jActivity: Supporting Mobile Web Developers with HTML5/JavaScript based Human Activity Recognition

Michael Hauber, Anja Bachmann, Matthias Budde, Michael Beigl
TECO, Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
{mhauber, bachmann, budde, michael}@teco.edu

ABSTRACT

Human Activity Recognition (HAR) using accelerometers has been studied intensively in the past decade. Recent *HTML5* methods allow sampling a mobile phone's sensors from within web pages. Our objective is to leverage this for the creation of individual activity recognition modules that can be included into web applications to allow them to gain context-awareness. In this work, *jActivity*, a first prototype of such a platform-independent *HTML5/JavaScript* framework is presented, along with experiments to determine the general feasibility and challenges for HAR in web applications. Our results indicate that the realization looks promising, albeit so far limited to certain devices/user agents.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Design; Experimentation; Measurement; Performance

Keywords

Activity Recognition; Context; Mobile Sensing; HTML5; JavaScript; Responsive Design; Progressive Enhancement

1. INTRODUCTION

Smartphones are increasingly widespread and continuously gain more sensor and processing capabilities. A lot of research has been conducted to leverage these powers for Human Activity Recognition (HAR). This work aims at facilitating HAR within web applications on mobile devices, using the new capabilities defined by the World Wide Web Consortium's (W3C) *Candidate Recommendation* for the *HTML5* standard. *jActivity* can provide developers with an easy way of integrating context-awareness into their applications – e.g. to create *responsive designs* – without having to implement and train classifiers themselves. Examples for this are a bicyclist's navigation app that can automatically increase the font size or audio volume during a bumpy ride, or an interaction app for the control in smart spaces [3], that directly enables people within the environment to interact with objects – e.g. via gesture recognition. A wide range of work has already been done on HAR using mobile phones [4, 5]. We build on performing the classification on the device itself and the training of classifiers on a server, as proposed by *ActiServ* [2]. Recently, *Google* also integrated an activity recognition API [1] into Android. First JavaScript APIs for mobile web browsers are emerging as well: *Webinos*, a platform to develop web applications for a wide range of devices, shows the capability of *HTML5/JavaScript* [7]. However, it focuses on security and does not offer a HAR component.

MUM '13 Dec. 02-05, 2013, Luleå, Sweden

DOI: `http://dx.doi.org/10.1145/2541831.2541873` .

2. SENSING FRAMEWORK

Fig. 1 shows the components and workflow of *jActivity*: Web developers include a HAR module into their web application and configure the set of activities that they wish to distinguish (Fig. 1, ①). When a user accesses the application (②), a classifier is loaded dynamically according to the user agent of the mobile device and the activity set specified by the developer (③). On the back-end runs an acquisition web app, to which users can submit labeled activity data (ⓐ). This is not mandatory in order to use the application, since the training data collection is crowdsourced and the collected data is employed for all developers' web applications that use *jActivity*. This way, a growing set of classifiers is trained (and continuously re-trained) using standard HAR methods (ⓑ), each one specific to sets of activities and user agents and available to the whole *jAcitivity* community.

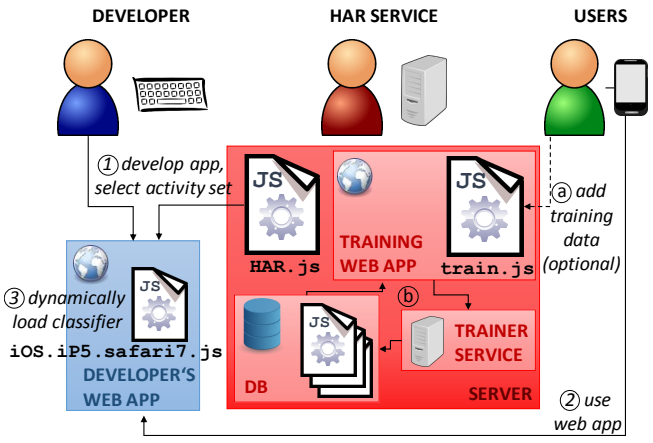

Figure 1: The *jActivity* HAR web framework.

OS	Browser	Device	DeviceMotion()			DeviceOrientation()			GPS
			mean	stdev.	median	mean	stdev.	median	
iOS	Safari 6	iPhone 4	20 Hz	0.14 Hz	20 Hz	20 Hz	0.14 Hz	20 Hz	yes
	Safari 7	iPhone 5	20 Hz	0 Hz	20 Hz	20 Hz	0 Hz	20 Hz	yes
Android 2.x	Opera 12	HTC Wildfire	—— *n/a* ——			—— *n/a* ——			yes
		HTC Desire	*9.71 Hz*	1.42 Hz	10 Hz	10.21 Hz	0.76 Hz	10 Hz	yes
Android 4.x	Opera 12	HTC Desire Z	*11.13 Hz*	1.24 Hz	11 Hz	3.90 Hz	4.07 Hz	3.50 Hz	yes
		Sony Xperia	72.66 Hz	**37.71 Hz**	92 Hz	29.49 Hz	**31.02 Hz**	27 Hz	yes
		Samsung Galaxy SII	—— *n/a* ——			9.98 Hz	0.14 Hz	10 Hz	yes
		Samsung Galaxy SIII	—— *n/a* ——			9.82 Hz	0.56 Hz	10 Hz	yes
		Samsung Galaxy S4	—— *n/a* ——			9.92 Hz	0.34 Hz	10 Hz	yes
		Google Nexus 4	15.65 Hz	0.80 Hz	16 Hz	15.22 Hz	1.45 Hz	16 Hz	yes
	Firefox 23	HTC Desire Z	*6.55 Hz*	3.52 Hz	8 Hz	40.40 Hz	8.95 Hz	38 Hz	yes
		Sony Xperia	*8.77 Hz*	0.54 Hz	9 Hz	98.07 Hz	3.64 Hz	98 Hz	yes
		Samsung Galaxy SII	96.43 Hz	2.78 Hz	97 Hz	96.42 Hz	2.85 Hz	97 Hz	yes
		Samsung Galaxy SIII	98.50 Hz	3.31 Hz	99 Hz	198.86 Hz	6.21 Hz	199 Hz	yes
		Samsung Galaxy S4	99.88 Hz	1.29 Hz	100 Hz	99.88 Hz	1.16 Hz	100 Hz	yes
		Google Nexus 4	103.63 Hz	**99.70 Hz**	117 Hz	136.71 Hz	**78.87 Hz**	167 Hz	yes
	Chrome 28	HTC Desire Z	*8.29 Hz*	**11.12 Hz**	0 Hz	6.25 Hz	**4.71 Hz**	9 Hz	yes
		Sony Xperia	20.30 Hz	7.41 Hz	23 Hz	9.84 Hz	0.47 Hz	10 Hz	yes
		Samsung Galaxy SII	23.73 Hz	1.13 Hz	24 Hz	9.90 Hz	0.47 Hz	10 Hz	yes
		Samsung Galaxy SIII	24.62 Hz	0.63 Hz	25 Hz	9.36 Hz	0.81 Hz	10 Hz	yes
		Samsung Galaxy S4	24.37 Hz	0.73 Hz	24 Hz	6.89 Hz	1.71 Hz	7 Hz	yes
		Google Nexus 4	—— *n/a* ——			8.68 Hz	1.34 Hz	9 Hz	yes
Windows Phone 8	IE 10	Nokia Lumia 820	—— *n/a* ——			—— *n/a* ——			yes

Table 1: *HTML5* event support and sampling rates on major mobile operating systems for different devices.

3. FEASIBILITY STUDY

As a first proof of concept, a prototype of the data gathering module was built and tested on different mobile platforms and devices (see Tab. 1). On each device, 30 to 60 seconds of raw sensor data were recorded on an otherwise idle phone, while steadily holding it in hand. The two *HTML5* events `DeviceMotion` (accelerometer) and `DeviceOrientation` (gyroscope and compass combined) were analyzed regarding their frequency, as web developers can not set the desired sampling rate. The achieved rates were calculated by dividing the data into 1-second-bins, counting the events and averaging over the time. We observed differences between the devices/user agents in the achieved sampling rates.

To detect many human activities, data should be gathered with at least 15-20 Hz [6]. On iOS/Safari, the possible rates seem to be fixed to 20 Hz, which we stably observed on several devices. Android devices perform very differently: All older phones did not deliver high enough sampling rates for activity recognition in our experiments (values *emphasized* in Tab. 1). On Android 2.x, some newer browsers were unavailable, and on some legacy devices, the *HTML5* events could not be accessed at all. On Android 4.x, sensor access worked for all tested devices, albeit with strongly platform-dependent performance. Using Firefox, much higher rates were mostly reached in comparison with Chrome and Opera.

The stability of the readout varied as well. While most platforms showed stable sampling rates, some device/browser combinations exhibited partial dropouts, delivering no sensor data for a second or more. This was observed for the *Nexus 4* in Firefox, the *Desire Z* in all browsers, and the *Xperia* in Opera (**bold** in Tab. 1). Another interesting observation was, that on Opera, sometimes a page refresh was necessary to get the sensor readout to work. Still, using the *Galaxy S*-series, all attempts at reading out the `DeviceMotion()` event failed. The same happened for the *Nexus* on Chrome. The current Internet Explorer version on Windows Phones 8 does not support the new *HTML5* events at all yet.

4. CONCLUSIONS

In this work, we proposed *jAcitivity*, a framework for the integration of HAR functionality in web applications. Our first experiments with the data gathering module indicate the general feasibility of sampling a phone's sensors for HAR using *HTML5/JavaScript*. However, the support has not fully been implemented for all user agents/devices yet. Especially legacy devices suffer from low sampling rates or no support at all, which is why it seems prudent to include context sensitivity using *progressive enhancement* strategies. Our future work will include the implementation of the full framework, comparisons to native activity recognition applications regarding performance and accuracy, as well as a user study. Also, the automatic training of user-personalized classifiers will be examined, similar to *ActiServ* [2].

5. REFERENCES

[1] Recognizing the user's current activity – android developers. `http://developer.android.com/training/location/activity-recognition.html`, visited 2013-10-18.

[2] M. Berchtold, M. Budde, D. Gordon, H. R. Schmidtke, and M. Beigl. Actiserv: Activity recognition service for mobile phones. In *ISWC 2010*. IEEE, 2010.

[3] M. Budde, M. Berning, C. Baumgärtner, F. Kinn, T. Kopf, S. Ochs, F. Reiche, T. Riedel, and M. Beigl. Point & control – interaction in smart environments: you only click twice. In *UbiComp '13 Adjunct*, 2013.

[4] J. Frank, S. Mannor, and D. Precup. Activity recognition with mobile phones. In *Machine Learning and Knowledge Discovery in Databases*. 2011.

[5] J. R. Kwapisz, G. M. Weiss, and S. A. Moore. Activity recognition using cell phone accelerometers. *ACM SIGKDD Explorations Newsletter*, 12(2), 2011.

[6] U. Maurer, A. Smailagic, D. Siewiorek, and M. Deisher. Activity recognition and monitoring using multiple sensors on different body positions. In *Wearable and Implantable Body Sensor Networks (BSN 2006)*, 2006.

[7] P. Vergori, C. Ntanos, M. Gavelli, and D. Askounis. The webinos architecture: A developer's point of view. In *Mobile Computing, Applications, and Services*. 2013.