# AirWare: Large Vocabulary, In-air Gesture Recognition with IR Proximity and Audio Doppler on Smartphones

**ABSTRACT**
Placeholder for the 150-word abstract.

*Placeholder for figure of app and user performing a gesture*

**Figure 1. Placeholder caption.**

**Author Keywords**
Doppler Sensing, Gesture Recognition, Infrared Proximity Sensing,

**ACM Classification Keywords**
H.5.2 User Interfaces: Input devices and strategies, Interaction styles; I.3.6 Methodology and Techniques: Interaction techniques

## INTRODUCTION AND MOTIVATION

Identifying in-air gestures on mobile devices has a rich history and has recently seen considerable attention by the UbiComp research community. The main premise for using in-air gestures is that they dramatically increase the input modality for mobile devices [cite]. This becomes important (1) when the mobile device is small and touch is harder to use without occluding screen content (such as a watch) or (2) in situational impairments, like wearing gloves or cooking, when hands get dirty and touching a smartphone or tablet is not desired [cite].

Dating back to XXXX, in-air gesture sensing was achieved using commodity cameras with a high degree of success [cite]. The RGB image of a user-facing camera was used to detect and follow hand movements. Even so, privacy concerns and the requirements of processing video (battery life, lag time) limited the impact of the technology [cite]. To mitigate these concerns, researchers have been innovative in how they sense hand motions. In Samsung's Galaxy S4 and S5, a dedicated infrared proximity sensor is used to detect hand motions above the phone, sensing velocity, angle, and relative distance of hand movements. The estimation is coarse, but allows for a number of panning gestures. Gupta *et al*. [cite] used an inaudible tone played on speakers and sensed the Doppler reflections to determine when a user moved their hands toward or away from the phone interface. Aumi *et al*. [cite], Sun *et al*. [spatacus], and Chen *et al*. [cite] extended this work to detect pointing and flick gestures toward an array of objects (including smartphones) using the Doppler effect. There have also been a number of innovative solutions that use infrared illuminating pendants, magnetic sensors [Keyu], side mounted light sensors [cite butler and hodges], and even the unmodified GSM antenna radiation [reynolds].

While previous work has shown the myriad of sensing possibilities, none of the technologies has simultaneously been able to (1) take advantage of embedded sensors on the smartphone *and* (2) provide a rich vocabulary of in-air gestures. The current gesture vocabularies only cover basic interactions (like panning), which does not provide a rich interaction modality, especially for various applications like gaming. To help bridge this gap, we present a prototype system, AirWare, that fuses the information from the onboard IR proximity sensor of a Samsung Galaxy S5 with the Doppler shifts detected by the microphone. Like previous work [cite, cite] we play an inaudible sine wave tone from the speakers and record from the microphone continuously. Using signal processing and machine learning we create a gesture profile and fuse the parameters of the S4 with the Doppler features to predict a vocabulary of 21 different in-air gestures. We conducted a user study with 8 participants that performed each gesture 10 times (load balanced in terms of presentation order). We show that, on average, AirWare can recognize the full gesture vocabulary with XX% accuracy and a reduced vocabulary of XX gestures, with YY% accuracy. Furthermore, we show that calibration can increase the accuracy of the gesture recognition to ZZ%. Our contributions are three-fold:

- We fuse Doppler gesture sensing techniques with the embedded IR proximity sensor input to achieve a large vocabulary of in-air gestures
- We validate the technology in a user study with ZZ participants, where participants are free to interpret how they perform different gestures
- We investigate personalized calibration to boost the recognition accuracy of the classifier, as well as providing an out-of-the-box gesture recognition system

## RELATED WORK

We present an abridged version of the related work. As discussed our work has overlap with the work of others using audible Doppler sensing [cite all of them]. However, AirWare is more ambitious in terms of the number of gestures we are attempting to classify, as well as different in terms of the fused sensor outputs investigated. Clearly, though, out work is influenced and informed by past approaches to Doppler audio sensing.

Raj *et al*. review the HCI uses of Doppler sensing from Ultrasonic range finders [cite]. Although this requires an extra sensor, it uses many of the same techniques as audio Doppler sensing. By using a set of "pings" in the environment the proximity of the hand (or any object) can be ascertained with high accuracy. AirWare shares some similarity in sensing techniques as we also employ a proximity sensor. Even so, our IR proximity sensor is less sensitive than the ultrasonic sensor and is embedded inside the Galaxy S5 smartphone.

We also take inspiration from many generic in-air gesture systems such as Song *et al*. [cite]. In this work, the rear facing RGB camera was used to detect eight different gestures behind the phone, in addition to tracking the hand position. The accuracy of the classification is quite robust (above 93% for all gestures) but the authors note that lighting conditions and processing time were confounding factors. NanoGest [cite], a gesture recognition SDK for iOS and Android, uses the front facing camera to detect four swipe gestures and a "wave." In [cite], Hilliges *et al*. user under mounted table top cameras to identify hand gestures above (and on) a semi-transparent surface. A thorough review of camera based gesture recognition can be found in [rautary]. Depth cameras have also been used extensively for in-air gesture recognition. Suarez and Murphy [cite] have thoroughly reviewed work in this area.

Though each of these works share similar aspect to AirWare, none of the previous works leverage the same sensors as AirWare nor do they attempt as large a gesture vocabulary as AirWare. This is an important aspect because it defines the privacy and potential impact of the gesture sensing technology. By using the on-board sensors of a smartphone, we ensure the technology remains low-cost and is highly scalable. By attempting a large vocabulary of gestures, the bounds of the system are more clearly evaluated.

*Placeholder for figure of zoomed spectrogram and normalized gestures. Need callouts of the frequency ranges used.*

**Figure 2. Placeholder caption.**

## THEORY OF OPERATION

Audio Doppler sensing is discussed in detail in a number of papers [cite]. Our method generally follows that of other papers. We play a 17 kHz sine wave from the speakers of the mobile phone, while continuously sampling from the microphone at 48kHz. When an object moves toward or away form a stationary phone, the microphone can detect Doppler frequency reflections given by:

$$\Delta f = f_0 \frac{v}{c}$$

where $c$ is the speed of sound, $v$ is the velocity of the hand, and $f_0$ is the frequency of the sine wave played from the speakers. Like previous approaches, we use the short time Fourier transform (STFT) to capture changes in the frequency over time. Specifically, we use a sampling rate of 48kHz, window size of 4096 samples (~85ms long, frequency resolution of 11.7Hz), hamming window, and overlap between windows of 3000 samples (~62ms). We save the STFT for one second of time data (discarding the initial startup windows), resulting in 40 frames of FFT data. An example of the STFT can be seen in Figure XX. To normalize and control dynamic range, we take the decibel magnitude of the STFT.

### Feature Extraction

In this stage, our aim is to create features from the STFT over one second that are representative of the gesture being performed. We start by finding the magnitude of the sine wave tone cross the entire STFT, $M_{0dB}(t)$, where t denotes the frame number {0, 1, … 39}. To estimate the change in frequency above and below the tone we take the maximum value of the magnitude in a range 23Hz to 46Hz above and below $f_0$, denoted as $M_{+dB}(t)$ and $M_{-dB}(t)$ for the high and low ranges, respectively. These ranges are highlighted in Figure XX. We then normalize the values using:

$$M_{\pm}(t) = \frac{M_{\pm dB}(t)}{M_{0dB}(t)}$$

to create features for the machine learning that are each 40 frames long. These values are also shown in Figure XX, callout. During preprocessing, we time align the features by taking the maximum of the feature $M_{\pm}(t)$, which corresponds to the point of maximum frequency deviation during the gesture. We keep two frames leading up the maximum of the frequency and five frames after the maximum, reducing the number of frames in each $M_{\pm}(t)$ to eight (~190ms). This results in 16 features from the STFT.

After processing the STFT, we process the features extracted from the Samsung Galaxy S5's Air sensor. The sensor uses a "push" style API where the app subscribes to notifications when the sensor is tripped. The notification includes the speed and angle of any detected movements (the angle is an average of the entering and exit angles). We post process this data during a one second window while the gesture is being performed. Every time we are notified of a movement, we increment one of four different integer values (up, down, left, right) based upon the angle of the movement. If the entering angle is between -45 and 45 degrees, we increment the "down" value. Other values are incremented by dividing the 360 angles into equal chunks of 90 degrees. We also save the relative speed of the gesture (an integer between ~40 and 100). This results in 5 features that we add to the existing 16 to get 21 features representing a gesture.

### Machine Learning Classification

We train a machine learning algorithm using the given features and labels from our experimental methodology (there are XX gestures, as explained in the next sections). We chose to use a support vector machine (SVM) classifier as implemented using python's *scikit-learn* [cite]. When selecting the parameters of the SVM we employ grid searching using 5-fold cross validation of the *training* data. That is, we divide the current training fold into 5 folds for running a grid search of the slack variable cost, kernel function, and gamma weighting parameter [cite] (test fold data is *never* used in the grid search or training of the SVM). We use accuracy to score the parameters of the grid search. We found from the grid searching that parameters are mostly chosen with low cost and a linear SVM kernel. Also note that we use different cross validation strategies to choose training and testing folds as explained in the results section.

## EXPERIMENTAL METHODOLOGY

To inform the design and validate the results of the machine learning classification, we conducted a user study with XX participants. Participants were recruited from undergraduate University classes and paid for their involvement (age range: XX-YY, Male: XX%). All but one participant was right handed. All participants used a smartphone daily and were familiar with touchscreen gestures. A large percentage (XX%) had also used some type of in-air gesture sensor before (like a Kinect or leap motion). During a session, participants were introduced to the AirWare data collection application. Users were told about each gesture in the vocabulary, but were not instructed on how to perform the gesture. This was done to avoid biasing the way in which a user might perform the gesture. If the participant asked what a gesture meant, we explained the gesture in words, but did not perform the gesture. We also did not instruct users where the different sensors were located on the phone. Participants were instructed to hold the smartphone in their non-dominant hand and perform gestures "above" the phone.

*Placeholder for figure of application with several examples of a subset of the gestures.*

**Figure 3. Placeholder caption.**

### Gesture Vocabulary

Participants performed XX different gestures as instructed on the screen of the phone. A gesture would flash on the screen and the user would perform the in-air gesture (Figure XX). All sensor data was saved to file locally on the phone for later processing. Users went through each gesture one time as practice (practice data not used in analysis) and then were presented with a random permutation of the gestures. In all, each participant performed XX iterations of each gesture for a total of ZZ iterations. The gestures selected were based upon an informal review of other in-air gesture sensing systems. They were: flick left/right/up/down (quick hand movement from wrist), pan left/right/up/down (hand flat, movement from elbow), push, pull, double tap (whole hand), whip, pluck, put, click (with finger, but not touching screen), double click (with finger), wobble (hand wobbled), circle, and erase (moving hand back and forth). Clearly, we expect considerable similarity in the chosen gesture vocabulary. However, we also chose a reduced vocabulary consisting of the following gestures: XX, YY, ZZ, etc. <fill in gestures>

## RESULTS

We break down our results by two variables: (1) if user calibration was used (across-subject versus within-subject model training) and (2) by gesture vocabulary size (full versus reduced). The across-subject training (*i.e.*, no calibration) uses leave-one-subject-out to train and test the machine learning model. Therefore, the model is trained without *any* data from a participant under test. The within-subject model only uses data from a given participant and training and testing split is achieved using three-fold cross validation. This is meant to simulate how the system might perform using training folds as data from a calibration protocol.

<insert results here>

## DISCUSSION AND LIMITATIONS

Though the results are promising there are limitations to our methodology. Firstly, participants were asked to perform gestures in a randomized order. They were not asked to complete a task using the gesture. As such, it is unknown if the user might begin altering the way they perform the gesture as they become more fatigued or are simply concentrating on the task and not the gesture. Secondly, participants were not trained to use the given in-air gesture vocabulary. With some initial training, it should be possible to make the gesture more consistent by explaining where the IR proximity sensor is located on the phone and having them perform some example gestures. Of course, by biasing the way the user performs the gesture, this might be at the cost of making the gesture less "natural feeling." To investigate how accurate someone might perform the gestures, we recruited two participants and showed them how to perform each gesture. Using within-subject training for these new participants we found that accuracy was boosted to XX% and YY% for each user on the large vocabulary gesture set.

Another limitation of the technology is user adoption. In-air gesture sensing, in general, has not yet caught on by "mainstream" smartphone users [cite]. This could be for a number of reasons including user fatigue, awkwardness, unfamiliarity, and past experience [cite hassani], which are general difficulties for any multi-modal interface [cite Dumas]. Even so, we argue that multi-modal interfaces are still exploratory (especially for mobile phones) and that time will tell the true impact of gesture sets such as the one discussed here.

## CONCLUSION

In conclusion, we presented AirWare, a technology that fuses the output of an embedded smartphone microphone and proximity sensor to recognize a gesture set of ZZ in-air gestures with XX% accuracy. Our user study indicates that a reduced vocabulary recognition system can be created using a generalizing model, but calibration per user is required to fully realize all XX gestures with high reliability.

## REFERENCES

1. Chen, K. Y., Ashbrook, D., Goel, M., Lee, S. H., & Patel, S. (2014, September). AirLink: sharing files between multiple devices using in-air gestures. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing* (pp. 565-569).
2. Zhao, C., Chen, K. Y., Aumi, M., Patel, S., & Reynolds, M. S. (2014, October). SideSwipe: detecting in-air gestures around mobile devices using actual GSM signal. *In Proceedings of the 27th annual ACM symposium on User interface software and technology* (pp. 527-534).
3. Butler, A., Izadi, S., & Hodges, S. (2008, October). SideSight: multi-touch interaction around small devices. *In Proceedings of the 21st annual ACM symposium on User interface software and technology* (pp. 201-204).
4. Starner, T., Auxier, J., Ashbrook, D., & Gandy, M. (2000, October). The gesture pendant: A self-illuminating, wearable, infrared computer vision system for home automation control and medical monitoring. *In Wearable computers, the fourth international symposium on* (pp. 87-94).
5. NanoGest: Air gesture recognition for iOS and Android. Webpage, accessed February 2015: http://www.nanocritical.com/nanogest/
6. Song, J., Sörös, G., Pece, F., Fanello, S. R., Izadi, S., Keskin, C., & Hilliges, O. (2014, October). In-air gestures around unmodified mobile devices. *In Proceedings of the 27th annual ACM symposium on User interface software and technology* (pp. 319-329).
7. Elgan, Mike (2014).Why In-theAir Gestures are Failing as a Mainstream User Interface, Editorial. Webpage accessed February 2015: http://www.eweek.com/mobile/why-in-the-air-gestures-are-failing-as-a-mainstream-user-interface.html
8. Hilliges, O., Izadi, S., Wilson, A. D., Hodges, S., Garcia-Mendoza, A., & Butz, A. (2009, October). Interactions in the air: adding further depth to interactive tabletops. *In Proceedings of the 22nd annual ACM symposium on User interface software and technology* (pp. 139-148).
9. Suarez, J., & Murphy, R. R. (2012, September). Hand gesture recognition with depth images: A review. *In RO-MAN, 2012 IEEE* (pp. 411-417).
10. Hassani, A. Z., van Dijk, B., Ludden, G., & Eertink, H. (2011). Touch versus in-air hand gestures: evaluating the acceptance by seniors of human-robot interaction. In *Ambient Intelligence* (pp. 309-313). Springer Berlin Heidelberg.
11. Löcken, A., Hesselmann, T., Pielot, M., Henze, N., & Boll, S. (2012). User-centred process for the definition of free-hand gestures applied to controlling music playback. *Multimedia systems*, *18*(1), 15-31.
12. Dumas, B., Lalanne, D., & Oviatt, S. (2009). Multimodal interfaces: A survey of principles, models and frameworks. In *Human Machine Interaction* (pp. 3-26). Springer Berlin Heidelberg.
13. Hinckley, K. (2003, November). Synchronous gestures for multiple persons and computers. In *Proceedings of the 16th annual ACM symposium on User interface software and technology* (pp. 149-158).
14. Sun, Z., Purohit, A., Bose, R., & Zhang, P. (2013, June). Spartacus: spatially-aware interaction for mobile devices

through energy-efficient audio sensing. In *Proceeding of the 11th annual international conference on Mobile systems, applications, and services* (pp. 263-276).

15. Gupta, S., Morris, D., Patel, S., & Tan, D. (2012, May). Soundwave: using the doppler effect to sense gestures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 1911-1914).
16. Aumi, M. T. I., Gupta, S., Goel, M., Larson, E., & Patel, S. (2013, September). Doplink: Using the doppler effect for multi-device interaction. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing* (pp. 583-586).
17. Bannis, A., Pan, S., & Zhang, P. (2014, September). Adding directional context to gestures using doppler effect. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication* (pp. 5-8).
18. Raj, B., Kalgaonkar, K., Harrison, C., & Dietz, P. (2012). Ultrasonic Doppler sensing in HCI. *IEEE Pervasive Computing*, (2), 24-29.
19. Chen, K. Y., Lyons, K., White, S., & Patel, S. (2013, October). uTrack: 3D input using two magnetic sensors. In *Proceedings of the 26th annual ACM symposium on User interface software and technology* (pp. 237-244).
20. Rubine, D. H. (1991). *The automatic recognition of gestures* (Doctoral dissertation, University of Toronto).
21. Rautaray, S. S., & Agrawal, A. (2015). Vision based hand gesture recognition for human computer interaction: a survey. *Artificial Intelligence Review*, *43*(1), 1-54.
22. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. *The Journal of Machine Learning Research*, *12*, 2825-2830.
23. Komer, B., Bergstra, J., & Eliasmith, C. (2014). Hyperopt-sklearn: Automatic hyperparameter configuration for scikit-learn. In *ICML workshop on AutoML*.
24. Wobbrock, J. O. (2006, April). The future of mobile device research in HCI. In *CHI 2006 workshop proceedings: what is the next generation of human-computer interaction* (pp. 131-134).