CLUSTERING AND SYNCHRONIZING MULTI-CAMERA VIDEO VIA LANDMARK CROSS-CORRELATION

Nicholas J. Bryan[1⋆] *Paris Smaragdis*[2,3] *Gautham J. Mysore*[3]

[1] Center for Computer Research in Music and Acoustics, Stanford University
[2] University of Illinois at Urbana-Champaign
[3] Advanced Technology Labs, Adobe Systems Inc.

ABSTRACT

We propose a method to both identify and synchronize multi-camera video recordings within a large collection of video and/or audio files. Landmark-based audio fingerprinting is used to match multiple recordings of the same event together and time-synchronize each file within the groups. Compared to prior work, we offer improvements towards event identification and a new synchronization refinement method that resolves inconsistent estimates and allows non-overlapping content to be synchronized within larger groups of recordings. Furthermore, the audio fingerprinting-based synchronization is shown to be equivalent to an efficient and scalable time-difference-of-arrival method using cross-correlation performed on a non-linearly transformed signal.

Index Terms— Video and audio synchronization, audio fingerprinting

1. INTRODUCTION

Through the proliferation of smartphones and low-cost portable electronics, video and audio recording devices have become ubiquitous. As a result, tens, hundreds, or even thousands of people can simultaneously record a single moment in history, creating large collections of unorganized and unprocessed audio and video recordings. To properly playback, edit, and analyze these collections, distinct event identification and time synchronization is required as shown in Fig. 1. Event identification groups or clusters all recordings of the same event together, while synchronization time aligns each video within a cluster. The synchronized recordings of each cluster provide multiple viewing angles and listening perspectives of a single event and collectively create multi-camera video.

In the past, both event identification and synchronization within large video collections have been considered difficult problems, requiring manual labelling and/or specialized hardware. Recent work, however, has shown great promise in using fingerprinting techniques on the audio signals of each video as a solution for both tasks. Presumably the first work towards the synchronization task using audio fingerprints is Shrestha et al. [1], based on the technique of [2]. Similarly, Kennedy and Naaman [3] use fingerprinting for synchronization with a discussion on event identification, but employ the fingerprinting strategy of [4]. Most recently, Cotton and Ellis [5] explicitly discuss audio fingerprinting for event identification, but do not address synchronization. For a general review of audio fingerprinting methods (not applied to video synchronization), please also see [6].

⋆This work was performed while interning at Adobe Systems Inc.

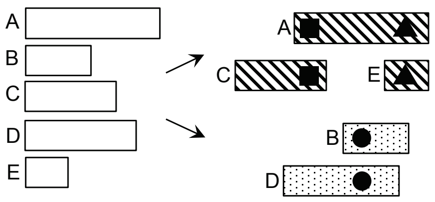


Fig. 1. Clustering and synchronizing an unorganized video/audio collection.

Within this work, we propose a method to both identify and synchronize distinct audio events using audio fingerprinting. The method is based on that of [3] and [4], but offers several improvements. Most notably, further improvements on event identification are discussed as well as a refinement method that resolves any inconsistent synchronization estimates within clusters that also allows non-overlapping content within larger groups to be synchronized. In addition, insight into landmark-based fingerprinting for the purpose of synchronization is presented, showing the procedure to be equivalent to time-difference-of-arrival (TDOA) using cross-correlation on a non-linearly transformed audio signal. The equivalence reformulates the fingerprinting technique into an efficient, scalable, and general TDOA method.

2. PROPOSED METHOD

While the clustering and synchronization method is collectively presented across [3] and [4], we describe the process from an alternative perspective. The modified interpretation allows the landmark feature extraction process and database search structure of [4] to be thought of as an efficient TDOA method outlined in four sections beginning with landmark feature extraction posed as a non-linear transform, followed by TDOA estimation, clustering, and computational issues.

2.1. Non-Linear Transform

The landmark feature extraction converts each audio signal $x(\tilde{t}) \in \mathbb{R}$ into a sparse high-dimensional discrete-time signal denoted by the landmark signal $\mathbf{L}(t)$. The transform first begins by computing the magnitude of the short-time Fourier transform (STFT) for each audio signal, typically downsampling the time axis via the STFT hop-size. Second, the onsets of local frequency peaks are computed from the STFT, resulting in time-indexed frequency values $f^i_{t_j}$, where $i = 1, 2, ..., N$, $j = 1, 2, ..., M$ and N and M are the number of frequency values and time indices respectively.

Third, the time-indexed frequency values are combinatorially paired to nearby values within a limited time-frequency region to create a set of unique time-indexed landmarks, each consisting of two frequencies and the time between (see [4] for more information). When $f^1_{t_1}$ and $f^2_{t_2}$ are paired, for example, the landmark $(f^1_{t_1}, f^2_{t_2}, t_2 - t_1)_{t_1}$ is produced, where the subscript t_1 denotes the start time of the landmark. The combinatorial pairing of the landmarks significantly increases the power of the landmark representation and is advantageous for both the clustering and synchronization. Fig. 2 shows an example spectrogram with a single landmark overlaid.

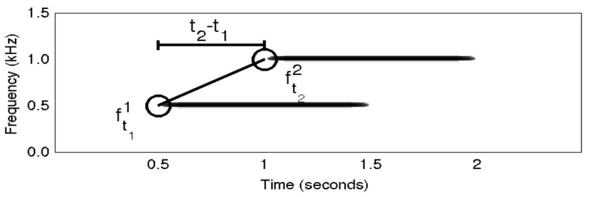


Fig. 2. An example landmark created from a 1kHz tone at starting .5 seconds and a 2kHz tone starting at 1 second.

Finally, each landmark is hashed (or quantized and packed) into a B-bit length integer value h, converting the landmarks to discrete time-indexed features analogous to a words of a text document. The hashes h and time indices t are then used to create the final $N = 2^B$-dimensional landmark signal $\mathbf{L}(t) \in \{0, 1\}^N$ by setting $\mathbf{L}(t, h) = 1$, with $\mathbf{L}$ initialized to all zeros. Typically, B can range from twenty to thirty, creating one million or more possible landmarks.

2.2. Time-Difference-Of-Arrival Estimation

To synchronize each file pair with one another, time-difference-of-arrival estimation (TDOA) via generalized cross-correlation is used. Assuming different recordings of the same event differ only by a time shift and additive noise on the signal $x(\tilde{t})$, the estimated time-difference-of-arrival or time offset between file i and j is computed as the time of the maximum of the cross-correlation signal $R_{\mathbf{L}_i,\mathbf{L}_j}(t)$,

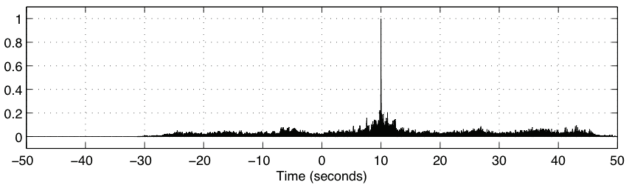
$$\hat{t}_{ij} = \arg\max_t \; R_{\mathbf{L}_i,\mathbf{L}_j}(t), \tag{1}$$

which defines the time shift $\hat{t}_{ij}$ needed to align the two signals appropriately. Instead of performing cross-correlation on the time-domain audio signal $x(\tilde{t})$, however, we perform the process on the derived landmark signal $\mathbf{L}(t)$ with a more efficient computational structure as addressed in Section 2.4. The cross-correlation between $\mathbf{L}_i$ and $\mathbf{L}_j$ for file i and j is defined by

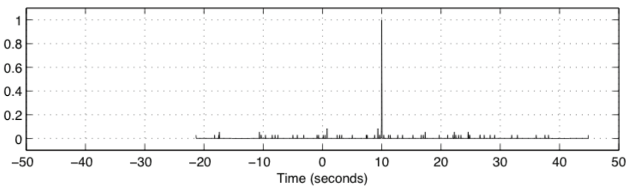
$$R_{\mathbf{L}_i,\mathbf{L}_j}(t) = \sum_{\tau=-\infty}^{\infty} \mathbf{L}_i(\tau)^{\mathrm{T}} \mathbf{L}_j(t + \tau). \tag{2}$$

For a given time τ, the inner product of the two binary vectors gives the number of matching landmarks in both signals. When summed up over all τ, the total number of matching landmarks is computed for a time-shift t between $\mathbf{L}_i$ and $\mathbf{L}_j$.

Fig. 3 shows two cross-correlation signals computed using $x(\tilde{t})$ (with absolute value) and $\mathbf{L}(t)$ between two different 60-second recordings of speech with an offset of 10 seconds. As seen, both correlation signals correctly identify the TDOA of 10 seconds within the time quantization of the STFT hop-size, but are indeed different otherwise. Depending on the required accuracy, the time resolution of the STFT may or may not be sufficient. If needed, a final time-domain cross-correlation post-processing can be computed on a small overlapping region of the two files to refine the time resolution of the landmark correlation with minimal sacrifice to computational cost.



(a) Normalized absolute time-domain cross-correlation.



(b) Normalized landmark cross-correlation.

Fig. 3. Example cross-correlation of speech signals.

2.3. Agglomerative Clustering

To identify distinct events within a larger collection, we use agglomerative clustering based on the landmark cross-correlation signals for each file pair combination. To do so, each recording or audio file is initialized as a separate cluster and then merged into successively larger clusters representing the different events of the larger collection. The two clusters of Fig. 1, for example, are defined by a merge between file A to C, A to E, and B to D.

The most straightforward way to decide if two files should be merged together is to use the maximum of the correlation

$$\hat{R}_{\mathbf{L}_i,\mathbf{L}_j} = \max_t \; R_{\mathbf{L}_i,\mathbf{L}_j}(t) \tag{3}$$

as a confidence score and compare it to a minimum threshold θ. If $\hat{R}_{\mathbf{L}_i,\mathbf{L}_j} \geq \theta$, a merge is accepted, otherwise it is rejected. For the task of recognizing different recordings of the same event across unseen datasets, however, more robust solutions are required to minimize the occurrence of false merges (or false positives), particularly when processing varying length files.

To improve this threshold-based decision rule, we keep track of the specific landmarks (or non-zero elements of $\mathbf{L}(t)$) that caused each TDOA estimate and compute various statistics on this set of landmarks to better inform the merge decisions and remove false-positive merges. Three useful additional decision rules include:

1. Reject merges with a small percentage of total matching landmarks (in both files) in the overlap region $\hat{o}$.
2. Reject merges with a small overall time range $\hat{r}$ defined by the set of matching landmarks.
3. Reject merges with a small overlap region $\hat{o}$.

Rejecting matches based on the percentage of total matching landmarks helps remove issues due to varying file lengths. In Fig. 4, the percentage of matching landmarks within the

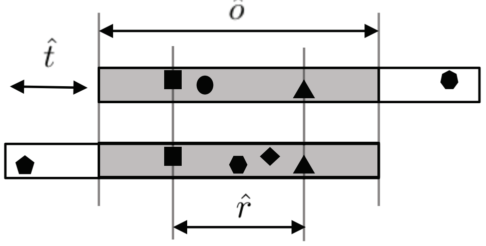


Fig. 4. Two different recordings of the same event.

top file is 66%, while the percentage of matching landmarks within the bottom file is 50%. Rejecting matches with a small time range, defined by the set of matching landmarks, eliminate merges resulting from densely packed landmarks in a small time region and no where else. Finally, rejecting matches with improbably small overlap regions can help further filter out erroneous matches. Although not discussed further, the frequency of matching landmarks over time and/or adaptive thresholds ([3]) on $R_{\mathbf{L}_i,\mathbf{L}_j}$ can also be used.

2.4. Efficient Computation

To cluster and synchronize a collection of P audio files, each pairwise cross-correlation is needed. When performed on the time-domain audio signal $x(\tilde{t})$, each correlation traditionally requires $O(N^2)$ operations or $O(N \log N)$ operations for fast FFT-based correlation, where N is equal to the file length. For large P, this quickly becomes burdensome and motivates landmark-based correlation. As a result, an alternative algorithm is used for computation.

To correlate two landmark signals and leverage the sparse discrete nature of $\mathbf{L}$, a hash table or map structure is created associating each non-zero landmark (map key) to a vector of tuples (map value), where each tuple stores the time of occurrence and file id (t, id) of its respective landmark. Once the map structure is created, we iterate over all keys of the map and find all values that have multiple unique file ids. These values are then used to compute time difference estimates between the given two files, which are then summed into the appropriate position of $R_{\mathbf{L}_i,\mathbf{L}_j}$ creating a non-negative discrete-time signal. The map structure computation allows time differences to be computed only for matching landmarks between files and reduces the number of operations for cross-correlation to approximately $O(N)$, where N is the number of unique matching landmarks between the two files (typically 10-100), plus the cost of building the map structure and linear time of feature extraction.

The significant savings is particularly noticeable when synchronizing and clustering large file collections with a small number of matching recordings per event. In this case, only matching landmarks will be found for files of the same event and little to no matching landmarks will be found for files of different events. In doing so, the process only computes cross-correlation signals between files of the same cluster and ignores all other irrelevant pairwise combinations, combinatorial reducing the computation.

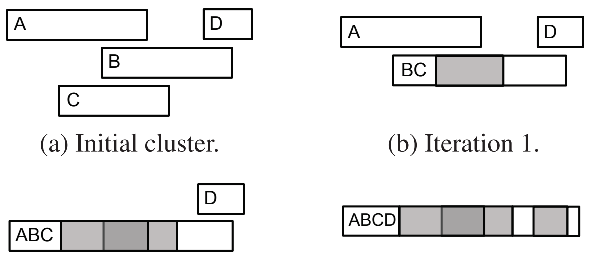
3. SYNCHRONIZATION REFINEMENT

Once initial synchronization and clustering is computed, synchronization refinement may be needed. Refinement is required for clusters of three or more in two cases: when inconsistent pairwise TDOA estimates do not satisfy all triangle equalities (e.g. $\hat{t}_{AC} \neq \hat{t}_{AB} + \hat{t}_{BC}$) of the cluster and/or when one or more TDOA estimates within any cluster is unknown caused non-overlapping. Both scenarios are common and must be addressed to adequately align all recordings within each cluster to a common time clock.

To appropriately handle these situations, we can refine the synchronization estimates within a cluster using the following "match-and-merge" algorithm and iteratively time-align all files within a cluster to a common clock:

1. Find the most confident TDOA estimate $\hat{t}_{ij}$ within the cluster in terms of $\hat{R}_{\mathbf{L}_i,\mathbf{L}_j}$ or similar confidence score.
2. Merge the landmark signals $\mathbf{L}_i(t)$ and $\mathbf{L}_j(t)$. First time shift $\mathbf{L}_j(t)$ by $\hat{t}_{ij}$ and then multiply or add the two signals together (depending on the desired effect).
3. Update the remaining TDOA estimates and confidence scores to respect the file merge.
4. Repeat until all files within the cluster are merged.

Step 2 and 3 can be either accomplished by recomputing the necessary cross-correlation signals and TDOA estimates or by simply time shifting the TDOA estimates and assuming the confidence scores will remain the same throughout the process. In either case, the match-and-merge process requires minimal computation and successively eliminates any inconsistent TDOA estimates or issues of non-overlapping files. In addition, no "master" reference recording is needed. In Fig. 5(a), we see four example recordings with various configurations of overlap. Over three iterative steps, each



(a) Initial cluster. (b) Iteration 1.

(c) Iteration 2. (d) Iteration 3.

Fig. 5. Match-and-merge synchronization refinement.

recordings of Fig. 5 (a) is gradually merged together and time aligned to a single reference clock as desired.

4. EVALUATION

Precision, recall, and the F_1-score is used to evaluate the pairwise merges of the clustering task, while manual listening tests were used to evaluate synchronization. The precision is the fraction of estimated merges retrieved that are correct compared to ground-truth, while recall is the fraction of correct merges retrieved. The F_1-score is the harmonic mean of the precision and recall. In addition, total compute time (seconds) and throughput (seconds of audio processed per second of computed time) are used to evaluate computational cost.

Datasets of both speech and music recordings were used for testing with hand labeled ground-truth. The *speech* dataset consists of 180 natural speech recordings taken from a film set with two separate recording devices. The recordings average 20-40 seconds in length and have 114 clusters: 54 clusters of one file, 54 clusters of two files, and 6 clusters of three files. The *music* dataset consists of 23 cell-phone recordings of three live music concerts of various styles, each averaging 3-6 minutes in length. In the music dataset, there is 1 cluster of 7 files, and 2 clusters of 8 files. Prior to computation, all recordings were time normalized to a sample rate of 8kHz.

The results are shown in Table 1[1]. Near perfect precision, recall, and F_1-score was achieved and all files were verified to be correctly synchronized. In terms of computation time, all datasets were clustered and synchronized in only a minute or two with high throughput compared to performing traditional FFT-based correlation on all pairwise file combinations. Also note the approximate linearity of the computation time of the proposed approach when processing both datasets independently versus the combined speech and music dataset, which is not present otherwise.

[1]Computed with Matlab and C++ code on 2.2 GHz Intel i7 MacBook Pro.

	Speech	**Music**	**Speech + Music**
Precision	100.0 %	100.0 %	100.0 %
Recall	97.0 %	100.0 %	99.2 %
F-Score	98.5 %	100.0 %	99.6 %

(a) Precision, recall, and F_1-scores.

	Speech	**Music**	**Speech + Music**
Proposed	47.0 / 164.6	41.1 / 146.5	90.1 / 152.7
Traditional	1550 / 5.0	197 / 30.5	3600 / 3.9

(b) Computation time (s) and throughput (s/s).

Table 1. Evaluation results.

5. CONCLUSIONS

We have presented a method to both identify and synchronize distinct audio events in time using landmark-based audio fingerprinting. Several improvements to prior work are discussed including further work on event identification and a synchronization refinement method that ameliorates inconsistent TDOA estimates that also allows non-overlapping content to be synchronized within larger groups of files. Furthermore, the audio fingerprinting-based technique used was shown to be equivalent to a time-difference-of-arrival (TDOA) method performed on a non-linearly transformed audio signal within the framework of cross-correlation, opening up the efficient and scalable synchronization method to general TDOA estimation problems. Evaluation demonstrated near perfect clustering and synchronization results in an efficient manner.

6. REFERENCES

[1] P. Shrestha, M. Barbieri, and H. Weda, "Synchronization of multi-camera video recordings based on audio," in *Proc. 15th Intl. Conf. on Multimedia*, 2007.

[2] Jaap Haitsma and Ton Kalker, "A Highly Robust Audio Fingerprinting System With an Efficient Search Strategy," *Journal of New Music Research*, vol. 32, no. 2, 2003.

[3] L. Kennedy and M. Naaman, "Less talk, more rock: automated organization of community-contributed collections of concert videos," in *Proc. 18th Int. Conf. on World Wide Web*, 2009.

[4] A.L. Wang, "An Industrial-Strength Audio Search Algorithm," in *Proc. 4th Int. Symposium on Music Information Retrieval (ISMIR)*, October 2003.

[5] C.V. Cotton and D.P.W. Ellis, "Audio fingerprinting to identify multiple videos of an event," in *Proc. ICASSP, 2010*, March 2010.

[6] Pedro Cano, Eloi Batlle, Ton Kalker, and Jaap Haitsma, "A review of audio fingerprinting," *J. VLSI Signal Process. Syst.*, vol. 41, pp. 271–284, November 2005.