Tempo Adaptation within Interactive Music Instruments in Mobile Phone

Marek Takáč and Alena Kovárová

Faculty of Informatics and Information Technologies,
Slovak University of Technology in Bratislava, Slovakia
xtakac@gmail.com, kovarova@fiit.stuba.sk

Abstract. Nowadays it is not unusual, that applications for mobile devices are using various sensors such as touch screen and accelerometer for their control. We aim to use these mobile sensors for music creation in a way that is intuitive to users. The main contribution of this paper is proposed formula for real-time tempo adaptation for a single user. We also dealt with the control design of musical instruments of different types (to make it similar to their real counterparts) so that users can easily adapt to the device. To test it, we have implemented an application simulating three musical instruments: piano, flute and drums ex-tended by a metronome, editable rhythms and other configuration settings in-cluding tempo adaptation.

Keywords: mobile device sensors, music, tempo adaptation.

1 Introduction

Mobile devices are undoubtedly one of the phenomena of our time. Mobile development is accessible to almost everybody, not only in developed countries, as almost everybody can now own some type of mobile device. As of 2012, it is relatively easy to obtain a device with a large screen, GPS navigation and with multiple integrated sensors. This technological shift significantly increases the range of applications on mobiles in terms of computing, but also in ways that users are interacting with the device.

All of these devices can be used for interactive music creation. Creating sound output using mobile device sensors is relatively young and, therefore, studied only in the last few years. This area of research can be further developed and that is the aim of this work.

Our first challenge was to create musical instruments with controls similar to those on real instruments so that the interface is intuitive for users. There are different types of musical instruments; we focus on the following three classes – keyboard, wind and percussions. We have explored the potential of today's mobile devices (in way of sensors combination) to achieve intuitive control, however the available computing power, device memory and sensors offer space to create additional features that a user might like. For our second challenge we decided to look at metronome, rhythms and automatic tempo adjustment.

P. van Emde Boas et al. (Eds.): SOFSEM 2013, LNCS 7741, pp. 540–551, 2013.

2 Related Work

2.1 Sensors for Mobile Music Performance

Modern mobile devices contain variety of sensors that can be used to generate music in very different ways. Michael Rohs and Georg Essl described this in their publication named Interactivity for Mobile Music-Making [1]. This work contains classification of available sensors in mobile devices that are relevant to making music according to different dimensions. A simplified matrix of the classified sensors is presented in Figure 1.

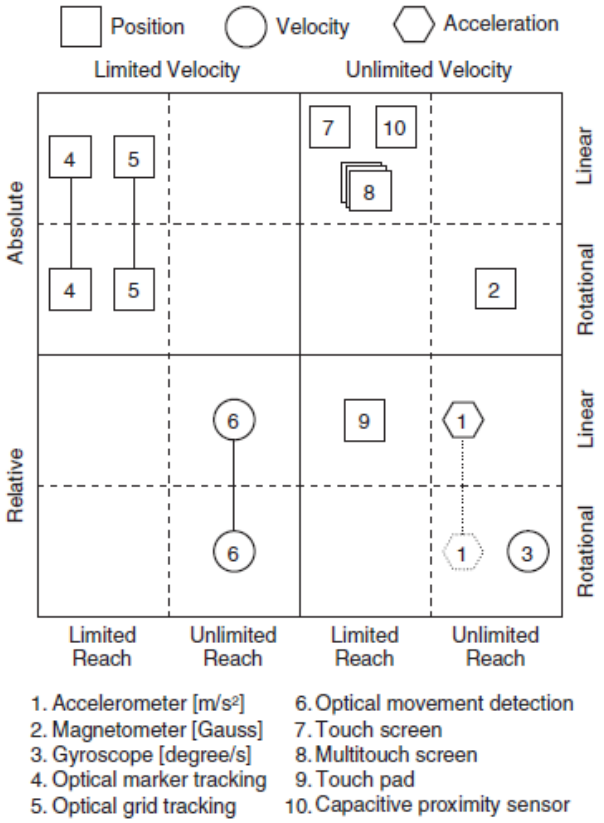

Fig. 1. Design space of sensors for mobile music performance [1]

In general, sensors can measure linear or rotational movement. Additionally, there are sensors that can measure both linear and rotational movement; these are represented as connections between cells in the matrix.

In context of mobile music creation it is also very important how particular sensors are utilized. Many sensors are usable only if they work within the available constraints of the user. This means that in some cases users are limited by sensor (e.g. in

optical sensing at low frame rates), which is shown in the figure by dimension limited/unlimited velocity. Some musical phrases, which rely on speeds faster than the sensor technology can reliably track, may be unavailable to the user. Sensors placed in unlimited velocity dimension are able to detect rapid movements so this limitation does not apply.

Dimension limited/unlimited reach means that sensor is limited (or not) by physical interaction space. For example, touch screen is limited by its size, or proximity sensor is limited by maximum sensing distance. In musical context this means that mapping specific sounds to interaction is more difficult on sensors with limited reach.

Data provided by sensors can be absolute or relative. Absolute data sensors are generally easier to work with in musical context because they can be easily mapped to particular tone pitch or volume. The same is possible with relative data, but it is necessary to propose more complex transformation of data to get specific sounds [1].

When we looked at existing applications, several interactive applications aimed on creating musical tones directly on mobile devices have been created. They mainly represent an individual musical instrument, like piano, guitar, acoustic drums, ocarina. In next lines we mention applications, which we consider as interesting (simply by using them) from their group of type. All of these applications are for iPhone OS, but similar application with the same control principles can be found also for other operating systems.

Keyboard musical instruments

- Piano+ ([1]. . . /piano+/id430119524)
- Virtuoso ([1]. . . /virtuoso-piano-free-2-hd/id304075989)
- WaveSynth ([1]. . . /wavesynth/id310846058
- Gyro Piano ([1]. . . /gyro-piano/id382551316

All of these applications are controlled mostly by a *touch-screen*. They usually include a simple graphical interface showing one or two octaves of piano keyboard on the screen. Some of them support multi-touch. In piano it is crucial to change between octaves – it is implemented either as a manual switch (touch the button) or, e.g. in Gyro Piano, it uses the *accelerometer* and *gyroscope* – to move across the piano keys the device has to be moved in the x-axis.

Wind musical instruments

- Smule's iPhone Ocarina ([1]. . . /ocarina/id293053479

This Ocarina [2] is probably the most famous application of this kind. It is controlled via the *touch screen* (multi-touch, by touching the screen at those places where are the holes and this contact will "cover" them) and blowing into a *microphone*. It also uses *accelerometer* to simulate vibrato of played tone.

Percussions

- Amazing Drums ([1]. . . /amazing-drums-lite/id424438240)
- Drums Deluxe Light ([1]. . . /drums-deluxe-light/id399326714))

[1] `http://itunes.apple.com/us/app/...`

- Gyro Air Drums ([1]…/gyro-air-drums/id383027843 or http://www.gyroairdrums.com/)
- vDrummer ([1]…/drums/id311549739 or http://www.v-drummer.com)
- Drum Meister ([1]…/drum-meister-pro-lite/id384869832)

Most of these applications play the drums and percussion through interaction with multiple sensors of the device. In the first case, the *touch screen* with displayed image of drum set. This allows a user to create sound by touching respective drum. The second and much more interesting way of sound creation is if a user can "play" on the instrument by inclination and "hitting" in space. For this purpose is used the built-in *gyroscope* that makes it possible to determine to which particular part of the instrument was "hitting" and also to recognize gestures representing the strike, but there is a drawback – there is not entirely accurate detection of beats of drums.

2.2 Tempo Adaptation Based on User's Interaction

There are several types of algorithms that detects tempo of the song. Tempo is measured in units called BPM (Beats per Minute), which determines the number of "beats" (or more precisely – quarter notes) per minute. The process which is behind the detection of beats (hence the tempo) is an audio signal analysis, including finding the positions with greater amplitude, which usually represents snare drum hits in tempo. Song can be split to parts and then the process of finding regional maxima can separate "real" beats in tempo from soft dynamics hits [3].

There are several algorithms that are applicable to tempo detection [4,5] but these algorithms are used to find out tempo of existing track and not during its formation as in our case. To detect tempo we are using our suggested algorithm (see Section 3.3), inspired by Goto and Muraoka [4]. This algorithm is based on regular analysis of IBI (inter-beat interval - interval between two successive strokes in rhythm). The tempo is not changed if the last interval is a multiple of the current interval. It is evident that human is not capable of playing tones accurately to milliseconds in the tempo so in this algorithm we use an error tolerance value. Tempo is changed if the IBI maintains approximately the same value specified number of times in a row, and is different from the current tempo.

2.3 Audio Output Playing

To have a sound (music instrument) we had to deal also with audio output. In principle, audio output can be played via MIDI interface on external device, operating as a sequencer or by direct playback of prepared audio samples stored on device. The first approach was used in application Camus [6]. However, this application is not designed to generate music, but only to modify existing songs with musical effects. But our application should use only the iPhone audio playback capabilities, therefore we have decided to use the sound samples. They are stored in memory of device as files containing samples of tones. The great advantage of this approach is that it possible to use high-quality music samples, which could significantly improve the quality of audio output. However, to minimize memory requirements (our average sample file has

84 kilobytes), application contains only a small number of samples (e.g. virtual piano contains only 20 samples out of 88). Then we can increase or decrease the frequency (pitch) of these samples by so-called pitch shifting algorithm. This algorithm adjusts pitch of any tone in playback in real time by a selected interval. Tone frequency change is also related the change of song duration. Pitch shifting algorithms are designed to adjust only frequency and sound duration is maintained (the reverse process of this algorithm is known as time stretching).

3 Interaction Improvements and Algorithms

3.1 Sensors for Mobile Music Performance

Today's mobile phones contain these useful sensors: touch screen, accelerometer, gyroscope, camera and microphone. We propose the following improvements:

Keyboard musical instruments: To move along the keyboard, the user can either use *swipe gesture* or shift the keyboard (detected by gyroscope) by changing the *tilt* of device which can be detected by 3-axis *accelerometer*.

Wind musical instruments: Our plan was to use a *camera* of device as a sensor which will detect "covering" of flute's thumbhole by real-time analysis of captured images. But in most of devices, the camera is in opposite position to microphone, therefore it would be very hard for user to control it. Moreover this type of instruments is used to have many holes. This is also a problem, because operating systems allows detecting only five touches at one time.

Percussions: Enhance it by *shaking* which can be detected by *accelerometer* – user's shaking gestures with device can play sounds of e.g. shaker or tambourine.

3.2 Sound Playback Algorithm

In addition to standard instruments mentioned in previous section (we call them "interactive"), there are also others that the user does not control directly – "non-interactive", and these instruments are used to create a sound playback, for example metronome or different rhythms. They play pre-defined sounds without user's interaction. During the playback the algorithm has to determine the time when concrete sound should be played. This can be done by simple formula:

$$time_n = time_{n-1} + \frac{240}{tempo} duration_{n-1} \tag{1}$$

where $time_n$ is n-th tone, $tempo$ is tempo of non-interactive instrument $duration_{n-1}$ is duration of n-1 tone. Using this recursive formula we can calculate the specific time when should be n-th tone played. This time is represented in seconds and so the numerator has to be 240, which is the multiplication of seconds in a minute with number four, because in music theory tempo is defined as number of quarter tones played in one minute. Our equation is similar to Reidsma et al. [7] equation 1 with conductor, but in our case the system follows user's tempo and does not try to lead him/her.

3.3 Automatic Tempo Adjustment Algorithm

Tempo of selected rhythm adapts during user interaction with one of three virtual instruments that results in a different tempo. As mentioned earlier, if new tempo is in same ratio (usually multiple of powers of two, but it is not a rule) to the original value, tempo change is considered natural and therefore it will not be changed. So the situation can be divided into two cases. In first case, if new tempo is in natural ratio to original it should not be changed and in second case, if is it not tempo is changed. To find this information was used and experimental fine-tuned following formula:

$$changetempo = \begin{cases} 1 \text{ if } \forall i \left| \frac{\text{newIBI}}{\text{oldIBI}} - multi_i \right| > \frac{\text{error}}{\text{multi}_\text{i}} \\ 0 \text{ else} \end{cases} \quad (2)$$

where *changetempo* is information whether is necessary or not to change the tempo, *newIBI* is time interval between the last played tones, *oldIBI* is time interval between the tones in the original tempo. Variable $multi_i$ stores *i*-th multiple of old tempo which can be considered in natural ratio with new tempo and error is time error value which can be tolerated musical timing of user. The result says that the tempo change is necessary if in tempo variable is number 1; otherwise new tempo is in natural ratio to original. In application we are using five multiples stored in variables multi1-multi5, namely the numbers: 1/4, 1/2, 1, 2 and 4. We also use value 0.2 stored in variable *error*, because during the tests it gives us the best and most natural results for tempo adaptation. From implementation point of view automatic tempo adjustment algorithm is based on observer design patterns.

4 Evaluation

To test our ideas we implemented three interactive musical instruments - piano, flute with five holes, drums and two non-interactive – metronome and drum rhythm with

Fig. 2. The basic screenshots of our application, from left: piano, flute, drums, settings of non-interactive instruments and automatic tempo

ability to adapt to user's tempo (see Figure 2). We implemented it for iPhone and we have used a number of frameworks (Core Motion, Core Media, AVFoundation, UIKit etc.) and for audio playback was used OpenAL API.

We conducted three types of testing – quantitative, qualitative and comparative. The number of testers was 10 – 3 women and 7 men, most of them of age 23 (the youngest was 19 and the oldest 48 years old). One of them had high IT skills and had experiences with iPhone. Other testers usually at least had experience with the touch screen but not with iPhone. Two testers did not have experiences with touch screen at all. We did not compare the adaptation, because there is no other application with such feature.

4.1 Quantitative Testing

The first test we conducted was aimed at tracking the time respondents needed to perform specified tasks. The results of this testing should point out how easy and intuitive interface our applications can be controlled. The average length of individual tasks reveals if there are some the poorly designed parts. The tasks were designed to cover most areas application functions (see Table 1).

Table 1. Tasks and their performance times realized by 10 testers

Task / Tester	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
1. Push „c3" key on the piano	8.9	20.6	11.6	7.4	8.2	8.2	7.5	13.7	5.9	6.8
2. Play a tone on the flute	7.9	10.2	40.5	9.5	5.6	35.4	6.7	106	14.7	19.6
3. Play a sound of shaker from percussion set	37.3	100	71.7	81.7	72.1	46.2	41.3	17.9	18.7	138
4. Record sound of drums in file "drums"	33.7	92.1	39.1	27.1	47.8	60	39.2	36.5	42.8	34.9
5. Play the file	7.0	15.4	6.3	3.5	4.1	43.2	3.9	4.1	38.2	5.5
6. Create a new rhythm using editor	45.4	94.9	90.7	58.1	48.8	72.9	45.1	79.1	71.5	82.4
7. Set this rhythm to the piano	46	55.7	45.3	8.1	63.7	34.5	13.7	130	53.1	56.1
8. Open the flute help screen	6.2	8.3	5.6	3.4	4.1	5.9	3.5	6.6	8.2	3.3

This test had to run first, because otherwise these people would already be familiar with graphical interface and control and thus would have been significantly distorted by time. Before carrying out a specific task, it was explained and then a tester was asked to realize it. The following diagram (Figure 3) shows the minimum, maximum and average time of testers' performance for each of the eight tasks.

As can be seen from the results, all respondents were able to perform each task under 140 seconds. The tasks also differed in the number of steps that had to be done to properly complete the task. It can also be seen that in addition to the task number three (play a sound of shaker on percussion set) and number six (create a new rhythm using editor) respondents were able to complete each task on average in under one minute. It also can be said that this was a task that most showed deficiencies in the GUI design, because testers usually looked for given functionality on another place or expected more highlighted settings.

The left side of Figure 3 shows which tasks had the biggest difference in the ratio between the minimum and maximum time. Generally speaking, this kind of tasks feel

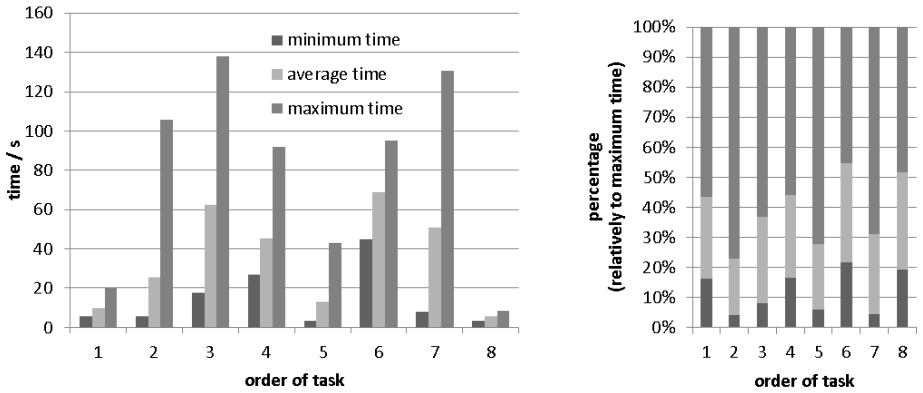

Fig. 3. Minimum, maximum and average time of testers' performance

completely natural to certain users (they have either prior experience or good intuition), however other users have considerable problems solving it. An example of such task is task number two, where a tester was asked to play a flute tone. Some respondents did not expect the tool to be controlled by blowing into the microphone, which revealed additional problems of application. Although the testers had no problem to check it in Help, they did not bother to do it. We solved this problem by adding a blowing icon on a flute screen.

Our qualitative test results are relatively good because the measured times showed that the application is fairly intuitive and easy to operate.

4.2 Qualitative Testing

The qualitative testing allowed testers to go through the whole application and then they were asked to fill out our questionnaire. Questions were asked about every major applications features (see Table 2).

Before each survey question we first presented to the tester all the (feature-) given functionality and how it can be controlled. Then the tester was free to test the intended functionality, which then he had to evaluate from 1 (worst) to 10 (best). On the next picture you can see a graph (Figure 4) showing the average grade for each question within a questionnaire. In addition, it shows the difference between the highest and lowest grade of the matter.

As can be seen, the average of all questions is located around the values of 7 and 8. This also means that on average respondents were very satisfied with each feature of the application. In this graph we also show the difference between the highest and lowest grades for testing. This difference indicates which questions were more and which less tester-subjective (dependent on the individual user's taste or subjective opinion). The biggest difference was the value 5 in next three questions: evaluation of piano control, flute control and the way of rhythm creation using the editor. On the other side, the lowest difference in combination with high average probably shows the best part of the system (see e.g. question. 9 - graphical design of applications).

Table 2. Questions and tester's grades of qualitative testing

Question / Tester	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
1. Piano GUI	7	7	7	8	7	8	7	8	8	9
2. Flute GUI	7	9	7	6	6	6	7	8	6	7
3. Drums GUI	8	8	8	7	7	7	9	7	7	9
4. Piano control	8	8	8	8	9	5	8	9	10	8
5. Flute control	6	8	9	6	7	9	4	8	8	8
6. Drums control	9	7	7	8	9	7	8	8	9	10
7. The way of rhythm creation using editor	6	9	7	7	7	7	4	7	8	8
8. Hierarchy of screens	8	7	7	8	6	5	6	8	6	6
9. Graphical design of application	9	8	8	7	8	7	8	9	9	9
10. Overall application control	7	7	7	8	7	6	7	8	7	8
11. The way of combination of interactive and non-interactive instruments	7	9	8	8	8	8	7	7	8	7
12. Realization of rhythm adaptation	9	8	8	7	9	7	8	9	8	6

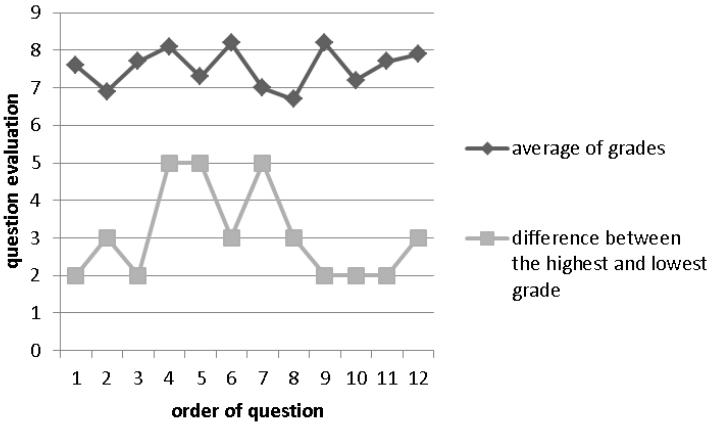

Fig. 4. Minimum, maximum and average time of testers' performance

These results indicate that the application can be considered as very good quality including GUI design, control and other features.

4.3 Comparative Testing

The last type of testing was based on a comparison of our application with other applications designed for similar purposes. Comparing functionality, we compared all three of our interactive instruments. Moreover, we let our testers to use two other different pianos and ask them to evaluate GUI, control and sounds.

Flute: Our Flute was compared with the Ocarina iPhone application, which utility is much more manageable than it is in our case (therefore no other tests provided), but on the other hand we consider our flute more graphically appealing.

Drums: Our drums were compared (obvious pros and cons, no other test provided) with applications Amazing Drums and Drums Deluxe Light, which both offer free versions of just ordinary acoustic set (see Figure 5).

Fig. 5. Amazing Drums (left) and Drums Deluxe Light (right)

Application Amazing Drums has very unattractive main screen (picture of drums) and the same goes for the sounds of this program. On the other hand, application Drums Deluxe Light creates sounds very similar to acoustic drums and the main screen also resembles the real drum set. The downside of this application (comparing to ours) is, that it does not show the visual feedback when played on any part of the drum set. But its advantage is that it offers a number of component sounds of the same drum that plays in dependence of touched space (e.g. in the middle of the drum or on the side of it). Both compared application can be controlled only by touch-screen.

Piano: Comparison of our piano program was conducted with applications Piano+ and Piano Virtuoso, both of which are available free on the App Store. Both applications provide only the functionality to play this instrument. The visual of their main screens shows Figure 6:

Fig. 6. Piano + (left) and Virtuoso (right) application

The comparison of our piano with two other applications, we proceed similarly as in the qualitative tests. Since we were not able to provide the absolute correctness of this test (testers and developer knew personally each other and testers have already completed two previous tests), we tried to at least establish the conditions that would minimize the impact on incorrect evaluation. Therefore we first demonstrated (functionality and control) all three applications to the testers. Then they were asked to play a

few of chords and tones in each application (to make the comparison was the most relevant). When testing the application, they have also highlighted the three main attributes (graphical user interface, control and sounds) that were evaluated in each application by grades from 1 (completely wrong) to 10 (excellent).

The testers' grades and questions are in Table 3. The average evaluation of these attributes for all three applications can be seen on Figure 7.

Table 3. Questions and tester's grades of qualitative testing

Question / Tester	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
Our piano GUI	7	7	7	8	7	8	7	7	8	9
Piano+ GUI	5	9	4	5	3	5	3	7	3	6
Virtuoso GIO	8	6	8	6	7	3	8	9	6	7
Our piano control	8	8	8	8	9	5	8	9	10	8
Piano+ control	6	5	4	5	5	7	3	3	2	4
Virtuoso control	8	6	6	7	6	5	8	6	7	6
Our piano sounds	9	8	8	8	9	8	7	7	7	8
Piano+ sounds	3	6	5	6	5	2	3	4	3	6
Virtuoso sounds	7	6	8	7	7	5	7	8	8	8

Figure 7 shows average grade for each of the tested feature in each of tested piano. As can be seen, our piano received the highest grades (in all three features) from our testers. In contrast, Piano+ with a relatively great distance determined in all features for the worst. Virtuoso received relatively similar grades as our piano; the noticeable difference is only in control.

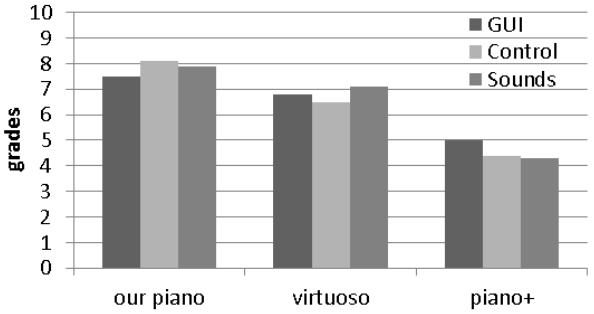

Fig. 7. Average evaluations of GUI, control and sounds in our piano, Piano+ a Virtuoso

These test results show that our instruments were created in a similar or better quality as well as in compared applications. The main difference still remains – our application combine more musical instruments and offer even rhythm adaptation.

5 Conclusions

In this paper we described our research which consists of study how mobile device can be used as a smart virtual music instrument. We proposed several improvements for three different types of musical instruments and implemented them (each is controlled by variety of sensors), with which we have conducted experiments. In addition, we focused on design and implementation of non-interactive instruments in which we had successfully implemented our formula for automatic tempo adjustment.

Our application was tested by ten users and three types of tests. From quantitative test (time of tasks performance) we realized some design errors but they were not critical (to improve intuitiveness, highlighting of some of the graphic elements or their move them to other place is needed). The qualitative test consists of a questionnaire covering all important parts of the system features. These were in average graded to be very good. At the end of testing, we asked testers to compare our piano with two other applications. Here our piano seems to have the best GUI, control and sounds. This is a very good result considering that those other two applications were designed only to play this one instrument.

It is crucial to note that during the test, the parts of the system that we have not found in similar types of applications were graded as excellent e.g., rate adaptation, a combination of two types of instruments or multiple music instruments within one application. Therefore, the quality of our application can be considered as equivalent to other existing applications or even comparably better thanks to its unique features including tempo adaptation.

This application could be used for entertainment or also as a simple educational musical instrument application.

Acknowledgement. This work was partially supported by the grants VG1/0971/11 and APVV 0208-10.

References

1. Essl, G., Rohs, M.: Interactivity for Mobile Music-Making. Organised Sound 14(2), 197–207 (2009)
2. Wang, G.: Designing Smule's iPhone Ocarina. In: Proceedings of the 2009 International Conference on New Interfaces for Musical Expression, Pittsburgh, Pennsylvania (2009)
3. Sauer, D., Yang, Y.-H.: Music-Driven Character Animation. ACM Transactions on Multimedia Computing, Communications, and Applications 5(4), 27:1–27:16 (2009)
4. Goto, M., Muraoka, Y.: Real-time beat tracking for drumless audio signals: Chord change detection for musical decisions. Speech Commun. 27(3-4), 311–335 (1999)
5. Tzanetakis, G., Essl, G., Cook, P.: Audio Analysis using the Discrete Wavelet Transform. In: Proceeding Conference in Acoustics and Music Theory Applications, AMTA 2001, Kiathos, Greece, pp. 318–323. Priceton University Press (2001)
6. Rohs, M., Essl, G., Roth, M.: CaMus: Live Music Performance using Camera Phones and Visual Grid Tracking. In: Proceedings of the 2006 International Conference on New Interfaces for Musical Expression, NIME 2006, IRCAM - Centre Pompidou, pp. 31–36 (2006)
7. Reidsma, D., Nijholt, A., Bos, P.: Temporal interaction between an artificial orchestra conductor and human musicians. Comput. Entertain. 6(4), Article 53, 22 p. (2008)