J Multimodal User Interfaces (2010) 4: 27–35
DOI 10.1007/s12193-010-0047-z

ORIGINAL PAPER

Human movement expressivity for mobile active music listening

Maurizio Mancini · Giovanna Varni · Jari Kleimola · Gualtiero Volpe · Antonio Camurri

Received: 8 March 2010 / Accepted: 28 July 2010 / Published online: 18 August 2010

Abstract In this paper we describe the SAME networked platform for context-aware, experience-centric mobile music applications, and we present an implementation of the SAME active music listening paradigm: the *Mobile Conductor*. It allows the user to express herself in conducting a virtual ensemble playing a MIDI piece of music by means of her mobile phone. The mobile phone detects the user's hand movement and molds the music performance style by modulating its speed, volume, and intonation.

Keywords Human-computer interaction · Active music listening · Movement expressivity

1 Introduction

In the recent past, mobile phones were used primarily for voice and SMS communication, so the 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.

The FP7 EU ICT SAME Project studies active music listening by developing mobile context-aware music applications for active, embodied experience of music in cooperative social environments using mobile phones.

Mobile phones are very widespread among the all age groups. However, this large availability does not correspond to a large use of mobile as interactive music performance devices. This work aims at presenting the SAME software platform for the creation of usable, effective mobile applications for active music listening, resting on the use of embedded mobile sensors and mobile network connectivity. For this aim we also introduce and describe in detail an application for active music listening called *Mobile Conductor*: the user modulates the rendering of music by moving her mobile phone in the space, that is, the mobile phone becomes a new type of active interface to interact with the system. The user movement expressivity, that is, the way in which the user moves (e.g., her arm speed and acceleration), drives the rendering of a music piece: for example abrupt movements produce a slight detune, slow movements slow down the playback and so on.

The reminder of this paper is organized as follows. In Sect. 2 we present an overview of systems for active music listening and research on human movement expressivity. Section 3 provides an overview of the SAME platform, while Sect. 4 explains the *Mobile Conductor* in detail. Fi-

M. Mancini (✉) · G. Varni · G. Volpe · A. Camurri
InfoMus Lab—Laboratorio di Informatica Musicale,
DIST—University of Genova, Viale Causa, 13, 16145 Genova, Italy
e-mail: maurizio.mancini@dist.unige.it

G. Varni
e-mail: giovannavarni@infomus.dist.unige.it

G. Volpe
e-mail: gualtiero.volpe@unige.it

A. Camurri
e-mail: antonio.camurri@unige.it

J. Kleimola
Aalto University School of Science and Technology, Department of Signal Processing and Acoustics, P.O. Box 13000, 00076 Aalto, Finland
e-mail: jari.kleimola@tkk.fi

nally, Sect. 5 discusses the outcomes and perspectives of this project.

2 Related work

The work presented in this paper is grounded on the active listening paradigm developed in the SAME Project. Research on active music listening is in its infancy, however initial results begin to be available.

The *Orchestra Explorer* by Camurri et al. [5] allows 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 a 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 Explorer* paradigm has been implemented in two different scenarios. In the first one users are tracked by fixed video cameras observing the sensitive space where user(s) 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 allows one to navigate a colored cursor on a map representing the orchestra in the phone display. Camurri et al. also propose a more sophisticated active listening paradigm, 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 nonverbal full-body movement and expressive gesture. An implementation of this system, named *Mappe per Affetti Erranti* [4], was presented during the science exhibition "Metamorfosi del Senso", held at Casa Paganini, Genova, in October–November 2007.

Further, interactive dance-music systems were proposed by Camurri [3] and Leman [16]. In such systems, the user(s) full-body rhythmic movements were analyzed in real-time and compared with the beat of a song. The results consist of the control of the quality of the orchestration, of the melodic pitch (de)tuning, and of the synchronization of the different voices of the song [3], and a visual feedback that corresponds to the synchronization of each group of users dancing in time with music [16]. Goto proposed a GUI-based system for intervening on prerecorded music with some original real-time signal processing techniques to select, skip and navigate sections of the recording [12].

Some projects addressed mobile music performances: *SonicCity* uses multimodal sensors allowing a single user at creating music and manipulate sounds by using the physical urban environment as interface; information about the environment and the user's actions are captured and mapped onto real-time processing of urban sounds [11]. *SoundPryer* is a peer-to-peer application of mobile wireless ad hoc networking for PDAs, enabling users to share and listen to the music of people in vehicles in the immediate surrounding [18]. The *SonicPulse* system is an application designed to discover in a physical environment other mobile music users and engage with them sharing and co-listening to music [1].

More recently, attempts have been done to use directly mobile phones as music instruments; for example the *CaMus* system [23], and its multi-user extension *CaMus2* [22], allow user(s) to experiment interactive music performance using the camera of mobile phones. Each camera tracks the position, rotation, and height of one or more marker sheets and sends these data via Bluetooth to a computer. The information is mapped to MIDI messages to connect to arbitrary MIDI-enabled sound software or hardware.

Several researchers focused their attention on the ability for machines to recognize human movement features which contribute to the understanding of human behavior and in particular of the communication of nonverbal information. Human behavior encodes not only explicit content, that is, "What is communicated", or the signs communicated by means of a gesture shape, but also expressive information, that is, "How it is communicated", the implicit information referring to the qualities of the performed gesture.

Researchers like Johansson [14], Ball and Breese [2] and Pollick [20] have investigated human movement features and encoded them into categories. Psychologists, musicologists, researchers on music perception and human movement, like Wallbott and Scherer [26], Gallaher [10], identified important features for characterizing qualities of the expressive movement. These include speed, amplitude, energy and so on. Wallbott demonstrated in [25] that body activity, expansiveness, and power are discriminating factors in communicating a large number of emotional states.

We also take into account theories from humanistic and artistic fields. In particular, in his "Theory of Effort" [15], the choreographer R. Laban identified movement qualities that are conveyors of high level expressive information. Further, embodied cognition music theories provide models based on audio features responsible to explain the communication of emotions through music, including features related to tempo, volume, pitch, articulation, and dynamics.

3 The SAME framework for active music listening

We introduce now the SAME networked platform, that is, an End-to-End framework (i.e., connecting several clients of a mobile service, producers and consumers of content) for context-aware, experience-centric mobile music applications, enabling embodiment and control of music content by

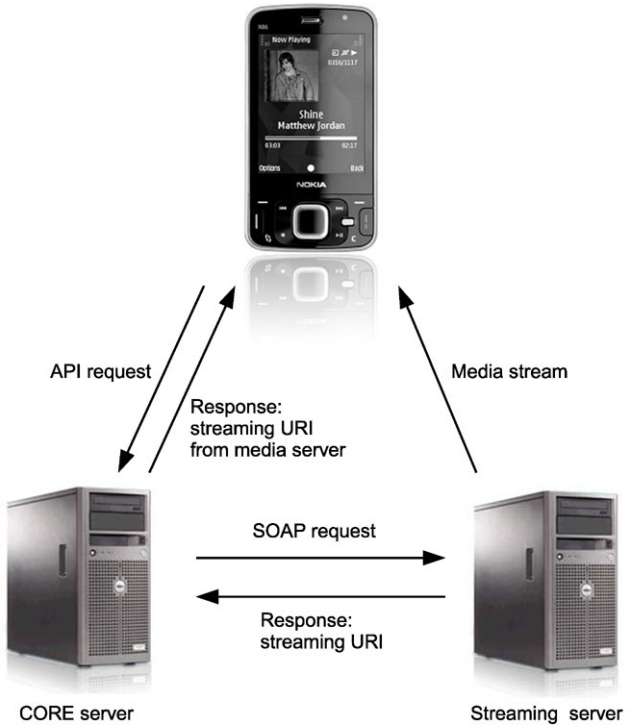

Fig. 1 The SAME platform in a streaming configuration. The mobile phone can send and receive data with two servers: the CORE server handles the phone requests, while the Streaming server deals with the content streaming

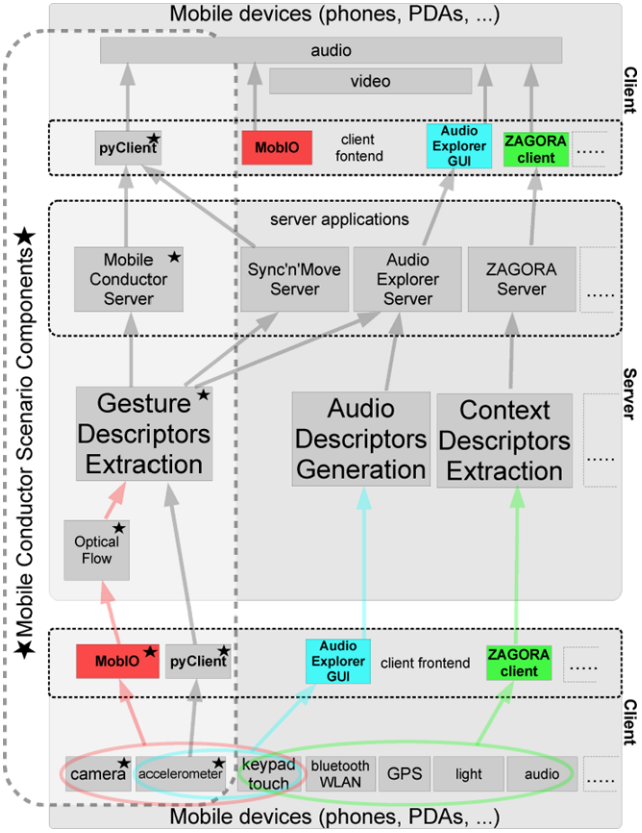

Fig. 2 The SAME platform architecture: mobile devices (*top* and *bottom part* of the figure) send and receive data to and from the SAME platform server (*middle part* of the figure)

user behavior. The platform has a modular variable structure and may include one or more remote or local servers, running software environments such as *EyesWeb XMI* (for eXtended Multimodal Interaction) [9], *Pure Data* [21] and *vvvv* [24].

The SAME servers 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. The servers can include services connected to the physical environment, e.g., the real-time processing of the signal from fixed position environmental cameras. The services include the mobile processing and management of the applications based on the interactions captured from the mobile phones.

3.1 Architecture overview

The SAME platform architecture is depicted in Fig. 2. Users interact with the platform by moving, shaking tapping on their mobile devices, e.g., mobile phones such as Nokia N85/N95 and PDAs. The mobile devices have keypads and embed sensors: accelerometers, cameras, GPS receivers, and so on. They also run applications (called *client frontend* in the lower part of Fig. 2), developed in the SAME framework, able to acquire and process data from the sensors and communicate with the SAME servers via a wireless or bluetooth network.

Once data is received by the SAME servers it is processed both to extract further information (see for example the *Optical Flow* module in Fig. 2) and to compute high level descriptors of *gesture*, *audio*, and *context* (see the *Gesture Descriptors Extraction*, *Audio Descriptors Generation*, and *Context Descriptors Extraction* modules in Fig. 2).

These descriptors are provided as input to several *server applications* that implement active music listening scenarios, enabling users, for example, to modify in real time the audio they are listening to by changing the way they interact with gesture. For a detailed description of these interaction scenarios see [7].

The multimodal audio/video output of the SAME servers is finally sent to the user mobile devices where it is rendered (see the *client frontend* in the upper part of Fig. 2).

In this paper we focus on one of the interaction scenarios implemented in the SAME platform: the *Mobile Conductor*. Its components are highlighted in the left side of Fig. 2. In the following, we describe the SAME platform data and modules: *the Sections and descriptions marked by the symbol ⋆ refer in particular to the Mobile Conductor interaction scenario*.

3.2 Platform data

The mobile devices we experiment within SAME are Symbian-based mobile phones, such as Nokia S60 devices, embedding sensors that provide different streams of data.

3.2.1 Camera ⋆

Nokia mobile phones running S60 applications are equipped with high and low definition cameras. In the SAME platform we implemented a protocol to send a stream of video frames through UDP. This technique allows one to send and receive low resolution black and white videos at 15 fps over a local wireless network.

3.2.2 Accelerometer ⋆

Such sensor is capable of measuring 3D acceleration in the range $[-2g, 2g]$ at 30 Hz. In the SAME platform, accelerometer data is read using Python scripts and C++ applications. Pre-processing, e.g., filtering out **g** and/or noise, is possible within the mobile itself or after receiving data in the application running on the End-to-End platform. We defined the following OSC packet format to exchange accelerometers data between mobiles and applications:

```
/data/acc/raw %x %y %z
/data/acc/norm %x %y %z
/data/acc/no-g %x %y %z
```

3.2.3 Keypad/touch

We aim to detect events corresponding to: (i) the user clicking on a particular position on the screen or (ii) the user clicking on a physical or graphical button. So, in the first case we transmit the raw touch screen clicking position, that is, x and y raw coordinates that are interpreted in some way by the receiving application. In the second case, we detect a particular event when touching the screen, for example the activation of a button, and we send this event, that is, the button information. In the same way, if we press a physical keyboard button of the mobile phone, we propose to send this information as an event.

3.2.4 Bluetooth/WLAN

Many PDAs and phones have one or both of these connectivity capabilities. That is, they are capable of detecting other mobile or fixed devices operating in the same area, e.g., room, building and so on. This feature allows us to send control messages to the mobile device to scan the local area for other devices with wireless capabilities to, for example, infer contextual information. Then the mobile device can send data messages to list the found devices.

3.2.5 GPS

SAME mobile phones are equipped with an internal GPS receiver. From the GPS position we can compute the mobile altitude, speed of movement, direction of movement and accuracy of all of these features.

3.2.6 Light

Some Nokia phones are equipped with a sensor for measuring ambient light and adapting the display backlight intensity. The information extracted by this sensor can be useful to determine whether the user moves for example from indoor to outdoor.

3.2.7 Audio

We are able to establish audio streams between the mobile devices and the SAME platform. Audio can be read from microphone, wave and MP3 files stored in the mobile. The format of such a streaming process is SHOUTcast, a free standard defined by NullSoft and implemented as a variant of the HTTP protocol.

3.3 Platform modules

3.3.1 MobIO ⋆

MobIO is a Symbian-based dynamic link library that provides a unified API to mobile devices such as camera, keypad, and screen. The API defines consistent parametrization and real-time dataflow interfaces among different classes of input and output objects, thereby simplifying multimodal application development.

This is semantically equivalent to the graphical programming paradigm employed for example in the *EyesWeb XMI*, *Pd* and *vvvv* environments, but because MobIO has programming language bindings, the syntax level is different.

The Mobile Conductor utilises MobIO to capture the raw video frames from the onboard mobile camera, to convert the frames into 64×64 pixel grayscale bitmaps, and finally, to stream the images in real-time to the *EyesWeb XMI* server application.

3.3.2 pyClient ⋆

The *pyClient* is a Python module running on the mobile phone that retrieves the acceleration measured by the embedded sensor. It can also perform several operations on it, for example: computation and sending of the user position in a 2D space, obtained by the integration of the x and y components of acceleration; audio streaming to and from other devices of the SAME platform.

In the *Mobile Conductor* scenario the pyClient is devoted to reading and sending the (x, y, z) acceleration to the Gesture Descriptors Extraction module, described in Sect. 3.3.4.

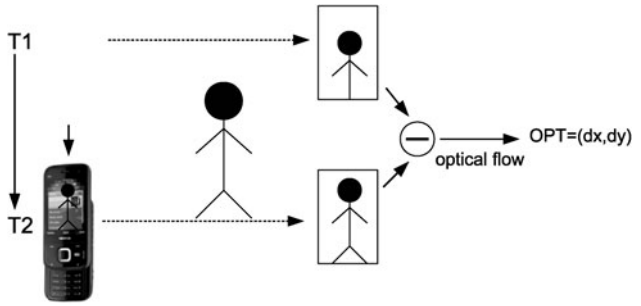

Fig. 3 Optical flow computation between two consecutive camera snapshots: we estimate the mobile phone movement direction and magnitude

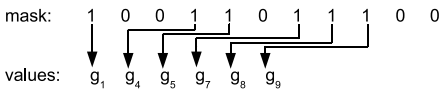

Fig. 4 The Gesture Description Frame: the mask identifies the gesture descriptors transmitted by the frame

3.3.3 Optical flow ⋆

By computing optical flow between two consecutive frames, one can estimate the mobile direction and quantity of movement along two axis, as illustrated by Fig. 3.

By comparing the pixels of two frames taken at times T1 and T2, we obtain a movement matrix in which the speed of movement of every pixel on the x and y axes is reported. If we compute the mean value of these speeds we obtain an estimation of the movement of the phone along x and y. The resulting values are transmitted over the network with an OSC message:

```
/data/opt %dx %dy
```

3.3.4 Gesture descriptors extraction ⋆

While users move their mobile phones, 11 descriptors are extracted depending on the user's movement and gestures: Motion, Impulsivity, Directness, Internal Motion, Contraction, Translational Motion, Symmetry, Fluidity, Periodicity, Space Occupation Area, Kinematical measures [6].

To transmit the above descriptors we implemented the Gesture Descriptor Frame (GDF), consisting of two elements: a masking vector and a values vector. The masking vector is a sequence of 11 elements corresponding to the current number of gesture descriptors included in the SAME platform. These elements can be either 0 or 1. The values vector is a sequence of integers in the range [0,65535] which represent the gesture descriptors values.

For each element in the mask, if the element is 0 then the corresponding element in the value vector is not present; if it is 1 then the corresponding element in the value vector is present. Figure 4 reports an example of GDF.

In the example of Fig. 4, 6 descriptors out of the 11 are enabled, thus only their values are sent, optimizing bandwidth. By defining a single GDF, we can describe the dynamic evolution of a gesture by sending a stream of GDFs, each frame identified by a time label. At the implementation level GDFs could be encoded in OSC messages to be transmitted between the End-to-End platform components.

The GDF format is similar to the GDIF format introduced by Jensenius et al. in [5]. The GDIF aims at establishing a standard description format for several aspects of a gesture: its low level physical data, descriptive information and high level features like velocity or intensity. Differences with GDIF are at transmission level: GDIF format consists of several synchronized streams that are transmitted in parallel; instead, we propose a variable length frame format that do not impose to establish a TCP network connection to stream data: in our format unavailable information is simply omitted and available information is packed and transmitted via UDP as a single frame in asynchronous way.

3.3.5 Audio descriptors generation

Here we briefly introduce the set of SAME descriptors for accessing the synthesis parameters of a sound synthesizer and to wrap the MIDI protocol as an OSC content format. These descriptors range from low-level descriptors such as frequency and amplitude to perceptual descriptors such as, for example, pitch, loudness, timbre and duration. These descriptors can be grouped together into meta descriptors allowing the user to set one-shot the sound attributes when she is playing on a synthesizer.

3.3.6 Context descriptors extraction

The SAME platform supports the extraction of descriptors for context-aware music applications and services. For example data from the accelerometer can be analyzed to infer the user current activity: walking, sitting, running, and so on. Or the audio captured by the internal microphone can be useful to retrieve information on the user location: street, car, disco, and so on.

The *Zagora server* application [7] exploits such features to detect the user environmental situation using audio analysis. Based on such a contextual information, Zagora generates a music playlist pertinent to the current location or activity. The user data is processed and transmitted by the Zagora client that also allows the user to retrieve the resulting playlist in a single click.

3.3.7 Server applications

Server applications are implementations of the SAME active music listening paradigm. The *Mobile Conductor* enables users to mold the execution of a virtual ensemble and

is described in detail in Sect. 4. The *Audio Explorer* allows users to interactively de-mix commercial stereo recordings into different channels while being streamed to their mobile devices, also offering interactive re-mixing possibilities based on previously separated channels. *Sync'n'Move* enables users to experience a novel form of social interaction based on music and gesture, using mobile phones and the SAME platform. Users move rhythmically (e.g., dance) wearing their mobiles. Context-awareness is particularly addressed in *Zagora*, a context-aware mobile music player, which detects the ambient situation using audio analysis and retrieves a playlist of suitable music.

3.4 Platform evaluation

A survey of music applications like those described in the previous Sections (Sync'n'Move, Audio Explorer, Zagora) have been presented and evaluated at the Agora Festival (IRCAM, Paris, France, June 2009) [7]. Starting from usability questionnaires filled up by about 30 participants we inferred that users generally liked the SAME platform applications. Usability feedback included criticism, suggestions, and proposals for improvements. Evaluation pointed out that the users are anxious to see social networking features implemented as a part of the applications. This is encouraging for future research on embodied social interaction envisaged in SAME. Finally, if from the one hand, the applications received an overall positive feedback on the other hand users appreciated active listening both as a new way for listening to music and also as an educational tool for gaining a better understanding of how music is made, structured, and performed.

4 The *Mobile Conductor*

The *Mobile Conductor* enables the user can express herself in conducting a virtual ensemble playing a MIDI piece of music. The mobile phone here allows us to detect the movement of the user hand and to mold the execution style by modulating the music speed, volume, and intonation. Figure 5 sketches the paradigm the *Mobile Conductor* is based on.

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 intonation of the musical instruments 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 music ensemble by experimenting all the three features above at the same time.

Fig. 5 An illustration of the *Mobile Conductor* paradigm

4.1 Technical description

Figure 6 depicts the implementation of the *Mobile Conductor*. The structure is an instantiation of the SAME platform shown by Fig. 2 in which we highlighted the corresponding modules. In the following a description of each component is provided starting from input (bottom part of Fig. 2) to output (top part of Fig. 2).

4.1.1 MobIO

In the *Mobile Conductor*, MobIO APIs are used in the video conversion module, which is responsible of capturing and converting the mobile camera frames into the EyesWeb image format, and in the UDP encapsulation module, which encodes and transmits the converted camera frames in the EyesWeb UDP format, one UDP packet per each captured video frame. The *video conversion* module receives raw input frames from the MobIO image acquisition class, in 2592×1944 pixel RGB format. It first crops the image to the size of 64×64 pixels and then changes the color model of the image to the 8-bit grayscale format. The *UDP encapsulation* module packs the image pixels into a single UDP packet, and sends it through the network to the SAME platform, running the EyesWeb XMI application. The UDP datagram has the following format:

```
struct DATAGRAM_DATATYPE_PACKET
{
__int64 packet_ndx;
size_t start_pos;
size_t size;
BYTE data[MAX_DATA_SIZE];
size_t uncompressed_data_size;
bool end_of_data;
};
```

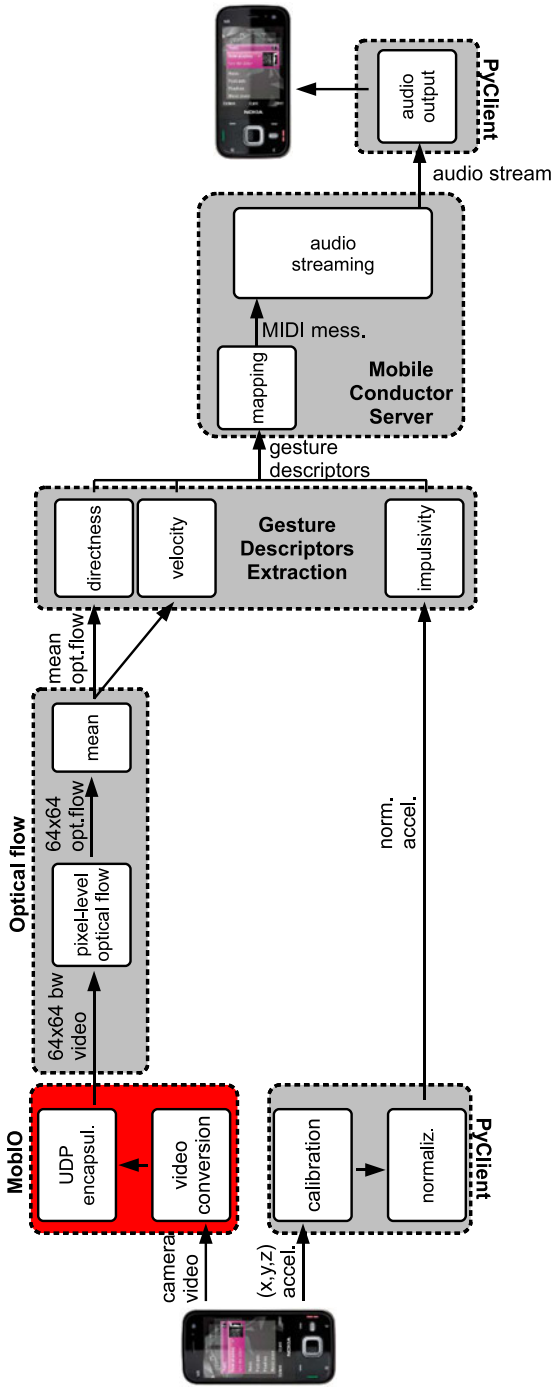

Fig. 6 *The Mobile Conductor* implementation

The maximum size of the entire EyesWeb datagram is 8000 bytes and, for efficiency reasons, we aimed to transmit each mobile camera frame as a single UDP packet. The packet starts with the EyesWeb image frame header:

```
struct DATATYPE_HEADER
{
TIME step_time;
TIME creation_time;
TIME presentation_time;
TIME media_time;
TIME duration;
TIME media_duration;
__int64 buffer_size;
};
```

This header is 32 bytes long and is followed by the 64×64 pixel grayscale image data. The total size of the packet is therefore $32 + 4096 = 4128$ bytes. That is, it is possible to fit the entire image frame into a single EyesWeb UDP datagram.

4.1.2 *pyClient*

The *calibration* and *normalization* modules are necessary since the accelerometer has a different *max* and *min* values of **g**, on every axis. Here we report the operations performed by the calibration block on the x axis. The same computation is necessary also on the other two axis:

$$A_{C_x} = A_{\text{raw}_x} - \text{IRD}_x; \quad \text{IRD}_x = \frac{gx^+ + gx^-}{2}; \tag{1}$$

in which:

- IRD_x is the *Inter Range Difference*, that is, the half of the difference between the maximum **g** measured on x^+ and x^-;
- A_{C_x} is the calibrated acceleration on x axis, that is, the acceleration obtained by subtracting the IRD_x from the raw acceleration on x, that is, A_{raw_x};

Finally the normalization block computes the absolute value of acceleration and nomalizes it in the range [0, 1] subtracting **g** in order to ignore it:

$$A_{\text{normalized}} = \frac{\sqrt{A_{C_x}^2 + A_{C_y}^2 + A_{C_z}^2} - \mathbf{g}}{A_{\text{MAX}}}; \tag{2}$$

where A_{MAX} is the maximal absolute value of acceleration detected by the phone.

4.1.3 *Optical flow*

The UDP encapsulated images containing video data captured by the user mobile phone are sent to the *pixel-level optical flow* module that works as explained by Fig. 7: for every pair of received frames we estimate the x and y displacement of each pixel and we store them in two matrices.

Then we compute the mean value of each matrix in the *mean* module, that is, we sum up all the matrix elements and we divide by the matrix cardinality. We obtain the mean optical flow on x and y for the two input frames.

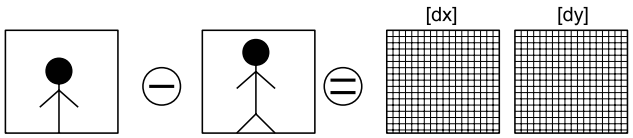

Fig. 7 Pixel-level optical flow: two subsequent frames are compared and for each pixel the horizontal and vertical displacement is estimated. The result values are stored in two matrices, one for *horizontal* (dx) and one for *vertical* (dy)

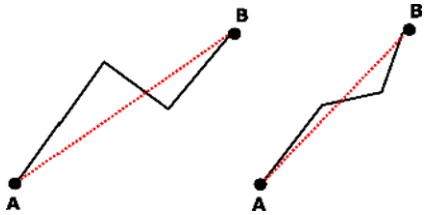

Fig. 8 Directness between A and B is measured as the ratio between the straight path connecting A and B and the actual one: an indirect path on the *left*; a more direct path on the *right*

4.1.4 Gesture descriptors extraction

The algorithms for the automatic extraction of descriptors have been implemented in the *EyesWeb XMI* software platform using the EyesWeb Expressive Gesture Library [6]. In the following we refer only to the gesture descriptors included in the *Mobile Conductor*:

- *Motion*: corresponds to the first order derivative of the optical flow computed on the mobile camera stream.
- *Impulsivity*: it can be measured as a combination of the *Movement Energy* ME and *Movement Duration* MD. ME is used to identify how long the gesture is in time: for example, using an adaptive threshold, when the ME value becomes greater than the threshold we identify the gesture beginning time; when the ME goes below the threshold we identify the ending time. Impulsivity can be formalized as the ratio between ME and MD.
- *Directness*: it 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, see Fig. 8.

4.1.5 Mobile Conductor server

The *Mobile Conductor* 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* module works as follows:

- "*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 audio output 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 the speed of movement, the higher the playback rate, and vice-versa.

The above mapping was chosen according to the state of the art in gestural expressive interfaces like [8, 17, 19]: shaking movements are usually associated to angry emotional states, that is, impulsive gestures [19]; wide movements are associated to louder music and vice-versa [8]; speed of movements and music are ofted directly mapped [17]. In the last years such a kind of mapping is also increasingly used in commercial products like Apple iPod [13].

Finally, the *audio streaming* module synthesizes MIDI output by modulating the speed, volume and intonation of the performance in real-time depending on the results of the above mapping. The MIDI output is finally sent to the *pyClient* module that produces the stereo audio rendering. The resulting audio is transmitted on the network through a Shoutcast server implemented in EyesWeb XMI and is received on the mobile phone.

5 Conclusion

This paper proposes a platform for mobile active music listening, which integrates mobile sensor data, such as acceleration and videocamera signal, to interpret user movements to manage in real-time the molding of pre-recorded music. The paper focuses on a concrete implemented scenario, based on the Nokia S60 family of mobiles, and in particular the *Mobile Conductor*. Its usability will be evaluated in the near future.

In targeting interactive applications future challenges are, e.g., synchronizing and combining the user input data streams into a more descriptive interaction stream. For this aim, we are working in finding whether it is possible to extract gesture descriptors from the mobile phone accelerometer streams, and whether it is possible to do this in realtime to enable active music listening experience.

In the near future, systems like the SAME platform will allow multiple user to interact between them with their mobile devices in a cooperative, collaborative and creative way.

The result of such interaction will be published on shared environments allowing other users to interact with it. The SAME platform is a first step toward this goal and the *Mobile Conductor* scenario demonstrates some of its capabilities.

Acknowledgements The authors would like to thank Carlo Andreotti (EyesWeb MIDI blocks implementation), Barbara Mazzarino (mobile conductor impulsivity feature extraction), and Alberto Massari (pyClient module design and implementation). The presented work is partially supported by the FP7 EU ICT SAME Project n. 215749 on active music listening (www.sameproject.eu).

References

1. Anttila A (2006) Sonicpulse: exploring a shared music space. In: 3rd international workshop on mobile music technology
2. Ball G, Breese J (2000) Emotion and personality in a conversational agent. In: Cassell J, Sullivan J, Prevost S, Churchill E (eds) Embodied conversational characters. MIT Press, Cambridge
3. Camurri A (1995) Interactive dance/music systems. In: Proceedings of international computer music conference
4. Camurri A, Canepa C, Coletta P, Mazzarino B, Volpe G (2007) 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
5. Camurri A, Canepa C, Volpe G (2007) Active listening to a virtual orchestra through an expressive gestural interface: the orchestra explorer. In: Proceedings of the 7th international conference on new interfaces for musical expression
6. Camurri A, Mazzarino B, Volpe G (2004) Analysis of expressive gesture: the eyesweb expressive gesture processing library. Lecture notes in computer science
7. Camurri A, Volpe G, Vinet H, Bresin R, Maestre E, Llop J, Kleimola J, Valimaki S, Seppanen J (2009) User-centric context-aware mobile applications for embodied music listening. In: Proceedings of the 1st international ICST conference on user centric media
8. Castellano G, Bresin R, Camurri A, Volpe G (2007) Expressive control of music and visual media by full-body movement. In: Proceedings of the 7th international conference on New interfaces for musical expression, pp 390–391
9. EyesWeb: http://www.eyesweb.org
10. Gallaher PE (1992) Individual differences in nonverbal behavior: dimensions of style. J Pers Soc Psychol 63(1):133–145
11. Gaye L, Mazé R, Holmquist L (2003) Sonic city: the urban environment as a musical interface. In: Proceedings of the 3rd international conference on new interfaces for musical expression
12. Goto M (2007) Active music listening interfaces based on signal processing. In: Proceedings of the 2007 IEEE international conference on acoustics, speech, and signal processing
13. iPod: http://www.apple.com/ipod
14. Johansson G (1973) Visual perception of biological motion and a model for its analysis. Percept Psychophys 14:201–211
15. Laban R, Lawrence FC (1947) Effort. Macdonald & Evans, USA
16. Leman M, Demey M, Lesaffre M, van Noorden L, Moelants D (2009) Concepts, technology and assessment of the social music game sync-in team. In: Proceedings of the 12th IEEE international conference on computational science and engineering
17. Mancini M, Bresin R, Pelachaud C (2007) A virtual head driven by music expressivity. IEEE Trans Audio Speech Lang Process 15(6):1833–1841
18. Östergren M, Juhlin O (2004) Sound pryer: truly mobile joint listening. In: 1st international workshop on mobile music technology
19. Paiva A, Andersson G, Höök K, Mourao D, Costa M, Martinho C (2002) Sentoy in fantasya: designing an affective sympathetic interface to a computer game. Pers Ubiquitous Comput 6(5-6):378–389
20. Pollick FE (2004) The features people use to recognize human movement style. In: Camurri A, Volpe G (eds) Gesture-based communication in human-computer interaction-gesture workshop 2003. LNAI, vol 2915. Springer, Berlin, pp 10–19
21. PureData: http://puredata.info
22. Rohs M, Essl G (2007) Camus2-collaborative music performance with mobile camera phones. In: Proceedings of the international conference on advances in computer entertainment technology (ACE)
23. Rohs M, Essl G, Roth M (2006) Camus: live music performance using camera phones and visual grid tracking. In: NIME '06: proceedings of the 2006 conference on new interfaces for musical expression. IRCAM, Paris, pp 31–36
24. vvvv: http://vvvv.org
25. Wallbott HG (1998) Bodily expression of emotion. Eur J Soc Psychol 28:879–896
26. Wallbott HG, Scherer KR (1986) Cues and channels in emotion recognition. J Pers Soc Psychol 51(4):690–699