Journal on Multimodal User Interfaces - Author version of the manuscript

Synchronizing Multimodal Recordings Using Audio-To-Audio Alignment

An Application of Acoustic Fingerprinting to Facilitate Music Interaction Research.

Joren Six · Marc Leman

Received: May 2015 / Accepted: August 2015

Abstract Research on the interaction between movement and music often involves analysis of multi-track audio, video streams and sensor data. To facilitate such research a framework is presented here that allows synchronization of multimodal data. A low cost approach is proposed to synchronize streams by embedding ambient audio into each data-stream. This effectively reduces the synchronization problem to audio-to-audio alignment. As a part of the framework a robust, computationally efficient audio-to-audio alignment algorithm is presented for reliable synchronization of embedded audio streams of varying quality. The algorithm uses audio fingerprinting techniques to measure offsets. It also identifies drift and dropped samples, which makes it possible to find a synchronization solution under such circumstances as well. The framework is evaluated with synthetic signals and a case study, showing millisecond accurate synchronization.

Keywords Multimodal Data Synchronization · Audio fingerprinting · Audio-to-Audio-Alignment · Music Performance Research · Digital Signal Processing

1 Introduction

During the past decades there has been a growing interest in the relation between music and movement, an overview of ongoing research is given in [5]. This type of research often entails the analysis of data from various (motion) sensors combined with multi-track audio and video recordings. These multi-modal signals need to be synchronized reliably and precisely to allow successful analysis, especially when aspects on musical timing are under scrutiny.

Synchronization of heterogeneous sources poses a problem due to large variability in sample rate and due to the latencies introduced by each recording modality. For example it could be the case that accelerometer data, sampled at 100Hz, needs to be synchronized with multi-track audio recorded at 48kHz and with two video streams, recorded using webcams at 30 frames per second.

Several methods have been proposed to address synchronization problems when recording multi-modal signals. The most straightforward approach to solve this problem is to route a master clock signal through each device and synchronize using this pulse. In [8] an SMPTE signal serves as a clock for video cameras and other sensor as well. In [7] a clock signal generated with an audio card is used to synchronize OSC, MIDI, serial data and audio. A drawback of this approach is that every recording modality needs to be fitted with a clock signal input. When working with video this means that expensive cameras are needed that are able to control shutter timing with a sync port. Generally available webcams do not have such functionality. The EyesWeb system [2] has similar preconditions.

An other approach is to use instantaneous synchronization markers in data-streams. In audio streams such marker could be a hand clap. A bright flash could be used in a video stream. These markers are subsequently employed to calculate timing offsets and synchronize streams either by hand or assisted by custom software. This method does not scale very well to multiple sensor streams and does not cope well with drift or dropped samples. Some types of sensor streams are hard to ma-

J. Six and M. Leman
Institute for Psychoacoustics and Electronic Music (IPEM)
Department of Musicology,
Ghent University,
Ghent, Belgium
E-mail: joren.six@ugent.be

nipulate with markers e.g. ECG recordings, which prohibits the use of this method. Although the method has drawbacks it can be put to use effectively in controlled environments, as is shown in [6,1].

In this article a novel low-cost approach is proposed to synchronize streams by embedding ambient audio into each stream. With the stream and ambient audio being recorded synchronously, the problem of mutual synchronization between streams is effectively reduced to audio-to-audio alignment. As a second contribution of this paper, a robust, computationally efficient audio-to-audio alignment algorithm is introduced. The algorithm extends audio fingerprinting techniques with a cross-covariance step. It offers precise and reliable synchronization of audio streams of varying quality. The algorithm proposes a synchronization solution even if drift is present or when samples are dropped in one of the streams.

2 Audio-to-Audio alignment

There are several requirements for the audio-to-audio alignment algorithm. First and foremost, it should offer reliable and precise time-offsets to align multiple audio-streams. Offsets should be provided not only at the start of the stream but continuously in order to spot drifting behavior or dropped samples. The algorithm needs to be computationally efficient so it can handle multiple recordings of potentially several hours long. Highly varying signal quality should not pose a problem for the alignment algorithm: the algorithm should be designed to reliably match a stream sampled at a low bit-rate and sample rate with a high-fidelity signal.

The requirements are similar to those of acoustic fingerprinting algorithms. An acoustic fingerprinting algorithm uses condensed representations of audio signals, acoustic fingerprints, to identify short audio fragments in large audio databases. A robust fingerprinting algorithm generates similar fingerprints for perceptually similar audio signals, even if there is a large difference in quality between the signals. Wang describes such algorithm in [15]. Wang's algorithm is able to recognize short audio fragments reliably even if the audio has been subjected to modifications like dynamic range compression, equalization, added background noise and artifacts introduced by audio coders or A/D-D/A conversions. The algorithm is computationally efficient, relatively easy to implement and yields precise time-offsets. All these features combined make it a good candidate for audio-to-audio alignment. This is recognized by others as well since variations on the algorithm have been used to identify multiple video's of an event [4] and to identify repeating acoustic events in long audio recordings [10]. The patent application for the algorithm [16] mentions that it could be used for precise synchronization, unfortunately it is not detailed how this could be done. Below a novel post-processing step to achieve precise synchronization is proposed.

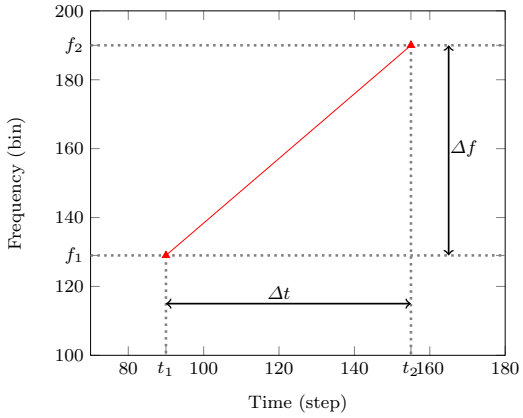

Fig. 1 A fingerprint consists of two peaks in a time-frequency representation .

Another approach to audio-to-audio synchronization is described in [11]. Their algorithm offers an accuracy of +-11ms in the best case and does not cope with drift or dropped samples. Two elements that are improved upon in the algorithm proposed below.

The algorithm works as follows. First audio is transformed to the time-frequency domain. In the time-frequency domain peaks are extracted. Peaks have a frequency component f and a time component t. The frequency component is expressed as an FFT bin index and the time component as an analysis frame index. The peaks are chosen to be spaced evenly over the time-frequency plane. A combination of two nearby peaks are combined to form one fingerprint, as shown in Figure 1. The fingerprints of the reference audio are stored in a hashtable with the key being a hash of f_1, Δt, Δf. Also stored in the hashtable, along with the fingerprint, is t_1, the absolute time of the first peak. Further details can be found in [15,14,16].

For audio that needs to be synchronized with a reference, fingerprints are extracted and hashed in the exact same way. Subsequently, the hashes that match with hashes in the reference hashtable are identified. For each matching hash, a time offset is calculated between the query and the reference audio, using the auxiliary information stored in the hashtable t_1. If query and reference

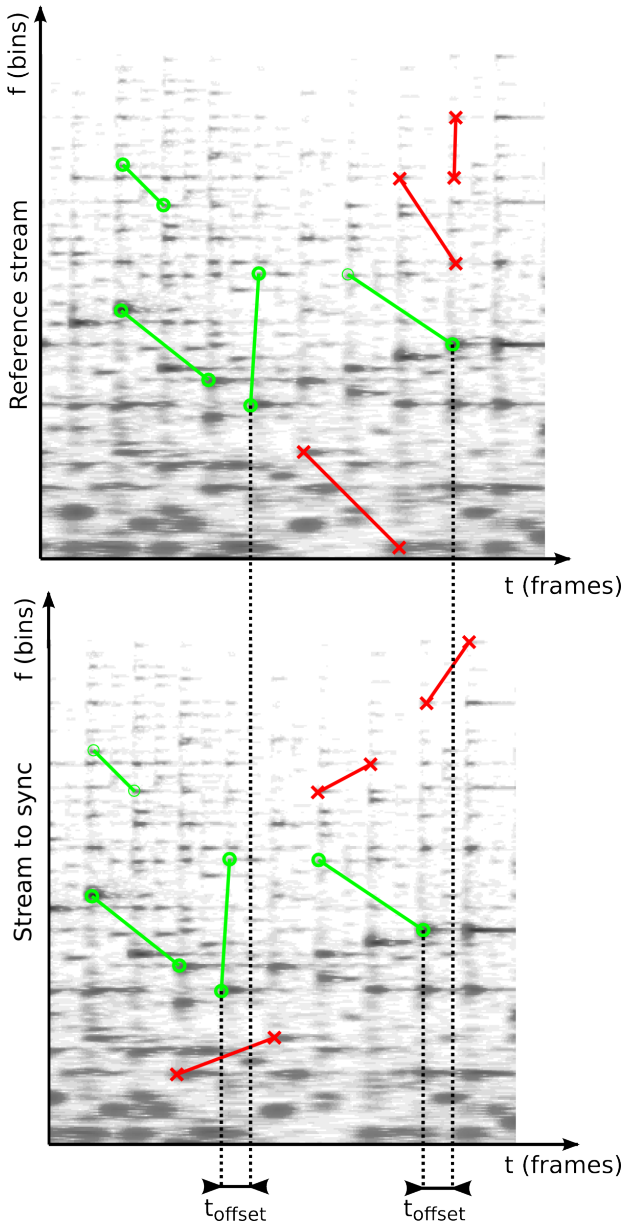

Fig. 2 Two streams of audio with fingerprints marked. Some fingerprints are present in both streams (green, O) while others are not (red, ×). Matching fingerprints have the same offset (t_{offset}), indicated by the dotted lines.

audio match, an identical time offset will appear multiple times. Random chance matches do not have this property. If a match is finally found, the time offset is reported. The matching step is visualized in Figure 2. In Figure 2 several matching fingerprints are found. For two of those matching fingerprints the time offset is indicated using the dotted lines. The figure also makes clear that the method is robust to noise or audio that is only present in one of the streams. Since only a few fingerprints need to match with the same offset, the other fingerprints - introduced by noise or other sources - can be safely discarded (the red fingerprints in Figure 2).

In this algorithm time offsets are expressed using the analysis frame index. The time resolution of such audio frame is not sufficient for precise synchronization. [1] To improve time resolution cross-covariance between audio blocks of the reference r audio and query q is calculated. For each time shift i the following is done: $(r \star q)_i = \sum_j r_j^* \, q_{i+j}$, with j the size of the audio block. The time shift i with the highest covariance determines the best match between the two signals. By adding the time shift, expressed in samples, to the audio block index times the audio block size, a sample accurate time offset can be determined.

The cross-covariance calculation step is not efficient $\mathcal{O}(i * j^2)$, so it should be done for the least amount of audio blocks possible. Since it is known from before at which audio block indexes similar audio is present - at audio blocks with matching offsets - in the reference and query, the calculation can be limited to only those audio blocks. The number of cross-covariance calculations can be further reduced by only calculating covariances until agreement is reached.

Until now there have been no precautions to deal with drift or dropped samples. To deal with drift the algorithm above is expanded to allow multiple time-offsets between the audio that needs to be synchronized and a reference. During the iteration of fingerprint hash matches, a list of matching fingerprints is kept for each offset. If list of matching fingerprints reaches a certain threshold the corresponding offset is counted as a valid. Drift can then be identified since many, gradually increasing or decreasing, offsets will be reported. If samples are dropped, two distinct offsets will be reported. The time at which samples are dropped or drift occurs is found in the list of matching fingerprints for an offset. The time information of the first and last fingerprint match mark the beginning and end of a sequence of detected matches.

3 Synchronizing data streams

With a working audio-to-audio alignment strategy in place, a setup for multimodal recording should include audio streams for each sensor stream. While building a setup it is of utmost importance that the sensor-stream and the corresponding audio-stream are in check. For a video recording this means that AV-sync needs to be guaranteed. To make sure that analog sensors are cor-

[1] If for example audio with an 8000Hz sample rate is used and each analysis frame is 128 samples, then time resolution is limited to 16ms.

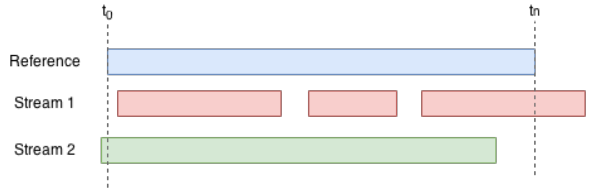

Fig. 3 A reference stream (blue) can be synchronized with streams one and two. It allows a workflow where streams are started and stopped (red) or start before the reference stream (green).

rectly synchronized with audio the use of a data acquisition module is advised. Generally these modules are able to sample data at sufficiently high sampling rates and precision. The main advantage is that such module generally has many analog input ports and by recording the audio via the same path it is guaranteed to be in sync with the sensor streams. In the setup detailed below (Section 4), for example, an USB data acquisition module with 16 inputs, 16 bit resolution and maximum sample rate of 200kHz is used.

To accommodate data-streams with an inherently low sampling - 8kHz or less - rate a method can be devised that does not sample the audio at the same rate as the data-stream. In the setup detailed below an accelerometer (Section 4) is sampled at 345Hz by dividing the audio into blocks of 128 samples, and only measuring one acceleration for 128 audio samples. Since the audio is sampled at 44.1kHz the data is indeed sampled at $44100Hz/128 = 345Hz$. Depending on the type of data-stream and audio-recording device, other sampling rates can be achieved by using other divisors. Conversely, high data sampling rates can be accommodated by using a multiplier instead of a divisor.

Note that during such recording session a lot of freedom concerning the workflow is gained. While one stream is recording continuously stopping and starting other recording modalities can be done without affecting sync. The order at which recording devices are started also has no effect on synchronization. This is in stark contrast with other synchronization methods. In Figure 3 this is shown.

Once all data is synchronized analysis can take place. If video analysis is needed, a tool like ELAN [17] can be used. If audio and sensor-streams are combined without video, Sonic Visualizer[3] is helpful to check mutual alignment. To store and share multimodal data RepoVIZZ[9] is useful.

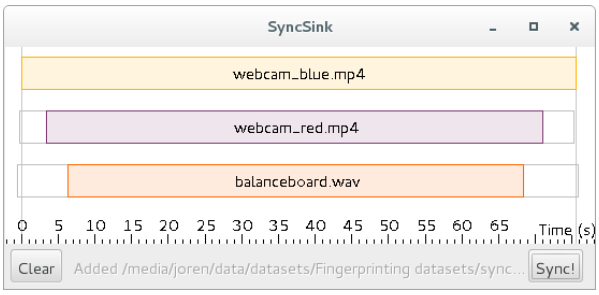

Fig. 4 A user-friendly interface to synchronize media and data files. First a reference media-file is added using drag-and-drop. The audio steam of the reference is extracted and plotted on a timeline as the topmost box. Subsequently other media-files are added. The offsets with respect to the reference are calculated and plotted. CSV-files with timestamps and data recorded in sync with a stream can be attached to a respective audio stream. Finally, after pressing `Sync!`, the data and media files are modified to be exactly in sync with the reference.

4 Results

To test the audio-to-audio alignment algorithm, it was implemented in Java. A user-friendly interface, called SyncSink[2], supporting drag-and-drop was developed as well. The SyncSink tool is depicted in Figure 4. The implementation of the algorithm is based on the open-source acoustic fingerprinting software Panako [14] and TarsosDSP[13,12]. Panako contains several modules that can be reused: the automatic conversion of audio in various formats to to PCM, the efficient FFT implementation and the spectral peak detection algorithm which uses a two dimensional min-max filter. The algorithm works best with an FFT of 512 points, and a step size of 128 samples for a sampling rate of 8000Hz. At least 7 fingerprints need to match before a synchronization solution is presented. With this parameter configuration, the worst synchronization accuracy is equal to $8000Hz/128 = 16ms$. Thanks to the cross covariance step, the best case synchronization is sample accurate. It is limited to $1/8000Hz = 0.125ms$.

To measure the accuracy of the time-offsets that are reported by the algorithm the following experiment is done. For each audio file in a dataset a random snippet of ten seconds is copied. The ten seconds of audio is stored together with an accurate representation of the offset at which it starts in the reference audio. Subsequently the snippet is aligned with the reference audio and the actual offset is compared with the offset reported by the algorithm. To make the task more

[2] SyncSink is included into the GPL'd Panako project available at `http://panako.be`

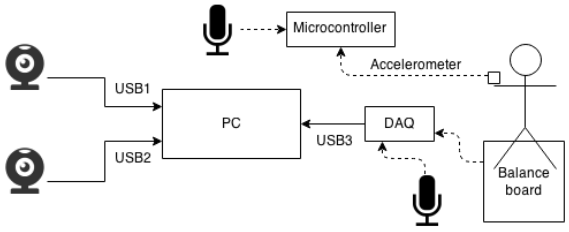

Fig. 5 Multimodal recording system diagram. Each webcam has a microphone and is connected to the pc via USB. The dashed arrows represent analog signals. The balance board has four analog sensors but these are simplified to one connection in the schematic. The analog output of the microphones is also recorded through the DAQ. An analog accelerometer is connected with a microcontroller which also records audio.

realistic the snippet is GSM 06.10 encoded. The GSM 06.10 encoding is a low-quality 8KHz, 8-bit encoding. This degradation is done to ensure that the algorithm reports precise time-offsets even when high-fidelity signals - the reference audio files - are aligned with low-quality audio - the snippets.

The procedure described above was executed for a thousand snippets of ten seconds. For 17 an incorrect offset was found due to identically repeating audio. It could be said that these offsets yield an alternative, but equally correct, alignment. For 10 snippets no alignment was found. For the remaining 973 snippets the offsets were on average 1.01ms off, with a standard deviation of 2.2ms. The worst case of 16ms (128/8000Hz) was reported once. Note that a delay in audio from 1-14ms affects spatial localization, a 15-34ms delay creates a chorus/flanger like effect and starting from 35ms discrete echos can be heared.

To get an idea how quickly the algorithm returns a precise offset, the runtime was measured. Four reference audio files were created, each with a duration of one hour and each with a corresponding query file. The queries consist of the same audio but GSM encoded and with a small time-offset. A query performance of on average 81 times real-time is reached on modest computing hardware[3] when synchronizing the reference with query streams.

This method was used in a study with dementia patients. The study aimed at measuring how well participants can synchronize to a musical stimulus. A schematic of the system can be found in Figure 5. The system has the following components:

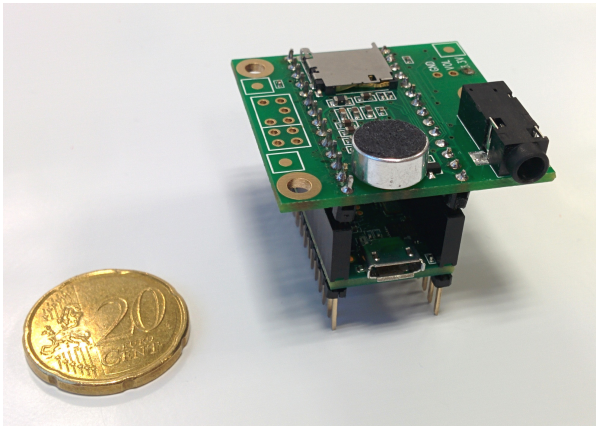

Fig. 6 A microcontroller fitted with an electret microphone and a microSD card slot. It can record audio in real-time together with sensor data.

- A balance board equiped with an analog pressure sensors at each corner.
- Two HD-webcams (Logitech C920), recording the balance board and ambient audio using the internal microphones.
- An electret microphone (CMA-4544PF-W) with amplifier (MAX4466) circuit.
- A data acquisition module with analog inputs. Here an Advantech USB4716 DAQ was used. It has 16 single-ended inputs with 16-bit resolution and is able to sample up to 200 kHz.
- A wearable microcontroller with an electret microphone (CMA-4544PF-W), MicroSD-card slot and an analog accelerometer (MMA7260Q) attached to it. Here we used the Teensy 3.1 with audio shield. It runs at 96MHz and has enough memory and processing power to to handle audio sampled at 44.1kHz in real-time. The microcontroller can be seen in Figure 6.

The microcontroller shown in Figure 6 was programmed to stream audio data sampled at 44.1kHz to the SD-card in blocks of 128 samples. Using the same processing pipeline the instantaneous acceleration was measured for each block of audio. This makes sure that the measurements and audio stay in sync even if there is a temporary bottleneck present in the pipeline. During the recording session this showed to be of value: due to a slow micro SD-card[4] audio samples were dropped twice in a recording session of 30 seconds. Thanks to the audio alignment algorithm it became clear that this happened after 3.1 seconds and after 23.9s. The sensor stream from zero to 3.1 seconds could not be matched

[3] An Intel Core2 Quad CPU Q9650 @ 3.00GHz 4 was used, with 8GB memory. A CPU that entered the market late 2008.

[4] To support real-time recording writing a 512 byte buffers should be faster than $256/44100Hz = 5.802ms$, on average.

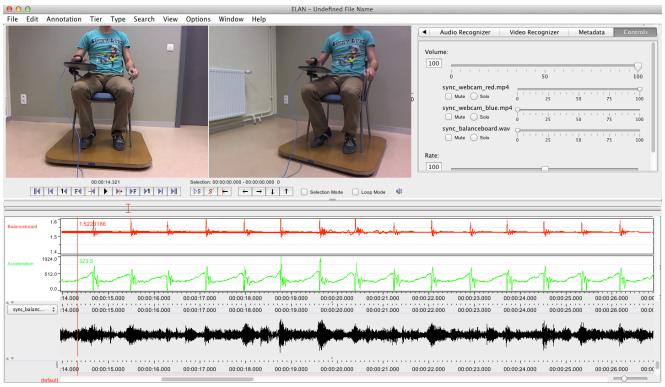

Fig. 7 The synchronized data from the two webcams, accelerometer and balanceboard in ELAN. From top to bottom the synchronized streams are two video-streams, balance-board data (red), accelerometer-data (green) and audio (black).

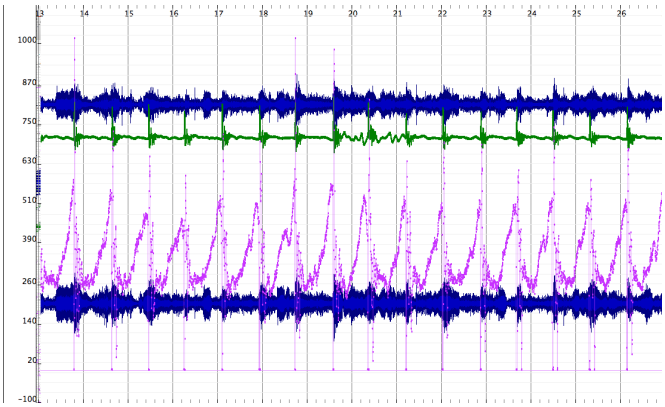

Fig. 8 Synchronized streams in Sonic Visualizer. Here you can see two channel audio synchronized with accelerometer data (top, green) and balanceboard data (bottom, purple).

reliably nor could the data from 23.9s to 30s. The bulk of the acceleration measurements, between 3.1 and 23.9 seconds, could however be synchronized correctly with the other data streams. A feat that would have been difficult using other synchronization means.

Once all data was transferred to a central location mutual time offsets were calculated automatically. Subsequently the files were trimmed in order to synchronize them. In practice this means chopping off a part of the video or audio file (using a tool like ffmpeg) and modifying the data files accordingly. The data of this recording session and the software used is available at `http://0110.be/syncsink`. The next step is to analyse the data with tools like ELAN[17] or Sonic Visualizer[3] (Figure 7 and 8) or any other tool. The analysis itself falls outside the scope of this paper.

5 Discussion

Since ambient audio is used as a synchronization clock signal the speed of sound needs to be taken into account. If microphones are spread out over a room the physical latency quickly adds up to 10ms (for 3.4m) or more. If microphone placement is fixed this can be taken into account. If microphone placement is not fixed it should be determined if the precision that the proposed method can provide is sufficient for the measurement.

The proposed method should not be seen as a universal synchronization solution but provides a method that can fit some workflows. Combining audio-to-audio alignment with other synchronization methods is of course possible. If for example motion capture needs to be synchronized with other sources, the motion capture clock pulse can be routed through a device that records the clock together with ambient audio making it possible to sync with other modalities. The same could be done for an EEG system and clock. This setup would make it possible to sync EEG with motion capture data, an otherwise difficult task. Combining the method with other synchronization approaches - e.g. synchronization markers - is also possible.

The current system is presented as a post-processing step but if the latencies of each recording system are relatively stable then there is potential to use the approach *real-time*. It would work as follows: each recording device would start streaming both audio and data. After about ten seconds audio-to-audio alignment can be done and the mutual offsets can be determined. Once this information is known the sensor data can be buffered and released in a timely manner to form one fused synchronized sensor data stream. The overall latency of stream is then at best equal to the recording modality with the largest latency. While streaming data, the audio-to-audio alignment should be repeated periodically to check or adapt offsets of sensor streams.

6 Conclusion

An efficient audio-to-audio alignment algorithm was presented and used effectively to synchronizing recordings and linked data streams. The algorithm is based on audio fingerprinting techniques. It finds a rough offset using fingerprints and subsequently refines the offset with a cross-covariance step. During synthetic benchmarks an average synchronization accuracy of 1.1ms was reached with a standard deviation of 2.2ms. A query performance of 81 times real-time is reached on modest computing hardware when synchronizing two streams. A case study showed how the method is used

in research practice. The case study combines recordings from two webcams and a balance board together with acceleration data recorded using a microcontroller which were all synchronized reliably and precisely.

References

1. Bannach, D., Amft, O., Lukowicz, P.: Automatic event-based synchronization of multimodal data streams from wearable and ambient sensors. In: EuroSSC 2009: Proceedings of the European Conference on Smart Sensing and Context, *Lecture Note in Computer Science*, vol. 5741, pp. 135–148. Springer (2009)
2. Camurri, A., Coletta, P., Massari, A., Mazzarino, B., Peri, M., Ricchetti, M., Ricci, A., Volpe, G.: Toward real-time multimodal processing: EyesWeb 4.0. In: AISB 2004 Convention: Motion, Emotion and Cognition (2004)
3. Cannam, C., Landone, C., Sandler, M., Bello, J.: The Sonic Visualiser: A visualisation platform for semantic descriptors from musical signals. In: Proceedings of the 7th International Symposium on Music Information Retrieval (ISMIR 2006). Victoria, Canada (2006)
4. Cotton, C.V., Ellis, D.P.W.: Audio fingerprinting to identify multiple videos of an event. In: IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP), pp. 2386–2389. IEEE (2010)
5. Godøy, R.I., Leman, M.: Musical gestures: Sound, movement, and meaning. Routledge, New York (2010)
6. Gowing, M., Kelly, P., O'Connor, N.E., Concolato, C., Essid, S., Feuvre, J.L., Tournemenne, R., Izquierdo, E., Kitanovski, V., Lin, X., Zhang, Q.: Enhanced visualisation of dance performance from automatically synchronised multimodal recordings. In: K.S. Candan, S. Panchanathan, B. Prabhakaran, H. Sundaram, W. chi Feng, N. Sebe (eds.) ACM Multimedia, pp. 667–670. ACM (2011)
7. Hochenbaum, J., Kapur, A.: Nuance: A software tool for capturing synchronous data streams from multimodal musical systems. In: International Computer Music Conference, pp. 1 – 6. ICMC (2012)
8. Jaimovich, J., Knapp, B.: Synchronization of multimodal recordings for musical performance research. In: K. Beilharz, B. Bongers, A. Johnston, S. Ferguson (eds.) Proceedings of the International Conference on New Interfaces for Musical Expression (NIME), pp. 372–374. Sydney, Australia (2010)
9. Mayor, O., Llimona, Q., Marchini, M., Papiotis, P., Maestre, E.: RepoVIZZ: A framework for remote storage, browsing, annotation, and exchange of multi-modal data. In: Proceedings of the 21st ACM international conference on Multimedia, pp. 415–416. ACM (2013)
10. Ogle, J., Ellis, D.P.W.: Fingerprinting to Identify Repeated Sound Events in Long-Duration Personal Audio Recordings. In: IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP), pp. 1–233. Hawaï(2007)
11. Shrestha, P., Barbieri, M., Weda, H.: Synchronization of multi-camera video recordings based on audio. In: Proceedings of the 15th International Conference on Multimedia, MULTIMEDIA '07, pp. 545–548. ACM, New York, NY, USA (2007)
12. Six, J., Cornelis, O., Leman, M.: Tarsos, a modular platform for precise pitch analysis of Western and non-Western music. Journal of New Music Research **42**(2), 113–129 (2013)
13. Six, J., Cornelis, O., Leman, M.: TarsosDSP, a real-time audio processing framework in Java. In: Proceedings of the 53rd AES Conference (AES 53rd). The Audio Engineering Society (2014)
14. Six, J., Leman, M.: Panako - A scalable acoustic fingerprinting system handling time-scale and pitch modification. In: Proceedings of the 15th ISMIR Conference (ISMIR 2014), pp. 1–6 (2014)
15. Wang, A.L.C.: An industrial-strength audio search algorithm. In: Proceedings of the 4th International Symposium on Music Information Retrieval (ISMIR 2003), pp. 7–13 (2003)
16. Wang, A.L.C., Culbert, D.: Robust and invariant audio pattern matching. US Patent US7627477 (2002)
17. Wittenburg, P., Brugman, H., Russel, A., Klassmann, A., Sloetjes, H.: ELAN: A professional framework for multimodality research. In: In Proceedings of Language Resources and Evaluation Conference (LREC) (2006)