Advancing Android Activity Recognition Service with Markov Smoother

Mingyang Zhong*‡, Jiahui Wen*‡, Peizhao Hu†, Jadwiga Indulska*‡
*The University of Queensland, Australia
School of Information Technology and Electrical Engineering
†Rochester Institute of Technology, USA
‡National ICT Australia (NICTA)
Email: mingyang.zhong@uq.net.au

***Abstract*—The rapid market shift to multi-functional mobile devices has created an opportunity to support activity recognition using the on-board sensors of these devices. Over the last decade, many activity recognition approaches have been proposed for various activities in different settings. Wearable sensors and augmented environments potentially have better accuracy, however performing activity recognition on user mobile devices has also attracted significant attention. This is because of less requirements on the environments and easier application deployment. Many solutions have been proposed by academia, but practical use is limited to testbed experiments. In 2013, Google released an activity recognition service on Android, putting this technology to the test. With its enormous market share, the impact is significant. In this paper, we present a systematic evaluation of this activity recognition service and share the lesson learnt. Through our experiments, we found scenarios in which the recognition accuracy was barely acceptable. To improve its accuracy, we developed ARshell in which we apply a Markov smoother to post-process the results generated by the recognition service. Our evaluation experiments show significant improvement in accuracy when compared to the original results. As a contribution to the community, we open-sourced ARshell on GitHub for application developers who are interested in this activity recognition service.**

I. Introduction

The number of mobile-connected devices grew to 7 billion in 2013, with over half a billion in one year [1]. The miniature, but powerful, multi-functional mobile devices have boosted the market shift from desktop to 'thin' client devices. Along with their slim design, many of these mobile devices are equipped with on-board sensors with various sensing capabilities, including location, acceleration, and orientation.

These added capabilities have enabled new uses of mobile devices. One of them, mobile device based activity recognition, have been an active area of research for almost a decade [2]. Many existing approaches are based on customised wearable sensors [3], environment augmentation [4] or their combination [5]. However, there is also a considerable amount of work on activity recognition based on sensor data from mobile devices. The typical procedures of activity recognition include data collection, feature extraction, classifier training, and activity recognition on a test dataset. Majority of the approaches to activity recognition make use of machine learning techniques to map patterns embedded in sensed data to human activities. To name a few, they include Naive Bayesian, Bayesian Network, Hidden Markov Model, Decision Tree, and Support Vector Machine. There are also hybrid models that combine multiple simple models to further improve recognition accuracy, such as [6], [7].

Developing accurate activity recognition algorithms requires the background, which is not a *must have* tool for developers. To simplify the use of these techniques, Google announced its Android activity recognition (AR) services in 2013. Therefore, rather than to go through the whole process of data collection, feature extraction and classifier training, software developers can utilise this AR service through an API. Initially four types of activities were supported: *Stationary*, *On Foot*, *Cycling*, *In Vehicle* and *Unknown*. In an update, three more activities were added: *Walking*, *Running* and *Tilting*. According to its documentations[1], the Android AR service makes use of low-power, on-board sensors to recognise the user's current physical activity with efficient energy consumption.

In this paper, we present a systematic qualitative and quantitative evaluations of this AR service, with a goal to investigate its accuracy, latency and complexity. Based on other referenced sources, together with our experiments, we demonstrate scenarios in which this AR service will perform poorly. We then propose our solution – ARshell – a post-processing step which uses a Markov smoother to improve the overall accuracy up to 19% across all recognition categories. We released ARshell as an open-source code on GitHub as a contribution to researchers and developers who might be interested in this Android AR service. In addition to improving the overall accuracy of the Android AR service, the ARshell API simplifies the task of using the AR service in Android related research projects and application development.

There are two contributions presented in the paper:

- A systematic evaluation of the Android AR service;
- An effective and lightweight post-processing method to significantly improve the AR accuracy.

The remainder of this paper is organised as follows. Section II presents the related work on activity recognition. Section III describes the evaluations of the Android AR service. This is followed by our proposed solution – ARshell – in Section IV. Section V concludes the paper.

[1] http://developer.android.com/training/location/activity-recognition.html

II. Related work

Earlier approaches in AR usually used multiple sensors attached at various positions of a human body, and researchers can characterise the specific movement of a region of human body with sensor data from that region. Bao et al. [8] proposed a method to recognise physical activities with multiple sensors attached at various positions on a human subject. The participants were asked to perform daily activities and the samples were manually annotated. The authors extract features from the annotated-data such as mean, energy and frequency-domain entropy, and then train and test multiple classifiers and find a Decision Tree to achieve the highest accuracy. Using wearable sensors leads to good activity recognition accuracy, however attaching multiple sensors on human subject is cumbersome and not practical to be used on a large scale.

In recent years, with the proliferation of mobile devices, especially smartphones, much effort has been made to leverage their on-board sensors for activity recognition. Smartphones usually incorporate multiple sensors including GPS, camera, microphone, accelerometers, light sensor and proximity sensor etc. Using these on-board sensors, mobile devices offer an opportunity for data mining sensor readings in order to provide activity recognition. There exist many proposals on performing activity recognition with mobile devices. Kwapisz et al. [9] presented the activity recognition system that collected labelled accelerometer data from 29 participants engaged in physical activities such as *walking*, *jogging*, *climbing stairs*, *sitting* and *standing* with the single device in their pockets. The time series data was aggregated into 10 s interval samples and features such as *mean*, *standard deviation*, *average absolute difference*, *average resultant acceleration*, *time between peaks* and *binned distribution* were extracted based on the collected data. Finally, multiple predictive models (decision tree, logistic regression, multi-layer neural networks) have been built for activity recognition. The authors concluded that walking and jogging achieve the highest accuracy while walking downstairs and upstairs are the most difficult ones to distinguish due to their similar patterns. Shoaib et al. [10] explored the role of gyroscope and magnetometer on smartphones for activity recognition. They experimented on four body positions using seven classifiers while recognizing six physical activities. They concluded that accelerometer and gyroscope complement each other in general, however magnetometer is not so promising due to its dependence on directions. Compared with the solutions for activity recognition on non-mobile devices, the recognition is relatively less accurate due to the limited number and type of sensors on mobile devices.

The previous works also proposed a variety of techniques to address the issues that come with the application of smartphones. For example, Hemminki et al.[11] proposed to extract gravity eliminated horizontal acceleration in order to achieve orientation-robustness considering the free-carrying of mobile phones. To be energy-efficient, the authors of [12] proposed an energy-efficient activity recognition based on prediction. The current and historical contextual information has been used to predict the possible future activities, and only a subset of the sensors are activated to distinguish the activities that are likely to happen. Maekawa et al. [13] addressed the issue of scalability of activity recognition. They employed the end user information to find other users with similar sensor data, and modelled the activities of the end user in an unsupervised way with the data from those similar users. In [14], the authors proposed a method to deal with personalization by training general and user-specific classifiers and using a meta-classifier to determine which classifier is more likely to provide correct predictions.

Although a substantial amount of research has been carried out on activity recognition on mobile devices, there was no activity recognition service available for Android application developers before the Android AR service. This service has the potential to revolutionise the development of mobile applications that offer better user experience. However the usefulness of this AR service depends on its accuracy, latency and complexity.

III. Evaluation of Android Activity Recognition

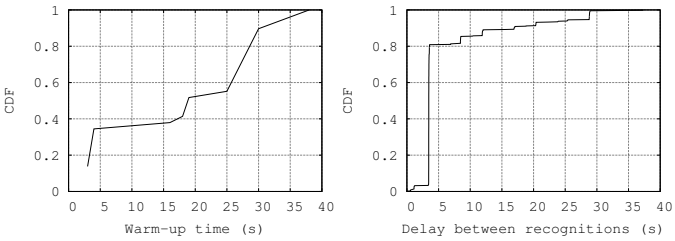
Google's support for the AR service on the Android operating system provides access to activity recognition for developers of Android applications. Through the AR API, mobile apps can incorporate activity recognition without dealing with complexity of pattern analysis on raw sensor data. According to the documentation, the AR service is bundled together with the location services and is part of the Google Play services APK. To access the AR service, a mobile app must be granted with a special permission[2], and be connected to the Play services APK. The recognition results are sent as updates through a callback function. Similar to the location update service in Android, mobile apps can specify a preferred update interval. Whenever a location update is available, the Android OS will trigger the callback function with the new location value. In addition, mobile apps should define methods for *starting* and *stopping* of the service and error handling. We perform our evaluations using a demo code provided, with an additional code for recording measurements.

A. Experimental setup

All of the experiments are carried out on actual Android devices. The details of a setup are as follows:

- Android devices: HTC Desire C with 600 Mhz processor and 512 MB memory, Samsung Galaxy Nexus with 1.2 GHZ dual-core processor and 1 GB memory, and Samsung NOTE II with 1.6 Ghz quad-core processor and 2 GB memory,
- Experiment duration: the experiments were carried out by the authors over two weeks using the aforementioned devices in various scenarios (*Stationary*, *Walking*, *Running*, *Cycling*, and *In Vehicle*). As for *Tilting*, it measures the relative change in gravity and is an instantaneous motion appearing in our measurement results as outlier.

[2]com.google.android.gms.permission.ACTIVITY_RECOGNITION



(a) Delay before 1_{st} data point (b) Delay between consecutive data

Fig. 1. CDF of two aspects of delay

Therefore, we treat *Tilting* as noise in our experiments. All the experimental data was recorded with a human label (as the ground truth). We sampled the data outputted from the Android AR service every second, and we collected data for two hours for each activity.

B. Delay

The update *interval* parameter is designed to be a trade-off to balance between freshness of the measurement values and power consumption. Developers should choose an appropriate value according to their application requirements. To better understand the delay characteristic of the AR service, we conducted experiments to investigate the *warm-up* time, which is the delay before the first recognised event from the AR service, and the delay between two consecutive recognised events reported by the AR service.

Fig. 1 shows the Cumulative Distribution Function (CDF) of the two aspects of delay. As for the warm-up time, we conducted multiple experiments to measure the average warm-up time and standard deviation. The results are 18.3 s and 13.9 s, respectively. Furthermore, Fig. 1(a) shows that the warm-up time can be as long as 30 s or more in about 10% of our experiments. The minimum wait time from starting the service to receiving the first recognition is on average 3 s.

The sensor data is generated continuously and streamed to the AR service for pattern matching. One might think that we should receive activity update after receiving new sensor data, but it is not the case. Most machine-learning pattern recognition based approaches make use of a window for sampling data. Pattern matching is applied on the sensor data within the scope of this window. Because of the lack of access to the source code of the AR service, we cannot determine the exact window size. Rather, we try to measure the time delay between two consecutive recognised events to approximate the minimum interval software developers can use for receiving activity updates. We conducted experiments with the interval fixed to 0 s, and collected data samples over an hour (roughly one thousand measurements). In our experiments, the absolute minimum delay between receiving two consecutive activity updates was around 0.5 s. As shown in Fig. 1(b), slightly over 80% of the data samples show a delay less than 3.5 s. Furthermore, the minimal delays below 3.4 s are rare cases that happen with the probability less than 4%. In some extreme cases, the delay can be expanded to as long as 35 s.

TABLE I
ACCURACY OF AR SERVICE

	Classified to: S	W	R	F	C	V	U	A_{aggr}
S	**52%**	5%	5%	10%	0	10%	18%	*52%*
W	0	**75%**	4%	**6%**	0	0	15%	*81%*
R	0	7%	**45%**	**28%**	10%	0	10%	*73%*
C	0	0	0	0	**68%**	0	32%	*68%*
V_N	3%	0	0	1%	1%	**88%**	7%	*88%*
V_S	7%	0	0	3%	7%	**41%**	42%	*41%*
A_{avg}								*67%*

Note: S - Stationary, W - Walking, R - Running, F - On Foot
C - Cycling, U - Unknown and Tilting
V_N - In vehicle with normal speed
V_S - In vehicle that is stopping or moves slowly

In another set of experiments, we investigated whether a larger interval will increase the accuracy of recognition. According to our results, different interval settings do not affect the accuracy. Thus, we used 10 s as interval for the rest of our experiments.

C. Accuracy

Accuracy is a direct measure of the usefulness of the recognition algorithm. In this subsection, we evaluate the performance of the Android AR service with various scenarios. Similar to the most of the proposed AR solutions, after collecting enough data samples over the recognition window, the AR service proposes the most probable activity and a list of probable activities, each with an attached confidence value ranging from 1 to 100. That is, $\{(a_{most}, cv_{most}), [(a_1, cv_1), (a_2, cv_2), ..., (a_n, cv_n)]\}$. If an activity a_{most} has confidence value cv_{most} of 100, it implies absolute certainty of the activity and the list of probable activities $[(a_1, cv_1), (a_2, cv_2), ..., (a_n, cv_n)]$ is *null* except for *Walking* and *Running*. As these two activities are the sub-activities of *On Foot*, the cv of one of these two sub-activities can be 100 if the cv of *On Foot* is 100. We gathered results from each activity and collectively presented them in a confusion matrix, as shown in Table I. In the confusion matrix, the first column contains the names of our labelled activities (or activity under test). The last column (indicated as A_{aggr}) is the aggregated accuracy of the AR service on correctly recognizing the activity under test. Columns between the first and last columns are the distribution of a correct and false classification for each activity. The label A_{avg} indicates the overall accuracy across all activities over all experiments.

We computed the accuracy of the AR service by comparing a_{most} with the corresponding human label activity a_{label} in a sequence of measurements. Thus, in a sequence of measurements reported by the AR service the aggregated accuracy, A_{aggr}, for an activity type $c \in C$ is calculated as

$$A_{aggr} = \frac{1}{N} \sum_{n=1}^{N} [a_{most} = a_{label}]$$

where $[a_{most} = a_{label}]$ equals to 1 if it is evaluated to true, 0 otherwise. The average accuracy A_{avg} across all activity types C is calculated as

$$A_{avg} = \frac{1}{C}\sum_{c=1}^{C} A_{aggr}$$

1) Stationary: There are two basic forms of the stationary scenarios: (i) when a user is sitting or standing still while holding the device under test in his/her hand or having it in a pocket, and (ii) when the device under test is rested on a stationary structure, such as a desk. The latter case achieves over 95% of accuracy. But it represents scenarios with no movement at all and is slightly less interesting compared to the former scenarios, which achieve 52% of accuracy. As the AR service achieves reasonable accuracy regarding the latter case, Table I only shows the detail of the former case. As we can see from the table, 48% of mis-classified activities are distributed across other activities. As mentioned in [10], the AR service potentially leverages the accelerometer and the gyroscope for recognizing this activity. We conjecture that these inaccuracies are due to the inability to keep mobile devices absolutely stationary. Software developers should be careful about the meaning of being *Stationary*.

2) On Foot: In the 2014 update, Google further improved the AR service to classify *Walking* and *Running* on top of *On Foot*, potentially with additional sensor inputs. Due to the different speed and motion when people walk or run, we investigated the accuracy of recognizing these two sub-activities. Data shown in the confusion table confirms our doubt. Other activities are reported even when users are running at a constant speed. A standard fallback is to report users being *On Foot* whenever sensor readings are not distinctive and cannot distinguish between the two sub-activities. When running at a lower speed, data reported from the acceleration is not as significant as for fast-running and is not as modest as for walking. Therefore, recognizing *Running* is less accurate when compared to *Walking*. Logically, both *Walking* and *Running* are sub-activities of *On Foot*. We consider *On Foot* as a correct recognition for these two sub-activities.

3) Cycling: As for *Cycling*, we noticed that around 32% of samples have been mis-classified as *Tilting*. *Tilting* reflects that the device angle relative to gravity changed significantly. This often occurs when a device is picked up from a desk or a sitting user stands up. When cycling on uneven roads, vertical acceleration is typically more significant compared to cycling on flat roads. Our experiment results show lower accuracy when travelling on uneven roads. In these cases, a large percentage of data has been mis-classified as *Tilting*.

4) In Vehicle: With regard to *In Vehicle*, the AR service performs better when the vehicle moves with a normal or relatively high speed than when the vehicle is stopping or moves slowly. Therefore, we labelled these two cases separately (labelled the vehicle moving with normal speed as V_N, and the vehicle stopping or moving slowly as V_S) in order to evaluate detailed accuracy. The accuracy of V_N (88%) is better than that of V_S (41%), because the horizontal acceleration for

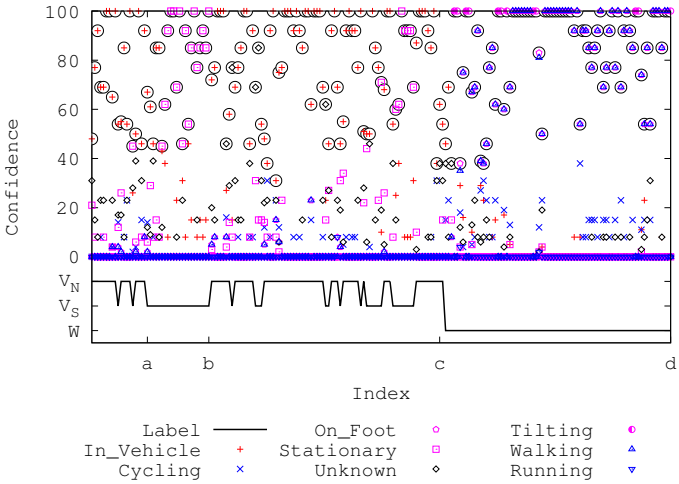


Fig. 2. Experiment on activity transition from *In Vehicle* to *Walking*.

V_S is less noticeable when vehicles are in slow motion or stationary.

After each interval, the AR service reports the list of probable activities, with corresponding confidence values. We conducted a detailed study on what happens when there is a transition from one activity to another. Fig. 2 shows the confidence values of each class of activities in the list of the probable activities reported by the AR service, while the bottom part of the figure presents the real activities labelled by human. We highlighted the most probable activity (with the highest confidence) with circles. We noticed that most mis-classifications for *In Vehicle* occur during the time when a bus is stopping or in stationary (e.g., in $zone1$ between a and b). The AR service reports the user is *Stationary*. It is logically correct. However, since the user is still inside the vehicle, it might be more appropriate to report *In Vehicle*. In $zone2$ (between b and c), we observed a significant increase of *Unknown* reported by the AR service, due to the frequent stopping of the vehicle. This is consistent with our results reported in Table I. At the same time, the confidence values of *In Vehicle* typically drop below 60, which means a high entropy of the posterior distribution. This is one of the insights that we used in our improvement of the AR service to smooth the outliers when detecting uncertainties. In $zone3$ (between c and d), we observed the transition from *In Vehicle* to *Walking* after some delays. We also observed occasional reporting of *Tilting* even when a user was walking on a flat surface road. In the following section, we describe the Markov smoother, which we incorporate to improve the accuracy of recognition.

D. Remarks

Based on our evaluation of the AR service, we share some of our other interesting findings and insights: (i) Because the AR service is bundled as part of the Location service API, our impression was that it might require a GPS device or network access. We conducted experiments with the SIM card removed, WiFi turned off and GPS disabled. These experiments confirmed that the AR service was still functional

and there was no impact on its accuracy. We conjecture that the AR service applies machine learning techniques based on multiple on-board sensors, such as accelerometer, gyroscope and compass. This is consistent with the API documentation; (ii) By adjusting the recognition interval, software developers can balance the trade-off between data freshness and battery consumption; (iii) In another set of experiments, we briefly studied the impact of irregular behaviours. As expected, mis-classifications occurred in all activities when a user shook the device under test in irregular pattern.

IV. ARshell: design and evaluation

To improve the recognition accuracy, we propose ARshell (Activity Recognition Shell), which is a shell that post-processes the results reported from the AR service. According to the aforementioned evaluations, as discussed in Section III, around 21% of errors are mis-classified as *Unknown*. When mis-classifications occur the confidence values are usually low. Our goal is to relate some of these *Unknown* to probable activities when the certainty is low.

A. Algorithm

As the AR service generates recognition results, ARshell compares these recognised activities (as the probable activities list sorted by their confidence values) to the previous activity generated from ARshell. We apply a Markov smoother to emphasise the temporal relationship embedded in the human activities. That is, current activity is more likely to continue into the next time window than transiting to a new one, unless new observations strongly suggest (with high confidence value) a different class of activity. This temporal characteristic of human behaviour has been justified for smoothing time slice sequences in previous work. For example, in [15] the authors manually set the transition probabilities between activity class to create Dynamic Bayesian Network for activity recognition.

To explain the algorithm of ARshell, we model the problems concerning transition between activities in Fig. 3. As shown in the Figure, we introduce the following notations:

- $y_{t-1}, y_t, ... \in Y$ is a list of activities that are generated by ARshell with different timestamps
- $x_{t-1}, x_t, ... \in X$ is a list of activity datasets that are reported by the AR service with different timestamps. Each x is a list of probable activities with their confidence values, which is modelled as a tuples list $[(a_{i,j}, cv_{i,j})]$, where $(a_{i,j}, cv_{i,j})$ is a probable activity and its corresponding confidence value. Tuples are sorted according to the descending order of cv. i is the timestamp, where j indicates the order in the probable activities list. For example, the AR result x_t contains the tuples list of $[(a_{t,1}, cv_{t,1}), (a_{t,2}, cv_{t,2}), (a_{t,3}, cv_{t,3}), ..., (a_{t,n}, cv_{t,n})]$. It should be noted that $cv_{t,1} \geq cv_{t,2}$, and $(a_{t,1}, cv_{t,1})$ is always the *most probable* activity in the AR service definition. Each $x_{t-1}, x_t, ... \in X$ is an input to ARshell for post-processing to determine the corresponding $y_{t-1}, y_t, ... \in Y$.

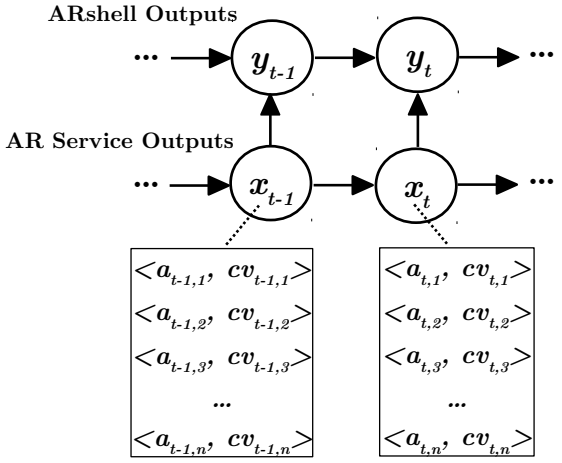


Fig. 3. Modeling of activity transition

There are four activity transitions that need to be addressed by ARshell: (i) *from Unknown to a specific activity*; that is, $x_{t-1} \rightarrow x_t$ where x_{t-1} is reported as *Unknown* and x_t is reported as one of the activities; in this case, the most probable activity $(a_{t,1}, cv_{t,1})$ is proposed for y_t, where $cv_{t,1}$ is the maximum confidence value; (ii) *from a specific activity to Unknown*; that is, in the same transition $x_{t-1} \rightarrow x_t$ where $y_{t-1} = x_{t-1}$ is reported as a specific activity and x_t is *Unknown*; in this case, we assume the last historical activity as the current activity $y_t = y_{t-1}$, with a self-transition probability ($P = 1 - \varepsilon$, where ε is a sufficiently small number [16]); (iii) *self-transition: a specific activity continues into next timestamp*; that is, in transition $x_{t-1} \rightarrow x_t$, if $x_{t-1} = x_t$, then $y_t = x_t$; and (iv) *activity switches from one to another*;

Whenever there is a transition like case (iv), we apply a Markov smoother to the AR service outputs. From our previous experiments, we identify that most mis-classifications (also considered as noises) have confidence values below 60, which we use for setting the activity transition threshold $threshold_{trans}$. As a result, when a transition occurs at timestamp t for x_t, we check the confidence value $cv_{t,1}$ of the most probable activity $(a_{t,1}, cv_{t,1}) \in x_t$. If $cv_{t,1} > threshold_{trans}$, $a_{t,1}$ is used as the output of ARshell ($y_t = a_{t,1}$). Otherwise, the Markov smoother backtracks to previous AR service outputs, and finds $\exists a_{t-1,k} \in x_{t-1}$ and $a_{t,1} = a_{t-1,k}$. We compute the $cv_{sum} = \sum(cv_{t,1}, cv_{t-1,k})$. If $cv_{sum} > threshold_{trans}$, then we say activity transition occurs and the new activity should be $a_{t,1}$. Therefore, we have $y_t = a_{t,1}$. Otherwise, we say the confidence value of two consecutive recognitions is not significant enough to determine an activity transition. In this case, ARshell will output $y_t = y_{t-1}$.

B. Accuracy

We tested the proposed ARshell on Android devices as described in Section III, and the recognition accuracy is presented in Table II. We observed a significant improvement regarding the accuracy for all the activities, except *Stationary*. In the scenario when evaluating *Stationary*, we sit down or

TABLE II
ACCURACY OF ARSHELL

	Classified to: S	W	R	F	C	V	U	A_{aggr}
S	**50%**	0	7%	0	0	36%	7%	*50%*
W	0	**95%**	0	**0**	0	0	5%	*95%*
R	0	0	**50%**	**44%**	0	0	6%	*94%*
C	0	0	0	0	**92%**	0	8%	*92%*
V_N	0	0	0	1%	0	**97%**	2%	*97%*
V_S	1%	0	0	1%	0	**89%**	9%	*89%*
A_{avg}								*86%*

Note: S - Stationary, W - Walking, R - Running, F - On Foot
C - Cycling, U - Unknown and Tilting
V_N - In vehicle with normal speed
V_S - In vehicle that is stopping or moves slowly

stand still with the testing devices held in our hands or in our pockets. The results of the AR service not only show no temporal pattern, but also occasional readings from the accelerometer become outliers that cause mis-classifications. It is very difficult to correctly classify these random patterns. In the future work, we plan to look into this issue further. In contrast, if we put the devices on a desk, which implies absolute stationary, we see that the accuracy increases to close 100%. This is consistent with our observation in the previous experiments. With regard to all the other activities, our method is able to leverage the temporal information to smooth the outliers and achieve high accuracy. For example, we observe (from Table I) a large percentage of mis-classifications when in stopping or slow moving vehicle; around 42% mis-classifications are *Unknown* (and a total of 59%). By applying the temporal constraints that we discussed, the proposed ARshell is able to reduce the total mis-classifications to 11%, as shown in Table II. The average accuracy has been improved by 19%, from 67% to 86%.

C. Prototype and API of ARshell

The proof of concept prototype has been developed on real Android devices, and in our current implementation, ARshell runs as a background daemon. As ARshell is based on the AR service, it inherits the warm-up time. ARshell provides two mechanisms for delivery of the activity recognition results: (i) a pull-based mechanism that replies with the latest activity when requested (support for per-second level), and (ii) a notification-based mechanism that updates the recognition result per interval; notifications are sent when an activity change is detected or per interval depending on how ARshell is configured.

We also streamline the API, so that software developers can easily integrate ARshell into their existing projects. With this ARshell API, only few lines of code are required for adding/linking the activity recognition functionalities. ARshell also supports backward compatibility with the AR service should developers need access to the original AR data. We open-source ARshell to the public on GitHub [3].

[3]https://github.com/myzhong/ARshell

V. CONCLUSION

In this paper, we shared the lesson learnt from the systematic analysis of the Android Activity Recognition service, and proposed ARshell that significantly enhances the activity recognition accuracy with the overall average improvement of 19%. The proposed ARshell offers a streamline API for mobile app developers to incorporate activity recognition into their existing projects or applications, and supports both pulling and notification based mechanisms for receiving recognition results. We developed ARshell and released it as an open source for use by the mobile app development community.

VI. ACKNOWLEDGEMENT

NICTA is funded by the Australian Government through the Department of Communications and the Australian Research Council through the ICT Centre of Excellence Program.

REFERENCES

[1] "Cisco visual networking index: Global mobile data traffic forecast update, 2012-2017," *Cisco white paper*, Feb 2013.
[2] L. Chen, J. Hoey, C. D. Nugent, D. J. Cook, and Z. Yu, "Sensor-based activity recognition," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 42, no. 6, pp. 790–808, 2012.
[3] T. Nguyen, D. Phung, S. Gupta, and S. Venkatesh, "Extraction of latent patterns and contexts from social honest signals using hierarchical dirichlet processes," in *Proc. of PerCom 2013*. IEEE, 2013, pp. 47–55.
[4] S. Lee, D. Ahn, S. Lee, R. Ha, and H. Cha, "Personalized energy auditor: Estimating personal electricity usage," in *Proc. of PerCom 2014*. IEEE, 2014, pp. 44–49.
[5] N. Roy, A. Misra, and D. Cook, "Infrastructure-assisted smartphone-based adl recognition in multi-inhabitant smart environments," in *Proc. of PerCom 2013*. IEEE, 2013, pp. 38–46.
[6] T. Maekawa, Y. Yanagisawa, Y. Kishino, K. Ishiguro, K. Kamei, Y. Sakurai, and T. Okadome, "Object-based activity recognition with heterogeneous sensors on wrist," in *Pervasive Computing*. Springer, 2010, pp. 246–264.
[7] T. Maekawa, Y. Kishino, Y. Sakurai, and T. Suyama, "Activity recognition with hand-worn magnetic sensors," *Personal and Ubiquitous Computing*, vol. 17, no. 6, pp. 1085–1094, 2013.
[8] L. Bao and S. S. Intille, "Activity recognition from user-annotated acceleration data," in *Pervasive Computing*. Springer, 2004, pp. 1–17.
[9] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, "Activity recognition using cell phone accelerometers," *ACM SigKDD Explorations Newsletter*, vol. 12, no. 2, pp. 74–82, 2011.
[10] M. Shoaib, H. Scholten, and P. J. Havinga, "Towards physical activity recognition using smartphone sensors," in *Proc. of UIC/ATC 2013*. IEEE, 2013, pp. 80–87.
[11] S. Hemminki, P. Nurmi, and S. Tarkoma, "Accelerometer-based transportation mode detection on smartphones," in *Proc. of the 11th Conference on Embedded Networked Sensor Systems*. ACM, 2013, p. 13.
[12] D. Gordon, J. Czerny, T. Miyaki, and M. Beigl, "Energy-efficient activity recognition using prediction," in *Proc. of ISWC*. IEEE, 2012, pp. 29–36.
[13] T. Maekawa and S. Watanabe, "Unsupervised activity recognition with user's physical characteristics data," in *Proc. of ISWC*. IEEE, 2011, pp. 89–96.
[14] B. Cvetkovic, B. Kaluza, M. Luštrek, and M. Gams, "Semi-supervised learning for adaptation of human activity recognition classifier to the user," in *Proc. of Workshop on Space, Time and Ambient Intelligence, IJCAI*, 2011, pp. 24–29.
[15] D. Wyatt, M. Philipose, and T. Choudhury, "Unsupervised activity recognition using automatically mined common sense," in *Proc. of AAAI*, vol. 5, 2005, pp. 21–27.
[16] S. Wang, W. Pentney, A.-M. Popescu, T. Choudhury, and M. Philipose, "Common sense based joint training of human activity recognizers," in *IJCAI*, vol. 7, 2007, pp. 2237–2242.