# Activity and Device Position Recognition In Mobile Devices

**Lenny Grokop** **Anthony Sarah** **Chris Brunner** **Vidya Narayanan** **Sanjiv Nanda**
Qualcomm, Incorporated
{lgrokop, asarah, brunnerc, vidyan, snanda}@qualcomm.com

## ABSTRACT
Activity recognition along with device position recognition can provide contextual cues suitable to infer user interruptibility and device accessibility. Our system fuses data from accelerometer and multiple light sensors to classify activities and device positions. Previously published results achieve robust activity recognition performance with multiple sensors attached to fixed body positions, a model suitable for use cases such as healthcare and fitness. We achieve comparable activity recognition performance using smartphones placed in unknown on-body positions including pocket, holster and hand. Results obtained from a diverse data set show that motion state and device position are classified with macro-averaged f-scores 92.6% and 66.8% respectively, over six activities and seven device positions. We demonstrate the performance of our classifier with an implementation running on the Android platform, that visitors can try out.

**Author Keywords** Activity recognition, device position, sensor fusion, context awareness

**ACM Classification Keywords** I.2.11 [Computing Methodologies]:Distributed Artificial Intelligence – intelligent agents.

**General Terms** Algorithms

## INTRODUCTION
Activity recognition typically refers solely to classifying the *motion state* of the user (eg. stationary, walking, running, etc.). When building an activity recognition system for mobile devices, the motion classifier must be robust to different device placements (e.g. in a pant pocket, bag, hand, on desk, etc.). The device position itself can be used to infer accessibility, estimate access time, estimate latency in user response, etc. and hence provides more context.

The system we have developed uses the accelerometer and various light sensors to classify both the users motion state and their device position. The architecture is versatile in that the system can be straightforwardly retrained to accommodate a different set of motion states or device positions. It can also freely adapt to changes in the desired latency or ROC operating point, on the fly. Most of the technical machinery that we have employed comes in the form of statistical learning techniques for training models of the sensor data. This is done offline using labeled activity data that we have collected from various test subjects.

Copyright is held by the author/owner(s).
*UbiComp'11,* September 17–21, 2011, Beijing, China.
ACM 978-1-4503-0630-0/11/09.

## RELATED WORK
Traditional wearable computing applications have tackled the activity recognition problem for sensors placed at fixed locations on the body [1,3,6,7,8]. These cases lead to certain simplifications that are not feasible for activity recognition solutions that operate on mobile devices, without placing additional requirements on the user. Some efforts to characterize device position for wearable sensor applications have be made in [4,5,9]. However, the positions are inferred in these when the sensors are fixed to some location on the body. More recently, there has been some work done with loosely placed device positions [10]. This work, while using the positions to classify the activities, does not output the device position itself and presents no results on position classification.

## PROBLEM FORMULATION
Motion state is to be classified into one of $K_M = 6$ categories: *walk, run, sit, stand, fiddle* and *rest*. In this instance, we define fiddle as any significant movement of the device in the users hand (presumably when it is not in use). Device position is to be classified into one of $K_P = 7$ categories: *shirt pocket, pant pocket, hand, bag, holster, desk* and *jacket pocket*. The desk class is defined as the device being at rest on a flat surface. These motion state and device position classes were chosen to span a range of common states that make up the bulk of daily activity.

The system is to produce classification outputs at a certain frequency, and with a certain latency. An intrinsic tradeoff exists between latency and accuracy as given the user state is constant, the larger the observation window the lower the probability of state confusion. Different applications thus have different latency/accuracy targets and the tradeoff can be appropriately determined based on the application.

## SOLUTION
The architecture of our system is summarized in **Figure 1**.

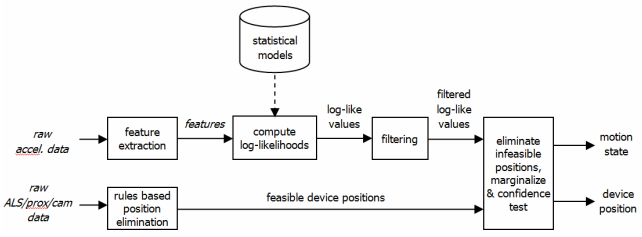


**Figure 1: System architecture**

### Overview
The primary branch processes accelerometer data and outputs both motion state and device position estimates.

The auxiliary branch processes the ALS, proximity sensor, and camera intensity data, to refine the position estimates.

At the heart of the primary branch is a Bayesian classifier. From every 1 second of raw accelerometer data, a feature vector is computed. The feature vectors are modeled using a Gaussian Mixture Model (GMM). The parameters of the GMM are learned offline from training data collected. The features used include some non-traditional ones such as MFCCs [2] that are seen to provide performance gains for both motion state and device position classification.

We view the class as a tuple (*motion state*, *device position*). We refer to the tuple elements as *subclasses*. We classify motion state by *marginalizing* over device position and vice-versa. For reliability, a confidence metric is computed and a decision is outputted from the classifier only when the confidence metric is above a threshold.

To accommodate multiple applications that have different latency targets, we combine classifier outputs to make an overall decision. This is done by filtering the log-likelihood values over time. This is computationally and conceptually simple, and is optimal for a constant user state.

The data from the ALS, proximity sensor, and camera intensity can be very helpful in determining the device position. The secondary branch examines the outputs of these sensors, and decides based on rules, whether one face of the device is occluded, both faces are occluded, or neither face is occluded. This information is then used to construct a list of feasible device positions.

The reason for the use of hard-coded rules in the secondary branch, as opposed to the machine learning/feature based approach adopted in the primary branch, is twofold. Firstly, the outputs to the light sensors available in Android, are fickle, with the ALS and proximity outputs being heavily quantized, and the camera output being controlled by an AGC mechanism that could not be disabled. Secondly, training a statistical model based on lighting conditions is a problematic endeavor. The results would be heavily dependent on alignment between the lighting conditions during training, and the lighting conditions present during system operation. In the current approach, the light sensors help greatly in a wide range of lighting conditions without hindering performance in the absence of light.

## DATASETS

Accelerometer, ALS, proximity sensor, and camera intensity were logged using a custom application developed on Android. Collection was performed using a total of 4 Android devices of varied models. Data was resampled to 60Hz in post-processing due to differing sampling rates.

Data was collected across a total of 9 subjects of various ages, composed of 8 men and 1 woman. Each subject performed two trials of each viable motion state/device position combination, with all of the devices.

## EXPERIMENTS AND RESULTS

The classifier is implemented on Android and is robust in uncontrolled settings. The leave-one-out cross-validation technique is used for evaluation. The train set consists of data from all subjects but one and the test set consists of the remaining subject. Results are generated for each combination of left-out subject and then averaged. With a latency target of 10s and 20% of the decisions discarded, we achieve a macro-averaged F-score of 92.6% and 66.8% for activity and device position recognition respectively.

## ACKNOWLEDGEMENTS

The authors would like to acknowledge the work of Abhijeet Bisain, Jim Dolter and Murali Akula in the development of the classifier.

## REFERENCES

1. Bao L. and Intille S., "Activity recognition from user-annotated acceleration data," in *Pervasive Computing*, Springer, 2004.
2. Deller J.R., Hansen J.H.L. and Proakis J.G., *Discrete-Time Processing of Speech Signals,* Wiley - IEEE Press, 1999.
3. Huynh T., et. al. "Discovery of activity patterns using topic models*," Proc. Ubicomp*, Seoul, Sep. 2008.
4. Kunze K. and Lukowicz P., "Dealing with sensor displacement in motion-based onbody activity recognition systems," *Proc. Ubicomp*, Seoul, Sep. 2008.
5. Kunze K. and P. Lukowicz, "Using acceleration signatures from everyday activities for on-body device location," in *Proc. 11th Symposium on Wearable Computers*, Oct 2007.
6. Lee S.W and Mase K., "Activity and location recognition using wearable sensors," in *IEEE Pervasive Computing*, vol 1., iss 3., pp. 24-32, Dec 2002.
7. Ravi, et. al, "Activity Recognition from Accelerometer Data," in *Proc. AAAI*, 2005.
8. van Kasteren T., et. al., "Accurate activity recognition in a home setting," *Proc. Ubicomp*, Seoul, Sep. 2008.
9. Vahdatpour A., et. al., "On-body Device Localization for Health and Medical Monitoring Applications," *Proc. PerCom,* Seattle, Mar. 2011.
10. Lu H., et. al., "The Jigsaw Continuous Sensing Engine for Mobile Phone Applications," *Proc. SenSys,* Zurich, Nov. 2010