Use the Force (or something) - Pressure and Pressure-Like Input for Mobile Music Performance

Georg Essl
Electrical Engineering & Computer Science and Music
University of Michigan
gessl@eecs.umich.edu

Michael Rohs
Deutsche Telekom Laboratories
TU-Berlin
michael.rohs@telekom.de

Sven Kratz
Deutsche Telekom Laboratories
TU-Berlin
sven.kratz@telekom.de

ABSTRACT
Impact force is an important dimension for percussive musical instruments such as the piano. We explore three possible mechanisms how to get impact forces on mobile multi-touch devices: using built-in accelerometers, the pressure sensing capability of Android phones, and external force sensing resistors. We find that accelerometers are difficult to control for this purpose. Android's pressure sensing shows some promise, especially when combined with augmented playing technique. Force sensing resistors can offer good dynamic resolution but this technology is not currently offered in commodity devices and proper coupling of the sensor with the applied impact is difficult.

Keywords
Force, impact, pressure, multi-touch, mobile phone, mobile music making.

1. INTRODUCTION
Multi-touch mobile smart phones such as the Apple iPhone and the Google Nexus One offer rich interaction capabilities that can be used to turn these devices into expressive musical instruments [3]. These devices offer a range of sensors such as a multi-touch screen, 3-axis accelerometers, magnetic field sending, high resolution color video cameras.

However certain interactions remain unsatisfactory. While there are many applications available for the iPhone which mimic pianos they all lack the defining characteristic of the piano-forte, which is the ability to play softly and loudly by changing the impact force exerted on the key. Current multi-touch mobile phones do not offer ways to directly sense these forces. The purpose of this paper is to report on early explorations of three possible techniques to add force or force-like sensing either by reappropriating existing sensor information on the device, or by adding external sensors.

The three techniques explored based on using the accelerometer data, the width information of touch points and an external force-sensing resistor which is added to the existing hardware.

NIME2010, 15-18th June 2010, Sydney, Australia

2. RELATED WORK
This work is very much in the spirit of earlier work that explores the sensory capabilities of mobile devices for musical expression. Early on mobile devices came with very limited interactive sensory capabilities. Atau Tanaka used custom hardware to add accelerometers and force sensing resistors (FSRs) to PDAs [9]. The FSRs were used to sense grip force. Unfortunately no details about the use of the FSR in this early prototype were published. Since then force as an input has seen little interest while other sensory capabilities have been systematically explored [3,4,7,8,9,12]. Multi-touch has emerged as an important form of input since the emergence of the iPhone, and numerous academic and commercial products use this capability in various ways [2,13,14]. Yet, the iPhone's API does not provide access to quantities that could be related to force sensing. At the same time in the context of expanding the interactive capabilities, techniques have emerged, which allow for sensing of forces on mobile phones. Also alternative hardware and software environments have since appeared on the market. This includes the Android operating system, which offers the programmer generally more generous access to low-level hardware capabilities than is supported for the iPhone.

Within the HCI community force has recently emerged as an interesting interaction dimensions on mobile phones [1]. Our force sensing setup is closely related to the prototypes described in these works [5].

3. THREE APPROACHES TO SENSING FORCES
We take a pragmatic approach of considering sensing technology already available in current smart phones, and in addition consider one possible way to extend current hardware to achieve this kind of sensing.

The three approaches are:

1. Accelerometer-based: The build-in accelerometer is used to sense the intensity of the impact.
2. Touch-width- based: The Android API gives access to the width of a touch event. This width is heuristically linked to intensity.
3. External force sensing resistors: A multi-touch device is augmented by external force sensing resistors similar to the Sandwich hardware [1,5].

It is clear that these approaches measure different physical quantities via different sensor technologies. They are also different in terms of the ease of setting up. Accelerometer-based sensing is suitable for a range of smart phones, as the

availability of built-in 3-axis accelerometers has become widespread in modern smart-phones. Touch-width based interactions can in principle be used with a number of touch-screen based devices. However the needed API is not conveniently accessible on some of the platforms in question yet. So does the public API of the iPhone not give access to the width information of a touch event, even though the hardware and low-level operating system APIs may well offer this capability. In Android this information is available however, and it remains to be seen if this becomes equally widespread as accelerometer data. The disadvantage of both of these approaches is the indirect nature of the sensing process. Currently available commodity hardware does not offer direct pressure sensing on the screen. Hence the third option, one that relies on direct pressure sensing, is least accessible and requires by far the most effort to set up.

3.1 Accelerometer-based Force Sensing

Acceleration has a close relationship to force through Newton's second law of motion, which couples the acceleration experienced by a body to a force exerted on it by the mass of the body: F=ma. Hence one can expect to be able to extract some force information from acceleration sensing.

The accelerometer used in the iPhone 3G is STMicroelectronics LIS331DL and offers a range of +/-2g with a resolution of 0.018g. This means that one receives just under 8 bit (signed) resolution from the sensor.

In order to explore this approach we conducted a series of experiments to measure the relation between accelerometer data (under a range of signal conditioning approaches) to force.

For this purpose we used the iPhone app Context Logger by Embedded Systems Lab to record streaming accelerometer data on the iPhone. We used a PCB 484b force-sensing hammer to be able to select impacts of equal force. The signal from this force hammer was recorded using a TDS2024B Digital Storage Oscilloscope.

3.1.1 The signal and signal conditioning

In order to explore the signal and find suitable ways to extract measures of impact force, we recorded series of impacts at various peak impact forces. Figure 1 shows our initial recording of accelerometer data for this purpose. The recording consists

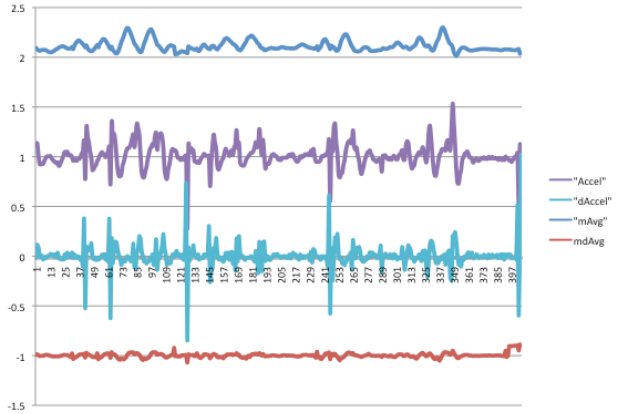

Figure 2 - Signal and various conditioning steps of 3-axis accelerometer data upon impact of sequences of increasing strength (repeated 4 times).

of 4 sequences of 4 strikes with increasing force. The purple curve is the raw data of the amplitude reported by all three axis combined. As can be seen it is difficult to see a good measure of force from the raw signal as the peaks and the slopes do not have a clear relationship to impact force. The light blue curve is the first order difference between neighboring measurements. This signal too does not give good measures, indicating that raw slope is not a suitable measure either. The dark blue curve is a moving average over 10 samples of the signal (offset by 1 for clarity). This signal shows a sensibly good representation of the impacts and in fact the gradient of the moving mean in practice proved to exceed peak estimation in accuracy.

This leads to an implementation of a piano-like instrument in the synthesis environment urMus [2] using a moving average estimate of force impact from accelerometer data.

In practice the dynamic range of this type of play is poor. Impacts of equal level often produce different levels of loudness. While stronger impact has a tendency to play louder, there seems to be very limited dynamic range and control.

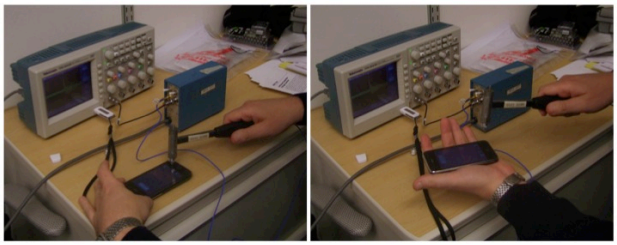

Figure 1 - Experimental Conditions for Variability Measurement of Accelerometer Impacts.

In order to explore the reason for this we recorded sequences of force strikes with the force hammer while simultaneously recording the accelerometer response to the impact in two conditions. The first condition was hand-held (see Figure 2) and the second condition was table-bound (Figure 3). We discarded all accelerometer recordings, which corresponded to a difference larger than 10% of the mean of all recordings. This means that the impact force for all recording was within 10 % for all data points.

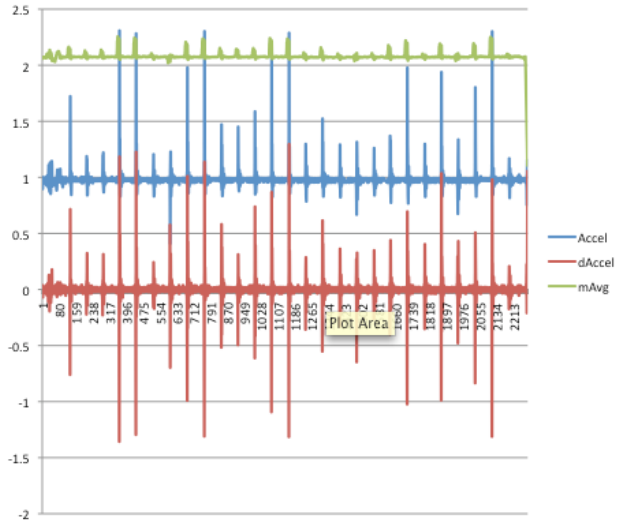

Figure 3 - Accelerometer data for force impact with at most 10 percent variation.

We then compared the output of the accelerometer data. This data can be seen in Figure 4 for the handheld case. For N=27 impacts within 10% measured force deviation we get the

following for accelerometer peak response: mean=1.628, stdv=0.421, min=1.168, max=2.308, median=1.4504. This is overall rather poor and suggests why the subjective playing experience is not stable. The table case shows similar results (min=1.16, max=2.44) and will be omitted. The accelerometer data seems to not be a good sensor for force impact due to the limited consistency. We do not know the reason for this behavior. The irregularity in response makes it unlikely that refined signal conditioning might be able to improve the sensing consistency.

3.2 Touch-Width-based Force-Like Sensing

The Android operating system specifies a range of sensor capabilities that can be supported by an Android-compatible phone. Among those capabilities is pressure and touch width information.

Some mobile smart phones such as the Google Nexus one and other Android-based phones support this functionality and return meaningful values to these when receiving a multi-touch event.

Unfortunately it is currently unclear what the precise technology is that supports this function. Hence we will treat the mechanism as black box. We will not assume that the value returned is a direct measure of pressure, but instead rely on experimentation to see the outcome.

Android offers "pressure" data via the MotionEvent getPressure() method. Android also does not specify the resolution of the value returned. Overall specs indicate a range of 0-1.0. We wrote a program to calculate the smallest observed differential between two pressure values different from zero. These measurements show that the minimal observed difference on a Nexus One Android phone is 0.003921, which corresponds almost exactly to an unsigned 8-bit resolution.

Figure 5 - Four distinct pressure levels on a Nexus One Android phone. Minimal pressure (top left), light pressure (top right), strong pressure (bottom left), flat finger (bottom right).

Using the result of getPressure() directly we found that one can distinguish between a soft and a hard impact, but with minimal differentiation between. However because in the Nexus One the value seems to be derived from the touch area one can increase the dynamic play substantially by flattening ones finger in play. In performance this does feel similar to after-touch play on a synthesizer keyboard. Extending ones playing style this way one can play 4 or more distinct loudness levels as depicted in Figure 4.

3.3 External Direct Force-Sensing

By adding a force sensor one can measure the forces between the mobile device and some background surface directly. There are earlier prototypes that offer this capability. For our purpose we use a reduced version of the Sandwich hardware [5], as depicted in Figure 5. Custom hardware is connected to the docking connector of an iPhone. The Custom hardware consists of an Arduino Nano plus auxiliary hardware to sense the data of a single force-sensing resistor (up to 4 are supported by the current prototype). This data is then made available to the iPhone through the docking connection. Earlier studies have shown that this setup has good dynamic range. We achieve a

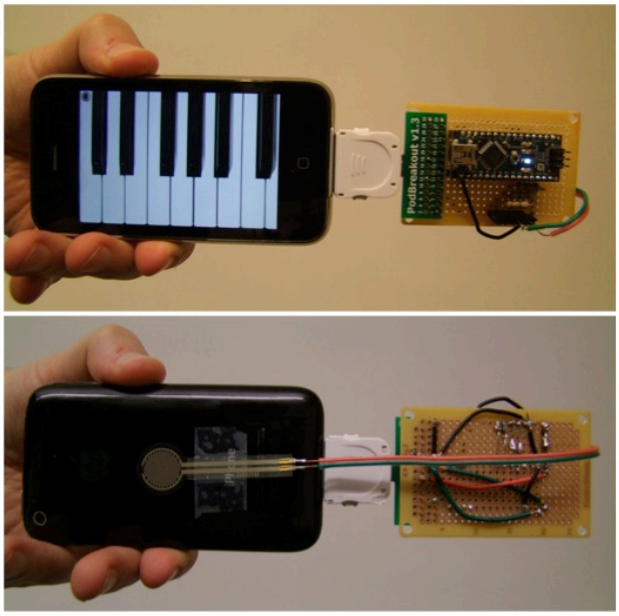

Figure 4 - The external hardware to support force sensing via a single force sensing resistor.

dynamic resolution of 9-10 bits.

Early experiments with the setup indeed show good dynamic range, but difficulty in practical use. The setup is very sensitive to calibration of the contact between the force sensor and the background surface. The device has to be held just right with the sensor against the back of the hand to achieve good dynamic responsiveness and just slight movement can either make it unresponsive or already biased by a base pressure. Clearly the sensor should best be placed just under the screen of the device with elastic compliance calibrated to give good and reliable input from the sensor. However if calibrated correctly the force data is stable and predictable. The Arduino driver supports a range of update speeds. In principle this rate can be increased to around 2000Hz until the serial protocol capacity is reached. For performance reasons with older iPhones we used 100Hz (or a delay of 10ms) and found this satisfactory for responsive updates.

4. DISCUSSION

On current devices the pressure input provided by Android phones like the Nexus One is the most attractive solution to provide impact based force input on mobile multi-touch devices. The dynamic behavior is sensibly stable and can be extended somewhat with flexible performance technique. However, the dynamic range is still relatively poor. In principle force-sensing resistors offer improved sensitivity for this purpose but proper integration of the sensor for reliable

application of the forces is a key problem that are difficult to address with an external prototype. We find that accelerometers are currently a poor sensor to use to detect impact-based forces. While the impacts themselves show clear signatures the problem of this method is the variability in the detected signal and the associated lack of a stable dynamic play.

5. CONCLUSIONS

We have yet to solve the problem of achieving good impact force dynamics on mobile smart phones. In this paper we discussed early experiments with three methods. We found that the most immediate solution, the use of built-in accelerometers shows poor consistency and hence is not attractive to use for this purpose. Pressure input as provided by some Android phones is a viable candidate to get some dynamic responsiveness. An early prototype with a force-sensing resistor shows promise, but the problem of properly integrating the sensor remains to be addressed to make this a reliable solution.

Future work is improved algorithms to support dynamic play with pressure input on Android phones as well as work on suspension of FSRs within the casing of mobile phones. Also currently the emphasis was on impact. We are interested in extending this work toward continuous pressure play such as soft gliding over a guitar string versus strong pressure glides.

6. ACKNOWLEDGMENTS

We thank Noel Perkins for generously lending us the force hammer and Edwin Olson for discussion and early help with data acquisition.

7. REFERENCES

[1] Stewart, C. Rohs, M., Essl, G. and Kratz, S. *Characteristics of Pressure-Based Input for Mobile Devices* In Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI 2010), Atlanta, Georgia, USA, April 10-15, 2010.

[2] Essl, G. *Urmus – An Environment For Mobile Instrument Design And Performance*, In Proceedings of the International Computer Music Conference, June 1-5, 2010.

[3] Essl, G., Rohs, M. *Interactivity for Mobile Music Making*, Organised Sound 14:2 197-207, 2009.

[4] Essl, G. and Rohs, M. 2007. *ShaMus – A Sensor-Based Integrated Mobile Phone Instrument*. Proceedings of the International Computer Music Conference (ICMC), Copenhagen.

[5] Essl, G. Rohs, M. Kratz, S. *Squeezing the Sandwich: A Mobile Pressure-Sensitive Two-Sided Multi-Touch Prototype* 22nd Annual ACM Symposium on User Interface Software and Technology (UIST), Victoria, BC, Canada, October 4-7, 2009.

[6] Geiger, G. 2003. *Pda: Real Time Signal Processing and Sound Generation on Handheld Devices*. Proceedings of the International Computer Music Conference, Singapore.

[7] Geiger, G. 2006. *Using the Touch Screen as a Controller for Portable Computer Music Instruments*. Proceedings of the International Conference on New Interfaces for Musical Expression (NIME'06). Paris, France.

[8] Misra, A., Essl, G. and Rohs, M. 2008. *Microphone as Sensor in Mobile Phone Performance*. Proceedings of the International Conference for New Interfaces for Musical Expression (NIME–08), Genova, Italy.

[9] Rohs, M., Essl, G. and Roth, M. 2006. *CaMus: Live Music Performance using Camera Phones and Visual Grid Tracking*. Proceedings of the 6th International Conference on New Instruments forMusical Expression (NIME), 31–6.

[10] STMicroelectronics LIS331DL Datasheet http://www.st.com/stonline/products/literature/ds/13951.pdf, retrieved on January 29, 2010.

[11] Tanaka, A. *Mobile Music Making*. In Proceedings of the International Conference on New Interfaces for Musical Expression, 154–6, 2004.

[12] Tanaka, A., Valadon, G. and Berger, C. *Social Mobile Music Navigation using the Compass*. Proceedings of the International Mobile Music Workshop, Amsterdam, 2007.

[13] Wang, G. *Designing Smule's iPhone Ocarina*. In Proceedings of the International Conference on New Interfaces for Musical Expression. Pittsburgh, 2009.

[14] Weinberg, G., Beck, A., Godfrey M. *ZooZBeat: a Gesture-based Mobile Music Studio.* In Proceedings of International Conference on New Instruments for Music Expression (NIME 09), Pittsburgh, PA, 2009.