Automatic Event-based Synchronization of Multimodal Data Streams from Wearable and Ambient Sensors

David Bannach[1], Oliver Amft[2], and Paul Lukowicz[1]

[1] Embedded Systems Lab, University of Passau
{david.bannach, paul.lukowicz}@uni-passau.de
[2] Signal Processing Systems, TU Eindhoven
oliver.amft@gmail.com

Abstract. A major challenge in using multi-modal, distributed sensor systems for activity recognition is to maintain a temporal synchronization between individually recorded data streams. A common approach is to use well defined 'synchronization actions' performed by the user to generate, easily identifiable pattern events in all recorded data streams. The events are then used to manually align data streams. This paper proposes an automatic method for this synchronization.
We demonstrate that synchronization actions can be automatically identified and used for stream synchronization across widely different sensors such as acceleration, sound, force, and a motion tracking system. We describe fundamental properties and bounds of our event-based synchronization approach. In particular, we show that the event timing relation is transitive for sensor groups with shared members. We analyzed our synchronization approach in three studies. For a large dataset of 5 users and totally 308 data stream minutes we achieved a synchronization error of 0.3 s for more than 80% of the stream.

1 Introduction

Multi-modal, distributed sensor systems have been proposed for a variety of wearable and pervasive computing applications. A major practical concern to use such systems is how to temporally synchronize data streams between individual sensors. In particular, methods that rely on sensor fusion (such as computing joint features from several sensors or jointly feeding a classifier) require that sensor signals are well synchronized.

The synchronization problem is a well known and widely studied distributed systems issue [3]. Without precautions, clocks of physically separate processors will run asynchronously [10]. A receiving node observes this as skew and drift in the incoming sensor data stream. Furthermore, networked nodes may startup or resume operation independently. This can causes offsets at stream receivers, similar to not handled data interrupts in wireless links.

Much effort has been devoted to lightweight clock synchronization methods for wireless sensor networks (see related work). In general these methods rely on

elaborate message passing and time distribution protocols among the network nodes. This paper does not aim to compete with, or improve upon this work. While in theory, it may be assumed that a sensor node could receive messages and run such synchronization protocols, in practice many on-body and pervasive sensing setups do not have appropriate capabilities. For one, leaving out the receiving capability reduces power consumption and makes node design simpler. Secondly, systems are often built using nodes from different manufacturers with insufficient built-in synchronization support. It is not common that commercial nodes provide access to node-internal communication routines.

Consequently, we look at a specific problem variation that occurs frequently in wearable and pervasive systems: sensor nodes merely stream data in a hierarchical topology and are not able to communicate with each other. This means that no message passing protocols can be run between the nodes. Instead, the synchronization can rely solely on the content of the data stream.

It is common practice in wearable and pervasive application studies to rely on event-based synchronization of data streams. To this end, 'synchronization actions' are defined, which are periodically performed by a subject during the recording (at least once at the start of a data collection). Such actions are designed to simultaneously activate multiple sensors and provide an easily identifiable data signature. Often encountered examples are clapping, jumping, and hitting a surface. The characteristic signature of a synchronization action can be used to align signal segments from different sensors. Currently, this is typically done through manual inspection of the data streams. This paper investigates a method to automate this process.

It is intuiting to rely on natural activity patterns to synchronize sensor data streams that are performed by users in a pervasive application. However, it is essential to develop a robust synchronization framework in the first place. Hence, we chose validation scenarios for this initial work, in which particularly selected synchronization actions had been inserted into sensor data streams. We subsequently discuss how our concept could extend on natural activities.

1.1 Related Work

Time synchronization is a well known and widely studied problem in wireless sensor networks, in particular for subsequent sensor data fusion. Surveys can be found in [8] and [10]. Typically, the solutions target a network-wide synchronization by aligning the clocks of all physically distributed nodes [10, 9]. For this purpose specific messages are exchanged among the nodes.

As detailed above, we target systems where a synchronization based on exchanging messages is not feasible due to unidirectional communication channels. Our approach relies on data signatures (events) embedded in data streams, hence it does not modify the source clocks. However, it ensures a relative synchronization between data sources at the fusion unit.

Instead of establishing a network-wide clock, various communication systems use a source synchronization approach for medium access. These techniques target to align message recipient(s) on a link with the clock provided by the sender,

through monitoring packet reception [7] or protocol-specific features, such as a preamble. Our approach is similar to source synchronization, as the fusion unit observes the relative offset in the incoming data streams.

More relevant research looks at correlation of different sensor data streams for various applications. This includes the use of correlations to determine that a set of devices are carried by the same person [5], as well as attempts to use correlation between sounds received by different sensors for indoor location [2].

Many researchers experienced clock synchronization failures in deployed sensor networks due to software bugs, communication failures, and frequent reboots, rendering large parts of data unusable [11, 4, 6]. Common solutions try to recover corrupted timestamps in a postmortem process. Werner-Allen [11] corrects errors caused by the time synchronization protocol on behalf of a base station that adds its own timestamps to the collected data. They build piecewise linear models to map local node time to global time without utilizing the actual sensor data collected by the nodes. Lukac [6] proposes data driven time synchronization by utilizing background noise in seismic sensing systems. By building a model of the propagation of micorseisms (seismic waves that travel through continents, originated by oceans) they can reconstruct timestamps in large datasets gathered by seismic networks. The Sundial system [4] uses light sensors to detect length of day and noon time and compares them offline with the astronomical model to estimate the correct global timestamps. While this approach is based on a single sensing modality they also apply a method to correlate rain events with soil humidity events for finding the correct day. This is very similar to our approach but we do not rely on additional information about the events to find the correct mapping and we explicitly allow event spotting errors.

1.2 Challenges of event-based stream synchronization

The conceptual solution for event-based synchronization is straightforward. We assume that each data item is time-stamped with a local clock (or stream sequence number). Once a first synchronization action is localized in all data streams, offsets between the local clocks or sequence numbers are computed. Upon locating a second event, differences between sensor clock frequencies can be computed. Subsequently, the data streams are aligned according to these differences. Additional events may help keeping streams synchronized over time and could increase synchronization accuracy.

In reality, there are a number of problems related to reliable, automatic spotting of synchronization actions. It is well known that recognizing short actions embedded in a continuous stream of arbitrary sensor data is a hard problem. To synchronize data streams based on events, actions must be spotted separately in each data stream. This is even more difficult than under a multi-modal spotting scheme, where signals from several sensors can be combined. Hence actions could be confused with other arbitrary actions. As a consequence, the synchronization algorithm has to cope with deleted (missed) and inserted (wrongly recognized) events in individual streams. Hence it is not a-priori clear which events should be aligned with each other.

Another particular spotting property is the timing variations among retrieved events. For an event-based synchronization approach, this variation can have two origins. On one hand, a spotting procedure can introduce a temporal jitter due to an assumed segmentation. Typically, this jitter is a fraction of the expected event size [1]. Secondly, spotting on independent sensors can impose different event patterns. For example, hitting a table with a fist will generate a short temporal acceleration at the wrist, while the table will vibrate longer with a delayed onset. Thus, even if the synchronization actions are correctly spotted, it is not always clear how to relate events in time.

1.3 Paper Contributions

This paper investigates solutions to the problems described above, facilitating a practical implementation of automatic, event-based synchronization. We demonstrate that synchronization actions can be identified across widely different sensors, such as accelerometers and optical motion tracking systems. We show how using repetitive events can improve synchronization results. Moreover, we show that the timing relation is transitive for sensor groups with shared members. The latter can be exploited for further reliability improvements and to synchronize larger multi-modal sensor networks.

2 Spotting and synchronization approach

Our approach consists of two steps. In the first step appropriately defined synchronization actions are spotted in each sensor data stream. Specifically, we spotted hand 'clap', 'push-release' of a button, and arm 'shake' actions as described below. In a second step our online synchronization algorithm is used to establish, which events in two different streams correspond to the same physical action. This step deals with missing and inserted events, as well as temporal jitter. The result of this step is a continuous alignment estimation corresponding to events derived from the participating data streams.

The required synchronization performance is application dependent. For a typical example of motion-related activities in daily life, such as considered in this work, stream alignment should be well below 1 s. Nevertheless, an alignment performance below 0.1 s is typically not needed. In our evaluation we selected a performance of 0.3 s as target alignment.

2.1 Event spotting

In our approach synchronization actions are associated with a single timestamp from each data stream. Nevertheless, even short physical actions such as a hand clap exhibit a temporal signal pattern. Figure 1(a) illustrates such patterns for acceleration and audio streams. Hence, we specifically modelled a synchronization point in the event patterns using signal features. Two sliding window algorithms were used to spot 'clap', 'push-release', and 'shake' gestures on individual

modalities. Figures 1(a)-1(c) show these events on four different sensing modalities (acceleration, force sensitive resistors (FSR), positioning sensors, and audio). Other event types or sensing modalities may require further spotting algorithms, which can be easily designed.

For both algorithms we obtained model parameters manually by analyzing pattern examples. The parameters were chosen to minimize event miss rate. This spotting approach might result in suboptimal event recognition performance. However, it was not the goal of this work to maximize event spotting performance, but to analyze practical operation of our stream synchronization algorithm.

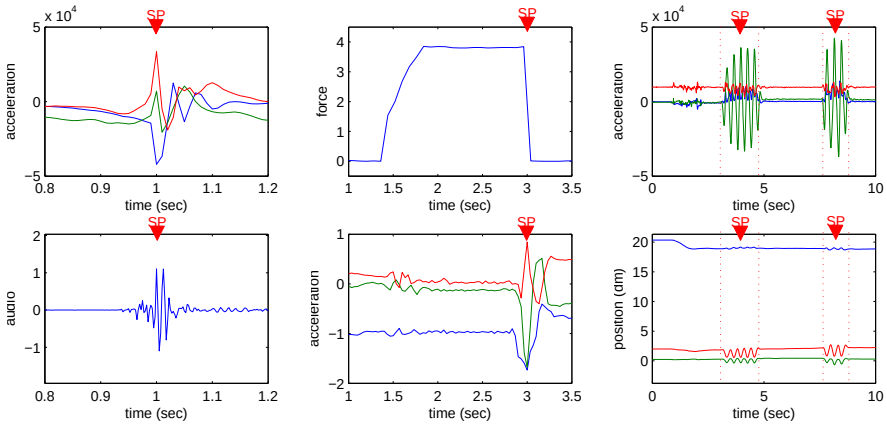
Trailing edge detection The patterns produced by 'clap' and 'push-release' events have similar characteristics. Their signal pattern have a stable phase of "silence" followed by a fast rise or fall (see Fig. 1(a) and 1(b)). The start of a rise/fall phase was considered as the synchronization point. The two events were distinguished by the features: minimal length of silence phase, signal range of silence phase, length of rise/fall phase, and signal range of rise/fall phase.

Shake detection The detection of shakes was implemented by tracking peaks (alternating local minima and maxima) over time. The features, minimal peak height, min/max of time between peaks, and number of peaks had been used. The begin and end of shake events may vary between sensing modalities (shown in Fig. 1(c)). Consequently, we defined the center point (midpoint between first and last peak timestamp) as synchronization point for shake events.

2.2 Event-based synchronization

While the event pattern structure (such as signal peaks for 'shake' events) could be used to analyze the synchronization between two sensor streams, we found these patterns to be too variable. Moreover, the patterns may differ between modalities, which would hamper their alignment. Instead our approach relies on a sequence of spotted events from both streams to estimate alignment. In particular, we consider the temporal distribution of the retrieved events as relevant synchronization property. Thus, as long as events do not occur with constant frequency, we expect that a match between event sequences of both streams can be found. Clearly, the probability for a correct event match will increase with the sequence length of matching events.

Algorithm: The synchronization algorithm compares two event sequences $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_m\}$ to estimate a sequence match. A sliding window with a size of 30 s was applied on the event sequence for online operation and retain multiple events for alignment analysis at any considered stream position. The procedure can be understood as shifting sequence B in time until an optimal match is found with sequence A.



(a) (b) (c)

Fig. 1. Aligned sensor signals: (a) hand 'clap' recorded from wrist mounted accelerometers and a microphone, (b) button push followed by sudden release recorded from an FSR under thumb and wrist mounted accelerometers ('push-release'), (c) two 'shake' motions recorded from accelerometers and a camera-based positioning system. "SP" indicates synchronization points returned by our spotting procedure and subsequently used for alignment.

Let

$$d(i,j) = t(b_i) - t(a_j),\ i \in [1,\dots n],\ j \in [1,\dots m] \tag{1}$$

be the n-by-m matrix of timestamp differences between all possible events of sequence A and B, and let

$$P = \{(i_1,j_1),\dots,(i_k,j_k)\},\ k > 1, i_l < i_{l+1}, j_l < j_{l+1}\ \forall l \in [1,\dots k-1] \tag{2}$$

be a matching path of length k between A and B. Each cell (i,j) in path P denotes a match between event a_i and b_j and therefore indicates an offset of $d(i,j)$ between A and B. Matches in P are unique ($i_l \neq i_{l'},\ j_l \neq j_{l'}\ \forall l \neq l'$) but not necessarily include all events ($i_{l+1} - i_l \geq 1,\ j_{l+1} - j_l \geq 1$). The latter allows coping with event spotting errors.

We analyze all potential paths (candidate paths) and select the best by applying a weighting function $w(P) = w_K(P) * w_V(P)$ on each candidate. Weights w_K and w_V were defined with an intend to maximize path length k and minimize variance of path distances $v = var(\{d(i_1,j_1),\dots,d(i_k,j_k)\})$. The synchronization algorithm provides a continuous estimation of relative stream offsets between two sources by selecting $P_{best} = max[w(P)]$.

The path search can be performed incrementally. For a given cell (i_l, j_l) a subsequent cell (i_{l+1}, j_{l+1}) in the path is determined by searching in the submatrix of (i_l, j_l):

$$j_{l+1} = \arg \min_{j' \in [j_l+1\dots m]} |d_1 - d(i_l+1, j')| \text{ and} \tag{3}$$

$$i_{l+1} = \arg \min_{i' \in [i_l+1\dots n]} |d_1 - d(i', j_{l+1})|, \tag{4}$$

where $d_1 = d(i_1, j_1)$ denotes a path's starting point. If $|d_1 - d(i_{l+1}, j_{l+1})|$ does not fall below a tolerance threshold this cell is discarded and searching is repeated in submatrix $(i_{l+1}, j_{l+1} + 1)$. In our evaluations a tolerance of 0.3 s was used to prune non-relevant paths. Multiple candidates of matching paths are found by starting a new search from each top and left border cell (see Fig. 2.2). This corresponds to shifting sequence B against sequence A.

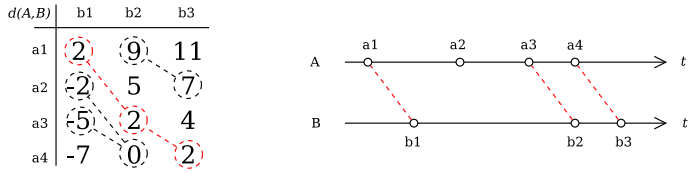


Fig. 2. Two event sequences with the corresponding matching paths highlighted in the timestamp differences matrix (left), and the best path displayed on the timeline (right).

For path weight $w_K = 1 - e^{-((k-1)/(2-k_E))^2}$ was used where k is the actual path length and k_E denotes the expected path length. As variance weight we used $w_V = 1 - e^{-0.5(v/R)^2}$, where regularizer R serves to control variance weight contribution. Weighting behavior is designed to maximize weight for particular parameter sets. For w_K a maximum is obtained at $k = k_E$, for w_V, variance should be minimized.

Parameters k_E and R can be used to adapt our synchronization algorithm. Expected sequence length k_E depends on the targeted synchronization performance, where larger k_E, hence longer sequences, will result in reduced synchronization errors. Parameter R controls the influence of temporal event jitter in w_V. For this work, we determined $R = 0.05$ in empirical tests and confirmed it in all evaluations discussed in Section 3.

It should be noted that our heuristic algorithm can miss a globally optimal path as Eqs. 3, 4 are sequentially evaluated. We consider our approach as a tradeoff limiting computational complexity compared to a full search which would involve evaluating all potential paths within a submatrix of cell (i_l, j_l). Our evaluation results confirm that this approach is feasible.

2.3 Specific synchronization properties

Transitivity Synchronization alignments, as determined with our approach, are transitive. If relative offsets between data streams A and B and between stream B and C are known, then the offset between stream A and C can be deduced. This transitive property allows that sensors or sensor subnets, which do not share common events with other sensors, can still be synchronized on behalf of mediating sensors. We demonstrate how this property can be exploited by analyzing synchronization pairs in Section 3. Moreover, such dependencies could be used to refine synchronization result, as potentially multiple synchronization sources become available.

Performance bounds We analyzed theoretical performance bounds for our synchronization algorithm with regard to event input. In particular, we considered the following essential properties: (1) effect of event sequence timing, (2) impact of the number of synchronization actions, and (3) temporal event jitter introduced through the spotting method.

A sequence of events $a = \{a_1, \ldots, a_n\}$ can be described in terms of time distances between subsequent events $s = \{s_1, \ldots, s_{n-1}\} = \{t(a_2)-t(a_1), \ldots, t(a_n)-t(a_{n-1})\}$. The worst-case scenario for our synchronization algorithm is a monotonous sequence of constant event distances, hence $s_1 = s_2 = \ldots = s_n$. In this specific case, an alignment can not be uniquely identified using our distance approach. However, in practice this situation will be rare and difficult to construct.

If we allow some degree of temporal variance in an input event sequence, we may describe s as a random variable with Gaussian probability distribution $p(s) = \mathcal{N}(\mu, \sigma)$. We denote $\hat{s}$ as an actual distance between spotted events that have jitter $\pm\omega$. Consequently, the probability for obtaining a specific event distance can be derived by:

$$P(s = \hat{s} \pm \omega) = \int_{\hat{s}-\omega}^{\hat{s}+\omega} p(s), \tag{5}$$

which has its maximum at $s^* = \mu$. Hence, if σ increases, the probability for a unique event match is raised, as $P(s)$ decreases. In contrast, if the temporal event jitter ω increases, $P(s)$ increases as well. This means, that it is more likely to randomly observe a specific event distance.

In the analysis above we considered a distance between two events only, hence $k = 1$. In a more practical setting, however, several synchronization actions may be considered for estimating alignment. Thus,

$$P_{Path}(\omega, k) = P(s_1 = \hat{s}_1 \pm \omega, \ldots, s_k = \hat{s}_k \pm \omega) = \prod_{i=1}^{k} P(\hat{s}_i \pm \omega) \tag{6}$$

shows the probability for a matching path of constant event distance. Hence, the chance of a random match decreases exponentially with increasing event sequence length k.

3 Evaluation

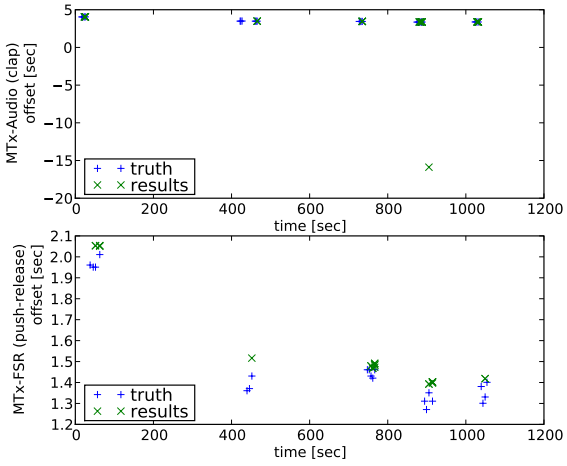
We performed three experiments to analyze the synchronization performance for different events and sensor modalities. All evaluations use sensor modalities that are relevant in ubiquitous applications and have been used in previous studies. Initially, two experiments where performed to investigate the synchronization approach with different sensor modalities and combinations (force-acceleration-audio, acceleration-positioning). In a subsequent evaluation, we investigate the performance of the synchronization algorithm in a large data set (totally 308 min. of 5 users) that was recorded for activity recognition in

daily living scenarios. All synchronization actions were annotated and refined in a post-recording step by visual inspections of the data streams. For all experiments the correct alignment was manually determined from signal inspections. Annotation and alignment information was used as ground truth.

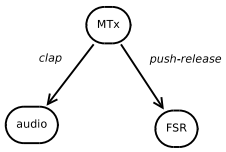
3.1 Evaluation 1: force-acceleration-audio

Data recording The dataset was recorded with a single wrist worn motion sensor (Xsens MTx), one FSR on the desk, and an ambient sound microphone. The MTx comprises 3D sensors for acceleration, rate of turn, and earth-magnetic field. In this investigation only 3D acceleration was considered. MTx were sampled with a nominal data rate of 100 Hz, FSR at 16 Hz, and audio at 8 kHz. For FSR and MTx a wireless connection (Bluetooth) was used to transmit data to a computer. Audio was acquired and timestamped on a second computer and streamed to the first computer for archival.

The total recording time was ∼20 minutes. During this time five sessions of 'push-release' gestures and six sessions of 'clap' gestures were performed by one person. Each gesture was repeated several times during a session resulting in a total of 20 'push-release' and 24 'clap' gestures. Between the synchronization sessions normal office activities were conducted and several times the wireless connection is intentionally broken by increasing the distance to the recording computer. These link interrupts served to simulate both, connection errors and temporary sensor failures.



(a)



(b)

Fig. 3. Evaluation 1: (a): Results for the force-acceleration-audio dataset. The offsets indicate the actual (truth) and estimated (result) alignments for the entire dataset. (b): Synchronization pairing scheme.

Synchronization procedure Using the transitive property of our approach, two synchronization pairs were established in this analysis: (1) MTx-FSR using 'push-release' and (2) MTx-audio using 'clap' gestures. The synchronization pairs are illustrated in Figure 3(b). The streams of each synchronization domain were independently aligned using our synchronization algorithm. We used the trailing edge spotting algorithm to recognize both event types in all streams.

Results The event spotting returned 95.5% of the gesture instances correctly. Only 4 instances from the MTx sensor were missed. In total 23 false positives were returned (26.1%), 19 of them by the clap detection on the accelerometer signals, which classified most 'push-release' events as 'clap' events. The remaining false detections were caused by the FSR spotting, which generated insertions at rising signal edges.

Figure 3(a) shows the synchronization algorithm results as a time plot. Green crosses indicate true offsets, manually determined at each event instance. Blue '+' signs mark the offsets estimated by our synchronization algorithm. At least one result was found for each session of 'push-release' gestures with an error of $\sim$0.1 s. One session of clap gestures did not lead to a result because of missed events. This session contained only three annotated event instances, which was also used for k_E. Consequently, if one event was not retrieved by the spotting algorithm, a correct match was no longer possible. The MTx-audio pair incurred one error with an error of -10 s due to event insertions.

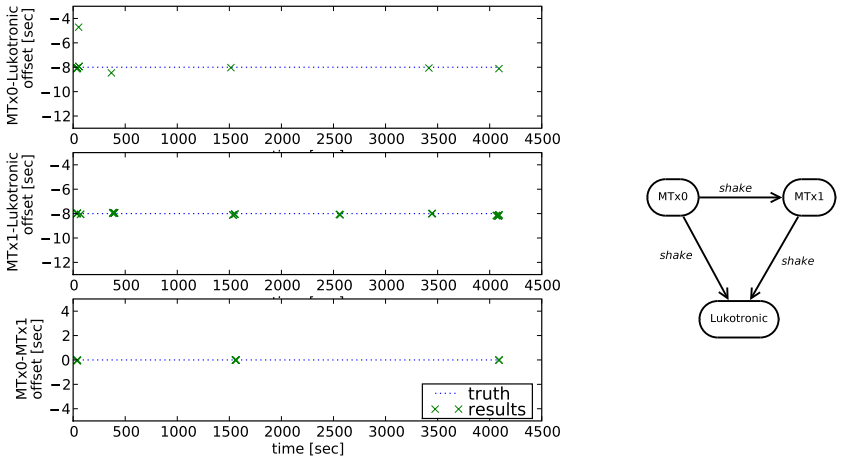
3.2 Evaluation 2: acceleration-positioning

Data recording We attached infrared markers of an optical motion capturing system (Lukotronic) onto two accelerometer-based motion sensors (Xsens MTx). The position data from the Lukotronic system was streamed over a wired network connection (TCP) to the recording computer. The acceleration data from the two MTx sensors was transferred over two independent wireless connections (Bluetooth). All sensing systems sampled with a nominal data rate of 100Hz. A dataset of $\sim$60 minutes was recorded during which individual units and both units together were shaken.

Synchronization procedure Three synchronization pairs were established: (1) MTx0-Lukotronic, (2) MTx1-Lukotronic, and (3) MTx0-MTx1. The synchronization pairs are illustrated in Figure 4(b). The streams of each synchronization domain were independently aligned using our synchronization algorithm.

We used the shake detection algorithm to spot 'shake' events for both sensor modalities. As described before, we considered the midpoint of a shake event as synchronization point. This approach compensated varying detections of peak begin and ends on both modalities.

Results The manual data stream alignment revealed that the camera system had an offset of 8 seconds relative to both accelerometers. This can be explained



(a) (b)

Fig. 4. Evaluation 2: (a): Results for the acceleration-positioning dataset. The offsets indicate the actual (truth) and estimated (result) alignments for the entire dataset. (b): Synchronization pairing scheme.

by different starting times of both system types and by differences in transmission delays. In addition to the stream offset, skew and drift seemed to be marginal for this recording.

The event spotting correctly returned all synchronization actions and did not incur insertion errors. Figure 4(a) shows the synchronization algorithm results as a time plot for all synchronization pairs. The plot can be interpreted as discussed for evaluation 1 above. The algorithm correctly detected that no offset existed at multiple time points on each synchronization domain. The algorithm incurred one error due to a similar event sequence found in one stream.

This experiment demonstrates that repetitive gestures (with an extent in time) can be used as synchronization actions. Moreover, it confirms the feasibility to use the transitive property for independent synchronization pairs. Sensors that are members of two independent synchronization pairs could use both to refine alignment results.

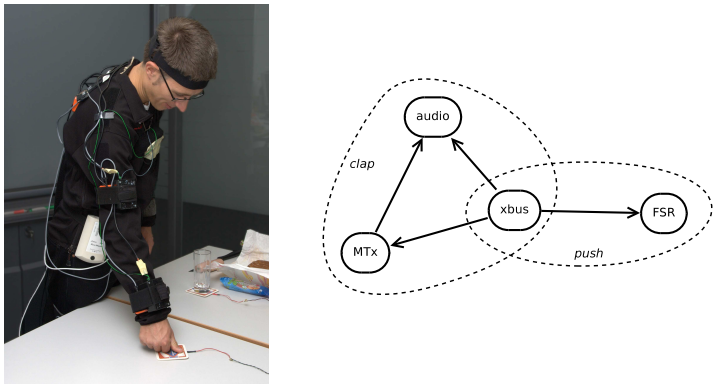
3.3 Evaluation 3: Office scenario

Data recording A setup consisting of a motion sensor at the right wrist (Xsens MTx), left wrist (Xsens MTx), and ambient sound microphone was worn by the users. In addition, four FSR sensors were mounted under pads that could sense if household utensils are placed on them were used. The right wrist motion sensor was attached to an Xbus, sampled at 30 Hz and interfaced to a laptop computer using the a wired connection. For the microphone a wired connection at 8 kHz sampling rate was used. The left wrist Xsens was connected through a wireless link (Bluetooth) at 30 Hz. The FSRs were synchronously sampled at 12.5 Hz by another (stationary computer). During the actual study the laptop that interfaced all wearable sensors, was carried by an experiment observer who

followed the test person. In total this setup had four unsynchronized data streams recorded by two computers.

Five test person were individually recorded for sessions ∼60 minutes. During these sessions the test persons were asked to perform various activities in four different scenarios: office (desktop work, printing, using fax), eating (preparing sandwich, making coffee, eating), gaming (installing a Wii console, gaming using the Wii remote), and leisure (reading magazines, speaking at the phoning).

Synchronization procedure Three times during the session (beginning, mid-time, end) the test persons were asked to perform synchronization actions consisting of 'push-release' of one FSR pads on the table using the right arm (activation of right MTx and FSR), and 'clap' with both hands (activation of both MTx and audio). Figure 5(a) illustrates the synchronization action for 'push-release'. The synchronization actions were repeated three to four times, resulting in a total of at least 45 relevant synchronization actions per test person (when counted each data streams separately).



(a) (b)

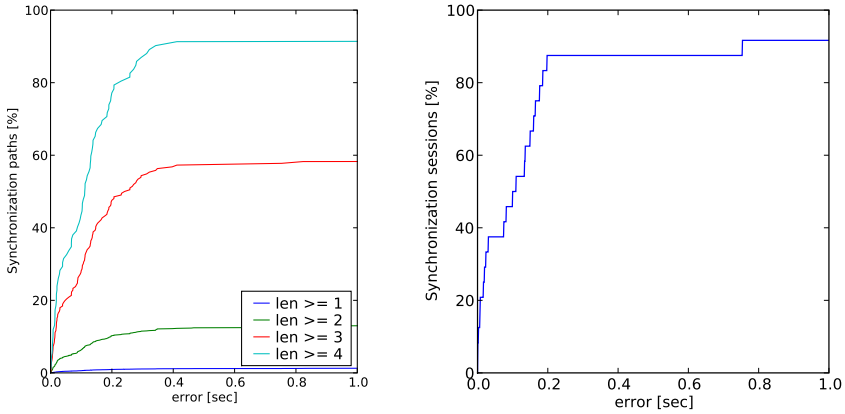
Fig. 5. (a): Test person during push-release gesture. (b): Synchronization pairing scheme for evaluation 3.

Four synchronization pairs were established: (1) right MTx-left MTx using 'clap', (2) right MTx-audio using 'clap', (3) left MTx-audio using 'clap', and (4) right MTx-FSR using 'push-release'. The synchronization pairs are illustrated in Figure 5(b). The streams of each synchronization domain were independently aligned using our synchronization algorithm. We used the trailing edge spotting algorithm to recognize both event types in all streams.

Results The event spotting algorithm was used with the same parameter set for all test persons. In total for all persons, 83% of the synchronization actions were correctly spotted. The procedure incurred 654 insertion errors (284%) with low inter-person variances. This result should be considered in relation to the recording time (total: 300 minutes), resulting in an average of two insertions per

minute. This high insertion rate was expected due to the simple spotting procedure and the limited model training. Elaborate spotting procedures, automated training, and person adaptation could improve the result. However, this was not the goal of this work. Nevertheless, this result represents a hard, real life test condition for our synchronization algorithm.

Figure 6(a) shows the cumulative error distribution for individual synchronization path lengths. This result was obtained by analyzing the synchronization error for all candidate synchronization paths. The result confirms that with increasing path length k the probability for large alignment errors decreases. When considering the targeted alignment error of 0.3 s, ∼55% of all cases for $k \geq 3$, and ∼90% of all cases for $k \geq 4$ are below this error bar. In contrast, for path length of $k \geq 2$ no useful synchronization performance is obtained. This means that it is very common to find two events in one stream that incidentally have the same distance (time difference) as two events in the paired stream. This observation is related to the relatively high insertion rate for this dataset and the deployed event spotter. However, path lengths of three or four events are sufficient to reach a reasonable synchronization performance. With the simple event spotters considered in this evaluation, we observed that $k = 4$ is needed in many cases.



(a) (b)

Fig. 6. Evaluation 3: Quantitative results of the synchronization. (a): Cumulative distribution results for all candidate synchronization paths and different path length k. (b): Cumulative distribution of synchronization sessions that have at least one alignment estimation result with regard to the synchronization error. The synchronization paths length k was not restricted for this representation.

Furthermore, we analyzed the error distribution with regard to the synchronization sessions that were identified by the event spotting. Figure 6(b) confirms that for more than 80% of the synchronization sessions an synchronization error of less than 0.3 s was achieved.

4 Conclusion and further work

Our experiments clearly show the potential of an automatic, event-based approach to synchronize distributed multi-modal sensor systems. We concluded from the results that our synchronization algorithm can achieve the initially set target of 0.3 s synchronization error. However, the evaluations also showed that in realistic scenarios achieving high synchronization performance is not trivial. In evaluation 3, a synchronization path length of four was required to achieve a performance greater than 80% for the targeted error level. This result indicates that an appropriate event spotting and subsequent reasoning across different synchronization pairs is the key to good performance. In addition, more investigation is needed on suitable synchronization actions for different sensor combinations. Our further work will attempt to include natural activities in the event-based synchronization approach. We expect that this step will help to further improve the synchronization performance.

References

1. 0. Amft and G. Tröster. Recognition of dietary activity events using on-body sensors. *Artificial Intelligence in Medicine*, 42(2):121–136, February 2008.
2. X. Bian, G.D. Abowd, and J.M. Rehg. Using Sound Source Localization in a Home Environment. In *Proc. of The 3rd Intl. Conference on Pervasive Computing*, pages 19–36. Springer, 2005.
3. G. Coulouris, J. Dollimore, and T. Kindberg. *Distributed systems*. Addison-Wesley Reading, Mass, 2001.
4. J. Gupchup, R. Musaloiu-E, A. Szalay, and A. Terzis. Sundial: Using Sunlight to Reconstruct Global Timestamps. In *Proc. of The 6th European Conference on Wireless Sensor Networks*, 2009.
5. J. Lester, B. Hannaford, and G. Borriello. "Are You with Me?"-Using Accelerometers to Determine If Two Devices Are Carried by the Same Person. *Lecture Notes in Computer Science*, pages 33–50, 2004.
6. M. Lukac, P. Davis, R. Clayton, and D. Estrin. Recovering Temporal Integrity with Data Driven Time Synchronization. In *Proc. of The 8th Intl. Symposium on Information Processing in Sensor Networks*, 2009.
7. P. MacWilliams, B. Prasad, M. Khare, and D. Sampath. Source synchronous interface between master and slave using a deskew latch. US Patent 6209072, March 2001.
8. F. Sivrikaya and B. Yener. Time synchronization in sensor networks: a survey. *Network, IEEE*, 18(4):45–50, 2004.
9. W. Su and I.F. Akyildiz. Time-diffusion synchronization protocol for wireless sensor networks. *IEEE/ACM Transactions on Networking*, 13(2):384–397, 2005.
10. B. Sundararaman, U. Buy, and A.D. Kshemkalyani. Clock synchronization for wireless sensor networks: a survey. *Ad Hoc Networks*, 3(3):281–323, 2005.
11. G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh. Fidelity and yield in a volcano monitoring sensor network. In *Proc. of The 7th USENIX Symposium on Operating Systems Design and Implementation*, 2006.