Smartphones as Drumsticks

Peter Barth, Henning Groß, Richard Petri

Hochschule RheinMain
University of Applied Sciences, Wiesbaden

Abstract

We present a mobile application using two smartphones as drumsticks. Based on accelerometer data the type of the beat (left, middle, or right) is derived. Because acceleration sensors are device specific, a device-specific support vector machine on each device is used to detect beats and type of beats. To attain good recognition on a variety of devices more quickly, the training process is simplified and backed by a server. For the sound generation the two devices are connected and a configurable drum sound for each beat on either device is played on one of the devices.

1 Introduction

Input devices such as the Nintendo Wii or the Playstation Move controllers revolutionized the game console market (LaViola et al. 2010). They established gestures as an easy to understand and easy to use human-machine interaction method. This paved the way for widespread adoption of computer games even to consumers without traditional computer literacy. In addition, smartphones have entered the life of consumers with over a billion mobile devices sold in 2010 and an increasing share of smartphones or phones with smartphone features (Gartner 2011). One of these features are integrated high frequency accelerometers (Essl & Rohs 2009). Based on the motion sensor data, gestures can be used as an alternative input method for games and entertainment on smartphones or industrial applications (de Souza et al. 2010). Accelerometers in smartphones have found first usage in remote control applications such as ShakerRacer (ShakerRacer 2007) and NiiMe (NiiMe 2008). While ShakerRacer uses the acceleration sensor data of a smartphone to steer a radio controlled car, NiiMe uses this data to steer an industrial robot or work as a game controller for computer games.

Related work in the field of gesture recognition with acceleration data was done by (Schlömer et al. 2008) which describes the recognition of gestures performed with a Wii-controller using Hidden-Markov-Models (HMM) and a Bayes classifier. An older but important work on gesture recognition using accelerometer data was done in 2000 by (Mäntylä et al. 2000) using HMM and Self-Organizing-Maps (SOM). (Kia Ng. 2002) and (Sawada 1997) explored gestural interaction for music-making.

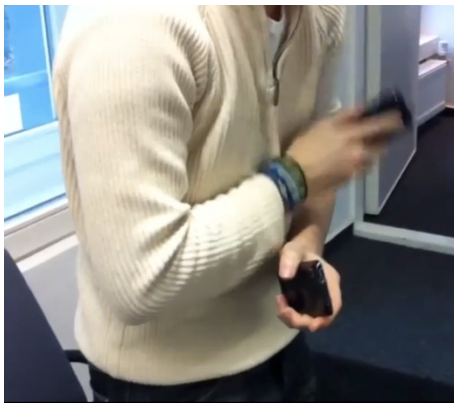

Figure 1: User playing drums with smartphones

In this paper we present a mobile application, which lets users play a virtual drum kit consisting of up to six different drums played by two Android devices as shown in figure 1. To this end, two devices are connected via Bluetooth, whereas only one device produces the sound. The decisions if and which gesture, meaning whether and which beat, has occurred are made on each device individually, based on trained support vector machines (SVM) adjusted for device specific behavior. Data for *device specific detection* can be recorded and shared with the help of a server-based training environment and loaded without changing the application. Hereinafter, we focus on the *detection* of the beats and only sketch sound generation. The mobile application has been made available on the Android Market and has been used for a *demonstration* captured as video, which is briefly explained before the conclusion.

2 Beat Detection

The main challenge of this mobile application is the recognition of the beats. It is has to be decided, whether there is a beat and which beat it is. We differentiate among three different possible beats, which for the right hand may represent the snare, the hihat, and the floor tom. It would be nice to differentiate among six different beats and thus accounting for crash and toms as well. In addition different strengths of the beats, such as soft, middle or hard, would help to make a more immersive experience. In principle, this could be accomplished with the presented architecture. However, experiments show that then recognition rates drop significantly and it becomes more difficult to accommodate for different devices. Thus, we concentrate on three different types of beat per drumstick and just one possible intensity (beat or no beat).

Figure 2: Device position while holding

To determine these beats, we examined raw sensor data of different devices while holding them in different positions and performing many similar beats successively. The user holds the device flat in her or his right hand, with the display pointing away from the palm, as illustrated in figure 2. A “middle” beat is performed by quickly stroking the device off the cuff around the z-axis (see figure 3). The “left” beat is a similar movement along the direction, which the display is facing, along the positive z-axis, and the “right” beat a movement in the opposite direction.

The detection of different beats based on accelerometers is bound to device specific sensor data and individual execution of beats by users. We illustrate device specific sensor data by having one person perform the same middle beat – within the limits of human accuracy – with four different devices in figure 4. The figure shows clearly, that the devices emit sensor data at different sampling rates. Table 1 lists the effective sampling rates of different Android Smartphones. Although the shapes of the associated curves backed by the input data points are similar, the actual absolute values as well as the relative peeks differ significantly.

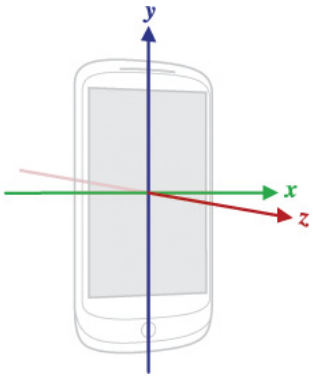

Figure 3: The Android default orientation for sensor data

To recognize a beat, we use a typical sensor pipeline architecture as shown in figure 5. Initially, the sensor data is captured from the hardware sensor. We start by smoothing the data. For each acceleration data point the weighted average of twice the current data, once the previous and once the next data point is computed. Then, the input data (x, y, and z separately) is normalized. Furthermore, we do not use the absolute value of the acceleration sensors, but the relative changes of the acceleration data, the slopes. Thus, the difference quotients of each point and its predecessor in the input stream are computed. Finally, the input data is scaled to always be between -1 (steepest descent) and 1 (steepest rise). The final scaling factor is computed based on one complete set of training data and thus device and training data specific. The training data must include extreme values in order for the final application to produce sensible results. As the shape of the curves constitutes the differentiating features of different beats a single value is not sufficient for recognition. In figure 3 we see that the main peeks at the left of the graphs – the beats – can be fully captured with around 10 consecutive values. Therefore, each individual input data tuple consists of 30 float values, which are the 10 most recent different quotients for each of the three axes (x, y, and z shown in figure 3). The actual recognition is then left to an artificial intelligence tool. As soon as a drum beat is recognized, the playing of a corresponding sound is triggered.

Device	Rate (Hz)
Samsung SPH-M900	118
Sony Ericsson U20i	94
Sony Ericsson E15i	71
HTC Desire HD	70
LGE LS670	68
HTC Desire	50
HTC Nexus One	27
LGE LG-P500	15
HTC Bee	9
Samsung SPH-M910	7
LGE Ally	5

Table 1: Effective sampling rates of different Android Devices

The chosen AI-method for gesture recognition is a support vector machine (SVM), which is good at recognizing “similar shapes”. The SVM (Chang and Lin 2011), which runs directly on the device, needs a pre-trained model to classify the tuples. The model is generated on a standard PC with data gathered and transmitted to the PC in a training phase. The collection of data can be done by each user of the mobile application. In that phase the user performs the gestures, while the device records the sensor data. After the raw sensor data – the sensor pipeline is not used at all in that phase – is transferred to the PC, the raw data can be processed. There is some tooling, that allows to estimate a useful classification and only requires to mark when the actual beat has occurred.

Figure 4: Comparison of acceleration sensor data of a middle beat. Left to right: HTC Desire, LG Vortex, Samsung Galaxy S and HTC T-Mobile G1

As shown in figure 4 a stroke can be detected by a high slope between two extremal values on the most significant axis. The marker for each stroke is set to the nearest point before the first extremal value. The tooling uses four points ahead and five points after the marker to create a tuple for classification. The five trailing points cause a small delay during detection, since sampling points after a performed gesture are used.

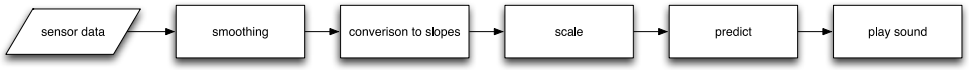

Figure 5: Sensor pipeline

For each of the possible beats relevant tuples are gathered and manually classified. To do this, the user is headed through a wizard, which collects this data as demonstrated in a video available in the online resources. This enables us to classify any tuple of slopes as one of the possible beats or, most of the time, no beat. In particular the “no beat” classification needs also several sample tuples. The interactive training tooling also helps this. It is sufficient to mark relevant tuples that should go as “no beat” into the sample set. Using the collected tuples as positively or negatively marked tuples for training, a model is computed and then deployed to classify a stream of tuples continuously. Using a sliding window may trigger duplicate detections since the shape of two consecutive curves is similar. To avoid duplicate detections, we temporarily disable the classification.

The generated models incorporate the characteristics of specific devices, such as the frequency of the data, but also the characteristics specific to the person's execution of the beats. While users instinctively adopt their drumming behavior to “what works”, this cannot be said from the hardware sensors. Thus, at least hardware-specific models need to be deployed to achieve better recognition rates. To gather the data for generating these models we use a community module collecting data from any smartphone where the mobile application is installed. The user is entering a training phase and is advised to perform the three different drumbeats in sync to a metronome, which is haptically emulated using vibration. The raw sensor data is captured while the user performs the training. This data is then sent to a server. There, it can be reviewed and processed to a specific model for this hardware profile. This step is obviously manual and cannot be done by a user. Afterwards the whole community is able to download, use, and rate the models on their devices.

3 Sound Generation

Sound generation for an application simulating a virtual drum kit has to be responsive and efficient. Without a quick response to the beat, there is a lag between the actual beat and the sound. Based on the SVM, the recognition itself is very fast and there is only a systematic lag based on the delay of using a few additional input vectors to better recognize the turning point. However, sound generation itself is computationally demanding and needs to occur spontaneous and quickly as soon as the event is triggered. Moreover, multiple sounds need to be played simultaneously, as the next beat may happen while the first is still playing.

To accommodate for the need to play multiple sounds quickly a thread pool pattern is used. A thread in this pool, waits for the trigger to play a specific sound and then handles the playback. The Android API conveniently incorporates an implementation of such a "sound pool". It is located on just one of the devices and plays the sounds associated with the gestures. Emitting the sound only on one device ensures a homogeneous experience. For example, a snare can be played with both smart phones and the sound is identical, as it should be. In addition delays that may occur are homogenous, which results in better playability. Furthermore, it is possible to record a session for later playback without additional effort.

To achieve sound generation on just one device, both devices are connected via Bluetooth serial connection. The latency is minimal (single digit millisecond) and thus below human recognition and an order of magnitude smaller than the systematic delay introduced by incorporating additional tuples after the actual beat. The architecture uses a classical master/slave idiom, where the sound-emitting device is the master. To minimize traffic, the slave device simply sends a sound identifier that instructs the master device to play a specific sound. Obviously, a smartphone can also be used without Bluetooth connection to emulate a single drumstick.

4 Demo

We demonstrate the application using two scenarios. In the first scenario, we just use one smartphone as one drumstick to make the three different beats. A user is able to perform a simple rhythm by using the middle beat twice (snare) and the left beat (hihat) or right beat (cymbal) in between. A demo is performed on a Samsung Galaxy S device. The other scenario is using two devices connected via Bluetooth. Both devices use different sounds for the beats. First, a simple rhythm is played. Then a new foot-mounted device is integrated to play another rhythm. Note, that as the devices are held differently – with different extremities – different training sets need to and have been used. A demo is performed on two Samsung Galaxy S devices and a HTC Desire device.

The application has been published on the Android Market. Within the first month there have been more than 10,000 downloads with more than 3,000 active installations. After about 6 months there are more than 30,000 downloads with more than 6,000 installations. The community module has attracted more than a hundred users, who have submitted training data

sets. Of the training data sets, more than 20 trained models have been generated, published, and rated.

5 Conclusion

This project provides a virtual drum kit, which employs acceleration data to recognize different beats. Recognition is done with a support vector machine accommodating for different devices by using different training sets. Immersive behavior is achieved by using two devices emulating two drumsticks but concentrating homogeneous sound emission on one device.

Accuracy depends on the hardware used. There is at best erratic success on unknown devices. Thus, training sets need to be manually fine tuned on different sensor types. The architecture and method itself is not limited to only three gestures and one strength of beat. In the future, allowing for e.g. six types and thus including another virtual row of instruments is possible. In addition, different strengths, such as soft, middle, and strong can be included. The modifications to the application will be minor. The main changes are in the training sets for the devices. It may be necessary to then train different sequences of beats as the characteristics of the curves change depending on the last used beat.

References

Chang C. and Lin C. (2011) *LIBSVM: A library for support vector machines.* ACM Transactions on Intelligent Systems and Technology Volume 2 Issue 3, 27:1 - 27:27. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm

Drumsticks for Android. Retrieved May 2011, from *http://www.youtube.com/watch?v=ZK474N21d-Y*

Drumsticks for Android with three Devices. Retrieved May 2011, from *http://www.youtube .com/watch?v=Z1q8_Uryuuo*

Gathering data for a new Android Drumsticks TrainingSet. Retrieved May 2011, from *http://www.youtube.com/watch?v=Wks4x9XuZps*

Gartner Group (2011). *Market Share Analysis: Mobile Devices, Worldwide, 4Q10 and 2010.*

Essl G. and Rohs M. (2009). *Interactivity for mobile music-making*. Org.Sound 14, 2, 197-207.

Kia Ng. (2002). *Interactive Gesture Music performance interface.* In Proceedings of the 2002 conference on New interfaces for musical expression (NIME '02), Eoin Brazil (Ed.). National University of Singapore, Singapore, Singapore, 1-2.

LaViola,Jr., Joseph J., and Marks, R. L. (2010). *An introduction to 3D spatial interaction with video game motion controllers*. ACM SIGGRAPH 2010 Courses, SIGGRAPH '10.

Mäntylä, V., Mäntyjärvi, J., Seppänen, T., Tuulari, E. (2000). *Hand gesture recognition of a mobile device user*. IEEE International Conference on Multimedia and Expo ICME2000 Proceedings Latest Advances in the Fast Changing World of Multimedia.

NiiMe (2008). Retrieved May 2011, from http://www.niime.com/

Sawada, H. and Hashimoto, S. (1997). *Gesture recognition using an acceleration sensor and its application to musical performance control*. Electronics and Communications in Japan (Part III: Fundamental Electronic Science).

Schlömer, T., Poppinga, B., Henze, N., Boll, S. (2008). *Gesture recognition with a Wii controller*. Proceedings of the 2nd international conference on Tangible and embedded interaction, New York, NY, USA: ACM

ShakerRacer (2007). Retrieved May 2011, from http://symbianresources.com/projects/shakerracer/ShakerRacer.pdf

de Souza, M., Carvalho, D., Barth, P., Ramos, J., Comunello, E., von Wangenheim, A. (2010). *Using Acceleration Data from Smartphones to Interact with 3D Medical Data*, 23rd SIBGRAPI - Conference on Graphics, Patterns and Images

Contact Information

Peter Barth, Henning Groß, Richard Petri
Hochschule RheinMain
University of Applied Sciences RheinMain

Unter den Eichen 5
D-65195 Wiesbaden