GraspTraker: Tracking smartphone grab posture with inaudible sound

Seungjoo Lee*
KAIST
juicelee@kaist.ac.kr

HyungJun Yoon*
KAIST
diamond264@kaist.ac.kr

1 INTRODUCTION

Mobile devices are used in countless user contexts, which varies with the type of the user and the situation. To provide the best user experience in mobile device use, the interaction method should be aware of the user's context and adapt to it in real-time. Thus, it is important to know the user context on mobile devices. Especially, for common smartphones, the hand posture of the user should be considered as a critical factor in user context, since touching is the most dominant interaction method. For instance, iGrasp [3] devised a hand posture-aware keyboard interface and showed its effectiveness by reducing 42% of typing time in its evaluation, which reveals the importance of hand posture awareness. However, our smartphones yet do not provide the user's hand posture context in real-time, thus these kinds of interfaces cannot be adapted accordingly.

To address this problem, several works have been proposed. Some of them [3, 7, 8] used additional capacitive touch sensors to sense how the users grab mobile devices. However, using external devices incurs additional costs and inconveniences to users. There have also been some works that overcame this limitation [6], which predict the hand posture by sensing the motion of the device and the touch of the user. Nonetheless, it is based on the user's already-taken actions including swipe and touch, thus there would be some time gap between the user's actual hand posture change and the prediction by the application, which can possibly degrade the user's QoE.

So we present an application in this study to track the user's hand posture, solving the limitations of the prior works. To do that, our application had to use built-in sensors to eliminate the cost and inconvenience of the user, and be able to immediately track the user's hand posture without time lag. For those purposes, we designed GraspTracker, with the key idea of using the differences in amplitude change at each frequency in the sound propagated through the smartphone's device, depending on the gripping hand posture. The sound emitted by the speaker and detected by the microphone can be divided into direct sound delivered directly and reflected sound that is not. Direct sound refers to sound transmitted through the smartphone device itself as a solid vibration, or directly from the speaker through the surrounding air. Reflected sound is the sound that is transmitted from the speaker to the environment and then reflected back to the microphone. Since direct sound includes sound propagated through the device hardware, the waveform recorded by the microphone depends on the physical state of the device. And the posture of the hand holding a smartphone is included in this physical state. Based on the principle that the frequency range absorbed or amplified depends on the shape and curvature of the hand, we designed a study to capture this acoustic property and classify it to the gripping posture of the hand.

For our goal, it is important to remove the reflected sound that confuses the classification from the sound recorded in the microphone, because it includes environmental features. While previous attempt has been made to detect gripping hand posture using audio signals [9], it has the limitation that it have not completely eliminated the effects of reflected sound. So, as the biggest contribution of our project, we present our new method for minimizing the effect of reflected sound using mobile system for the hand posture classification. And by using that method, we implemented an Android application, GraspTracker. In order not to disturb people, the GraspTracker outputs FMCW sounds using inaudible sounds. At the same time, it uses the smartphone's camcorder microphone and primary microphone together to record sound, and processes the FFT on the recorded sounds. Finally, the GraspTracker is trained to minimize the effect of the reflected sound through the machine learning classifier from the FFT results and to classify the current gripping hand posture. In the following sections, we will detail the challenges in designing this application, their solutions, and evaluation on our implemented application.

2 RELATED WORK

In this section, we first introduce works that recommend an appropriate interface based on the user's context, especially hand posture, to improve the quality of the user's interaction. The purpose and motivation of these studies is the same as ours, and when we succeed in tracking hand posture, recommending an appropriate interface, it can be used as an application of GraspTracker. And we will introduce previous studies of GraspTracker to determine the hand posture of a

*Both authors contributed equally to this project.

user holding a mobile device. The studies fall into two broad categories: papers that use external sensors in addition to the mobile device itself to identify hand posture, and those that do not.

Grasp-based User Interfaces For modern mobile devices, it is important to provide a context adaptive interface in order to give the user the best experience, and there had been many studies related to it [5]. And what we focused on of these were the papers that studied grasp-based interfaces in smartphones. iGrasp [3] solved the problem of users who are uncomfortable typing in the traditional way when using a large screen mobile device with various grasps. They attached an external sensor to the device to determine the posture of the hand. It then helps the user to type keyboard input more easily by adjusting the position of the keyboard displayed on the screen according to the inferred hand posture. There is a similar study named iRotateGrasp [2], which tracks the user's grasping hand posture and rotates the orientation of the smartphone's screen according to the user's orientation. The study also tracks the hand posture according to the touch input coming into the sensor installed on the edge of the smartphone.

Sensing Grasp with Additional Hardware The iGrasp and iRotateGrasp above both attach additional hardware to the mobile device to sense the hand grasp. For iGrasp, they developed a hand grasp sensing prototype targeting the iPad, with 23 touch sensors on the right and left sides of the iPad's back. The hand grasp is estimated based on inputs from 46 capacitive sensors. In addition to this, there were Hand Sense [13] and Graspables [12] to estimate hand posture by connecting additional capacitive sensors to mobile devices. iRotateGrasp tracked grasp on iPod, by attaching a total of 44 sensors which contain 32 light sensors to the edge and backside of it. Other attempts have been made to Touch Active [10] using additional hardware that utilizes acoustic signals. They attached a vibration speaker and a piezo-electric microphone to the mobile device to predict hand posture. All of the above studies categorized grasping type in detail and detected classified posture with high accuracy. However, they commonly require the user to use additional devices. This makes the user to pay an additional cost and feel more inconvenient than only using the mobile device itself. Therefore, these works were limited to users' broad use in the real world, and in GraspTracker, we wanted to overcome these limitations.

Sensing Grasp with Built-in Sensors GripSense [6] differs from the above works in that it has succeeded in hand grasp classification without additional hardware. As the features used for the classification, GripSense senses the motion of the gripping hand using gyroscope and accelerometer which are the built-in sensors in the smartphone, and senses the touched area or swipe direction on the touch screen. However, GripSense distinguishes a limited number of hand grasp types compared to previous grasp detecting applications, and has a fatal disadvantage that grasp detection has time gap from user's actual grasp since posture prediction requires user's swipe action as feature. SmartGrip [9] is a work that can distinguish more types of hand grasps, in real time, without such touch events. It emits the FMCW signal from the speaker and analyzes the sound measured by the microphone using the direct sound that differentiates according to the gripping type of the hand and use it as a feature of the classification. Since sound playback and recording is available as a basic feature of a smartphone, it does not require any additional devices, and the FMCW signal is very short in length, allowing the user to track the hand posture closer to real-time. SmartGrip, however, has not succeeded in separating the direct sound completely from the recorded sound input. To improve the accuracy of the methods they used, they needed a way to remove the effects of the reflected sound from the microphone as much as possible. We will introduce our own method for eliminating the effects of reflected sound in the next section, and describe GraspTracker, an application built using it.

3 DESIGN AND IMPLEMENTATION

GraspTracker identifies user's smartphone grasp posture using built-in sensors on smartphone. Fig 1 shows the system overview of *GraspTracker*. It generates FMCW audio signal from the earpiece speaker of the smartphone, and records the sound with two different microphone. For each recorded sound, FFT (Fast Fourier Transform) is performed to get the frequency response and extracts feature from the result. Finally, *GraspTracker* classifies the user's grasp posture using extracted features. In the remainder of this section, we will discuss detailed explanation and design rationale of each components.

3.1 Sound propagation model

To distinguish different grasp posture, *GraspTracker* uses inaudible sound. In this subsection, we analyze how the emitted sound propagates and reaches the microphone.

The emitted sound can be classified into two categories, *direct sound* and *reflected sound* as shown in Fig 2. *Direct sound* goes straight from the speaker to the microphone, and propagated through device body and air. The direct sound propagated through air is affected by the skin of hand on the propagating path. As human skin absorbs acoustic signal and the absorption rate is different for each frequency[1], each grasp posture attenuates certain frequency ranges.

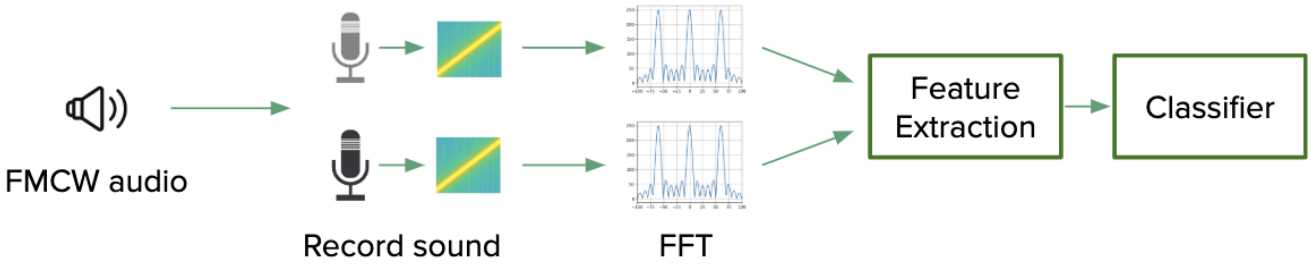

Figure 1: Overview of *GraspTracker*

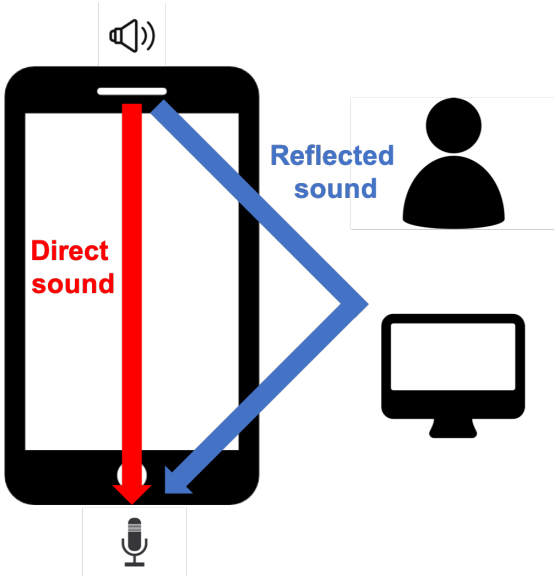

Figure 2: Propagation of sound emitted from earpiece speaker.

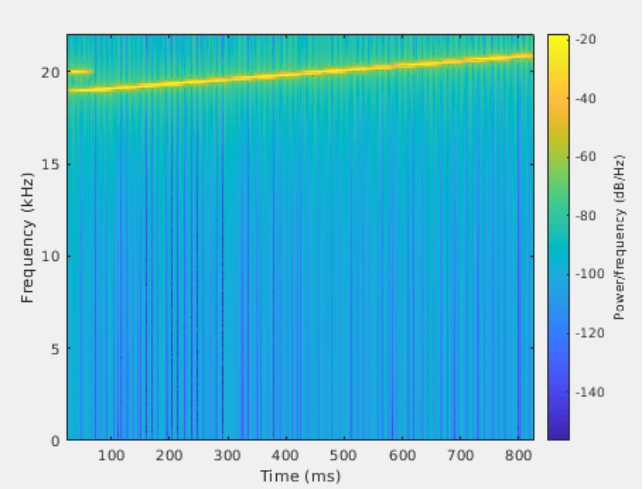

Figure 3: Spectrogram of designed sound

The direct sound propagated through device body is influenced by the user's grip, which changes damping and boundary conditions[4, 11]. Different damping and boundary conditions changes the propagating sound, which enables to distinguish grasp posture.

Dissimilar from direct sound, reflected sound is caused by the reflection of emitted sound with surrounding objects. The reflected sound can result in different frequency response even with same grasp posture as it has the additional information about surrounding environments.

3.2 Sound signal design

To classify the grasp posture, it is important to get the rich information from the sound, which means generated sound should sweep large frequency ranges to distinguish each grasp posture. Simultaneously, the sound should not be disruptive or audible to user.

For this, we designed the sound that sweeps from 19khz to 21khz lineally and lasts 0.8s with python module *pydub*. Spectrogram of the designed sound is shown in Fig 3.

3.3 Eliminating the reflected sound

With the designed sound signal, each grasp figure responses in unique way. Left and middle graphs of Fig 4 shows the FFT result of different grasp posture when the designed sound is played. Due to the change on damping, boundary condition, and attenuation caused by the skin, each grasp posture can be distinguished.

However, the response changes when the environment is changed. Middle and right graphs of Fig 4 show the responses when the grasp posture is same but environment is

different. Even if the grasp posture is same, the response is quite different because of the reflected sound.

To address this issue, *GraspTracker* leverages another microphone which is common in modern smartphones. In specific, *GraspTracker* plays the designed sound from the earpiece microphone and records it from two different microphone, one is usual microphone that is located at the bottom of the smartphone, and the other is top microphone that can be accessed by *CAMCORDER MICROPHONE* on Android. The bottom microphone records the direct sound affected by the hand posture and reflected sound, and the top microphone records the direct sound that is not affected by the hand and reflected sound. Here, recorded reflected sound of two microphones are similar than the direct sound even if the position of two microphones are different, hence the reflected sound can be cancelled out.

3.4 Classification

Feature extraction We use FFT coefficients to capture the distinct frequency response from each grasp posture. However, raw FFT coefficients cannot be used because too many features is not good for classification. Even if the designed sound's duration is about $0.8s$, *GraspTracker* records the sound for $6s$. This is because the sound is not immediately played on Android smartphone due to scheduling problem. If we use sampling frequency of $44100hz$, the number of FFT coefficients in $19khz$ to $21khz$ range is 24000, which is too many. Thus, we averaged the amplitude of FFT coefficients of $25Hz$ window, resulting 123 features for each microphone, 246 features in total.

Classifier We use Support Vector Machine ($c = 1.0$, polynomial kernel with degree=3) as a classifier, provided by *scikit-learn* python module. We employ SVM classifier since it needs not much training data and less prone to overfitting, as it finds an optimal hyperplane for classification. Moreover, it outperforms other types of classifiers in our evaluation such as logistic regression and random forest.

3.5 Implementation

GraspTracker is implemented as an Android application with *Kotlin*. We generate the sound through earpiece microphone, collect the sound data from the two microphones, and save the data as an csv file with the Android application. Then, we build an classifier and evaluate with *scikit-learn* python module.

To show the feasibility of the real time classification, we also implement a demo application that extracts feature and classifies the grasp posture on device in real time with *Weka* library. The time taken for the feature extraction and classification is relatively shorter than the time it takes to make a sound, demonstrating that it is suitable for use in real time.

Table 1: GraspTracker Classification Accuracy

	2-channels	top-channel	bottom-channel
LogReg	49%	31%	41%
SVM	60%	33%	56%
RF	63%	30%	51%

4 EVALUATION

To evaluate the classification accuracy of the implemented GraspTracker, we designed experiments under various conditions. For all evaluation processes, we used Google Pixel, and they are performed in a quiet in-lab environment.

We first set up to seven labels for evaluation: grasping with right-hand / left-hand / two-hands / right-hand-landscape / left-hand-landscape / two-hands-landscape / on-desk. And we conducted experiment for two users, one user collected 30 data, and the other user collected 20 data per each label, so we collected a total of 350 data. The data of each label collected by each person is divided into 4/5 of training data and 1/5 of testing data. That is, 280 data were used for training, and by using 70 testing data, we measured the accuracy. In one data, there were 123 from each microphone channel extracted from the recorded sound. We trained the data with seven kinds of labels corresponding to these 246 features through a machine learning classifier. We adopted three methods as classification algorithms: logistic regression, SVM, and random forest. We also looked at the classification accuracy when only one microphone channel was used, i.e. 123 features, to determine whether receiving two channels of microphone input would be effective in increasing the accuracy of the GraspTracker. All of the results of these evaluations are shown in Table 1.

First, the results in Table 1 show that using both channels is effective for GraspTracker. In classification via SVM, accuracy is 56% using only the primary microphone at the bottom of the device, 33% using only the camcorder microphone at the top, and 60% using both microphones. Accuracy increases in the order of using top-channel only, bottom-channel only, and using both microphones, and the trend is the same in other machine learning algorithms. This result fits well with the hypothesis we built when we designed GraspTracker. Since the camcorder microphone at the top of the device is very close to the speaker, it will consist of reflected sound and direct sound that is less affected by the hand holding the device. Therefore, top-channel records cannot be a good feature to distinguish hand posture. On the other hand, in the case of the bottom-channel, it can be a clear feature compared to the top-channel because it is greatly influenced by the gripping hand while sound is propagated through the device in the speaker. But the bottom-channel still contains the sound reflected in the record. In order to minimize this effect, we proceeded the classification using the top-channel

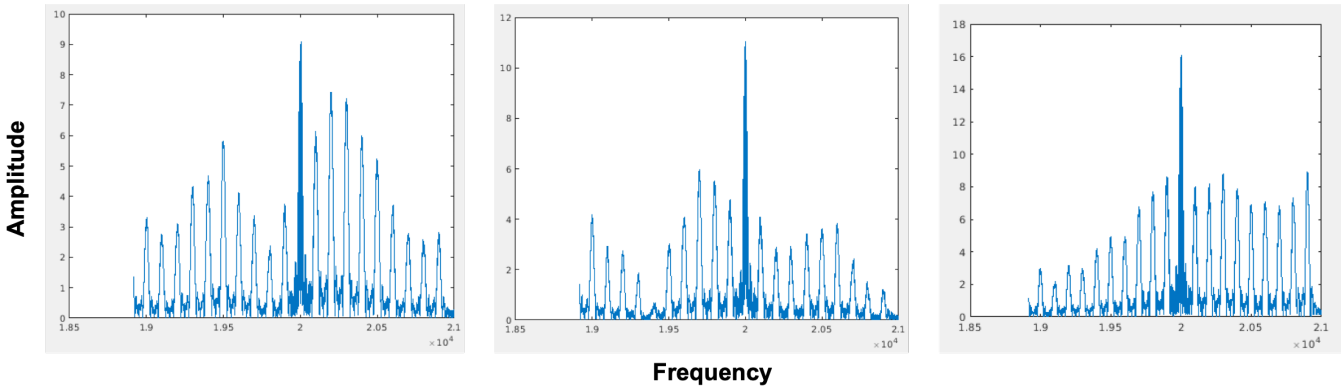

Figure 4: FFT result of different grasp pose and environment with designed sound. Left figure is left hand posture with environment 1, middle figure is two hand posture with environment 1, and right figure is two hand posture with environment 2.

and bottom-channel together as features, and the classification using 2-channels had higher accuracy as expected. This can be interpreted as the classifier trained the effect of the reflected sound in the bottom channel to cancel the features in the top channel. Next, by comparing the accuracy of classification according to the machine learning algorithm, we can find that SVM and random forest algorithm have higher accuracy than logistic regression. Random forest algorithm has 3% higher accuracy compare to SVM, however, 3% is not a meaningful value because we conducted our test with small amount of test dataset. For the top-channel and bottom-channel, SVM has higher accuracy than random forest, but we used 2-channels for GraspTracker, so we adopted the most accurate random forest as the classification algorithm.

5 DISCUSSION

We built a application *GraspTracker* that tracks gripping hand posture with inaudible sound, using only built-in sensors. For the key algorithm of GraspTracker, we proposed a unique method to extract direct sound from smartphone microphone input. Our work contains the evaluation for direct sound extraction method.

Using *GraspTracker*, we enable adjusting the smartphone interface regarding the most important user context, which is grasp posture. Thus, increasing user's QoE.

Limitations The most obvious limitation of the GraspTracker we have developed so far is that the application does not have a high enough accuracy for real-world use. The method currently used to offset the reflected sound in the GraspTracker is to store two kinds of microphone inputs and merge the two into the appropriate coefficients through machine learning. And the SVM, random forest, and logistic regression that we used all have the characteristic that they do not output high enough accuracy when there is not enough data set to use for train. And the lack of this training dataset is the biggest problem of our work.

Since we did not collect enough dataset, we could not try deep learning that requires a lot of data, and even the classifiers we used as a substitute did not show noticeably high accuracy. There may be other reasons for low accuracy, but because of the lack of data, we cannot even be convinced that the accuracy is low for other reasons.

Next, in connection with the lack of a dataset, GraspTracker's evaluation was processed with that dataset that have lacked diversity. In the evaluation part we used only one type of device (Google Pixel). However, speakers and microphones are hardware that is easy to have different characteristics for each device, so for proper evaluation, it is necessary to evaluate whether the same algorithm works well on different devices. Our hypothesis also suggests that GraspTracker's classification algorithm can be affected differently depending on the shape of the user's hand. However, for the user test, only two participants were evaluated. For a wider range of practical uses of GraspTracker, more than 10 different participants should have been tested.

Finally, the current GraspTracker has not been validated for use in wild environments. Since there was no evaluation of the wild environment, it is unknown whether both datasets and algorithms used for current evaluation will work. The FMCW signal of the inaudible sound we designed may be influenced by external noise, and the case of much movement of the phone such as shaking is not considered. Also, in real life, there are various types of postures where people grip mobile phones. In our case, we have evaluated 7 types in total, so for the real use of GraspTracker, we need to investigate grasp poses as a user test in a wild environment.

Future work To improve the usability and accuracy of the *GraspTracker*, here we discuss the future work. First, we

can extract only the direct sound propagated through the device body. As mentioned earlier, grasp posture changes the damping and boundary conditions[4, 11], we can get a sound that is not affected by the surrounding environment at all if this sound can be solely extracted. As the propagation speed of the sound is much faster in solid, this approach would be possible.

Second, we can employ neural network to improve the accuracy. Currently, features for classification have both information about grasp posture and surrounding environment. As neural networks can extract complex features that is more relevant to grasp posture, the accuracy would be improved. However, we need to collect much more data to train it.

Lastly, if hand posture detection is always-on and generate inaudible sound through earpiece microphone all the time, it would restrict the user's activity and drain lots of battery. We can employ other sensors such as accelerometer and gyroscope to detect possibility of grasp posture change. If the posture change is detected, then detect the exact posture with *GraspTracker*.

6 INDIVIDUAL CONTRIBUTIONS & WHAT WE LEARNED

- Equal contribution
  - Brainstormed the idea, and designed the whole algorithm of GraspTracker.
  - Found related work, and shaped up the topic.
  - For the documentation, wrote proposal and final report for each part.
  - Conducted entire experimental setup and user testing process for evaluation.

- Seungjoo Lee
  - Suggested using sound signal to identify the grasp posture.
  - Experimented with recorded sound signal using various methods like spectrogram, FFT to determine how to extract relevent features.
  - Implemented an android code that performs FFT and extracts feature from them.

- HyungJun Yoon
  - Designed the FMCW audio structure and implemented the code to output the audio.
  - Implemented an android code that outputs an audio signal from the speaker and records it on two microphones.
  - Set up the environment for machine learning training and testing through scikit-learn, and conducted coding and evaluation for SVM / random forest / logistic regression.
  - Implemented the machine learning part of GraspTracker android code.

REFERENCES

[1] Oda F Ackerman E. 1962. *Acoustic absorption coefficients of human body surfaces.* Technical Report.

[2] Lung-Pan Cheng, Meng Han Lee, Che-Yang Wu, Fang-I Hsiao, Yen-Ting Liu, Hsiang-Sheng Liang, Yi-Ching Chiu, Ming-Sui Lee, and Mike Y Chen. 2013. iRotateGrasp: automatic screen rotation based on grasp of mobile devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 3051–3054.

[3] Lung-Pan Cheng, Hsiang-Sheng Liang, Che-Yang Wu, and Mike Y. Chen. 2013. iGrasp: Grasp-based Adaptive Keyboard for Mobile Devices. In *CHI '13 Extended Abstracts on Human Factors in Computing Systems (CHI EA '13)*. ACM, New York, NY, USA, 2791–2792. https://doi.org/10.1145/2468356.2479514

[4] Ewins D. 2000. *Modal testing: theory, practice, and application.* Research Studies Press.

[5] Mayank Goel, Leah Findlater, and Jacob Wobbrock. 2012. WalkType: using accelerometer data to accomodate situational impairments in mobile touch screen text entry. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2687–2696.

[6] Mayank Goel, Jacob Wobbrock, and Shwetak Patel. 2012. GripSense: using built-in sensors to detect hand posture and pressure on commodity mobile phones. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*. ACM, 545–554.

[7] Beverly Harrison, Kenneth P Fishkin, Anuj Gujar, and Carlos Mochon. 1998. Squeeze me, hold me, tilt me! An exploration of manipulative user interfaces. In *In Proceedings of CHI'98*. Citeseer.

[8] Kee-Eung Kim, Wook Chang, Sung-Jung Cho, Junghyun Shim, Hyunjeong Lee, Joonah Park, Youngbeom Lee, and Sangryoung Kim. 2006. Hand grip pattern recognition for mobile user interfaces. In *Proceedings of the National Conference on Artificial Intelligence*, Vol. 21. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 1789.

[9] Namhyun Kim, Junseong Lee, Joyce Jiyoung Whang, and Jinkyu Lee. 2019. SmartGrip: grip sensing system for commodity mobile devices through sound signals. *Personal and Ubiquitous Computing* (2019), 1–12.

[10] Makoto Ono, Buntarou Shizuki, and Jiro Tanaka. 2013. Touch & activate: adding interactivity to existing objects using active acoustic sensing. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*. ACM, 31–40.

[11] Richardson MH Schwarz BJ. 1999. *Experimental modal analysis.* CSI Reliability week 35(1):1–12.

[12] Brandon T Taylor and V Michael Bove Jr. 2009. Graspables: grasp-recognition as a user interface. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 917–926.

[13] Raphael Wimmer and Sebastian Boring. 2009. HandSense: discriminating different ways of grasping and holding a tangible user interface. In *Proceedings of the 3rd International Conference on Tangible and Embedded Interaction*. ACM, 359–362.