Simply use the force
Implementation of RF-based gesture interaction on an android phone

Christoph Rauterberg
Institute of Computer Science
Georg-August-Universität Göttingen
christoph@rauterberg.eu

Mathias Velten
Institute of Computer Science
Georg-August-Universität Göttingen
matze@mathias-velten.de

Stephan Sigg, Xiaoming Fu
Institute of Computer Science
Georg-August-Universität Göttingen
{sigg, fu}@cs.uni-goettingen.de

Abstract—Various approaches exist to detect gestures and movements via smartphones. The absence of reliable ad-hoc on-line gesture detection from environmental sources inspired this project for on-line hand gesture detection on a smartphone using only WiFi RSSI. We highlight our line of work and explain problems at hand to provide information for possible future work. We will furthermore introduce pcan_showcase, a smartphone application, that is able to detect movement in front of the smartphone by reading the WiFi RSSI and use this information to control a Slideshow.

I. INTRODUCTION

Over the last decade, smartphones have taken up an important role in our daily lives. With this growth in wealth of opportunities, the investigation of control mechanisms for smartphones has become an interesting research topic. The exploitation of the wide range of sensors that are included in mobile phones has spurred a multitude of interaction principles. Gesture recognition over the wireless interface is, however, yet widely unexplored for mobile phones. Gesture recognition from wireless signals was demonstrated with expensive software defined radio devices [1], [2]. Recently, it has also been shown how RSSI information from WiFi packets captured at a mobile phone can be exploited to detect gestures in proximity of the phone [3]. While the data in the latter work was processed offline, in this paper we present the first implementation of RSSI-based gesture recognition that runs completely on a mobile phone. Furthermore we will describe lessons learned and give an outlook on possible future work, that could improve our results.

II. IMPLEMENTATION

The task at hand was to achieve a reliable detection of movement and nearby gestures through RSSI signal strength information and to develop a show-case application on an Android device to demonstrate its usage. The apk for the proposed application is available via\textsuperscript{1}. The source code can be found in the according pcan_analyzer repository\textsuperscript{2}. The work flow of the application is illustrated in Figure 1.

A. Platform Requirements

The setup for the proposed pcan_analyzer application requires a rooted smartphone for access to system resources. For this purpose, we used Cyanogenmod \textsuperscript{3} on a Nexus One phone. Furthermore, we utilised the BCMon\textsuperscript{4} firmware in order to be able to access the WiFi interface of the phone in monitor mode. For this, it is mandatory that the hardware of the phone employs one of two widely instrumented Broadcom chipsets (bcm4329 and bcm4330). We also employed the library libsuperuser\textsuperscript{5} in order to become able to issue terminal commands as root from within an application.

B. Data processing

1) Recording data: As we wanted to run an online analysis on our Android-phones, we were limited in our choice of programming languages. We used the bash-program tcpdump and called this program within the application. We recorded only information about the RSSI and SSID on the phone.

Furthermore, we propose a simple bash script to extract the RSSI values from this file and provided these values for analysis, as there are no possibilities with Java to read the RSSI values directly in sufficient frequency. The main problem with tcpdump was to run commands with root privileges without blocking the UI. As a solution, we utilised libsuperuser\textsuperscript{5} in combination with the AsyncTask class from the Android Developer Tools.

2) Data analysis: The recorded data is exemplarily depicted in Figure 2. We can see from the figure that in the disturbed recording, when the device was shielded by a hand or other body part, significantly fewer packets arrive. We have, however, not been able to identify significant change in the distribution of signal strength in this simple case\textsuperscript{6}. In a sandboxed scenario with only one access point reachable, a statistical analysis where we consider the count of data points

\begin{itemize}
  \item \textsuperscript{1}http://bcmmon.blogspot.de/2012/09/working-monitor-mode-on-nexus-one.html
  \item \textsuperscript{2}http://su.chainfire.eu/
  \item \textsuperscript{3}In contrast to previous work, RSSI values have not been separated for packet type or access point
\end{itemize}

\begin{figure}[ht]
\centering
\includegraphics[width=\textwidth]{fig1.png}
\caption{The work-flow of the pcan application}
\end{figure}
Fig. 2: A noisy data set (left), compared to an undisturbed data set (right).

in a fixed sample window and the variance in the recorded data proved to be very efficient. Yet, observations in situations where more than one network, i.e. access points surround the device, showed, that a combination with more advanced machine learning techniques is required.

C. pcan_analyzer application

We now propose the pcan_analyzer application. It is a simple showcase application to demonstrate the working of this method: a slide-show, that displays several pictures. Once the application is running and detects movement in front of the phone, it displays the next picture. If there is no movement detected, the picture will not be changed. The UI of the application is displayed in Fig. 3. Before the application is started, the user will be asked to train the application: To distinguish between normal and noisy data, we need an example of those first. This is done with $t = 10$ seconds of normal and $t = 10$ seconds of noisy data. After the training is complete, the user can start the slideshow.

1) Once the slideshow is started, we now use a time slot of $t = 2$ seconds to record data. The RSSI values are extracted via the proposed bash script.

2) We implemented the possibility to filter the packets for access points, which is useful when executed in an environment with many unsteady Wi-fi networks around. If the number of surrounding WiFi networks grows, the variance and the total number of received packets is alternating much more, which can be perceived a s noise for the recognition process.

3) After recording, we perform a statistical analysis of the data. We compute the variance in the recorded data, as well as the number of recorded data points.

4) Furthermore, after training, we compute a threshold: We define a 25% window around the variance and the number of data points recorded in the training for normal data. Every data slice that falls out of this threshold is interpreted as noisy data.

III. RESULTS

In the sandboxed scenario described above, pcan_analyzer is able to detect the movement of a hand in front of the phone. In surroundings where more than one access point is available, the performance of the application is conditioned on the current fluctuation of the network: When in a training situation noisy data is learned through the movement of a hand in front of the phone, a current low in the number of packets for example can lead to a false negative during the recording of data.

With only statistical analysis, we were able to detect movement in front of the phone and with that control the proposed show case application, with an accuracy $> 90\%$ in a sandboxed scenario! In other practical scenarios, the accuracy has proven to be the same or lower.

A. Demonstration setup

The proposed application, is available online and will be part of any given presentation. Users will be able to interact with a show case application during the presentation by controlling the application via gestures. For all purposes, we consider the GNU Free Documentation License as an open source license.

IV. OUTLOOK ON FUTURE WORK

To improve the results it would be necessary to implement a library for capturing the packets in Java and provide access to the RSSI data. In this way, it would be possible to analyze data on-the-fly without recording in chunks and storing it on the SD-Card. The analysis could be optimized for the specific environment at runtime by using Machine Learning technologies to determine a good size of chunks to analyze instead of hardcoding them in the source code. Furthermore we would like to monitor the performance and the power consumption on the phone and compare this to other gesture recognition approaches, that are implemented on smartphones. Afterwards we strive for the evaluation of this method for controlling other devices, e.g. laptops or other pervasive devices.

REFERENCES

