in a hand, on a table, or in motion [27], and researchers have also used *external sensors* for grip detection [9,22,30]. Leveraging the built-in inertial sensors, vibration motors, and touchscreen for grip and pressure detection has not, until now, been explored.

We evaluated GripSense in a controlled study with 10 participants. Our findings show that GripSense differentiates between device usage in hand or on a flat surface with 99.7% accuracy and various hand postures with 84.3% accuracy and, on an average, makes a decision within 5 "interaction steps"—actions taken by the user that give GripSense information. Based on data collected in our controlled study, we built a three-level pressure detection model for inferring pressure input. GripSense differentiates between three levels of pressure with 95.1% accuracy. Lastly, we also developed three applications that 10 participants used in different settings to qualitatively evaluate the usefulness and usability of GripSense.

The main contributions of this paper are: (1) a new artifact called *GripSense* embodying multiple approaches to sensing mobile device hand-postures using only the built-in sensors present in commodity touchscreen devices; (2) empirical results from an evaluation of GripSense showing that it robustly detects four postures *i.e.*, single-handed operation with left thumb, with right thumb, two-handed operation with either index finger, and operation on a flat surface; and (3) the accurate sensing of three levels of pressure applied to the touchscreen using those hand postures.

RELATED WORK

Our work draws motivation from prior research on exploring solutions for making mobile device interactions easier in different situations and contexts as well as technologies that understand a user's interaction behavior to add new input capabilities to the device.

Situational Impairments and Hand Postures

It has been emphasized by a number of researchers that the devices need to have knowledge of a user's context or situation to provide better support to the user by making interfaces intelligent and invisible (*e.g.,* [18,27]). Recent design approaches have also emphasized this; for example, proactively sensing context is a design principle of *ability-based design* [33], which seeks a better match between interfaces and the abilities of the people who use them. There also has been extensive research in the domain of activity recognition to have a better understanding of the context of a user-in-motion. Choudhury *et al.* [3] developed a small wearable device with number of sensors for activity recognition. Laerhoven and Cakmakci [23] leveraged an accelerometer attached to a phone for recognizing different user motions like walking, climbing stairs, *etc.* Schmidt *et al.* [27] leveraged accelerometers to detect, in addition to user movement, whether a device is held in the hand, is on a table, or is in a suitcase. GripSense contributes to this research area by detecting in which hand and in which hand-posture a device is being used.

Prior to GripSense, others have also proposed techniques for detecting hand postures. Kim *et al.* [22] and Harrison *et al.* [9] used capacitive touch sensors to differentiate between numerous grips. Taylor and Bove [30] additionally leveraged accelerometers to dynamically detect changes in a user's grip for improved interactions. Our system, in contrast, requires no additional instrumentation of a modern smartphone to robustly detect handling grips. The main trade-off for this capability is that the user needs to be interacting with the device for GripSense to make inferences.

Understanding hand posture is important for making devices more intelligent to situational impairments caused by them. Holz *et al.* [16] have evaluated systematic error in target selection due to change in finger posture. Wobbrock *et al.* [34] studied a number of hand postures and evaluated front- and back-of-device finger performance with mobile devices. A number of researchers [14,21,31] suggest that although users prefer single-handed operation while using smartphones, traditional mobile interfaces are designed for two-handed operation. Karlson *et al.* [20] studied such interfaces and evaluated how they impede thumb-based usage. Azenkot and Zhai [1] found that different hand postures induced different touch patterns and affected overall performance while typing on a mobile touchscreen keyboard. AppLens and LaunchTiles [21] attempted to design interfaces mindful of the limited precision and range-of-motion of the thumb. GripSense, using only on-device sensors, robustly detects whether a user is using his left thumb, right thumb, or either index finger to interact with a device.

Inertial Sensors and Force Input

Inertial sensors like accelerometers and gyroscopes have become ubiquitous. Numerous researchers have leveraged these sensors to improve device performance. Joshi *et al.* [19] used them to reduce image blurring due to touch-induced vibrations while using a camera. WalkType [7] used on-device accelerometers to adapt smartphones' keyboards to users' walking movements to reduce text entry errors. Phielipp *et al.* [25] used accelerometers and the sequence of button presses to detect remote control user identity. GripSense derives motivation from such work and uses similar sensors to improve mobile device interaction.

There has been long and consistent interest in augmenting mobile devices with pressure input. Iwasaki *et al.* [17] measured typing pressure on laptop keyboards using on-device accelerometers. Pressure Widgets [26] used a pressure-sensitive stylus for adding pressure-based interactions to PDAs. Clarkson *et al.* [4] instrumented a flip-phone with force sensitive resistors (FSR) to infer continuous pressure applied by the user. Essl *et al.* [5] combined FSR with accelerometers and touch size, inferred from the touchscreen, as a proxy for pressure applied on phone. Force Gestures [12] added detection of tangential forces in a similar setup for richer interactions. Unlike GripSense, most of these efforts required additional device instrumentation.

As with GripSense, there is significant previous work that does not require custom instrumentation. Hinckley *et al.* [14] and Heo and Lee [13] used smartphone accelerometers to leverage touch-induced vibrations as a proxy for pressure. In contrast with our work, these approaches do not get a continuous measure of pressure applied by the user. They provide a coarse proxy of the pressure and only infer the initial velocity with which a user's finger strikes the screen. Although useful in a number of situations, the granularity and frequency of these inferences is limited. Heo and Lee [13] also found that the requirement of increased speed-of-contact results in higher target selection error. We believe this limitation does not impede GripSense.

Grasping and Squeezable Interfaces

Fitzmaurice *et al.* [6] coined the term "graspable user interfaces." Such interfaces allow devices to become more context-sensitive and thereby improve "expressiveness or the communication capacity" of the computer. SqueezeBlock [8] embodies this idea and demonstrates a device that provides feedback by varying its "squishiness." Wimmer *et al.* [32] leveraged computer vision and optical fibers to detect grasping pressure on mobile device surfaces. Harrison *et al.* [9] leveraged FSRs to detect squeezing pressure. These projects required additional device instrumentation to infer grip pressure. GripSense requires no additional hardware for its graspable user interfaces and is a completely software-enabled solution provided one has a commodity smartphone. Another software-only solution, one by Strachan and Murray-Smith [29], used muscle tremor as a proxy for pressure sensing in a squeezable interface. GripSense uses similar phenomenon and combines it with motor-induced vibrations to achieve more fine-grained estimation of pressure.

DESIGN OF *GRIPSENSE*

GripSense uses multiple sources of information to detect a user's hand posture and the amount of pressure exerted in a variety of these postures. Among these sources is the data from device's built-in gyroscope. In case of hand posture detection, the gyroscope is used to measure the direction and amount of rotation of the device in all three axes. For the detection of exerted pressure, the gyroscope is used to measure specific damping characteristics of touch- and motor-induced vibrations. Another source of information is touchscreen interaction data. In this section, we outline the concept and theory behind GripSense.

Inference	Features Used	Sensor Event	Latency
Table *vs.* Hand	Gyroscope (Low frequency in all axes)	Touch Down	1
Thumb *vs.* Index Finger	Gyroscope (Low frequency in *x*- and *y* axis	Touch Down	3
	Swipe Shape	Touch Up	
	Touch Size	Touch Down	
Left Thumb *vs.* Right Thumb	Gyroscope (Low frequency in *y*-axis	Touch Down	5
	Swipe Shape	Touch Up	
	Touch Size	Touch Down	
Pressure in hand	Gyroscope (Low Frequency)	Touch Down	1
	Gyroscope (High Frequency) + Motor		
Pressure on table	Gyroscope (High frequency) + Motor	Touch Down	1
Squeeze	Gyroscope (High frequency) + Motor	Held in Hand	0

Table 1. Summary of all inferences made by GripSense and when and which features were used for each of them.

Inferring Hand Posture

GripSense uses touchscreen interaction and device rotation information to infer whether the phone is (*a*) in a user's left hand and operated with left thumb, (*b*) in a user's right hand and operated with right thumb, (*c*) in either hand and operated with the index finger of the other hand, (*d*) on a flat surface, or (*e*) being only grasped by the user and not operated. Karlson *et al.* [21] discussed how limited precision and extent of the human thumb impedes one-handed mobile touchscreen interaction. We use this information in GripSense to detect hand postures. We use a combination of three features: (1) relative variance in rotation, (2) change in touch size, and (3) direction of arc for finger swipes. These features were extracted on a Samsung Nexus S smartphone running Android OS 2.3.

Rotation of the Device. The first feature is the rotational movement of the device as the user touches the screen. In a one-handed interaction, the phone rotates in response to touches at the top of the screen more than it does to touches at the bottom of the screen (Figure 2). This is to compensate for the limited range of the thumb; fingers move the device as the thumb extends to reach the top of the screen.

In contrast, touches at the bottom of the screen result in less angular motion because that area is usually within the thumb's range. When the user interacts using their index finger, there is no difference in the angular motion from touches at the top or the bottom of the screen. If the device is on a table then there is no change in any of these parameters before the touch event is registered.

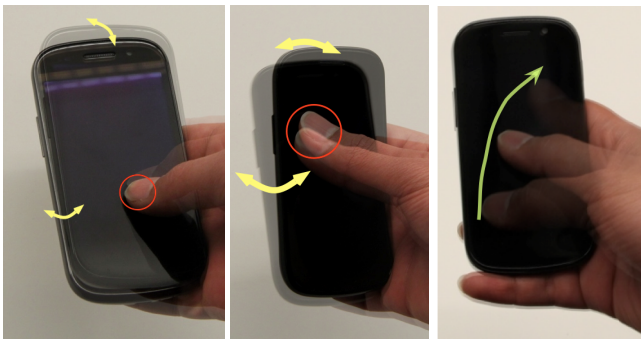

Figure 2. (left) Minimal device rotation in *x*- and *y*-axis, and smaller touch size when the user touches nearby with the thumb. **(center)** Significantly more rotation in *x*- and *y*- axis and larger touch size when the far quadrant of the screen is touched. **(right)** The shape of the swipe arc in the case of right thumb. (All of these phenomena are mirror-imaged for the left thumb.)

To leverage these insights, we store the angular velocities around the *x*-axis sampled at 1 kHz from the gyroscope in a quarter-second buffer. The data in the buffer is passed through a low-pass filter to isolate the low frequency angular velocities. We record the last two angular velocities observed for touches in the top third of the screen and the bottom third of the screen (determined from a pilot with four users). If the difference in variance of angular velocities for touches in the top is five times greater than for touches in the bottom of the screen, we assumed that it was thumb-based interaction.

If the difference in the variances does not exceed the threshold for three consecutive touches, then we bias our final decision towards selecting "index finger."

Similarly, when a user holds the phone in their left hand and interacts with their thumb, touches on the right of the screen cause more angular motion than touches nearer to the palm, again because of the compensation for the limited motion range of the thumb (Figure 2). In the case of the right hand, more motion is seen from touches on the left of the screen. If a thumb-based interaction is inferred, we use a similar approach as before, except now we log the variance in the *y*-axis of the gyroscope for touches on the left third of the screen and the right third of the screen. If the variance in angular velocity of the last two touches on the left side is greater than that on the right side, then we assume the phone is in the right hand (left hand if the variance on the right is greater). Moreover, if the difference in angular velocities is more than ten times greater in consecutive touches, we set a "high confidence flag" which is used to bias our final decision towards using this feature (discussed later).

Touch Size. The second feature is based on the change of size of touch in different regions of the touch screen. We hypothesize that in one-handed interaction when the user interacts with the left and right sides of the screen, the size of the touch changes because of the shape of the thumb and rotation of the device in the user's hand. The touch size on the same side as the thumb will be smaller than the touch size on the far side away from the thumb (Figure 2).

For this feature, we divide the screen into six (2×3) parts and keep track of last two touch sizes. Note that the Android platform provides a method to get the touch size on the screen. This method is supported by most Android smartphones available in the market. We compare touch sizes in the left third and right third of the screen for the same third of the screen height. If the difference in the mean of the touch sizes is more than 25%, we bias the system towards a thumb-based interaction. If the larger tap size is on the left side, then the system believes it is right thumb, and vice versa. Moreover, if the difference in touch sizes is more than 40% for consecutive touches, the heuristic sets a "high confidence flag." If the difference is less than 25%, it biases toward index finger-based interaction.

Shape of the Swipe Arc. This feature is only applicable when the user swipes on the screen. Because of the shape and position of the thumb, users often draw an exaggerated arc instead of a relatively straight line. Karlson *et al.* [21] observed similar arcs and analyzed how an interface can be more effective by limiting interaction within this arc. We use this arc as our "signal" to detect the user's hand posture. While using the phone with the index finger there is no consistent arc. However, with the thumb there is a consistent, exaggerated arc to the right or left depending on which thumb is being used. Figure 2 shows the arc formed by the right thumb while performing a bottom-to-top swipe. A mirror image of this arc will form in the case of the left thumb.

If the difference in coordinates of the start and end position of a vertical swipe are more than 5% of the screen resolution, GripSense biases itself towards one of the two thumb postures. Even so, we observed that sometimes a thumb-based swipe does not result in an arc. Instead, the phone experiences angular motion in the hand. For example, a right-handed swipe from bottom to top results in a counter-clockwise rotation. These two phenomena combine to form a robust heuristic for handling posture detection in the case of swipes. As with the other two heuristics, the final intra-heuristic decision is made when the system biases toward the same posture twice in a row.

Making the Final Decision. If swipes are present, we use majority voting on the output of each heuristic to decide the posture. If all three votes disagree, the posture is marked as "unknown." In the absence of swipe, a final decision is made only if both touch size and rotation heuristics agree or if the "high confidence flag" in one of the heuristics is set. If both heuristics come up with different decisions, then the system chooses the heuristic with a "high confidence flag."

If both confidence flags are set or no confidence flags are set with disagreement, the posture is set to "unknown."

Detecting Pressure Applied to the Touchscreen

GripSense uses the gyroscope and vibration motor to classify the user's touchscreen touches into three pressure categories: *Light, Medium* and *Heavy*. We hypothesize that if we trigger the built-in vibration motor when a user touches the screen (similar to what is already done in a number of smartphones to provide haptic feedback), the user's hand absorbs a portion of these vibrations. Our experiments show that this vibration absorption is proportional to the amount of pressure being applied to the screen (see Figure 3). This damping effect is measured using the on-device gyroscope. We primarily look for the damping of vibrations induced by the vibration motor. We also observed that as the amount of force exerted by the user on the touchscreen increases, there is a subtle oscillating motion between the user's thumb and the four fingers that rest on the back of the device (see the low pass signal in Figure 3). Strachan and Murray-Smith also observed and leveraged this phenomenon [29]. We hypothesize that this oscillation occurs because the user's thumb and fingers try to compensate continually for pressure exerted and this oscillation has much lower frequency compared to that induced by the vibration motor. This subtle motion is not dependent on the vibration motor. In order to make a robust classification of a user's touch intensity, we use both of these features.

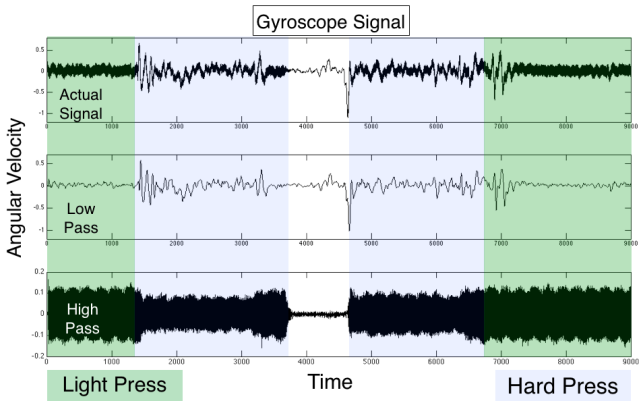

Figure 3. (top) Gyroscope signal when user presses light, then hard, then waits for a second and presses hard and soft again. **(middle)** The lower frequencies generated from touch-induced vibrations *increase* with increase in pressure. **(bottom)** Motor-induced vibrations are diminished as the amount of pressure exerted increases.

The touch-induced vibrations observed in GripSense are different from those used by prior work [13,14]. This prior work used exaggerated vibrations observed when a user "whacks" the phone with higher than usual velocity, moving the phone backward with respect to the finger stroke. This backward motion is proportional to the force with which finger strikes the screen. This technique provides an effective but coarse proxy of the pressure exerted. In contrast, GripSense leverages the subtle shaking of the phone as a user's thumb or finger (depending on the posture) and hand in which the phone is held try to compensate for pressure exerted by each other. An effective combination of these touch-induced vibrations with damped motor-induced vibrations give a much more authentic fine-grained and continuous proxy of pressure exerted on the screen.

We built a custom application on an Android Nexus-S smartphone, wherein any touch triggered the phone's built-in vibration motor. We then gathered angular velocities around the three axes through the built-in gyroscope with a 1 kHz sampling rate (Figure 3, top). Touch-induced vibrations were obtained by passing the signal through a low pass filter (Figure 3, middle). The motor-induced vibrations were obtained by passing the original signal through a high pass filter (Figure 3, bottom).

It is clear from the bottom plot in Figure 3 that in the case of a hard press (blue background), there is an exaggerated damping effect due to vibrations absorbed by the user's hand. We quantify this damping using the 90th percentile of the high-frequency component of the observed signal. For the low frequency signal, we quantify the movement of the phone using the signal variance.

Essl *et al.* [5] have earlier used the size of touch on the screen as a proxy for pressure exerted, and we also use this as a feature in our system. Our analysis shows that this feature alone is a poor measure of pressure (only about 60% accurate in our pilot study), but when combined with vibration analysis it can marginally improve performance by 1.4%, on average. Our pilot study also showed that amount of motor-induced vibrations absorbed by the hand and thumb or finger was also dependent on the location of touch on the screen. Hence, we divided screen into a 4×6 matrix in portrait mode and added "touch zone" as another feature for pressure level classification.

We buffer the gyroscope data at 1 kHz in a 500 ms buffer and analyze it every 250 ms (Figure 4). The data then passes through low pass and high pass filters and appropriate variances and 90th-percentiles are calculated. These features, along with touchscreen features (*zone* and *size*), were used to classify to pressure level using the Weka machine learning toolkit. Weka was used to generate J48 Decision Trees with pruning confidence set to Weka's default (0.25).

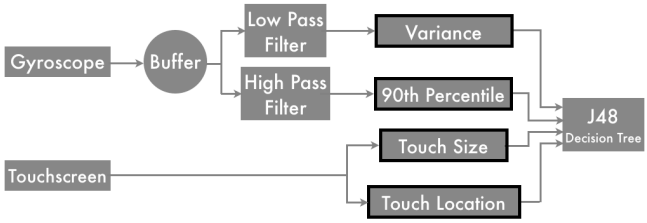

Figure 4. Block diagram of the major components of GripSense's pressure detection module. Low frequency variance, 90th percentile of higher frequencies, touch size and location are the features used for classification.

Squeeze and Grasp GesturesUsing similar techniques as for quantifying pressure exerted on a touchscreen, we implemented a method to detect squeeze or grasp gestures. For example, imagine quickly silencing a phone while it is still

in a pocket or in a purse by squeezing it and without the need for fully retrieving the phone. Although grasping provides a significant amount of damping to the motor-induced vibrations, there was no significant variance in low frequency component of the gyroscope data; therefore, only higher frequencies were analyzed and their 90th percentiles were used as features for Weka's J48 decision trees.

EVALUATION

Participants

The performance of GripSense was evaluated in a controlled study. Ten participants (6 males, 4 females) ranging in age from 21 to 32 years (M=26.9, SD=3.6) were recruited. All participants had more than 10 years of experience with computers and self-rated as intermediate to expert computer and smartphone users.

Apparatus

Participants used separate custom Android applications for posture and pressure detection. These applications were deployed on a Samsung Nexus S. The device's angular velocities were recorded at 1 kHz using the built-in gyroscope. The ground truth for pressure detection was obtained using thin-film force sensitive resistors (FSR), affixed to the touchscreen. This was only used to train users to exert different pressures while receiving feedback from the FSR; it was not used to train any of the algorithms.

Pressure Detection Procedure

Pressure Detection. The machine learning system for pressure detection was modeled and evaluated on data collected from participants in a 45-minute study each. Participants were asked to tap the screen using three different and discernible pressure levels with different hand postures detectable by GripSense: (1) thumb-based operation, (2) index-finger operation, (3) on a table, (4) only held by the user but not operated. Twenty taps for each pressure level using each posture were recorded for all participants. Apart from the taps, the participants were also asked to perform seven longer, continuous touches lasting 10 seconds each for each pressure level. These longer touches were recorded because in our pilot study we realized that low frequency touch-induced vibrations lasted for nearly 3 seconds and started attenuating thereafter as the hand became used to the pressure level. Hence we added these gestures in our data collection mode to make for more robust models.

The three different pressure levels were explained to participants and they were asked to practice them. The ground truth from the FSRs was visualized on the screen for feedback during training. Once participants were comfortable entering three distinct levels of pressure, they were introduced to the data collection interface and procedure. Although all participants were comfortable distinguishing for themselves three pressure levels, absolute pressure values were not uniform and varied across participants. The order of postures in which data was collected was randomized to prevent unwanted effects from fatigue.

The procedure for collecting data to evaluate grip pressure was the same as that for collecting touchscreen pressure. Participants were asked to grip the device with three varying intensities. The least pressure was slightly less than what participants would apply while using their phone in general. The middle grip pressure was meant to approximate how participants would normally hold their phones. In the highest-pressure level, participants were asked to grip their phones tightly.

The participants were asked to press a button to trigger the vibration motor. The motor went off 5 seconds after the button was pressed. This gave user ample time to get the phone into the correct grip. Then the device vibrated for 10 seconds. Five such tasks were run for each pressure level and for each participant. Although we collected pressure-based interaction data for our postures separately, using our posture-detection heuristics, applications can seamlessly switch between models for different postures at runtime.

Posture Detection Procedure and Applications

Our heuristics look at different touchscreen interactions like taps and swipes to infer grip; hence, we developed two separate custom Android applications to evaluate performance. One application, *Contact Selection App*, used more swipes than taps; the other, *Text Entry App*, used only taps. The 10 participants recruited for pressure detection data collection also participated in evaluation of these two applications. These applications were evaluated in a separate 30-minute session.

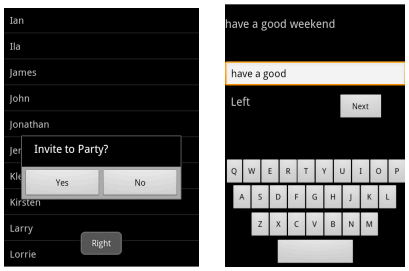

Figure 5. (left) Contact Selection App. Swipe-intensive application that helps to quantify swipe heuristic performance. **(right)** Text Entry App. Tap-intensive app helped in evaluating performance in absence of swipes. The left area of the screen just below the text field prompts the user with current hand posture.

The Contact Selection App presented participants with a list of 100 random names. The investigator asked each participant to select 50 of these names in a random order. Participants were asked to randomly invite some of the names to a fictitious party through a dialog box (Figure 5, *left*). After every 5 name selections, the app prompted the participant to switch postures. The order of postures was randomly generated. The list and dialog box ensured that participants performed a good combination of swipes and taps. The dialog box also ensured participants had a mixture of taps on both the left and right sides of the screen.

The Text Entry App required only tapping. Participants were presented with 15 short English language phrases randomly selected from MacKenzie and Soukoreff's phrase set [24]. At the end of each phrase, the application instructed participants to switch to a new randomly selected posture (Figure 5, *right*). The performance analyses for both of

these applications and also for pressure detection models are presented in the next section.

RESULTS

Posture Detection

Accuracy results for posture detection are shown in Figure 6, *left*. This figure shows the confusion matrix for the three hand postures while using the Text Entry App. The *y*-axis has the actual posture and *x*-axis has the prediction made by GripSense. We do not show the performance of detection of the device being on a table or grasped because it was relatively straightforward and the accuracy for that detection was 99.42%.

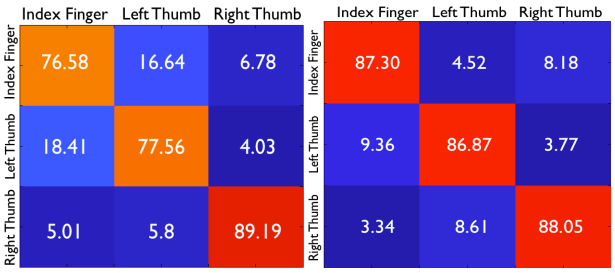

Figure 6. (left) Confusion matrix for classification of hand postures using Text Entry App. The *x*-axis shows the classification and the *y*-axis shows the actual posture. **(right)** Confusion matrix while using the Contact Selection App.

When evaluated using our Text Entry App, GripSense was able to detect correct posture in 81.11% cases. It made a decision after about 5 interaction steps. The number of interactions required to make a decision could be decreased but only at the cost of accuracy. With 5 required interaction steps, GripSense was able to make a decision often when the user completed the first word of a phrase.

The performance of GripSense, expectedly, improved further while using the Contact Selection App, as it permits the use of our third heuristic, the shape of the swipe arc. The accuracy improved to 87.4%. The confusion matrix for the three hand postures is shown in Figure 6, *right*. In this case, GripSense was able to make a decision on hand posture after about 4 interaction steps.

Pressure Detection

Figure 7 shows the accuracy of GripSense in detecting pressure exerted on phone in different postures. The participants were asked to use three levels of pressure and the overall accuracy across all postures was 95.05%.

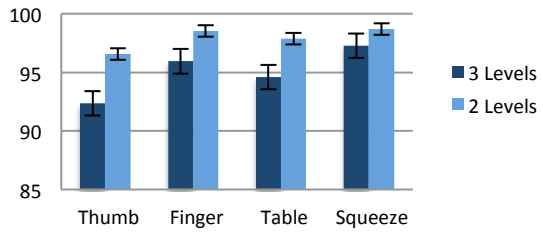

Figure 7. Squeeze gestures performed the best. Reducing the number of pressure levels to two improved accuracy further. Errorbars are standard error.

As discussed earlier, inclusion of Android's built-in touch-size method as a feature provided marginal improvement in performance. We performed an analysis to see how much effect this feature had on the performance of GripSense. The accuracy of GripSense without touch size was found to be 93.46%.

It may be that for many applications, two levels of pressure are adequate so we evaluated performance for two levels of pressure. As expected, the performance of GripSense's pressure detection improved significantly to 97.91%.

We discussed earlier how the touch-induced vibrations attenuate over the duration of the touch. So we collected data for both taps (momentary touch) and long 10 second touches as well. The long touches facilitate functions like zooming-in and out of pictures, maps, *etc.* During our pilot study we realized that different users have different ways of holding the phone and have different types of hands. Hence, we developed personalized pressure classification models for all participants. This meant that participants had to take part in a relatively long data collection study. So we investigated how much data was enough to train the system for acceptable levels of accuracy.

Figure 8 shows the progression of improvement in the average accuracy of GripSense to sense three distinct levels of pressure as we increase the amount of data used for development of models. The investigation has been divided into different accuracies for different tap and continuous gestures in variety of postures. The *x*-axis shows the percentage split in training and test data. Larger values on the *x*-axis signify more training data. It is clear from the figure that in the case of long continuous touches, GripSense does not require a lot of training data. Even for momentary taps, in total, only 20 taps were recorded for each pressure level. In most cases, the classification accuracy reaches above 80% by the 30% split mark, meaning that only six taps for each pressure level are required to train a device per user.

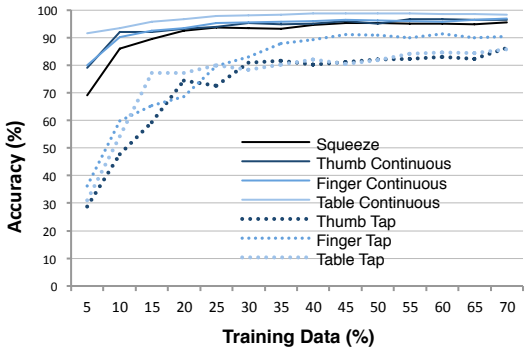

Figure 8. Improvement in average accuracy of pressure level detection for different gestures and touch-types with increasing size of training data. Continuous touches reached maximum performance with significantly less training data.

APPLICATIONS FOR PRESSURE DETECTION

To put GripSense through its paces, we implemented separate Android applications to take advantage of GripSense's posture- and pressure-sensing techniques. Our applications were motivated from the challenges associated with situa-

tional impairments [28] and the question of how interfaces may accommodate using only one hand, having one's device in a bag or purse, having limited screen space, *etc*.

The Maps Application

There are circumstances where it is difficult to use multiple fingers on a mobile device. Users often do not have both hands available for interaction. To explore these circumstances, we developed a map application in which user could zoom in by pressing harder on the screen, and could lightly press to zoom out. Hinckley *et al.* [14] have developed a similar application where they used hard and soft onset of taps for similar interactions. As noted previously, our approach differs because we do *continuous* pressure monitoring, not just detection of pressure at touch-onset, and therefore, our users do not have to impact the screen with increased velocity.

In order to assess the usability of our Maps app, we presented participants with a fully zoomed-out view of a world map and asked them to zoom-in to a set location (*e.g.*, Mexico). The participants needed to zoom in until they could see street names, and then they had to zoom-out until they could see the entire country again. To mimic a situational impairment, we required participants to hold a coffee mug in one hand and operate the device in their other hand. We also asked participants to perform same task on an app that did not have pressure-based input capabilities, and they had to use *pinch-to-zoom* to interact with the map using only one hand. After completion of a task on each system, we asked participants to make Likert-scale ratings based on the NASA TLX perceived workload index [10,11]. All 10 participants preferred using GripSense's pressure-sensitive maps, as it was much easier to navigate the map with one hand using pressure input to zoom in and out than to use one hand with pinch-to-zoom. On average, the GripSense-based app did better than traditional maps app on all counts on the Likert scale (Figure 9). A number of participants also liked the fact that the focal point of the zoom did not move while using pressure-based input, which is usually not the case with *pinch-to-zoom* implementations.

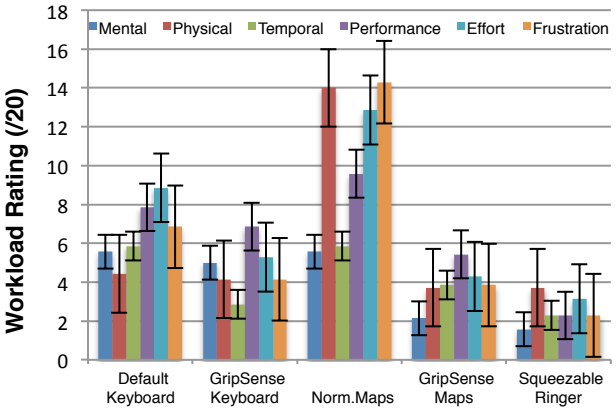

Figure 9. Perceived workload ratings show the GripSense-based applications resulted in relatively low workload. Lower ratings are better. Error bars are standard error.

The Keyboard Application

One obvious utility of having pressure-based input is alternative input. Researchers have used similar input modalities for mimicking right-clicks [14], changing keyboard modes [4], and so on. We use pressure input information to change letter case on a touchscreen keyboard. Users can press harder to enter uppercase letters and press lighter to enter lowercase letters. Participants were presented with 5 phrases randomly selected from the MacKenzie and Soukoreff phrase set [24]. Forty percent of characters in each of these phrases were randomly converted to upper case. We also presented participants with a parallel app having the same interface and task, with the only difference being an absence of pressure-based input. Instead, the app had a separate shift key for uppercasing letters.

The keyboards in both apps were modified to not show any typing errors if participants pressed within three keys of the intended key. We made this choice because the aim of this application was not to measure the user's typing accuracy, but to measure GripSense's pressure-detection accuracy. Participants were asked to use this app while holding the device in one hand and interacting with the index finger of the other hand.

After the completion of tasks on each of the two keyboards, we asked participants to fill out the same Likert scales as for the Maps app. The difference in performance of the two keyboards was not as dramatic as it was in case of the Maps app. Nonparametric Wilcoxon signed-rank tests indicate that our Keyboard application required significantly less perceived workload on the *temporal* and *frustration* Likert scales ($p<.05$). Responses for *effort* showed a trend in favor of our Keyboard application ($p=.07$). Responses for the *mental*, *physical*, and *performance* scales were not significantly different.

The participants were divided on which keyboard they thought allowed them to type faster. For example, P4 said, "[GripSense] felt like it took a lot less time to enter the text because I did not have to keep switching modes." Whereas, P7 said, "While using the pressure one I felt that typing was slower. I had to think more." A Shapiro-Wilk W test of normality indicates the *Time* measure differs significantly from normal ($W=0.95$, $p<.01$). However, a Kolmogorov's D test indicates the *Time* data does not depart significantly from lognormal ($D=0.07$, $p=.15$). Therefore, we log-transformed our *Time* measure before running a repeated measures ANOVA. Although the mean time taken for GripSense was a little less on average than the traditional Shift-based keyboard (14.98 s, $SD=5.29$ *vs.* 16.31 s, $SD=4.55$), the difference was not statistically significant ($F_{1,6}=2.69$, $p=.15$).

Squeeze Application

We developed a fake phone ringer app to check the utility of our squeeze gesture. This application plays a ringtone and goes into silent mode when user squeezes the device. This application was tested by asking participants to keep the phone in their pocket or bag and once the app starts

ringing, to reach inside the bag or pocket and squeeze the device. This squeeze action sends the device into silent mode and mimics the behavior of sending the caller to voicemail.

As is evident from the Likert-scale measures, participants frankly loved this application. Many wanted it on their personal phones immediately. P7 said, "There was something satisfying about squeezing the phone and having the vibration stop instantly." P1 said, "When can I get this? This is a really cool feature that I think would be very useful in all kinds of contexts."

DISCUSSION

GripSense successfully infers a user's hand postures and pressure exerted on device in these postures with high accuracy. But, there is slight degradation in the performance of GripSense when deciding between left-handed thumb-based operation and index finger-based operation. We believe this degradation is due to the system confusing one-handed and two-handed operation. This problem is mitigated to a large extent in the swipe-heavy application, Contact Selection App. In the case of the keyboard application, the user's interaction was largely limited to the bottom half of the screen and our current heuristics depend on analyzing differences in device movement when interacting with different parts of the screen. We believe this degradation would not be felt in real world applications that require users to interact with various parts of screen and hence provide much richer data for the algorithms presented in this paper. However, the keyboard app provides ample interaction switching between the left and right sides of the screen, so GripSense's performance is not dramatically affected.

Extended use of the touchscreen, gyroscope, and vibration motor can have significant power implications. But GripSense only leverages these sensors when the device is interacted with. We did not do a direct analysis of energy use, but anecdotally we did not observe any significant reduction in battery life during our user studies. If needed, future iterations of such system could employ a multistage approach by sampling at a low frequency first and then higher rates, as needed.

Because GripSense uses the vibration motor to sense the pressure exerted, the motor is triggered only when the user interacts with the touchscreen or in case of an infrequent event (*e.g.* incoming voice call). We explicitly asked participants about the effects of vibration on their experience with the system. The majority of participants did not feel that their experience deteriorated due to this vibration. Significantly lower levels of frustration in our exit survey after using GripSense-based applications also support this finding.

We implemented our algorithms on a Samsung Nexus S running the Android OS. Although the basic premise would remain the same, our pressure detection algorithms might need to be adjusted somewhat for different phones because of different physical characteristics. The variability of the sampling rate and resolution of different devices may also require algorithmic adjustments on some phones. Current inertial sensors present on commodity mobile devices are not high resolution and the techniques presented in this paper can benefit a great deal from improved resolution. The high performance exhibited by GripSense, particularly in the case of pressure detection, could be even better with improved sensor hardware in the future.

Our use of the built-in motor to produce vibration means that almost half of our features are coming from a relatively high-frequency source. Hence, techniques presented here for pressure detection do not suffer from the usual limitations of inertial sensor-based techniques like the presence of external sources of vibration, *etc.* Although no formal study was conducted to measure the effects due to external vibrations, an informal test was conducted to estimate the efficacy of pressure sensing while sitting as well as walking; results were comparable for both postures. In the case of posture detection, the combination of inertial (gyroscope) and non-inertial (touchscreen) sensors should help mitigate this issue.

As demonstrated by our results, the combination of touch-induced and motor-induced vibrations means that these techniques can be reliably implemented when the device is on a flat surface. Hence these algorithms can be ported to tablets as well, which are used relatively more on a desk when compared to a smart phone. Modern game controller manufacturers can also leverage these techniques with a simple software upgrade to add pressure sensitivity to their devices, as game controllers already have vibration motors and inertial sensors.

In our evaluation of pressure sensing in a variety of postures, we only had three levels of pressure. Three levels were chosen to make it easy for users to discern different pressure levels with acceptable levels of accuracy. We believe our algorithms actually can infer more than three levels of pressure, amply demonstrated by the fact that even though the range of pressure applied by different participants was different, the system maintained high levels of accuracy. That said, a more continuous regression to pressure is possible and algorithms can be built on top of this work that have more than just quantized levels of pressure.

CONCLUSION

The dynamic usage environments of mobile devices can lead to situational impairments that may be overcome with better device awareness and enhanced interaction techniques. In this paper, we presented GripSense, a system that leverages various capabilities of mobile devices like the touchscreen, inertial sensors, and the vibration motor to infer users' hand postures and the amount of pressure exerted on device in these postures. GripSense differentiates between device usage in-hand or on a flat surface with 99.7% accuracy and various hand postures with 84.3% accuracy and, on average, makes a decision within 5 "interaction steps". GripSense differentiates between three levels of pressure on different areas of the device with 95.1% accu-

racy. A controlled study with 10 participants qualitatively evaluated the desirability of GripSense with the help of three custom applications. Users reported lower perceived workload ratings for GripSense-based applications than for conventional alternatives. GripSense represents an important step in extending the capabilities of our mobile devices to be more aware, responsive, and useful.

ACKNOWLEDGEMENTS

We thank Peter Hornyack for his help in prototyping.

REFERENCES

1. Azenkot, S. and Zhai, S. Touch Behavior with Different Postures on Soft Smartphone Keyboards. *Proc. Mobile HCI 2012*, (2012).
2. Cheng, L.-P., Hsiao, F.-I., Liu, Y.-T., and Chen, M.Y. iRotate: Automatic Screen Rotation based on Face Orientation. *Proc. CHI 2012*, (2012).
3. Choudhury, T., Consolvo, S., Harrison, B., et al. The Mobile Sensing Platform: An Embedded Activity Recognition System. *Pervasive Computing, IEEE 7*, 2 (2008), 32-41.
4. Clarkson, E.C., Patel, S.N., Pierce, J.S., and Abowd, G.D. Exploring Continuous Pressure Input for Mobile Phones. .
5. Essl, G., Rohs, M., and Kratz, S. Use the Force (or something) - Pressure and Pressure- Like Input for Mobile Music Performance. *Organised Sound*, June (2010), 15-18.
6. Fitzmaurice, G.W., Ishii, H., and Buxton, W.A.S. Bricks: Laying the foundations for graspable user interfaces. *Proc. CHI 1995*, ACM Press/Addison-Wesley Publishing Co. (1995), 442-449.
7. Goel, M. and Findlater, L. WalkType: Using Accelerometer Data to Accommodate Situational Impairments in Mobile Touch Screen Text Entry. *Proc. CHI 2012*, (2012).
8. Gupta, S., Campbell, T., Hightower, J.R., and Patel, S.N. SqueezeBlock: using virtual springs in mobile devices for eyes-free interaction. *Proc. UIST 2010*, ACM (2010), 101-104.
9. Harrison, B.L., Fishkin, K.P., Gujar, A., Mochon, C., and Want, R. Squeeze me, hold me, tilt me! An exploration of manipulative user interfaces. *Proc. CHI 1998.*, ACM Press (1998), 17-24.
10. Hart, S.G., California, M.F., and Staveland, L.E. Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research. .
11. Hart, S.G. Nasa-Task Load Index (NASA-TLX); 20 Years Later. *Proc. of the Human Factors and Ergonomics Society Annual Meeting 50*, 9 (2006), 904-908.
12. Heo, S. and Lee, G. Force gestures: augmented touch screen gestures using normal and tangential force. *Proc. CHI EA 2011*, ACM (2011), 1909-1914.
13. Heo, S. and Lee, G. Forcetap: extending the input vocabulary of mobile touch screens by adding tap gestures. *Proc. Mobile HCI 2011*, ACM (2011), 113-122.
14. Hinckley, K. and Song, H. Sensor synaesthesia: touch in motion, and motion in touch. *Proc. CHI 2011*, ACM (2011), 801-810.
15. Hinckley, K., Pierce, J., Sinclair, M., and Horvitz, E. Sensing techniques for mobile interaction. *Proc. UIST 2000*, ACM (2000), 91-100.
16. Holz, C. and Baudisch, P. Understanding touch. *Proc. CHI 2011*, ACM (2011), 2501-2510.
17. Iwasaki, K., Miyaki, T., and Rekimoto, J. Expressive typing: a new way to sense typing pressure and its applications. *Proc. CHI EA 2009*, ACM (2009), 4369-4374.
18. Johnson, P. Usability and mobility; interactions on the move. *First Workshop on Human-Computer Interaction with Mobile Devices.* (1998).
19. Joshi, N., Kang, S.B., Zitnick, C.L., and Szeliski, R. Image deblurring using inertial measurement sensors. *ACM Transactions on Graphics 29*, 4 (2010), 30:1-30:9.
20. Karlson, A.K. and Bederson, B.B. *Understanding Single-Handed Mobile Device Interaction*. 2006.
21. Karlson, A.K., Bederson, B.B., and SanGiovanni, J. AppLens and launchTile: two designs for one-handed thumb use on small devices. *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM (2005), 201-210.
22. Kim, K.-E., Chang, W., Cho, S.-J., et al. Hand Grip Pattern Recognition for Mobile User Interfaces. *Proc. AAAI'06*, (2006).
23. Van Laerhoven, K. and Cakmakci, O. What shall we teach our pants? *Wearable Computers, The Fourth International Symposium on*, (2000), 77-83.
24. MacKenzie, I.S. and Soukoreff, R.W. Phrase sets for evaluating text entry techniques. *Proc. CHI EA 2003*, ACM (2003), 754-755.
25. Phielipp, M., Lee, R., and Hightower, J. Fast , Accurate , and Practical Identity Inference Using TV Remote Controls. *Artificial Intelligence*, (2010), 1827-1832.
26. Ramos, G., Boulos, M., and Balakrishnan, R. Pressure widgets. *Proc. CHI 2004*, ACM (2004), 487-494.
27. Schmidt, A., Aidoo, K., Takaluoma, A., Tuomela, U., Van Laerhoven, K., and de Velde, W. Advanced Interaction in Context. In H.-W. Gellersen, ed., *Handheld and Ubiquitous Computing*. Springer Berlin / Heidelberg, 1999, 89-101.
28. Sears, A., Lin, M., Jacko, J., and Xiao, Y. When Computers Fade Pervasive Computing and Situationally-Induced Impairments and Disabilities. *HCI International 2*, (2003), 1298-1302.
29. Strachan, S. and Murray-Smith, R. Muscle Tremor as an Input Mechanism. *Proc. UIST 2004*, (2004).
30. Taylor, B.T. and Bove Jr., V.M. Graspables: grasp-recognition as a user interface. *Proc. CHI 2009*, ACM (2009), 917-926.
31. Weberg, L., Brange, T., and Hansson, A.W. A piece of butter on the PDA display. *Proc. CHI EA 2001*, ACM (2001), 435-436.
32. Wimmer, R. FlyEye: grasp-sensitive surfaces using optical fiber. *Proc. of TEI'10.* ACM (2010), 245-248.
33. Wobbrock, J.O., Kane, S.K., Gajos, K.Z., Harada, S., and Froehlich, J. Ability-Based Design: Concept, Principles and Examples. *ACM Transactions on Accessible Computing 3*, 3 (2011), 9:1-9:27.
34. Wobbrock, J.O., Myers, B.A., and Aung, H.H. The performance of hand postures in front- and back-of-device interaction for mobile computing. *International Journal of Human-Computer Studies 66*, 12 (2008), 857-875.