TapSkin: Recognizing On-Skin Input for Smartwatches

Cheng Zhang, Abdelkareem Bedri, Gabriel Reyes, Bailey Bercik
Omer T. Inan, Thad E. Starner, Gregory D. Abowd
chengzhang, akareem.bedri, greyes, baileybercik, thad, abowd@gatech.edu; omer.inan@ece.gatech.edu
Georgia Institute of Technology

ABSTRACT
The touchscreen has been the dominant input surface for smartphones and smartwatches. However, its small size compared to a phone limits the richness of the input gestures that can be supported. We present TapSkin, an interaction technique that recognizes up to 11 distinct tap gestures on the skin around the watch using only the inertial sensors and microphone on a commodity smartwatch. An evaluation with 12 participants shows our system can provide classification accuracies from 90.69% to 97.32% in three gesture families – number pad, d-pad, and corner taps. We discuss the opportunities and remaining challenges for widespread use of this technique to increase input richness on a smartwatch without requiring further on-body instrumentation.

Author Keywords
Smartwatch interaction; acoustic sensing; inertial sensing; machine learning; input technology

ACM Classification Keywords
H.5.m. Information Interfaces and Presentation (e.g. HCI): Miscellaneous

INTRODUCTION
Most wearable devices such as the smartwatch, by necessity, are relatively small compared to traditional computing devices (e.g., laptop, smartphone). Input technologies for traditional computing devices cannot be easily replicated on wearable devices because of its size and form factor disparity. For instance, wearing an additional keyboard, even an optimized one [17], is inconvenient for most users and may limit the functionality of the hand. Therefore, providing an input technique for wearable devices that is both lightweight and always available is of much interest to the HCI community.

As a modern representative of wearable devices, the smartwatch uses a built-in capacitive multitouch screen as the surface for both input and output. Unfortunately, the relatively small size of the screen limits the richness of interactions.

ISS'16, November 06-09, 2016, Niagara Falls, ON, Canada

DOI: http://dx.doi.org/10.1145/2992154.2992187

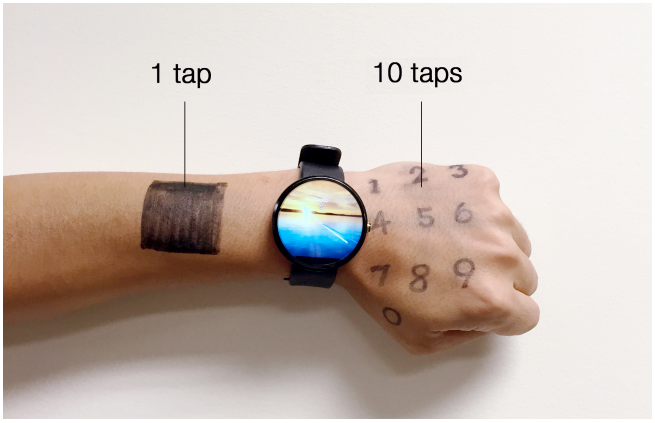

Figure 1. The TapSkin interaction technique. Drawings on the skin are only for clarity to indicate where input events can be performed.

Many successful interactions on smartphones cannot be reproduced on smartwatches, such as multitouch gestures. Furthermore, these interaction issues, such as a large finger occluding parts of the screen, may seem insignificant on a smartphone, but are exaggerated on a smartwatch.

These issues motivate us to look at the area surrounding the smartwatch, specifically the skin on the back of the hand, as a potential input surface. By extending the input outward to the skin, the user can view the screen while interacting, similar to the indirect interaction provided by a trackpad on a laptop. Although approaching the skin as the input surface has been previously explored [12, 29], those approaches have exploited additional instrumentation of the body to achieve the interaction. We demonstrate providing a rich set of tapping gestures that work with the current sensing capabilities of a smartwatch, specifically its microphone and inertial sensors.

We present TapSkin, a novel interaction technique that provides a family of tap gestures (up to 11 tap locations) on the skin around the wrist area without additional instrumentation, by only using the inertial measurement unit (IMU) and the microphone on a commodity smartwatch. Our approach offers a more practical solution for detecting on-skin tap gestures for a wrist-mounted device.

We provide the following contributions in this paper:

- The design and implementation of a family of tap gestures on the skin around the wrist area with the IMU and microphone on a smartwatch.

- An evaluation of the tapping gestures with 12 participants in a lab-based environment.
- User-dependent, independent, and user adaptive models are built to evaluate the challenges of personalizing on-skin gesture models for different participants.
- An extensive discussion of the challenges for the future practical deployment of this technique.

RELATED WORK

Though smartwatch and wearable devices have only started becoming popular in the past few years, providing interactions for portable computers has been explored by the community for many years. Therefore, we will first review the relevant work on smartphones and then discuss research about novel interactions on the wearable devices and smartphones. We conclude by examining past work that has used the skin as an input surface.

Interaction for smart phones

Compared with the traditional computers (e.g., desktops or laptops), the smartphone has a relatively small screen and is usually in the possession of its owner. Therefore, they share many similar interaction design challenges with the smartwatch.

Starting in the early 2000's, researchers have explored extending interactions on smartphones [13]. Some of the research focused on extending the input experience on the touchscreen by designing new gestures [26], improving the touch input accuracy [4], and detecting touch pressure [9]. Other researchers extended the interactions beyond the touchscreen, by providing novel motion gestures [27], or recognizing gestures on the back or the side of the case [2, 19, **?**].

Interaction for wearable devices

Early explorations of input for wearable computing used an additional wearable device with physical buttons to provide text input (e.g., Twiddler [18]). These methods were able to provide reliable and fast input with training. More recent work has explored instrumenting different parts of the body, such as the fingers [5, 21, 34] or the arm [6, 12, 28] to provide a more limited input vocabulary, typically a small set of discrete gestures. However, all these works require the user to wear an additional device on the body, which is inconvenient and impractical for always-on devices such as smartwatches.

Interaction for smartwatches

As wrist-mounted devices (e.g., health activity monitors and smartwatches) dominate the wearable device market, more interest has been raised on improving the interactions with them, especially the input on the smartwatch. The approaches used have varied from increasing the size of the touchscreen by adding additional screens around the wrist [16], to using an active stylus which allows finer manipulation than the fingers [31], or designing specific finger/hand gestures [3, 33, 30, 23].

Different parts of the watch have also been appropriated for interaction to provide a richer input vocabulary. Perrault [24] and Funk [8] turned the watch band into an input surface for gestures and text input. Nenya [1] used a magnetic ring to provide gestures for the smartwatch by detecting a changing magnetic field. Xiao [32] showed the feasibility of using the screen as a joystick to provide more input events. WatchOut [36] extends the interaction to the case and band of the watch with only IMU sensors.

The input space for a smartwatch can also be extended by considering the larger area around the watch. Kim *et al.*[14] demonstrated a gesture watch using an array of IR proximity sensors to detect intuitive and gross hand gestures above the watch face.

Approaching the skin as the input surface

Though not demonstrated in the context of wrist-worn device interaction, Harrison *et al.*'s demonstration of OmniTouch [11] showed how depth sensing can be used to enhance interaction around the hand and wrist, and specifically using the skin as the interaction surface. Indeed, a person's skin is a good alternative input surface for wearable devices[29], because of its proximity to the smartwatch.

Obviously, defining the skin as the input surface has great potential in the space of gesture design. To recognize gestures on the skin, a variety of sensing modalities have been explored, including capacitance [25], vision [11], muscle activation via electromyography [28], infrared proximity sensors [15], motion/inertial and force [6], electricity[37] as well as sound [7, 12, 20, 22].

These solutions all require the user to wear additional device to accommodate the input technique, which is not ideal. Our solution, TapSkin, will demonstrate the feasibility of using a combination of sensors already in a smartwatch and a smartphone to detect a family of on-skin gestures, providing a more practical solution.

DESIGN AND IMPLEMENTATION OF TAPSKIN

TapSkin is demonstrated on commodity smarwatches (Sony SmartWatch3, Moto 360) using available sensor data from the gyroscope, accelerometer, and microphone. A Google Nexus 6 phone is used to running the recognition algorithm. We first explain the intuition behind moving from skin taps to recognized gestures in TapSkin. Then, we discuss the design of our TapSkin gesture sets. Lastly, we present details of the gesture recognition implementation we used for our validation experiments.

Theory of Operation

When touching the skin around the wrist, sound propagates from the point of contact towards the microphone in the smartwatch. We observed that the sounds of tapping on different locations of the hand are distinguishable from each other. As Figure 2 shows, tapping on the number 3 (depicted as N3 on the top right of Figure 2 and shown in Figure 1) would have more energy at higher frequencies compared with the tapping event on the number 1 (depicted as N1 on the top left of Figure 2 and shown in Figure 1). This is because the thickness of the tissue, the ratio of fat, and the structure of the bone underneath are all different at different locations along

the back of the hand and along the top of the forearm. For instance, the position of number 3 is closer to the knuckle, which has more bone underneath. Therefore, the sound of a tap at that position has more energy at lower frequencies.

In addition, the act of tapping will induce a movement of the arm that the smartwatch is mounted on. As Figure 3 shows, tapping on different locations on the back of the hand, would introduce different kinds of movement. For instance, tapping on the number 7 and 9 (graphs N7 and N9 in Figure 3) would first have a positive peak on the x-axis of the gyroscope and then a negative peak, while tapping on the number 1 and 3 (graphs N1 and N3 in Figure 3) would first generate the negative peak along the y-axis and then a positive peak.

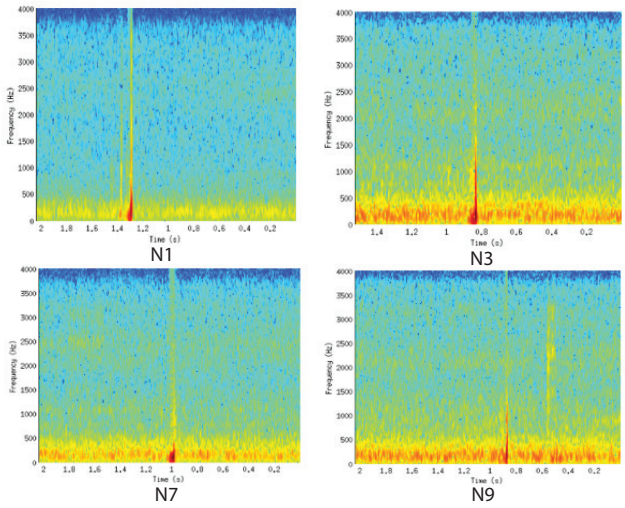

Figure 2. Comparing the acoustic response of taps at different location around the smartwatch.

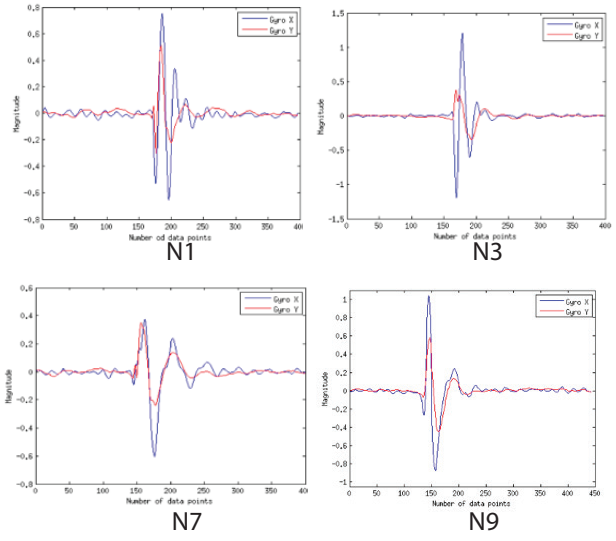

Figure 3. Comparing gyroscope sensor response of taps at different locations around the smartwatch.

These visually apparent differences across the various sensing modalities strongly suggest that with appropriate data acquisition and analysis, we should be able to distinguish different locations of a tap on the skin around the smartwatch. Therefore, we choose to use the gyroscope and accelerometer to characterize arm displacement and the microphone to record the acoustic response of the tap.

Gesture Design and Applications

With this intuitive and theoretical motivation, we first consider three sets of tapping gestures that can map into useful input: the NumPad, the DPad, and the CornerPad. Figure 4 depicts these three gesture sets. For each set, there is also a single tap on the top of the forearm beyond the smartwatch (depicted as a gray rectangular patch).

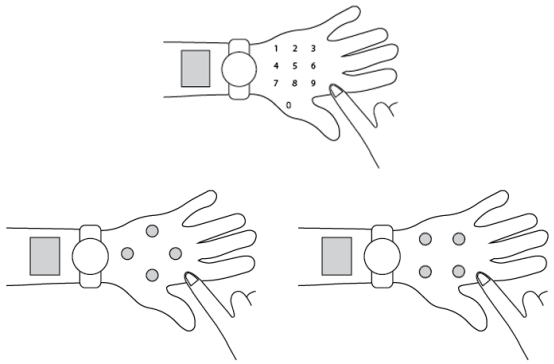

Figure 4. The TapSkin gesture sets: NumPad (top), DPad (bottom left), and CornerPad (bottom right).

NumPad

The NumPad gesture family simulates the layout of a number pad, which provides 10 distinct tap locations on the back of the hand and one tap location to the left side of the watch (as seen in Figure 4). It can potentially be used for entering numbers or even text (through multi-tap or T9), which is currently difficult on a small touch screen. The tap gesture on the left of the watch (TapLeft) is designed for two purposes: one is to serve as the activation gesture, informing our recognizer when it should pay attention for skin gesture input. The other is to undo the last gesture, such that the user can revise their input.

DPad

As Figure 4 shows, DPad contains the TapLeft gesture as well as four tap gestures located at the top, bottom, left, and right parts of the back of the hand. It can be used for directional controls or scrolling through a list of items on the smartwatch. For instance, the most common operation on a smartwatch touchscreen is to scroll down the menu vertically and horizontally. This operation can be naturally mapped to the four tap gestures using DPad. The TapLeft gesture is used for activation as well as for confirming the selection of a menu item or application.

This main advantage of the TapSkin DPad will be seen with applications that display a lot of information. DPad will reduce occlusion while still providing an intuitive means of interaction.

CornerPad

The CornerPad includes the four corners on the back of the hand. The two gestures on the left are close to the wrist joint, and the other two gestures on the right are close to the knuckles at the base of two fingers. These four distinct tap gestures can be used as shortcuts to certain applications on a smartwatch. The TapLeft gesture again serves as the activation gesture into CornerPad.

Gesture Recognition

We apply a traditional machine learning pipeline to recognize the gestures, which consists of two steps: gesture event detection and gesture classification. To identify events, we first apply a sliding window of 1.2 seconds with 75% overlap on the synchronized data streams of the gyroscope, accelerometer, and the microphone. Within each window, we detect whether there is a gesture event. If a gesture is detected, we pass the event window for feature extraction, which is then passed to a Support Vector Machine (SVM) to classify the gesture events. We use the sequential minimal optimization (SMO) of SVM provided by Weka for building our training models and for real-time classification in our prototype system [10].

Event Detection

To detect whether there is a gesture event in each 1.2 second sliding window, we inspect the window to determine if either the gyroscope or accelerometry data passes an empirically-determined threshold value based on the maximum energy in the window. If one of the thresholds is satisfied, we continue to the next step in the pipeline to define an *event window*.

Within the original sliding window used for event detection, we locate the position of the maximum absolute value of each sensor signal. Note that the max points may be at different times for each sensor. We then define an event window of data values for each sensor stream balanced around that maximum point. The size of event window is 0.5 seconds both IMU data (accelerometer and gyroscope) and acoustic data. If the maximum point is too close to the edge of the original sliding window, we advance into the previous or next sliding window. We now assume there is a gesture within that event window, so the raw data is passed along for feature selection.

Feature extraction

For each event window, we calculate a vector with 286 features.

For each axis of the linear acceleration and the gyroscope data, we first derived virtual sensors by taking the derivative of each axis. Then for each axis of the raw sensor and virtual sensors, we extracted a set of statistical features including the maximum, minimum, mean, median, standard deviation, root-mean-square(RMS), variance, zero-crossing rate, and the values of peaks and their differences. To capture the relationship between the different axes of each IMU sensor, we also calculate the ratio and differences of the energy, first and second absolute peak values of each two axes on each sensor. We also perform a Fast Fourier Transform (FFT) on the gyroscope data (50 points), which results in 24 features (abandoned the first bin) for each axis. Since the frequency of our gesture is under 6kHz, we only add the values of first three bins of the FFT results and the entropy across the values of different bins into a feature vector.

For the acoustic data, we extracted features in the frequency domain. We divided the window into 30ms frames with 50% overlap. We first extract 26 Mel-frequency cepstral coefficients (MFCC) features, based on the observation that the sounds generated by these on-skin gestures are mostly under 4kHz, which is similar to the range of the human voice. For each frame of data, we also calculate the FFT and average the results across different frames. We only keep the first 30 values of the FFT results, which represent the magnitude of frequencies between 0Hz to 500Hz, which we determined empirically as the most informative frequency band for the sound of our gestures. The center of mass across the frequencies is also added to the feature vector.

The feature vector is used to train a support vector machine to recognize which gesture has been performed.

EVALUATION

We conducted a user study to understand how our technique would work across different people and how we could build optimal machine learning models for each gesture set. We collect gesture examples first and apply offline analysis to build our models. We evaluate the system offline instead of using a real-time prototype. By collecting gesture examples first and performing an offline analysis, we can better explore the classification problem. At present, there is a practical limitation for running the data processing pipeline in real-time on the watch, so we offload that computational task to a paired smartphone. Therefore, the sensor data needs to be transmitted from the watch to the phone. However, continuously transmitting acoustic data via Bluetooth in real-time is limited by the bandwidth between the mobile phone and smartwatch. The data transmission would introduce a non-negligible latency while the system is running the real-time. This is the current limitation of the system implementation. However, as the processing power of the smartwatches are expected to increase over time, we would expect the whole system to run exclusively on the watch in the near future.

Apparatus

We evaluated our technology on a Sony SmartWatch 3 (Android Wear) for data collection and a Nexus 6 (Android 5.1) for running our machine learning algorithm. We specify the highest sampling rate available for the gyroscope (200Hz) and the linear acceleration (250Hz), but we intentionally lower the sampling rate for the microphone to 8kHz, as the sound generated by the tap event on the skin is mostly under 4kHz, and the lower sampling rate reduces power consumption. The lower sampling rate also minimizes the data transmission time between the watch and the phone, reducing the latency for eventual real-time classification results.

Study Design

To evaluate TapSkin, we recruited 12 participants (6 male) from our university campus with an average age of 26 years.

The study was conducted in an open space at a university building, where the sound from the air conditioner as well

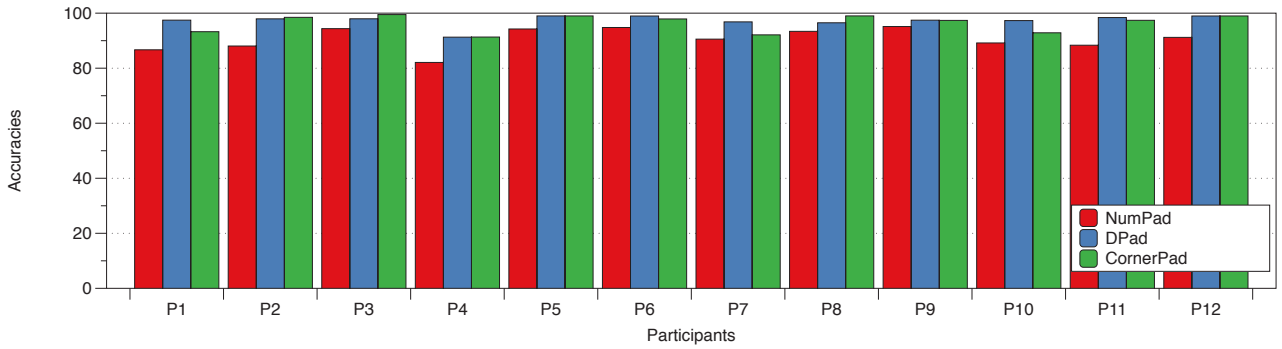

Figure 5. The classification accuracy for each participant

as people's talking and movement can be heard. Each participant was asked to complete surveys before and after the study gauging their experience. At the beginning of the study, we asked the participants to wear the watch on the left wrist and use the right hand to perform the gesture. We advised them to wear the watch closer to the hand and adjust the band tightly but comfortably on the wrist. For consistency and helping participants remember the locations of each tap gesture, a number pad as seen in figure 1 was drawn on the back of each participant's hand.

A researcher then demonstrated the entire gesture set and asked the participant to practice. The formal study consisted of 8 sessions. During each session, the participant performed all the gestures while sitting in front of a table, where they could rest their elbow during the study. The participants were asked to take breaks for a few minutes between sessions to reduce fatigue. During these breaks, they could walk around or take the watch off if desired. The whole study lasted approximately 45 minutes on average.

During each session, they were asked to follow the stimuli displayed on the screen of the watch to perform the NumPad gestures. DPad and CornerPad are a subset of the NumPad gesture family. The sequence of the stimuli was randomized. Five instances of each gesture were collected per session, and 11*5 = 55 total gesture events were collected per session. Each gesture event was saved into 2-second raw data files for further analysis. In total, we collected 5*11*8 = 440 gesture instances from each of the 12 participants. We recorded video of the sessions for later analysis.

Results

We first removed instances where the researcher observed the participant tapping at the wrong position or missing the event. Only the remaining gesture instances were used for the offline analysis. As a result, we collected 5264 gesture instances from 12 users in total.

During the study, each participant provided gesture instances through 8 separate sessions. To understand how the system works when the training data came from different sessions, we applied a "leave-one-session-out" method to evaluate the system. For each participant, we iteratively used the instances from one session as the testing data, while using the instances from the rest seven sessions as the training data to build the model.This step was repeated eight times for each participant and we reported the overall results.

Gesture Event Detection Results

In our data processing pipeline, the first step is to detect the gesture event. If we failed to detect any gesture event in any sliding window in the offline files representing a gesture instance, we counted this instance as a false-negative error. If we detected more than one gesture event representing a gesture instance in the files' sliding windows, we counted these instances as false-positive errors. We did not run these instances with errors through gesture classification. As a result, 9 false-positive errors and 162 false-negative errors were detected out of 5264 gesture instances (NumPad). The false positive rate are 0.17%, 0.17%, 0.21%, and the false negative rate are 3.08%, 2.68%, 2.84% respectively for NumPad, DPad and CornerPad.

NumPad Classification Results

The average accuracy across 12 users for NumPad in the leave-one-session-out evaluation is 90.69%. The accuracy for each participant is shown in Figure 5. Participant 9 (P9) provided the gesture instances with the highest accuracy (95.14%) and P4 provided the instances resulting in the lowest accuracy at 82.09%. We reviewed the video from P4's data collection session and found that the way the participant performed the tap gesture was apparently lighter than others. We further examined the data and discovered that the microphone barely captured the sound of some gesture events, resulting in the lower reported accuracy. We present the confusion matrix of NumPad in Figure 6.

The lowest precision of 85.62% was provided by N8. Most of the confusion exists between adjacent tap locations. For instance, 3.05% and 7.19% of the instances of N8 were misclassified as N7 and N9. The N0 and LeftTap are the most accurate gestures at 97.15% and 96.55%, respectively. We believe this is due to the fact that the position of these two gestures are distinct from others on the back of the hand. The closer the two tap locations are, the more similar displacement patterns are during the tap captured on the wrist.

DPad Classification Results

The DPad gesture family is a subset of NumPad gestures. The four tap gesture is mapped to the N2, N4, N6, and N8 in the NumPad gesture set. The layout of the these four tap locations

	N0	N1	N2	N3	N4	N5	N6	N7	N8	N9	TapLeft
N0	97.15%	0.22%	0.44%	0.44%	0.00%	0.00%	0.00%	0.66%	0.88%	0.22%	0.00%
N1	0.22%	85.81%	6.88%	1.29%	2.80%	0.43%	0.22%	1.08%	0.00%	0.43%	0.86%
N2	0.00%	4.70%	86.75%	5.56%	0.43%	1.71%	0.85%	0.00%	0.00%	0.00%	0.00%
N3	0.43%	0.22%	3.70%	93.26%	0.00%	0.87%	1.30%	0.00%	0.22%	0.00%	0.00%
N4	0.22%	0.43%	0.43%	0.00%	90.91%	5.19%	0.43%	0.87%	0.00%	0.65%	0.87%
N5	0.00%	0.44%	1.54%	0.44%	3.96%	89.43%	2.86%	0.44%	0.88%	0.00%	0.00%
N6	0.00%	0.21%	0.21%	1.27%	0.21%	3.82%	93.84%	0.00%	0.21%	0.21%	0.00%
N7	1.08%	0.87%	0.43%	0.43%	1.52%	0.22%	0.00%	89.59%	3.69%	1.74%	0.43%
N8	1.74%	0.00%	0.00%	0.22%	0.65%	1.31%	0.22%	3.05%	85.62%	7.19%	0.00%
N9	0.85%	0.21%	0.00%	0.21%	0.63%	0.42%	1.27%	0.21%	7.40%	88.79%	0.00%
TapLeft	0.00%	0.43%	0.43%	0.22%	0.86%	0.22%	0.00%	0.86%	0.22%	0.22%	96.55%

Figure 6. Confusion Matrix for NumPad

is similar to a control pad, which can be used to navigate in the up, down, left and right directions.

The average accuracy for DPad across 12 participants is 97.33%. P4 and P5 provided the gestures with the lowest accuracy (91.28%) and highest accuracy(98.99%), respectively. As the confusion matrix in Figure 7 shows, TapLeft has the highest accuracy 97.84% and N4 offered the lowest at 96.75%.

	N2	N4	N6	N8	TapLeft
N2	97.22%	0.64%	1.71%	0.43%	0.00%
N4	0.65%	96.75%	1.30%	0.43%	0.87%
N6	0.64%	1.27%	97.66%	0.42%	0.00%
N8	0.65%	1.09%	0.65%	97.17%	0.44%
TapLeft	0.43%	1.51%	0.22%	0.00%	97.84%

Figure 7. Confusion Matrix for DPad

CornerPad Classification Results

The layout of the CornerPad gestures is mapped to the four corners on the back of the hand. By locations, they are mapped to the N1, N3, N7, and N9 gesture of the NumPad set. The average accuracy across 12 participants is 96.64%. The gesture instances from P3 achieved the highest accuracy of 99.48%. P4 offered the gesture instances with the lowest accuracies of 91.32% in the cross evaluation.

The confusion matrix of the CornerPad is shown in Figure 8. The most accurate gesture is N9 with an accuracy of 98.52% and the least accurate gesture is N7 with an accuracy of 94.36%. Interestingly, we noticed the TapLeft gesture was mostly confused with N1 and N7, which are the closest tap locations to TapLeft in the CornerPad.

	N1	N3	N7	N9	TapLeft
N1	94.41%	2.15%	1.51%	0.65%	1.29%
N3	0.87%	98.04%	0.43%	0.65%	0.00%
N7	1.08%	0.65%	94.36%	3.25%	0.65%
N9	0.42%	0.63%	0.42%	98.52%	0.00%
TapLeft	0.65%	0.43%	1.08%	0.00%	97.84%

Figure 8. Confusion Matrix for CornerPad

DISCUSSION

Evaluation with more practical models

The results of leave-one-session-out validation provide a good estimate of how well the classifier performs on recognizing different gestures across different sessions. However, in practice, building a real-time gesture recognition model would face more challenges, such as how many training instances are needed to build a model for a person, whether the model can be built for user-independent operation, and how many calibration instances would be necessary to improve the classification accuracy. To answer these questions, we conducted another set of experiments by building user-dependent, user-independent(leave-one-participant-out), and user-adaptive models. Similarly, we simulate the real-time classification pipeline by applying a sliding window of 1.2 seconds with 75% overlap for all the files we collected in all the models we built in this section.

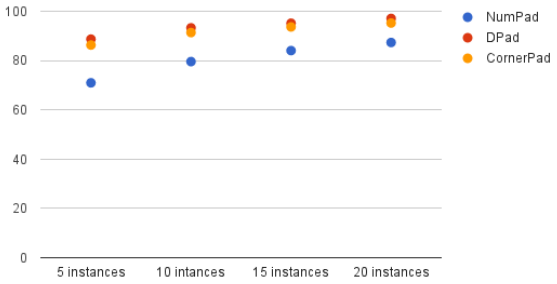

Figure 9. Accuracy of different number of training instances for user-dependent models

User-Dependent Models

User-dependent models only use the data from the same participant to train a model which will be then used for classifying his/her own gestures. Usually, a model trained with more training instances provide higher recognition accuracies, but also require longer time to collect the data from the participant, which can be annoying. Therefore, to investigate how many instances are needed to train a user-dependent model, we built user-dependent models for each participant by using the first 5, 10, 15, and 20 collected instances per gesture from that same participant to train an SVM model. We used the remaining instances of that participant to test the model. Evaluating TapSkin with user-dependent models across the data collected in different sessions would be similar to how the system would work in practice. A user provides the data

in the first few sessions, which will be used to build models evaluated in the later sessions. The results are shown in Figure 9. When only using 5 instances to build a model, the overall accuracy for NumPad, DPad and CornerPad across 12 participants were 70.97%, 88.69% and 86.3%. The highest accuracies were 87.33%, 97.09%, and 95.23% respectively for NumPad, DPad and CornerPad, when we trained the user-dependent models with 20 instances per gesture for each participant.

User-Independent and User-Adaptive Models

Compared with user-dependent models, user-independent models(Leave-one-user-out) do not require the user to provide any training data before using the technique. It would improve the user experience by reducing the workload on the side of the user, but it places a higher standard for the reliability of the gesture models.

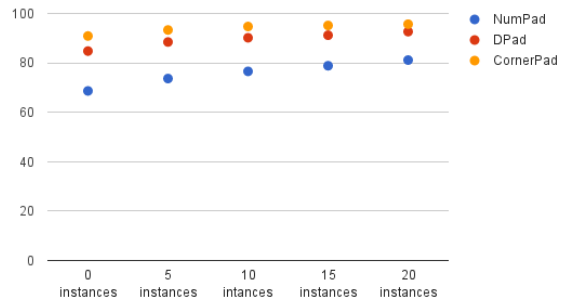

Figure 10. Accuracies of User-independent and user-adaptive Models. Along the x-axis, a value of 0 instances is the same as a user-independent model.

We used the data of 11 participants from the total 12 participants to train a support vector machine, which is then tested with data from the remaining participant. The average accuracy across 12 participants for the NumPad, DPad and CornerPad are 68.65%, 84.78% and 90.35% as shown in Figure 10.

The user-adaptive model falls in between the user-independent and user-dependent models, considering the amount of effort required for each user to provide training instances. It strengthens the user-independent models by adding a few number of instances per gesture from an individual user as demonstrated by [35]. We evaluated our technique on user-adaptive models by incrementally adding 5, 10, 15, and 20 instances per gesture from each participant to the user-independent training set. As Figure 10 shows, the more personalization data is added to the models, the higher the average accuracies are. The highest accuracies are reached when 20 instances are added to the models, which are 81.13%, 92.64%, and 95.66% for NumPad, DPad and CornerPad.

We notice that the accuracies of user-adaptive models with 20 instances per gesture are lower than the accuracies of user-dependent models built with similar number of training instances. We believe this is due to tap locations being different from person to person based on different hand sizes. For example, tapping at the same location of the hand (e.g., knuckle) can generate a different sound from person to person. Therefore, adding more training data from others may not necessarily help improve the recognition accuracy.

However, the user-independent model may work for a smaller set of gestures. CornerPad achieved 90.35% accuracy with user-independent models. With 5 and 10 instances added, the accuracy further increased to 93.32% and 94.72%. We believe it is because the distances between the tapping locations in CornerPad are longest, which potentially provides more distinguishable characteristics in different gestures. A practical approach is to ask the user to test each gesture before using it and only provide a few calibration gestures for those gestures that may be misclassified.

Addressing the false-positive and negative errors

One important practical challenge for this gesture recognition technique for wearable devices is how it works when being deployed in a real device which is worn by the user all day long. In real scenarios, this may trigger false-positive (FP) and false-negative (FN) errors, which is critical to the perceived user experience.

Therefore, we conducted a more detailed analysis into these errors. In the above experiment, our system triggered 9 (0.17%) FP and 162 (3.08%) FN errors out of 5264 gesture instances. We first looked into the 162 FN errors. We found only 13 of the FN errors were caused by not passing the predefined thresholds.The rest 149 FN errors were introduced by not being able to capture the whole 0.5 feature extraction window after the peak localization. After reviewing the video and the sensor data, we found three reasons.

The first one is that if the participant (E.g. P4) made weak taps, the peak localization may not capture the peak as expected. The second reason is that the participant performed the gestures either too early or too late, such that only part of the gesture fell into the 2-second window which were saved into sensor files. The third reason is that the audio data was not synchronized with IMU data while saving to files, which introduces a misalignment on the time-stamp. As a result, one of the sensor (mostly the audio) files failed to capture the whole gesture. This happens more when the watch is over heated after performing the heavy computing workload for a certain period of time. As the watches become more powerful in the future, it is reasonable to expect that the last two kinds of FN errors can be reduced by running the whole system online exclusively on the watch.

Although our system only triggered 9 FP errors, avoiding the FP errors is still quite a challenging task. On one hand, the results caused by FP errors can be very frustrating to a user. For instance, if TapSkin is used to control the dial pad, FP errors may result in unwanted phone calls when the user even does not notice it at all. On the other hand, all the gesture instances were captured in a daily office area in a static posture(sitting), where only gentle noise (e.g. talking, walking, air conditioning) existed compared with outdoor noise. To address this FP issue in real scenarios, one way is to apply more

advanced signal processing technique (e.g. band-pass filter) and build another classifier in the gesture event detection step with carefully collected noise data. The other method is to use an activation gesture for TapSkin. Only when an activation gesture is detected, the system begins recognizing the whole gesture set. The activation gesture should be both intuitive to perform, has less confusion with other gestures, and providing low FN and FP error rates. We observed that the tap on the left side of the watch satisfies these criterion by providing the highest classification accuracy and relatively low FN and FP error rates. Therefore, we chose this gesture as the activation gesture for all TapSkin gesture families. In practice, in order to further reduce the false-positive errors, a double tap instead of a single tap on the left of the watch can be used as the activation gesture.

Improving the Sensing Unit

TapSkin uses both inertial sensing (gyroscope, accelerometer) and acoustic sensing (microphone) to detect the tap events on the skin. To understand which sensing unit is more informative in recognizing the on-skin tap gestures, we conducted an additional experiment by using only the data from IMU or the microphone in a traditional 10-fold-cross-validation for the NumPad gesture set on each participant. The overall results across 12 participants are presented in Table 1. The table shows that the IMU data provides more discriminative power than the acoustic data, but together still provide significantly better classification results.

Table 1. Accuracies with different sensor combinations

	IMU	Acoustic	Both
Accuracy	88.82	65.72	92.96

We are currently using the built-in microphone on a smartwatch to capture the on-skin sound, which is not designed to capture with the body transmit ed sound. Potentially, the quality of the acoustic information can be improved by using a more sensitive sensor (e.g. contact mic) that can be better coupled to the body rather than the air.

Improving the Gesture Design

The current gesture design is based on the layout of a number pad (NumPad), a control pad (DPad) and the corners of a square shape (CornerPad). To help the experimental participants learn the tap locations, we drew the layout on the skin, a solution that may not be desirable in practice. One challenge is to design a gesture set that is both memorable as well as machine recognizable.

One way to enhance recognition is to provide a small set of tap gestures, which are located at greater distances between each other (e.g., DPad, CornerPad). Therefore, the user can easily memorize the location of each tap gesture. A drawing layout may not be needed in such a case. Since there is a larger space between the two tap locations, variance between different taps for the same gesture can be tolerated by the system and used for recognition.

Another method to prevent drawing icons on the skin is to utilize the natural layout of the hand and map gestures to different parts of the body. For instance, each hand consists of 4 knuckles. If we map each of them with a tap gesture, a user can easily repeat the tap even without any marker on the skin.

However, painting on the hand may not always be unacceptable for some users, if the drawn picture is well designed. For instance, many people have already a tattoo on their skin. Therefore, we can redesign the whole drawn layout of TapSkin which makes it more like a tattoo, such that people will not feel uncomfortable with a 'Tattoo" style layout on their skin.

System Limitations and Future Work

There are several important limitations of our results and system to consider before applying TapSkin to real applications.

First, in our validation experiment, gesture instances were collected while the participants were holding their arms in a static position. In practical scenarios, the users may use TapSkin while they are in motion (e.g., while walking). Tapping on the skin in motion would influence the inertial signals, as well as potentially the accuracy of the tapping locations. Our current technique may exhibit higher classification errors and more false-positive errors. However, we can build separate models for when the user is in motion (or different states of motion, such as stationary, walking or running). To classify a gesture, we need to first classify the user's motion state (using something like Google's Activity Recognition service) and choose the appropriate classifier for that motion state. To reduce the potential false-positive errors while the user is in motion, a high-pass filter can be applied to the IMU data to remove the low-frequency motion(such as walking). And a separate binary classifier for noise detection can be built for event detection too. We plan to further explore this issue in the future.

Second, all the processing work is not processed solely on the watch but using an external phone. The data transmission between the watch and the phone introduces a high latency, which influence the perceived user experience. We plan to further improve the user experience by optimizing the processing pipeline to reduce the delay.

Another limitation of the work is that the sound generated by the tapping event may change in different situations. For instance, if the hands sweat or are otherwise moistened, the wet skin will change the acoustic signature of the tap, providing an unknown influence on the system's performance. We also have not considered the effect of clothing covering the arm or gloves on either the tapping or receiving hand. This introduces an even more complicated recognition problem beyond determining the motion state of the user. We note that these issues also cause problems with the normal touchscreen interaction, suggesting a more general problem for future research on wearable interaction techniques.

Figure 5 shows variance on the classification accuracies across different participants. After reviewing the video, we found that certain participants made weak taps on the back of their hand, tending to result in a higher amount of false-negative errors. The overall classification accuracies were also influenced by the strength of the tap being too light. This can be one potential limitation of TapSkin.

Somewhat related to this last point, we intentionally instructed our participants to wear the smartwatch near the wrist in a tightly fit fashion. This is helpful for collecting useful IMU data, but may cause a problem for those who dislike a tight-fitting wrist-mounted device. We are aware that some users like wearing the watch loosely, where the watch may slide along the arm. Such a movement may cause both motion and acoustic noise, which introduces more false-positive and classification errors. This is another limitation of our technique.

Furthermore, the current gesture family only contains tap gestures. However, we have received encouraging results by applying the similar technology on detecting slide gestures on the back skin of the hand. We would expect in the future to be able to expand the gesture set to include more elaborate sliding sequences, ultimately allowing a user to do a Grafiti-style text input language.

CONCLUSION

In order to increase the richness of input to a smartwatch without introducing any further on-body instrumentation, we introduced TapSkin, which uses the on-board sensing of the smartwatch to classify a variety of tapping gestures. Our results for classifying tap events are good enough for everyday use in a variety of application scenarios. We can identify up to 11 on-skin tap gestures (Number Pad with an activation patch) around the wrist area with an accuracy of 90.69% in a leave-one-session-out validation with 12 participants. These results increase to 97.33% and 96.64% for reduced tap locations of a DPad and CornerPad, each with 4 tap locations and an activation patch. While there are still some limitations to our approach for practical deployment, we provided an extensive discussion to address those issues for the practical deployment.

ACKNOWLEDGMENTS

We would like to thank Ning Xu for the help on the video and visual graphs, all the reviewers for their detailed feedback and the participants in the user study. This work is supported by Georgia Tech Wearable Computing Center Engagement Grant and the Intel Science and Technology Center for Pervasive Computing (ISTC-PC).

REFERENCES

1. Ashbrook, D., Baudisch, P., and White, S. Nenya: Subtle and eyes-free mobile input with a magnetically-tracked finger ring. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, ACM (New York, NY, USA, 2011), 2043–2046.
2. Baudisch, P., and Chu, G. Back-of-device interaction allows creating very small touch devices. In *Proceedings of the 27th international conference on Human factors in computing systems - CHI 09*, Association for Computing Machinery (ACM) (2009).
3. Bernaerts, Y., Druwé, M., Steensels, S., Vermeulen, J., and Schöning, J. The office smartwatch: development and design of a smartwatch app to digitally augment interactions in an office environment. In *Proceedings of the 2014 companion publication on Designing interactive systems*, ACM (2014), 41–44.
4. Bi, X., Li, Y., and Zhai, S. FFitts law. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI 13*, Association for Computing Machinery (ACM) (2013).
5. Chen, K.-Y., Lyons, K., White, S., and Patel, S. utrack. In *Proceedings of the 26th annual ACM symposium on User interface software and technology - UIST 13*, Association for Computing Machinery (ACM) (2013).
6. Dementyev, A., and Paradiso, J. A. Wristflex. In *Proceedings of the 27th annual ACM symposium on User interface software and technology - UIST 14*, Association for Computing Machinery (ACM) (2014).
7. Deyle, T., Palinko, S., Poole, E. S., and Starner, T. Hambone: A bio-acoustic gesture interface. In *2007 11th IEEE International Symposium on Wearable Computers*, Institute of Electrical & Electronics Engineers (IEEE) (oct 2007).
8. Funk, M., Sahami, A., Henze, N., and Schmidt, A. Using a touch-sensitive wristband for text entry on smart watches. In *Proceedings of the extended abstracts of the 32nd annual ACM conference on Human factors in computing systems - CHI EA 14*, Association for Computing Machinery (ACM) (2014).
9. Goel, M., Wobbrock, J., and Patel, S. Gripsense. In *Proceedings of the 25th annual ACM symposium on User interface software and technology - UIST 12*, Association for Computing Machinery (ACM) (2012).
10. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. The weka data mining software: an update. *ACM SIGKDD explorations newsletter 11*, 1 (2009), 10–18.
11. Harrison, C., Benko, H., and Wilson, A. D. Omnitouch: wearable multitouch interaction everywhere. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, ACM (2011), 441–450.
12. Harrison, C., Tan, D., and Morris, D. Skinput: appropriating the body as an input surface. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (2010), 453–462.
13. Hinckley, K., and Horvitz, E. Toward more sensitive mobile phones. In *Proceedings of the 14th annual ACM symposium on User interface software and technology*, ACM (2001), 191–192.
14. Kim, J., He, J., Lyons, K., and Starner, T. The gesture watch: A wireless contact-free gesture based wrist interface. In *2007 11th IEEE International Symposium on Wearable Computers*, Institute of Electrical & Electronics Engineers (IEEE) (oct 2007).

15. Laput, G., Xiao, R., Chen, X. ., Hudson, S. E., and Harrison, C. Skin buttons. In *Proceedings of the 27th annual ACM symposium on User interface software and technology - UIST 14*, Association for Computing Machinery (ACM) (2014).

16. Lyons, K., Nguyen, D., Ashbrook, D., and White, S. Facet. In *Proceedings of the 25th annual ACM symposium on User interface software and technology - UIST 12*, Association for Computing Machinery (ACM) (2012).

17. Lyons, K., Starner, T., Plaisted, D., Fusia, J., Lyons, A., Drew, A., and Looney, E. Twiddler typing: one-handed chording text entry for mobile phones. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM (2004), 671–678.

18. Lyons, K., Starner, T., Plaisted, D., Fusia, J., Lyons, A., Drew, A., and Looney, E. W. Twiddler typing. In *Proceedings of the 2004 conference on Human factors in computing systems - CHI 04*, Association for Computing Machinery (ACM) (2004).

19. McGrath, W., and Li, Y. Detecting tapping motion on the side of mobile devices by probabilistically combining hand postures. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*, ACM (2014), 215–219.

20. Merrill, D., and Raffle, H. The sound of touch. In *CHI'07 Extended Abstracts on Human Factors in Computing Systems*, ACM (2007), 2807–2812.

21. Nanayakkara, S., Shilkrot, R., Yeo, K. P., and Maes, P. Eyering: a finger-worn input device for seamless interactions with our surroundings. In *Proceedings of the 4th Augmented Human International Conference*, ACM (2013), 13–20.

22. Nandakumar, R., Iyer, V., Tan, D., and Gollakota, S. Fingerio: Using active sonar for fine-grained finger tracking. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, ACM (2016), 1515–1525.

23. Oakley, I., Lee, D., Islam, M. R., and Esteves, A. Beats. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems - CHI 15*, Association for Computing Machinery (ACM) (2015).

24. Perrault, S. T., Lecolinet, E., Eagan, J., and Guiard, Y. Watchit. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI 13*, Association for Computing Machinery (ACM) (2013).

25. Rekimoto, J. GestureWrist and GesturePad: unobtrusive wearable interaction devices. In *Proceedings Fifth International Symposium on Wearable Computers*,

26. Roudaut, A., Huot, S., and Lecolinet, E. TapTap and MagStick. In *Proceedings of the working conference on Advanced visual interfaces - AVI 08*, Association for Computing Machinery (ACM) (2008).

27. Ruiz, J., Li, Y., and Lank, E. User-defined motion gestures for mobile interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (2011), 197–206.

28. Saponas, T. S., Tan, D. S., Morris, D., Balakrishnan, R., Turner, J., and Landay, J. A. Enabling always-available input with muscle-computer interfaces. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, ACM (2009), 167–176.

29. Weigel, M., Mehta, V., and Steimle, J. More than touch: Understanding how people use skin as an input surface for mobile computing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, ACM (New York, NY, USA, 2014), 179–188.

30. Wen, H., Ramos Rojas, J., and Dey, A. K. Serendipity: Finger gesture recognition using an off-the-shelf smartwatch. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, ACM (2016), 3847–3851.

31. Xia, H., Grossman, T., and Fitzmaurice, G. Nanostylus. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology - UIST 15*, Association for Computing Machinery (ACM) (2015).

32. Xiao, R., Laput, G., and Harrison, C. Expanding the input expressivity of smartwatches with mechanical pan, twist, tilt and click. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems - CHI 14*, Association for Computing Machinery (ACM) (2014).

33. Xu, C., Pathak, P. H., and Mohapatra, P. Finger-writing with smartwatch: A case for finger and hand gesture recognition using smartwatch. In *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*, ACM (2015), 9–14.

34. Yang, X.-D., Grossman, T., Wigdor, D., and Fitzmaurice, G. Magic finger: always-available input through finger instrumentation. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*, ACM (2012), 147–156.

35. Zhang, C., Guo, A., Zhang, D., Southern, C., Arriaga, R., and Abowd, G. Beyondtouch: Extending the input language with built-in sensors on commodity smartphones. In *Proceedings of the 20th International Conference on Intelligent User Interfaces*, IUI '15, ACM (New York, NY, USA, 2015), 67–77.

36. Zhang, C., Yang, J., Southern, C., Starner, T., and Abowd, G. D. Watchout: Extending interactions on a smartwatch with inertial sensing. In *2016 20th IEEE International Symposium on Wearable Computers* (2016).

37. Zhang, Y., Zhou, J., Laput, G., and Harrison, C. Skintrack: Using the body as an electrical waveguide for continuous finger tracking on the skin. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, ACM (2016), 1491–1503.