A sensor pairing and fusion system for a multi-user environment

Jari Kleimola, Maurizio Mancini, Giovanna Varni, Antonio Camurri, Carlo Andreotti, Longfei Zhao

***Abstract*—This paper proposes a system for sensor pairing and fusion in an interactive multi-user environment. Using the system, we integrated mobile accelerometer and fixed position optical tracking methods, and implemented two active music listening applications based on the movement interaction from one or more users. We found that an acceleration domain similarity index between the two tracking methods is able to pair the raw interaction streams in near real-time, and that concurrent sampling of the streams allows for easy sensor fusion. Although these algorithms still require further refinement, we believe that combining accurate position with accurate acceleration data is beneficial for novel interactive applications.**

***Index Terms*—interactive multisensor systems, mobile phones, music, sound synthesis, tracking**

I. Introduction

In the recent past, when mobile phones were used primarily for voice and SMS communication, phone microphone and keypad sufficed to fill the interaction needs of most users. Today, a growing number of phones embed accelerometers, cameras, compass, GPS, and touch sensors, which extend the number of available input modalities, context processing capabilities, and use scopes well beyond the original communication scenarios. Mobile phones are becoming fundamental players in user-centric media applications: they can be used as multimodal interfaces, or personal transducers having many uses for our everyday lives.

However, these extended input capabilities are rarely utilized in contemporary interactive applications. To date, interactive applications are mostly exploiting fixed external sensors (for example, fixed cameras for motion or color tracking) that provide "traditional" kind of input streams, such as exact coordinate positions in a well defined coordinate system.

Most embedded mobile sensors are not well suited for producing this type of traditional input. For example, a phone camera can be used to compute approximate coordinate positions using optical flow, but the resulting data is often too inaccurate. Likewise, an integrated accelerometer can provide high resolution acceleration data that is precise, but when double-integrated to obtain the coordinate positions, the cumulative errors in the processing phase result in inexact data.

This work aims at providing methods to support the creation of usable, effective mobile applications, resting on the balance between traditional techniques based on environmental sensors (e.g., fixed video cameras) and embedded mobile sensors. That is, methods implementing sensor *pairing* and *fusion*. For example, traditional camera-based tracking fails to detect minimum acceleration deviation while gesturing: precise coordinate position, but poor acceleration data can be obtained, due to noise in double derivation. On the other hand, accelerometers provide precise acceleration data but poor coordinate position due to noise in double integration.

The work we are presenting in this paper is based on the assumption that we are acquiring synchronized data from different sensors from multiple users. Performing *sensor pairing* corresponds to identify to which user each data refers to. *Sensor fusion* occurs when we are able to use the paired combined data from different sensors referring to every single user.

More precisely, this eNTERFACE project addresses the following research problems that are currently investigated by the SAME project [1]:

- How to combine mobile and fixed position camera – based data tracking with phone embedded accelerometer tracking, in order to integrate and exploit the best properties of both approaches;
- To study whether it is possible to extract high level motion features by exploiting this integration;
- To analyse and compare the pros and cons of the two data acquisition methods.

For these aims, we propose a system for sensor pairing and fusion in a multi-user environment. In particular, we implemented two active music listening scenarios: the Audiovisual Air Instruments and the Mobile Conductor.

In the direction of defining novel active listening paradigms [1], Camurri et al. [2] recently developed a system, named Orchestra Explorer, allowing users to physically navigate inside a virtual orchestra, to actively explore the music piece the orchestra is playing, to modify and mold in real-time the music performance through expressive full-body movement and gesture. By walking and moving on the surface, the user discovers each single instrument and can operate through her expressive gestures on the music piece the instrument is playing. The interaction paradigm developed in the Orchestra Explorer is strongly based on the concept of navigation in a physical space where the orchestra instruments are placed. The Orchestra Ex-

Manuscript received November 19, 2009. This work was supported in part by the European Union (EU) as part of the 7th Framework Programme with the SAME project (ref. 215749).

Jari Kleimola is with the Dept. of Signal Processing and Acoustics, Helsinki University of Technology (TKK), Espoo, Finland (jari.kleimola@tkk.fi).

Maurizio Mancini, Giovanna Varni and Antonio Camurri are with the Infomus Lab, Dept. of Communication, Computer and System Sciences (DIST), University of Genova, Italy (maurizio.mancini@dist.unige.it, Giovanna.varni@gmail.com, antonio.camurri@unige.it).

Carlo Andreotti and Longfei Zhao are with the Dept. of Communication, Computer and System Sciences (DIST), University of Genova, Italy.

plorer paradigm has been implemented in two different scenarios. The first is based on users tracking by fixed video cameras observing the sensitive space where user(s) can navigate and interact with the sections of the virtual orchestra. A second version is based on mobile systems (Nokia S60 family of mobiles), where the onboard 3D accelerometer is used to navigate a colored cursor on a map representing the orchestra in the phone display.

Camurri et al. propose also a more sophisticated active listening paradigm [3], where multiple users can physically navigate a polyphonic music piece, actively exploring it; further, they can intervene on the music performance modifying and molding its expressive content in real-time through non verbal full-body movement and expressive gesture. An implementation of this system, named Mappe per Affetti Erranti, was presented in the framework of the science exhibition "Metamorfosi del Senso", held at Casa Paganini, Genova, in October – November 2007. In that occasion Mappe per Affetti Erranti was also used for a contemporary dance performance.

The reminder of this paper is organized as follows. Section II provides a system overview, while sections III through V explain the system in detail. Sections VI and VII present the implemented application scenarios. Finally, Section VIII discusses the outcomes and perspectives of this project.

II. SAME project framework for Active Music Listening

Our system is developed in the SAME networked platform, which is an end-to-end framework (i.e., between clients of a mobile service, producers and consumers of the content) for context-aware, experience-centric mobile music applications, enabling embodiment and control of music content by user behaviour. The platform includes one or more remote or local servers, running software environments such as EyesWeb XMI [4], Pd [5] and vvvv [6].

Remote SAME servers may provide services related to the access and retrieval of audiovisual content (e.g., music content, see fig. 1), as well as the remote support of users. Local SAME servers may include services connected to the physical environment, as in our case, the real-time processing of the signal from a fixed position camera. The services include also the ability to process and manage the applications based on the interactions captured from the mobile phones.

The mobile phones may embed MobIO environment, which is a multimodal mobile IO framework developed in the SAME project.

A. System description

In this work, we propose a system for sensor pairing and fusion using the components of the SAME project framework. The hardware setup consists of mobile phones (e.g., Nokia N85 and N95), which are connected to a PC via a wireless network. The PC interfaces also a fixed position video camera, a projector, a MIDI synthesizer [7] and a loudspeaker setup.

The PC runs a local SAME server, hosting three applications (see fig. 2). *EyesWeb* is responsible for video tracking, gesture recognition, feature extraction and MIDI playback. *Pd* is used as a sound synthesizer, while *vvvv* serves as the graphics engine.

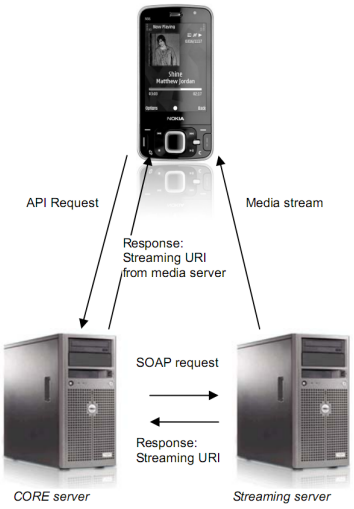

Figure 1: SAME platform in a streaming configuration. The mobile phone can send and receive data with two servers: the CORE server handles the phone requests, while the Streaming server deals with the content streaming.

The mobile phones run an embedded application utilizing the *MobIO* framework. The application is responsible for capturing the accelerometer and mobile camera streams from the phone embedded sensors, and routing them to the local SAME server applications for further processing and audiovisual synthesis tasks.

The components are connected together using IP-based Open Sound Control (OSC) [8] and EyesWeb proprietary streaming video protocols.

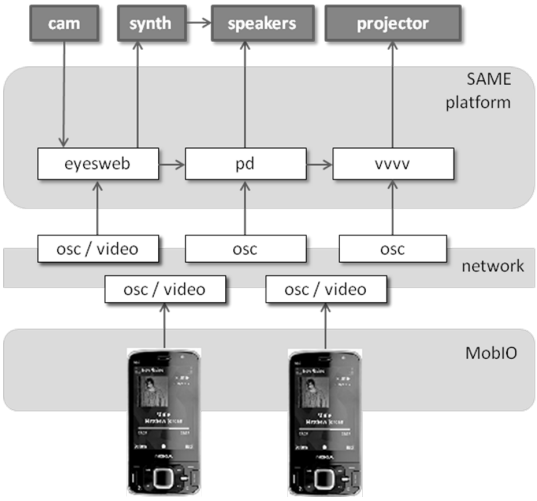

Figure 2: SAME project framework in a local configuration.

Figure 3 depicts the conceptual diagram of the system. Low level features are captured from the mobile phones, either by using the sensors of the phone itself, or by using a fixed camera tracking the position of the phone. The high level features (impulsivity, curvature and so on) are computed from these data. The audiovisual output of the system is then computed by

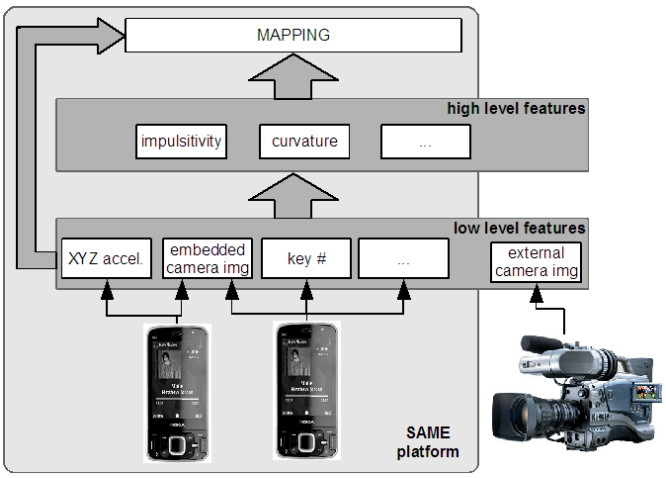

Figure 3: A system for sensor pairing and fusion in a multi-tracking environment.

the result of a mapping process.

III. LOW AND HIGH LEVEL FEATURES

This section describes the low level data acquisition and high level formulation tasks utilizing MobIO and EyesWeb environments.

A. Inputs from the mobile phones

The accelerometer and camera sensors of Symbian-based mobile phones, such as Nokia S60 devices, are accessible through C++ APIs that – unfortunately – depend on the operating system version of the phone. The MobIO framework models these sensors as black boxes, attempting to hide the version differences between the different APIs. The boxes are also equipped with input and output ports in order to have a consistent parametrization interface across different classes of objects, and more importantly, affording the construction of dataflow networks from the interconnected boxes. This concept is semantically equivalent to the graphical programming paradigm employed in EyesWeb, Pd and vvvv environments, but because MobIO has programming language bindings, the syntax level is different.

MobIO calls these black boxes *Units* that can be interconnected to form *Patches*. In this work, the patch encapsulates dataflow paths for streaming the accelerometer data as OSC messages (Acc → OSCWriter → UDPOut), streaming the video to the EyesWeb (Cam → EyWImageStreamOut), processing the video and streaming the optical flow out as OSC (Cam → MeLib → OSCWriter → UDPOut), and detecting and generating keypress events (Key → OSCWriter → UDPOut). The Acc, Cam, Key and UDPOut units encapsulate the raw input and output APIs of the Symbian SDK. The OSCWriter unit generates OSC-formatted messages within /acc, /key and /opt namespaces. The EyWImageStreamOut unit converts the raw video stream of the phone camera into a 64x64 pixel grayscale bitmap stream before forwarding it to the EyesWeb. Finally, MeLib unit interfaces a phone hosted motion estimation library [9], capable of producing a stream of positional changes in the horizontal, vertical, distance and rotational dimensions.

B. Inputs from the fixed and mobile camera

The video signal provided by a fixed camera is processed in real-time by an EyesWeb application (or *patch*). It identifies blobs corresponding to the mobile phones. Each mobile phone screen shows a different color, see Figure 4. A color-based tracking allows us to follow the 2D position of the phones frame by frame in the video stream. The tracking module takes as input a list of the blobs detected in the current frame of the video stream and compares them with the blob list detected in the previous frame and stored in an internal buffer of the module. A weighted distance area criterion is used to match and track blobs frame by frame.

The mobile camera sends video stream to an EyesWeb patch that performs the optical flow computation on this stream. The flow is computed on each video frame compared to the previous ones. For each pixel of the frame we obtain the movement vector (dx,dy) that represents how much and in which direction the pixel has moved between the two frames. Then we compute the frame mean optical flow on the X and Y axis separately as the arithmetical mean of the X and Y components of each pixel movement vector.

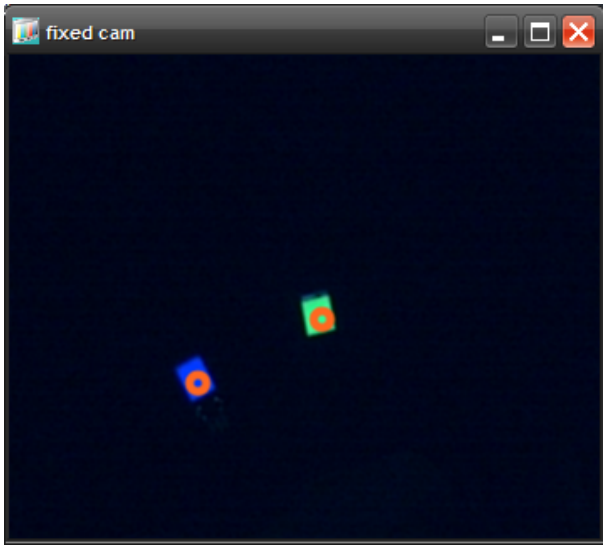

Figure 4. EyesWeb tracking the mobile phones.

C. Expressive Motion Features

The algorithms for the automatic evaluation of motion features, have been implemented in the EyesWeb software platform (www.eyesweb.org) using the EyesWeb Expressive Gesture Library [4]. From the low-level features described above, we computed the following motion features:

- *Impulsivity* (computed from mean optical flow) indicates whether or not movement presents sudden and abrupt changes in energy. We define impulsive gestures as those characterized by short duration and high magnitude.
- *Directness* (computed from absolute acceleration) is a measure of how direct or flexible a trajectory between two

points in space is. We obtain it by comparing the length of the minimal distance between the two points with the length of the actual gesture trajectory.

• *Velocity* (computed from absolute acceleration) corresponds to the first order derivative of the gesture trajectory coordinates.

IV. Sensor pairing and synchronization

This section describes two alternative algorithms for sensor pairing and synchronization.

A. Real-time Pairing and Synchronization

The pairing of the acceleration measured from the mobile phone and the acceleration from the computer–vision based tracker engine was developed by means of a real-time EyesWeb XMI application.

Two users are in front of the fixed camera and they frontally move, respectively, (i) a mobile phone showing a green marker and (ii) a red marker. The user with the mobile performs a shaking movement, whereas the other user freely moves the marker. The absolute acceleration is computed both from the accelerometer and from the second order derivatives of the barycenter coordinates of the colored markers tracked by EyesWeb. Then, a *similarity index* among these absolute accelerations is extracted providing a criterion to pair the accelerometer with the corresponding colored blob.

, There are 3 streams coming from the acquisition: (i) acceleration from the mobile accelerometer; (ii) acceleration from the green marker tracking (also related to the mobile movements); (iii) acceleration from the red marker tracking. A generalized auto-correlation function is computed for each stream and then it is normalized to zero mean and unitary standard deviation. Finally, the application performs a correlation coefficient (the Correlation Probability of Recurrence [14]) among the generalized auto-correlation function of the mobile acceleration and the generalized auto-correlation functions of the tracked markers acceleration and a numeric comparison is done. The marker having the highest correlation coefficient is associated to the accelerometer in the mobile phone.

B. Offline Pairing and Synchronization

Regarding to the limitations of hardware and software, significant noise is present, which is not easy to cancel or at least reduce to an acceptable level (in terms of system usability). For example, because of the pre-emptive nature of the operating system, the 3D accelerometers embedded in mobile phones have time-varying sampling that produce non-uniform latency. Likewise, without an embedded orientation sensor it is impossible to determine the angle between the phone and the fixed position camera. In some conditions, these errors lead to unexpected pairing results, and therefore the analysis is split into pre-processing and matching parts.

The *pre-processing* part aims to remove useless information from the original data, to smooth and to find out the potential information, to let them available for subsequent evaluation algorithms. Because we perform the analysis in the acceleration domain, the position extracted from the fixed camera tracking is double-differentiated in this part. Also, considering the orientation of mobile phone is undetermined when moving, we introduce Energy Space by using quadratic sum of accelerations to ignore the orientation.

In the *matching* part, three algorithms (Wing, Covariance and Row) are used to check the similarity between the two acceleration data sets. Custom coefficients are given to these algorithms according to different hardware setups. If one set of data from accelerometers meets the highest total score with another set of data from the camera, they're determined to be a pair.

- The *Wing algorithm* is an extension of peak detection, it evaluates the similarity using the shapes of signals. A wide and high wave denotes a strong action. The probability for the other one to follow should then be larger. So the Wing Algorithm is related with the sum of such probabilities on each wave.

- The *Covariance algorithm* returns the correlation coefficients calculated from an input matrix [*x*, *y*], where *x* is the acceleration from accelerometer Energy Space, and *y* is the acceleration from camera in Energy Space.

$$r_{xy} = \frac{\sum x_i y_i - n\bar{x}\bar{y}}{(n-1)s_x s_y} = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n\sum x_i^2 - (\sum x_i)^2}\sqrt{n\sum y_i^2 - (\sum y_i)^2}}.$$

- The *Row algorithm:* Rate based on Observe Window Algorithm evaluates the strength similarity of the signals based on magnitude ratio in observe window of Energy Space. Observe Window can remove small fluctuation, which makes the rate comparing method more robust.

The pairing method turned out to be reliable and robust when the 2D movement was strictly perpendicular to the camera. To extend to more general 3D movement, the orientation of the mobile phone itself is needed. And for dedicate hardware, parameters determined based on some tests will lead to a better synchronization performance.

V. Communication among system modules

The system modules are interconnected using IP-based protocols (see fig. 2). In particular, the input and output namespaces of the two implemented active music listening scenarios are described using OSC address strings and their numerical arguments. On the input side, the interaction data consists of three-axis acceleration (/acc ax ay az), two-dimensional position (/pos x y) and key press (/key code state) events. The arguments of the /acc messages are normalized to floating point g-forces ranging from -2g to 2g. The position is expressed as a floating point coordinate pair in a (0,1) times (0,1) plane, roughly covering the reachable area in front of the user. The key presses from the mobile phone keypad are con-

verted to ASCII, with a Boolean state indicating down or up position of the key.

The interaction streams from the two mobile phones are transmitted separately, and prefixed by a mnemonic name tagged to the mobile. For example, the accelerometer stream of the first phone is transmitted as "/blue/acc x y z", while the position of the second would be given as "/green/pos x y". The tags can either be fixed at the phone end, or created dynamically according to the pairing procedure described in Section IV A.

The mobile video stream is transmitted as an EyesWeb proprietary streaming protocol. The payload of the stream consists of uncompressed grayscale images.

VI. Scenario 1: Audiovisual Air Instruments

In this scenario, the user plays a virtual musical instrument by gesturing in the air. The instrument is controlled either by impulsive vertical hand gestures (strokes), or more relaxed horizontal actions (sweeps), and it responds by producing synchronized aural and visual feedback.

In the installation, the user stands in front of a large projection screen and a loudspeaker setup, holding two mobile phones in his hands. The mobile phones are used as input transducers, or virtual drum sticks / mallets, generating raw interaction streams. The system analyzes the input streams in order to extract the stroking and sweeping gestures, tracks the positions of the phones, and renders the interaction using aural and visual rendering engines.

The aural output is generated by a software sound synthesizer, producing pitched and non-pitched percussive FM timbres [9] in response to vertical strokes, and low-pitched scanned synthesis timbres [11] in response to the sweeping gestures. The audio synthesizer is polyphonic (i.e., capable of generating multiple notes simultaneously), and multi-timbral (capable of synthesizing multiple sounds at once).

The visual output engine is also capable of rendering multiple objects and visual timbres simultaneously (see fig. 5). The strokes are visualized as animated circles that are enlarged and faded away in synchrony with the percussive sound. The scanned synthesis instrument is visualized as a slowly rotating 3D pearl of interconnected plates. The plates are animated also in the vertical dimension according to the force imposed by the interaction gesture.

A. Mapping

Table 1 summarizes the mapping procedure of the Audiovisual Air Instruments scenario. Raw accelerometer (/acc), position (/pos) and key press (/key) input streams are routed to a processing component that translates the input events into the language of the rendering engines. The processing involves gesture extraction from the /acc and /key streams, sensor fusion of the /pos coordinates and /acc energy, and finally, sending the detected and parametrized gesture events (/stroke x y E or /sweep x y E) to the rendering engine for final mapping into the native rendering speak (/note pitch velocity).

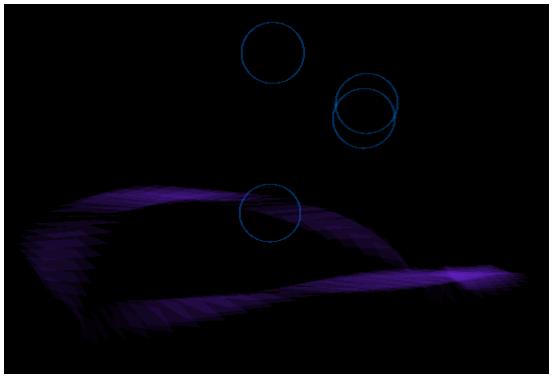

Figure 5. Graphical output of the Audiovisual Air Instruments scenario.

In addition, the state of the physical model of the scanned audio synthesis engine is transmitted periodically to the graphics engine as an array of 128 samples (/wavetable T).

Table 1. Multistage mapping in the audiovisual air instruments scenario.

raw input	gesture	fusion	rendering
/acc ax ay az	/stroke	energy E	/note pitch vel
/key code s	/sweep	-	
/pos x y	-	x y	
			/wavetable T

B. Processing

A Pd patch receives raw /acc, /key and /pos streams as sequential OSC packets arriving at a single UDP port. The patch first separates the streams of the two mobiles (/blue and /green) into identical parallel chains of execution. Processing continues thereafter according to the next token of the OSC address string.

For /acc packets, the patch first calculates the energy of the movement ($E = \mathrm{sqrt}(ax^2 + ay^2 + (az-1)^2)$), and generates a new stroke event when the calculated value exceeds a threshold of 1.57. The algorithm enters then into a wait state, and does not generate new events until a 100 ms time delay has expired. The triggered stroke event is extended with position information by sampling the current value of the /pos stream. The x and y positions of the phone and the calculated energy value E are transmitted to the rendering engines as '/stroke x y E' messages.

The accelerometer data is also used in forming horizontal sweeping gestures. This gesture is triggered by pressing and holding the middle navigator button of the mobile phone. While pressed, the position stream is sampled constantly at 10 Hz rate. The accelerometer energy is calculated at the same rate, resulting in a new '/sweep x y E' message that is transmitted to the scanned synthesis engine at each sampling instant.

C. Rendering the audio and graphics

The '/stroke x y E' messages are converted into '/note pitch velocity' messages (with a fixed one second duration) and routed internally inside the Pd to a *dssi~* external [12] hosting

a *hexter* plugin [13]. Hexter is a DSSI plugin emulating Yamaha's DX7 FM synthesizers, and it is capable of rendering original DX7 patch files and operating in multi-timbral and polyphonic playing modes. Experimental versions of the dssi~ and hexter modules were ported to the Windows platform as part of this work. In the final mapping stage, the x, y and E arguments of the message are transformed either into the pitch or velocity parameter of the '/note pitch velocity' message. For example, after informal user evaluation we noticed that a marimba timbre worked best when x was mapped to the pitch and E to the velocity. We experimented also by quantizing the x-range into four zones, each triggering a different sound from the synthesizer (y was then mapped to the pitch and E once again to the velocity).

The stroke messages are also transmitted to a vvvv patch, which extracts the x and y arguments of the message, and renders a circular Rope primitive at the central position denoted by the extracted arguments. The scaling and the alpha channel values of the rendered item are animated using an ADSR node.

The '/sweep x y E' messages are routed inside Pd to a custom scansynth~ external developed in this work. The external implements a scanned synthesis oscillator with an internal physical model consisting of 64 mass-spring-damper elements chained into a form of a circular string. The synthesized sound is produced by scanning a slowly changing wavetable at audio rates. The wavetable update rate is parametrized, and is typically around 20 Hz. During the update, the vertical positions of the masses are sampled into the amplitude values of the wavetable, which is then played back at a low pitch in order to produce a wide background sound layer for the more percussive FM timbres generated by stroking. The x argument of the message selects a single mass from the circular string, the y argument is bound to the vertical displacement parameter of the mass, and the E argument to the initial velocity of the mass. Thus, by sweeping, the user is able to induce disturbances to the physical model, thereby exciting it to produce sound.

The sweep message is not transmitted to the graphics engine. Instead, each update of the wavetable transmits the entire newly scanned wavetable to a vvvv patch, which updates its internal graphical model accordingly. The model consists of simple semitransparent Quads arranged into a circular 3D spread of length 64. Each wavetable sample value is mapped to the Y-translation parameter of the Quad. In order to make the model more organic, displaced Quads are also rotated around y-axis using a Damper node. The entire structure is also rotated slowly around y-axis by using a LFO node.

The FM and scanned synthesis modules are able to generate sound simultaneously. The 2D circles and the 3D string model are also coexistent in the graphics engine, and layered on top of each other.

VII. Scenario 2: Mobile Conductor

The Mobile Conductor scenario enables active experience of music: the user can express herself in conducting virtual musicians playing a prerecord MIDI piece of music. The mobile phone is here used to detect the movement of the user hand and to mold the execution style by modulating the music speed, volume and intonation.

Figure 6: an illustration of the Mobile Conductor paradigm.

The user holds the phone in her hand. By performing quick or slow gestures she determines the music playback rate (e.g., quick gesture stands for quick playback and vice versa). Linear and straight hand trajectories raise the performance volume, whereas spread and curved trajectories lower it. The user can further process and manipulate the audio content: for example, if the user shakes the mobile phone by performing sudden stroke gestures, then the execution intonation of the active musical instrument is modified by an impulsive perturbation (e.g. a temporary pitch detune). Thus, the user can conduct and manipulate with assigned degrees of freedom the virtual musicians by experimenting all the three above features at the same time.

The music rendering is either based on loudspeakers or directly on the mobile phone using its headphones. In the latter case, the music is sent to the network and is received on the mobile phone where the user can listen to it by headphones.

A. Mapping

The Mobile Conductor scenario uses both the three-axis acceleration data, sent via OSC messages, and the image captured by the mobile camera sent through UDP protocol. The mapping is organized in two steps: (i) from low to high level motion expressive features; (ii) from high level features to MIDI messages.

First steps consists in the following:

- *from absolute acceleration to a measure of "impulsivity"*: we compute the squared sum of the three acceleration components along the X, Y and Z axis; in this way we can filter out the gravity component by simply subtracting 1 to the resulting acceleration. Impulsivity of movement is defined as a ratio between the movement acceleration amplitude over its duration, see Section IV for more details.
- *from movement coordinates to the measure of "directness"*: 2D movement coordinates are computed from the optical flow on the image stream sent by the mobile camera. Starting from 0.5 to 4 seconds buffers

of coordinates in the X,Y plane we build movement trajectory, using temporal segmentation algorithms, then we evaluate how such trajectory is spread with respect to a straight line connecting the first to the last position in the trajectory.

The second step provides the MIDI messages to be sent to the audio output module:

- *"impulsivity" mapping to MIDI pitch change*: when impulsive movements are detected, the system produces a sequence of pitch change MIDI messages, in the range of +/- 25 around the regular pitch of the music piece. The result is that if the user shakes strongly the mobile phone the output audio becomes out of tune for a while, consistently with the sudden, impulsive "shake" of the device.
- *"directness" mapping to MIDI volume change:* as the trajectory performed by the user with her mobile phone approximates a straight line, that is, the "directness" feature is high, we send MIDI messages that increase the volume of the music piece and vice-versa.
- *"speed of movement" mapping to MIDI speed change*: the speed of movement comes from the derivation of movement coordinates; this is mapped on MIDI messages that modify the music piece playback rate. The higher speed of movement, the higher playback rate, and vice-versa.

B. Outputs to the user: Midi audio

The Mobile Conductor scenario produces a MIDI output of a music piece. MIDI allows us to easily modulate the speed, volume and intonation of the execution in real-time.

For this purpose, we implemented two EyesWeb blocks: Midi File Reader and Midi Message Generator (see fig. 7).

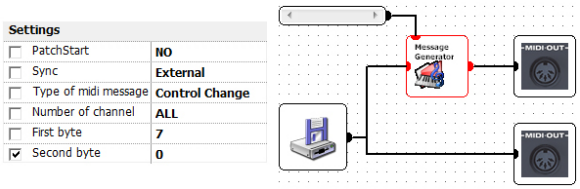

Figure 7: EyesWeb patch to control playback volume . The settings table shows that a "Control Change" message has to be generated for "ALL" channels. The message's second byte carries the volume value and it can be changed in run-time with the slider at the top left of the patch.

The *Midi File Reader* has the following output modes:

- Multi-channel output: all the messages of a channel are sent trough out that channel's dedicated output line.
- Mixed-channel output: all the midi messages read from the input file are sent trough out the output line without taking care of the message's channel.
- Buffered output: the output is buffered and sent out after the buffer becomes full. The size of the buffer can be set manually, or calculated automatically from the external audio clock signal.

Moreover, there are three synchronization modes:

- Internal clock: the execution period is calculated from the values read from the input midi file such as the "Time division" value located in the file header chunk and the various Tempo values carried by the "Set Tempo" meta-event messages.
- External audio clock: a new input pin is created which can be connected to an external audio clock signal (such as an audio buffer signal). The timing properties of the midi track (such as Tempo and Tick length) are read from the midi input file like in the previous synchronization mode, but this time the output mode is automatically switched to "Buffered output" and the length of the midi buffer is calculated to match the values acquired from the external audio signal. In this way, each time an audio buffer is generated from i.e. an mp3 reader block, a midi buffer with the same length is generated too. This is very useful to synchronize a midi file with an external media source.
- User specified: the "Time division" and "Set tempo" meta events are ignored, and the execution period is set manually.

The *Midi Message Generator* allows to manually generate some midi messages. The user must specify the type of the message, he can choose between: Note Off, Note On, Polyphonic key pressure, Control Change, Program Change, Channel pressure, Pitch Bend and Real Time Message. Then he can specify the values of the two byte message's payload. The block checks the midi channel number and, if the value is between 1 and 16 it creates the message for the specified channel and sends it out like in the original behavior; if the midi channel number instead is set to "ALL" the block creates and sends trough the output pin sixteen midi messages, one for each midi channel. This block can be used i.e. to modify the volume or the "pitch bend" of every midi channel at the same time.

VIII. Discussion

In this work we developed a system for sensor pairing and fusion in the specific case of interactive mobile applications. We presented two implementations on active music listening, based on movement interaction from one or more users, integrating onboard accelerometers and external video cameras. We investigated low level motion descriptors, and utilized these in the formulation of higher level, expressive motion features and gestures. We then designed the mapping of movement on music, sound synthesis and animation.

The low level input streams are captured from mobile phone embedded accelerometers and cameras, and from a fixed position camera tracking the color of the mobile phone screens. The fixed camera with blob tracking provides 2D position information, the mobile camera with optical flow techniques is used to compute the velocity, while mobile accelerometer provides the acceleration data.

Targeting multi-user environments, we were interested in finding ways to do sensor pairing, i.e., to identify which two separate anonymously tracked streams originate from a particular device. We found that by calculating a similarity index between two acceleration streams (as obtained from the mobile

accelerometers and from a double differentiated blob tracking stream), it is possible to pair the streams of the two tracking mechanisms. On the other hand, targeting interactive applications, we were also interested in doing sensor fusion, i.e., synchronizing and combining the paired streams into a more descriptive interaction stream. Here we found that an event based push protocol, such as OSC, allows for easy implementation. Unfortunately, we cannot conclude anything about the limits of the acceptable synchronization latency, because it depends on the output parameter mapping.

We were also interested in finding whether it is possible to extract high level motion features from the mobile phone accelerometer streams, and whether it is possible to do optical flow natively inside the mobile phone. We were able to derive impulsivity of motion successfully, although the slow sampling rate of the accelerometers (37 Hz) - coupled with the buffering delay of the impact gesture recognition - made the system feel unresponsive in the test applications, in particular when triggering new notes. The optical flow from the mobile phone was even slower (15 fps) than the accelerometer stream. The same 15 fps applied also to the raw video stream captured from the mobile camera.

The implemented application scenarios revealed that the mapping process is sequentially distributed between the system components. We found that it is most convenient to normalize the parameter ranges as early in the sequence as possible (for example, transforming the accelerometer readouts into |2g| range inside the mobile phone itself, instead of transmitting sensor-specific values). This could be addressed for example by utilizing the MobIO abstraction mechanisms. Considering the output stage, the scanned synthesis algorithm, possibly with remote wavetable implementation, seems to be an attractive mobile audio synthesis method.

In conclusion, it is possible to pair the fixed camera-based blob tracking streams with the accelerometer-based mobile tracking, but that is not a trivial task. However, although the slow sampling rates of the mobile phone sensors pose some limits to the applicability, combining the exact position with the exact acceleration data proved to be clearly beneficial, and therefore we plan to further refine the similarity index algorithm.

Contribution and Acknowledgment

The contributions of the authors are as follows. Jari Kleimola (project coordination, MobIO framework, audiovisual air instruments), Maurizio Mancini (project co-coordination, mobile conductor), Giovanna Varni (Sensor pairing in EyesWeb, mobile conductor), Antonio Camurri (project revision), Carlo Andreotti (EyesWeb MIDI blocks implementation), and Longfei Zhao (offline Matlab analysis of sensor pairing).

The authors would like to thank Barbara Mazzarino for mobile conductor application feature extraction, Corrado Canepa for EyesWeb application design, and Gualtiero Volpe and Paolo Coletta for project revision.

References

[1] SAME project (Sound and Music For Everyone Everyday Everywhere Everyway), [Online] homepage at http://www.sameproject.eu/

[2] A. Camurri, P. Coletta, G. Varni, and S. Ghisio. Developing multimodal interactive systems with EyesWeb XMI. In Proceedings of the 7th International Conference on New Interfaces for Musical Expression (NIME 2007), pages 305-308. ACM New York, NY, USA, 2007

[3] A. Camurri, C. Canepa, P. Coletta, B. Mazzarino, G. Volpe, "Mappe per Affetti Erranti: a Multimodal System for Social Active Listening and Expressive Performance", In Proceedings of the 8th International Conference on New Interfaces for Musical Expression (NIME 2008), Genova, Italy, June 5-7, 2008.

[4] A. Camurri, B. Mazzarino, and G. Volpe, "Analysis of expressive gesture: The eyes web expressive gesture processing library," *LECTURE NOTES IN COMPUTER SCIENC*E, 2004. [Online]. Available: http://www.springerlink.com/index/RFT1L2J1UM7W2H86.pdf

[5] M. Puckette, *The Theory and Technique of Electronic Music*, World Scientific Press, 2007.

[6] *vvvv: a multipurpose toolkit*, [Online] http://vvvv.org {18.11.2009}.

[7] MIDI Manufacturers Association, *Complete MIDI 1.0 Detailed Specification*, November, 2001.

[8] *Open Sound Control*, [Online] http://www.opensoundcontrol.org {18.11.2009}.

[9] P. Sangi, J. Hannuksela, J. Heikkilä, "Global motion estimation using block matching with uncertainty analysis", in *Proc. 15th European Signal Processing Conference* (EUSIPCO 2007), Poznan, Poland, 2007, pp. 1823-1827. [Online] Implemetation available at http://research.nokia.com/research/projects/nokiacv/NCVOverview.html {18.11.2009}.

[10] J. Chowning, "The Synthesis of Complex Audio Spectra by Means of Frequency Modulation," *Journal of the Audio Engineering Society* 21(7), 1973.

[11] B. Verplank, M. Mathews and R. Shaw, "Scanned Synthesis", in *Proc. Int. Comp. Music Conf.* (ICMC'00), Berlin, Germany, 2000.

[12] J. Bullock, *DSSI and LADSPA host for PD*. [Online] Available at http://puredata.info/Members/jb/dssi~/view {18.11.2009}.

[13] S. Bolton, *hexter: Yamaha DX7 modeling DSSI plugin*, [Online] Available at http://dssi.sourceforge.net/hexter.html {18.11.2009}.

[14] N. Marwan, M.C. Romano, M. Thiel, and J.Kurths, "Recurrence plots for the analysis of complex systems", *Phys. Rep*. 438: 237-329, 2007.