Enabling Multimodal Mobile Interfaces for Interactive Musical Performance

Charles Roberts
University of California at Santa Barbara
Media Arts & Technology
charlie@charlie-roberts.com

Angus Forbes
University of Arizona
School of Information: Science, Technology, & Arts
angus.forbes@sista.arizona.edu

Tobias Höllerer
University of California at Santa Barbara
Dept. of Computer Science
holl@cs.ucsb.edu

ABSTRACT

We present research that extends the scope of the mobile application *Control*, a prototyping environment for defining multimodal interfaces that control real-time artistic and musical performances. Control allows users to rapidly create interfaces employing a variety of modalities, including: speech recognition, computer vision, musical feature extraction, touchscreen widgets, and inertial sensor data. Information from these modalities can be transmitted wirelessly to remote applications. Interfaces are declared using JSON and can be extended with JavaScript to add complex behaviors, including the concurrent fusion of multimodal signals. By simplifying the creation of interfaces via these simple markup files, Control allows musicians and artists to make novel applications that use and combine both discrete and continuous data from the wide range of sensors available on commodity mobile devices.

Keywords

Music, mobile, multimodal, interaction

1. INTRODUCTION

"Multimodal" interfaces are broadly defined to combine multiple signals from different modalities in order to generate a single coherent action[5]. Although there are many examples in the electronic music community of interfaces that include synchronization mechanisms, for instance [2], it is more common for musicians to map individual modalities to individual parameters than it is to employ multimodal fusion. That is, while the detection of discrete features involving the analysis of various multimodal signals is one important aspect of multimodal interfaces, a second important aspect for musical applications is enabling the direct mapping of continuous sensor output to expressive control parameters.

The use of multimodal signals for the control of musical parameter spaces is found in various domains, from avant-garde electronic compositions to commercial implementations of multimodal musical interfaces. In David Rosenboom's "Portable Gold and Philosophers' Stones (Music From Brains In Fours)", first performed in 1972, four performers don a combination of EEG electrodes, body temperature monitors and galvanic skin response monitors[18]. Signals from the body temperature and galvanic skin response monitors control the frequency ratios of four droning pulse waves. These waveforms are then fed into a resonant bandpass filterbank, in which the center frequency and output amplitude of each band is controlled by an analysis of the EEG signals. This creates a modulating output where individual frequencies are slowly emphasized and muted over the course of the eighteen minute piece in response to biological signals obtained from the performers.

In 1994, the Yamaha Corporation released an influential commerical product, the VL-1 physical modeling synthesizer, that made extensive use of multimodal signal processing. The sonic realism possible with physical models is highly dependent on the quality of control signals being fed to the model; accordingly, the VL-1 is a duophonic instrument with up to twelve parameters of the model exposed for simultaneous control using a standard piano keyboard, a breath controller, multiple pedal inputs and various wheels, knobs and sliders.

The musical interest in exploring different modalities to expressively control large parameter spaces has recently expanded to include the affordances offered by mobile devices. Atau Tanaka, perhaps best known for his work using electromyography and other bio-sensors in performance, has more recently included mobile devices in his performances. He is attracted to the "autonomous, standalone quality - the embodied nature - of a mobile instrument" that allows the expressive mapping of sensors to musical parameters controlling audio generated on the devices themselves[22].

Our research provides electronic musicians with a vehicle for exploring multimodal interfaces on mobile devices. We extend the open-source application *Control*[16], available for both iOS and Android devices[1], to include a variety of modalities not previously present in Control or any other mobile interface designer. Interfaces in Control are created with simple JSON markup and can be extended via JavaScript to aggregate signals from multiple modalities into discrete outputs and to implement other complex behaviors.

2. RELATED WORK

Although there are a number of toolkits for multimodal interface development on desktop computers, such as [20, 4], there are currently no applications available that provide simple APIs for reading the myriad of sensors available on mobile devices. Our research draws upon previous work that enables complex interactivity through the use of multiple sensors.

An early inspiration for many mobile applications focusing on interface creation for musical control is the JazzMu-

[1]Access to many of the sensors described in this paper is currently only available in the iOS version of the app.

NIME'13, May 27 – 30, 2013, KAIST, Daejeon, Korea.

tant Lemur[7]. The Lemur was a standalone touchscreen device first introduced in 2006. It featured unlimited multitouch tracking and allowed users to design their own custom interfaces to run on it. In addition to a wide variety of touchscreen widgets the Lemur also featured a physics engine and scripting capabilities that afforded complex interactivity. The software from the Lemur was ported to run on iOS devices in late 2011, however, the current version of Lemur only collects input from the touchscreen and accelerometer while ignoring the other sensors found on mobile devices. Many related iOS apps for interface creation (TouchOSC[24], mrmr[11], etc.) also have this shortcoming.

An impressive application related to Control is *urMus*[6] which provides access to a variety of sensors available on mobile devices. urMus enables users to define interfaces for expressive musical control of remote applications as well as synthesis algorithms to be executed on mobile devices; as such it provides a complete system for designing NIMEs while Control instead focuses on controlling networked computers and MIDI devices. Both audio synthesis and interfaces in urMus are defined programmatically using Lua. Some aspects which differentitate the Control application from urMus in regards to multimodal interfaces are the inclusion of speech recognition, a more robust video analysis module, support for aftertouch, and a simpler environment for creating interfaces using markup.

A number of toolkits for augmented reality are available on the market, each of which provide support for the analysis of a live camera feed, for the tracking of particular features, and for the rendering of auxiliary information or models onto the original scene representation captured by the video camera. Some AR toolkits claim to support multimodal interaction, for instance [23, 14], but in fact support it only implicitly through its integration with the mobile operating system or via another toolkit, such as Unity3D mobile game engine[25]. An execption to this is [8], which provides tracking through the integration of "gravity aware AR" which measures orientation using the device's accelerometer. That is, the combination of two signals is used to improve both feature tracking and then the rendering of information using that tracking. However, this functionality is not directly accessible to the user nor able to be manipulated by a user via an interface. Other toolkits provide multimodal use case scenarios, such as integrating GPS data with object recognition so that a site-specific data set can be pre-loaded in order to improve the recognition capabilities[23, 21]. However, these modalities are accessed sequentially and the toolkits do not explicitly support the concurrent control of multiple signals.

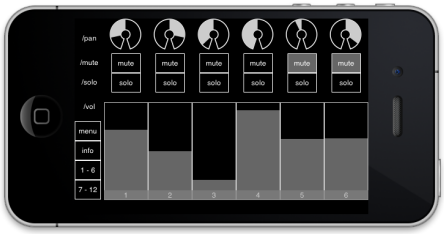

Figure 1: A mixer interface in Control with volume, pan, mute and solo controls for six channels.

3. AN OVERVIEW OF CONTROL

Anonymous is an application for iOS and Android that allows users to define interfaces that output OSC and MIDI messages. It is built on top of the PhoneGap framework [1], which enables cross-platform mobile applications to be built using web technologies. Interfaces created in Control look the same on iOS and Android tablets and phones. Users define interfaces in Control using JSON (Javascript Object Notation), a simple markup language. Advanced users can extend interfaces with JavaScript to enable interactions of greater complexity[17, 9]. A more detailed account of the features and implementation details of Control can be found in the original publication on the software[16].

4. CONTRIBUTIONS

Prior to this research, Control only had access to the inertial sensors on the mobile devices, enabling users to make use of raw accelerometer and gyroscope output and orientation information calculated using a variety of sensors. Our work on extending Control newly provides access to the microphone, camera (both front and back) and magnetometer and integrates feature extraction for audio and video signals to simplify their use in performative contexts. The feature extraction algorithms provided include speech recognition, pitch detection, amplitude detection, changes in the brightness and color distribution of video, and blob tracking. We also added support for a crude yet surprisingly musical approximation of aftertouch that is derived from the surface radius of individual touches.

4.1 Audio Input and Analysis

4.1.1 Musical Feature Analysis

Feature analysis of audio input enables many important musical mappings. As one example, the amplitude of audio entering the microphone on a mobile device can both trigger musical notes when the volume crosses a certain threshold and also control their amplitude. By measuring the fundamental frequency of pitched sounds entering the microphone we enable users to sing the pitches of notes they would like to hear electronically synthesized. Control provides feature extraction algorithms to measure both the pitch and amplitude of incoming audio signals.

Amplitude is measured by observing the highest sample value in an audio window or by calculating the root mean square of all samples. The amplitude can be determined concurrently with the pitch of periodic signals. Pitch is detected by measuring the number of zero crossings in each audio window or by finding the harmonic product spectrum (HPS) of the signal; users may specify which method to use depending on their particular application.

4.1.2 Speech Recognition

Control uses speech recognition provided by the PocketSphinx library from Carnegie Mellon University[3]; this library is wrapped by the OpenEars framework[13] to make it easier to deploy under iOS. Custom dictionaries can be defined on a per interface basis using JSON markup and are dynamically generated when the interface is loaded. Whenever a word or phrase in the dictionary is recongized, Control will output a UDP message to the address matching the name of the Speech object (in this case `/speechRecognizer`) and pass the recognized word or phrase as an argument to the message. Users can also define a custom callback to be called whenever a word or phrase is recognized; this callback can be used to change aspects of the interface itself in addition to sending messages to remote destinations. The Speech object included with Control also includes `listen()` and `stopListening()` methods that can be used to control when speech recognition is engaged. Finally, the dictionary used by the speech object can be changed dynamically at

runtime via UDP messages or by scripts triggered from inside the interface.

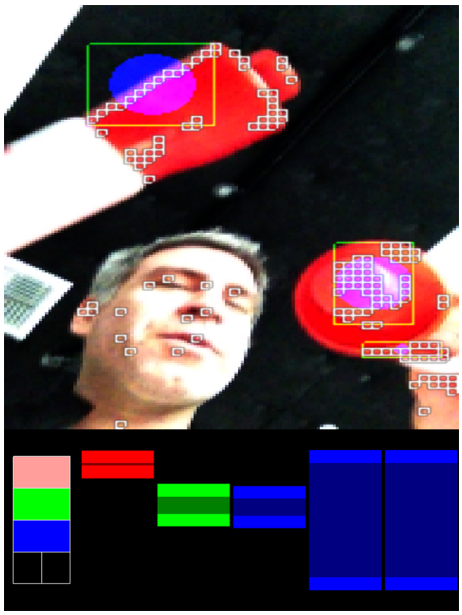

Figure 2: A screen capture of Control's blob detection module. The user has selected a light red pixel by touching the screen and attentuated the threshold in the red channel so that blobs are detected only on the marker caps.

4.2 Video Input and Analysis

Many researchers and performers have investigated gesture tracking and other feature detection from video input for interactive performance. For example, [19] uses a live video feed of conducting gestures to control musical parameters; [12] incorporates optical tracking to enhance music and dance performance; and [10] analzyes a live video feed of facial movements for expressive musical control. Our video analysis module more generally allows users to analyze a real-time feed of video data from the camera and apply the results to arbitrary functionality. In particular, we have implemented a custom blob-tracking algorithm, similar to [15], as well as a detector of changes in average brightness and color. Information about the number of blobs, the overall brightness, as well as the size, centroid position, average color, average brightness, and density of each blob can be used to control musical parameters. The blob-tracking interface emphasizes ease-of-use for real time tracking and thresholding of one or multiple pixel regions. The user can pick a pixel range simply by touching the screen to retrieve the specific color or brightness values at that spot, and then adjust the range of values in each channel to include or exclude adjacent pixels. Further real-time control is available so that users can adjust the minimum and maximum density of a blob as well as the minimum and maximum size of a blob. The live-video feed, the threshold map of pixels within a particular pixel range, and the blobs discovered within these pixel ranges are blended together in a single window so that the user has real-time feedback of which blobs are currently active, as well as information about how to adjust the parameters to track additional blobs. This makes it easy for the user to immediately identify interesting features in various environments, even if they change during a performance. Moreover, since most mobile devices prohibit programmatic access to camera calibration (e.g., preventing the automatic updating of white balance), this is particularly important as pixel values may shift unexpectedly when lighting conditions change.

5. MULTIMODAL FUSION IN CONTROL

A premise of our research is that musical applications require the ability to control large parameter spaces concurrently and expressively; they are less likely to require the fusing of control signals from multiple modalities into individual actions. However, there are certainly use cases for multimodal fusion in the digital arts. As a simple example, consider an emulation of a wind instrument. As mentioned previously, the microphone can detect when a user is blowing into it; we can then trigger musical notes whenever a certain amplitude threshold is crossed. However, for a typical wind instrument it is not only necessary for the performer to blow into the instrument to generate a note; they must also press down on a keypad or valve. The scripting support in Control greatly facilitates the programming of such input fusion. All events, whether they are button presses or blob detections, trigger the execution of event handlers in JavaScript. Users can easily assign custom event handlers to widgets that evaluate the state of other sensors and perform actions accordingly. As one example, almost all sensors possess an "onvaluechange" event handler that is triggered whenever the state of a sensor changes. Other event handlers available for the touchscreen widgets provide the ability to detect when touches enter, move through and leave the boundary box of widgets.

6. CONCLUSIONS

Although current mobile devices feature a variety of sensors that makes them ideal for multimodal interfaces, there has been a need for an application that allowed users to take full advantage of these sensors in an integrated fashion. Our research extending Control provides such an application. As Control has already been downloaded over 40,000 times on iOS alone, the updates we are providing will immediately make the creation of multimodal interfaces more accessible to existing users and encourage users downloading the application for the first time to explore the inherent multimodality of their devices.

Our future work with multimodal interface creation will primarily consist of adding new feature extraction algorithms. Some examples to explore include methods to determine the velocity of a finger strike on the touchscreen using the microphone and accelerometer, signal processing to determine the tempo of rhythmic audio entering the microphone and video processing techniques to use the camera and flash for photoplethysmography. Ongoing research more general to Control involves letting users create interfaces from within the app itself using drag-and-drop methods, which would further simplify the interface creation process empower users who might be intimidated by the process of creating interfaces using markup.

Control is open source and can be downloaded from a git repository located on GitHub[2].

7. REFERENCES

[1] PhoneGap - Home. `http://phonegap.com/`, accessed February 2013.

[2] B. Bell, J. Kleban, D. Overholt, L. Putnam, J. Thompson, and J. Kuchera-Morin. The multimodal music stand. In *Proceedings of the 7th international*

[2]https://github.com/charlieroberts/Control/tree/develop

conference on New interfaces for musical expression, pages 62–65. ACM, 2007.

[3] Carnegie Mellon University. http://cmusphinx.sourceforge.net/2010/03/pocketsphinx-0-6-release/, accessed May 2012.

[4] P. Dragicevic and J. Fekete. Icon: input device selection and interaction configuration. In *Companion proceedings of the 15th ACM symposium on User Interface Software & Technology (UIST'02), Paris, France*, pages 27–30, 2002.

[5] B. Dumas, D. Lalanne, and S. Oviatt. Multimodal interfaces: A survey of principles, models and frameworks. *Human Machine Interaction*, pages 3–26, 2009.

[6] G. Essl. Designing mobile musical instruments and environments with urmus. *Proceedings of the 2010 Conference on New Interfaces for Musical Expression*, 2010.

[7] JazzMutant Lemur. http://www.jazzmutant.com/lemur_overview.php, accessed May 2012.

[8] D. Kurz and S. Benhimane. Gravity-aware handheld augmented reality. In *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on*, pages 111–120. IEEE, 2011.

[9] Q. Liu, Y. Han, J. Kuchera-Morin, M. Wright, and G. Legrady. Cloud bridge: a data-driven immersive audio-visual software interface. *Proceedings of the 2013 Conference on New Interfaces for Musical Expression (NIME 2013)*, 2013.

[10] M. Lyons. Facial gesture interfaces for expression and communication. In *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, volume 1, pages 598–603. IEEE, 2004.

[11] mrmr. http://mrmr.noisepages.com/, accessed May 2012.

[12] J. A. Paradiso and F. Sparacino. Optical tracking for music and dance performance. In *Optical 3-D Measurement Techniques IV*, pages 11–18. Herbert Wichmann Verlag, 1997.

[13] Politepix. http://www.politepix.com/openears, accessed May 2012 2012.

[14] Qualcomm's Vuforia SDK. https://ar.qualcomm.at, accessed May 2012.

[15] C. Rasmussen, K. Toyama, and G. Hager. Tracking objects by color alone. *DCS RR-1114, Yale University*, 1996.

[16] C. Roberts. Control: Software for End-User Interface Programming and Interactive Performance. *Proceedings of the International Computer Music Conference*, 2011.

[17] C. Roberts, B. Alper, J. Morin, and T. Höllerer. Augmented textual data viewing in 3d visualizations using tablets. *3D User Interfaces (3DUI), 2012 IEEE Symposium on*, pages 101–104, 2012.

[18] D. Rosenboom. *Biofeedback and the arts, results of early experiments*. Aesthetic Research Centre of Canada, 1976.

[19] A. Salgian, M. Pfirrmann, and T. Nakra. Follow the beat? understanding conducting gestures from video. *Advances in Visual Computing*, pages 414–423, 2007.

[20] M. Serrano, L. Nigay, J. Lawson, A. Ramsay, R. Murray-Smith, and S. Denef. The openinterface framework: a tool for multimodal interaction. In *CHI'08 extended abstracts on Human factors in computing systems*, pages 3501–3506. ACM, 2008.

[21] String Augmented Reality. http://www.poweredbystring.com, accessed May 2012.

[22] A. Tanaka. Mapping out instruments, affordances, and mobiles. In *Proceedings of the 2010 Conference on New Interfaces for Musical Expression (NIME 2010)*, pages 88–93, 2010.

[23] Total Immersion's D'Fusion Suite. http://www.t-immersion.com, accessed May 2012.

[24] TouchOSC by Hexler. http://hexler.net/software/touchosc/, accessed May 2012.

[25] UNITY: Game Development Tool. http://unity3d.com, accessed May 2012.