Realtime signal synchronization with acoustic fingerprinting

Ward Van Assche

Supervisor(s): Joren Six, Marleen Denert

Abstract—**Many experiments executed at IPEM use sensors such as accelerometers and pressure sensors. A common problem after these experiments is the synchronization of data of each sensor. The current synchronization system requires each sensor to be connected to a microphone recording the sound of the environment. With efficient audio-to-audio alignment techniques the latency can be detected very accurately. The current synchronization system is performed as a post-processing step. A real-time and more user-friendly solution is desirable. This paper explores the possibilities to do this. To use the current synchronization algorithms in real-time they had to be changed and optimized in different ways. The new system resulted in a Max/MSP module which makes it possible to run the synchronization in real-time without writing a single line of code.**

Keywords— **Signal Synchronization, Audio Alignment, Real-Time, Acoustic Fingerprinting, Cross-covariance, Digital Signal Processing**

I. Introduction

EXPERIMENTS in various research areas use sensors and video cameras to capture the environment. Before sensor data and video recordings can be analyzed properly they have to be synchronized accurately in time. This is no easy task because of various reasons. The first problem is that the sensor datastreams can be heterogeneous: the sample rate can vary and the resulted data can be very different (video data or numerical samples). Another difficulty is to cope with the fact that some data sources are unreliable. This can lead to dropped samples and drift which cause unexpected latency changes. The most techniques require a post-processing step: the data has to be synchronized manually or by software *after* the experiment. Because this is impractical and time-consuming a technique which can avoid this is desirable.

The most straightforward way to perform the synchronization is by adding markers. This technique is described in article [1]. The way how a marker is placed depends on the type of the stream. A short sound can add a marker in an audio stream, a bright flash can do the same in a video stream. The latency can be found by calculating the difference between the marker positions. The usability of this method is limited because of its poor scalability. The synchronization of a large number of streams can be very challenging. Dropped samples and drift can only be detected when the markers are repeated each time interval, which is impractical. The actual synchronization using this technique is only possible as a post-processing step.

Article [2] describes a method which uses a clock signal to synchronize streams in real-time. However this method avoids the post-processing step it does not fit the requirements: each device (sensor, video camera) should accept a clock-signal as input. Because these devices are very expensive this method is not feasible for the stated problem.

Article [3] discusses a similar problem where its approached in an entirely different way. The described method does not try to synchronize the streams directly. Instead, the (recorded) environment sound is embedded in each stream. This ploy reduces the initial problem to audio-to-audio alignment. Because the recorded environment sound is almost the same for each stream the problem is much easier to tackle. The audio-to-audio algorithms described in the article perform very well. The latency can be detected with a precision less than 1ms. However the article only describes a post-processing approach, the used algorithms look very interesting.

The next part of this paper will describe a method to synchronize streams in real-time using the algorithms mentioned in the previous article. The word “real-time” can be ambiguous in a signal-processing context. Therefore it’s important to specify some requirements for a real-time system. Because the latency-detecting algorithms need some amount of audio in order to determine the latency it’s impossible to immediately output the synchronized signals. In this paper the real-time restriction refers to the fact that the synchronization algorithms are executed while the sensors are collecting data. The post-processing step should be avoided.

II. Determining latency

The method described in article [3] uses two latency detecting algorithms which are very complementary. By combining them, latency can be detected very accurately.

A. *Acoustic fingerprinting*

Acoustic fingerprinting is a fast and robust technique for comparing audio fragments initially described in article [4]. The technique uses fingerprints based on spectral peaks. Each fingerprint contains condensed information based on typical audio properties. This technique allows finding similar audio fragments ignoring noise and other disturbing background sounds.

However the initial application was identifying an audio recording using a huge database containing a myriad of fingerprints it’s also possible to compare the fingerprints of recordings mutually. The latency can be detected by calculating the offset between the fingerprints. The precision varies around 16 ms to 32 ms, this depends on the used parameters.

B. *Calculating the cross-covariance*

The cross-covariance (also referred to as cross-correlation) is a calculation which measures the degree of similarity between two time sequences. Because an audio signal is a time sequence this calculation can be used for determining the latency. When the cross-covariance value is calculated for each possible shift between two audio fragments the shift with the highest result determines the latency.

This method can determine the latency to the nearest sample. The precision in milliseconds depends on the sample rate: when the sample rate is 8000 Hz the maximum precision is $1/8000\text{ Hz} = 0.125\text{ ms}$.

A disadvantage to this method is its performance. The time complexity of the algorithm is $O(n^2)$ where n is the number of samples in each signal. Finding the latency between two audio fragments of 10 s at a sample rate of 8000 Hz would asymptotically result in $6.4 \cdot 10^9$ calculations, which is unfeasible on a regular computer.

C. *Refining the results*

The two previously described algorithms are very complementary. Acoustic fingerprinting allows finding the latency very fast and robustly. The cross-covariance algorithm can detect the latency between two tiny pieces of audio very accurately. These advantages can be easily combined.

By determining the raw latency with acoustic fingerprinting, the number of samples used in the cross-covariance calculation can be limited. By cutting the raw latency from the corresponding audio fragment, the new latency is reduced. When the acoustic fingerprinting algorithm uses a precision of 32 ms, the cross covariance should be calculated on two audio fragments of at least 256 samples (when the sample rate is 8000 Hz). This asymptotically results in 65 536 calculations, which is much more feasible than without the acoustic fingerprinting step.

III. Buffering streams

Sed nec tortor in libero rutrum pellentesque[5] et gravida turpis. Phasellus gravida neque vitae elit fringilla, a efficitur purus sollicitudin. Proin lacus est, suscipit sed nibh ac, hendrerit eleifend leo.

IV. Latency filters

Sed nec tortor in libero rutrum pellentesque[5] et gravida turpis. Phasellus gravida neque vitae elit fringilla, a efficitur purus sollicitudin. Proin lacus est, suscipit sed nibh ac, hendrerit eleifend leo.

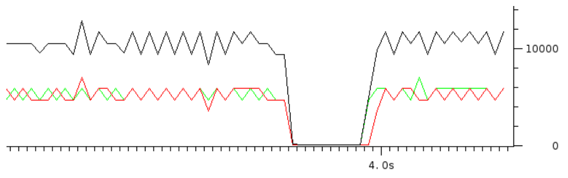

Fig. 1. Detailed capture of the stream at the moment of a handover between two simulated AP's with a strong signal. Notice the gap of 100 ms.

V. Synchronization

Sed nec tortor in libero rutrum pellentesque[5] et gravida turpis. Phasellus gravida neque vitae elit fringilla, a efficitur purus sollicitudin. Proin lacus est, suscipit sed nibh ac, hendrerit eleifend leo.

VI. Results

Sed nec tortor in libero rutrum pellentesque[5] et gravida turpis. Phasellus gravida neque vitae elit fringilla, a efficitur purus sollicitudin. Proin lacus est, suscipit sed nibh ac, hendrerit eleifend leo.

VII. Conclusion

Sed nec tortor in libero rutrum pellentesque[5] et gravida turpis. Phasellus gravida neque vitae elit fringilla, a efficitur purus sollicitudin. Proin lacus est, suscipit sed nibh ac, hendrerit eleifend leo.

References

[1] David Bannach, Oliver Amft, and Paul Lukowicz, "Automatic event-based synchronization of multimodal data streams from wearable and ambient sensors," in *Smart sensing and context*, pp. 135–148. Springer, 2009.

[2] Javier Jaimovich and Benjamin Knapp, "Synchronization of multimodal recordings for musical performance research.," in *NIME*, 2010, pp. 372–374.

[3] Joren Six and Marc Leman, "Synchronizing Multimodal Recordings Using Audio-To-Audio Alignment," *Journal of Multimodal User Interfaces*, vol. 9, no. 3, pp. 223–229, 2015.

[4] Avery Li-Chun Wang, "An industrial-strength audio search algorithm," in *ISMIR 2003, 4th Symposium Conference on Music Information Retrieval*, 2003, pp. 7–13.

[5] Joren Six, Olmo Cornelis, and Marc Leman, "TarsosDSP, a Real-Time Audio Processing Framework in Java," in *Proceedings of the 53rd AES Conference (AES 53rd)*. 2014, The Audio Engineering Society.