LiveSync: a Method for Real Time Video Streaming Synchronization from Independent Sources

Marcello N. de Amorim, Ricardo M. C. Segundo, Celso A. S. Santos

Federal University of Espírito Santo, Brazil
novaes@inf.ufes.br , rmcs87@gmail.com , saibel@inf.ufes.br

Abstract. This work features the LiveSync, a method that allows synchronization of live video streams from multiple sources, such as YouTube, Vimeo, and other video streaming platforms. The proposed method aims to cover scenarios where automatic techniques face difficulties to provide correct video synchronization: multiple users capturing videos on the same event, and streaming them through different platforms at live events. LiveSync uses human perception and judgment to determine synchronization points between video streams related to the same event. A human can determine the time delay between a pair of videos captured separately and put them together. Moreover, if multiple user contributions in this issue are properly aggregated, a set of video streams can be efficiently synchronized. The outcome of this cooperative process is the temporal alignment of video streams, allowing the generation of synchronized presentations. This work also presents the Dynamic Alignment List, that is an abstract datatype defined to store and to manage human contributions, as well to perform relevant operations over these data, such as to infer additional synchronization points by using transitivity properties.

Categories and Subject Descriptors: H.4.4.3 [**Information Systems Applications**]: Crowdsourcing; H.3.2.7 [**Human-Centered Computing**]: Synchronous editors

Keywords: crowdsourcing, live video, synchronization

1. INTRODUCTION

Live streaming of user-generated videos (UGV) continuously grows on the Internet, both in numbers and relevance boosted by platforms such as Facebook, Youtube, and Vimeo. In this scenario, it is relevant to find out methods capable of synchronizing them in a way that a presentation of multiple streaming can be presented: as in a video mosaic, or with audio and camera personalization (the spectator chooses which video and camera he wants to watch together).

Automatic methods are widely used to reach video synchronization. Although, they demand a vast training database input as well as controlled conditions, such as standardized structure and marks to work properly [Shrestha et al. 2010; Wang et al. 2014; Bohez et al. 2016]. These methods usually present good results synchronizing well-structured video productions, professional coverage for sports events and other modalities of planned videos. However, they face challenges attempting to synchronize heterogeneous videos in situations where they must deal with wide baselines, camera motion, dynamic backgrounds, and occlusions [Schweiger et al. 2013]. When we specifically focus on live transmission scenario, even more restrictions are imposed, and few works approach this restriction. Common solutions use Inter-Destination Video Synchronization (IDMS) [Montagud et al. 2012], that relies on synchronization protocol within clients, capturing devices and servers. In our scope, however, we consider heterogeneous devices and heterogeneous platforms that don't use these protocols, creating the demand for an innovative approach for UGV live-streaming video synchronization.

The authors would like to thank FAPES, CAPES and CNPq for financial support of this research.

UGVs are produced from the user's point of view, so different users covering the same event tends to result in heterogeneous video streams, with different angles, qualities, audio content, and other characteristics that make automatic methods unappropriated. To achieve synchronization for UGV live streaming, this work introduces the LiveSync, a method that explores the human ability to associate heterogeneous videos using their senses (vision and hearing). LiveSync is based on the Human Computation paradigm [Von Ahn 2005], using human intelligence to perform tasks usually hard to machines but easy to people. In specific, we use a crowdsourcing approach, where a problem is divided into tasks, and the execution of these tasks is delegated to the crowd composed of individuals engaged in the solution process. The partial result delivered by each contribution is registered, and an outcome is generated by processing and aggregating these contributions.

Using a crowdsourcing approach to solve the user-generated live video streaming synchronization problem, we present a divide-and-conquer method. The group of N videos to be synchronized is divided into N/2 pairs, which are sent to cooperators. Each crowd member must provide a unique synchronization point for each pair of videos. These synchronization points are aggregated and used to determine time relations among all videos streams, making possible their synchronized presentation.

This work extends a previous work: "LiveSync: a tool for Real-Time Video Streaming Synchronization from Independent Sources" [de Amorim et al. 2017] that has been honored with the best paper award of the XVI Workshop of Tools and Applications (WFA) of the XXII Brazilian Symposium on Multimedia and Web Systems (WebMedia 2016), with a major focus on method instead of the tool's implementation. It is organized as follows: Section 2 describes the method for LiveSync, Section 3 details Dynamic Alignment List, Section 4 presents the implementation of the method through the LiveSync Tool and finally, Section 5 concludes with some remarks about this work and further research.

2. LIVESYNC

Live synchronization includes scenarios where a spectator has access to an event that is live streamed by more than one source. These sources are independent, so their videos do not have initial resources that allow their automatic synchronization, requiring a real-time video analysis to generate synchronization points [Segundo and Santos 2015]. An example is the coverage of a sporting event, such as a soccer match. In the event, multiple people can take their cell phones and start streaming it. At home, other people can watch these videos. However, the videos from various sources won't be presented synchronized if placed together. It is necessary a form to synchronize these UGVs. In this sense, we use the crowd to achieve it. The correlated videos are grouped in a MashUp application that connects users and video sources. This MashUp application contains all necessary functionalities allowing synchronization and presentation of the multiple videos. This way, spectators can be used to synchronize videos they are watching, contributing to enhance the overall experience of all spectators.

In a live presentation, it is assumed that the content must be consumed right after its generation, or it would not be a live content. In this case, it is of high importance that the synchronization method can be performed in presentation time, allowing the integration of live contents. The way synchronization is performed in live video poses a single requirement: there is no need to analyze an entire video to find synchronization points, only an estimated stream delay must be considered. This happens because as the event is happening in real-time (live), sources are also live and, therefore, the lack of synchronization during playback is caused by video streaming delays. In this specific case, synchronization through the crowd may be achieved by using tools that allow a user to manipulate time from videos: he can, for instance, keep pausing videos alternately, until he finds them synchronized. This is a consequence of having a reduced search space when limited to stream delays between videos.

The process to live synchronization works as follows: (i) multiple live sources for the video coverage of an event are created and made available online; (ii) spectators start to watch the videos; (iii) a spectator selects and synchronizes two videos with the help of a manipulation tool producing a relation time (Δ) between videos; (iv) Δ become a candidate synchronization point; (v) other spectators evaluate the result from (iv), and if considered correct, the synchronization information is shared with other spectators interested in watching the same group of videos.

As an example, consider two independent sources that are providing video streams of a live event. A MashUp application allows spectators to watch both videos at the same time in his device. However, the videos are not synchronized and the user perceives this problem. He accesses the application option to synchronize the videos. After he achieves a synchronous result, his contribution is sent to the application server (Coupler). This contribution is time difference between videos. The server will feed other users that choose to watch the same videos with this information included, to provide a synchronized presentation of the video MashUp. If the user thinks the content is not synchronized yet, he can adjust it himself and send another contribution to refining the synchronization information. Note that not all spectators are required to collaborate to achieve synchronization. If a synchronization made by a single member of the crowd is accepted as accurate, it can be transferred to the remaining viewers, this way each one will have his content locally synchronized.

Fig. 1 presents an overview of the LiveSync main components: Video Sources, Coupler and MashUp Player. The sources are live video platforms such as YouTube and Twitch. Video streams are transmitted from the source to the MashUp Player. When required, additional information such as the uptime is also sent from the source the player. The MashUp Player runs as a client web application in the user browser. The user access the web page and the application is loaded on their device. The application consults the Coupler to receive the videos paths and synchronization information. If the user acts as a worker and synchronizes a video, he sends the synchronization information back to the Coupler that updates its relations. Lastly, the Coupler runs as a service, waiting for requisitions and workers' contributions. In Fig. 1, the communication among components used in our implementation are: (1) a communication API provided by the Sources, where only the player receives information; and (2) WebSocket technology, where that information flow can be sent from/to the Coupler.

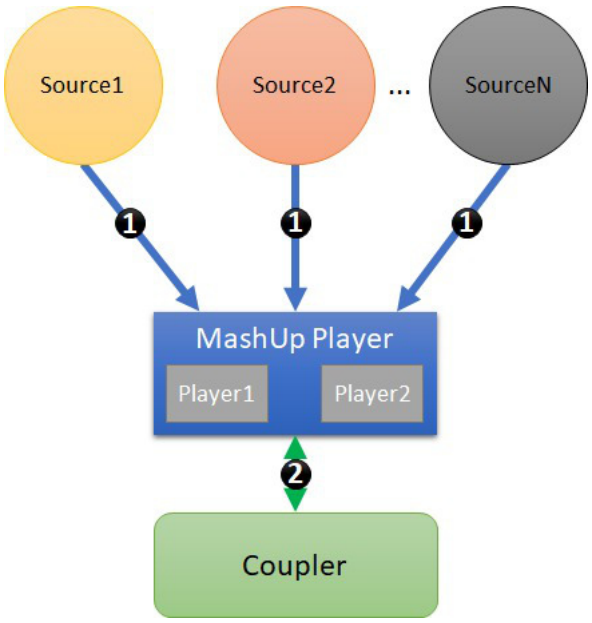

Fig. 1. LiveSync Component Communications

2.1 Video Sources

Video Sources are third-party video streaming platforms, such as YouTube Live, LiveStream, TwitCast, Twitch and Ustream. Each platform has its own protocols and video formats, making difficult the integration among them. However, we made it possible using an application in the middle (MashUp player). Making multiple platforms available maximizes the number of video streams that can be related to an event transmission.

2.2 MashUp Player

MashUps are applications generated by the combination of content, presentation or other applications functionalities from disparate sources. They aim to combine these sources to create useful new applications or services to users. The LiveSync combines videos from different sources with synchronization information from the Coupler to reproduce a synchronous presentation of these videos.

The MashUp Player is used to both presenting video synchronously and collecting the synchronization values. Using the Mashup player, spectators can:

—Select videos they wish to watch synchronously;

—Synchronize videos that have no synchronization information;

—Refine synchronization that is not "good enough"; and

—Add new video sources to the event coverage.

2.3 Coupler

The Coupler is responsible for storage, distribution, and calculation of synchronization points among video streams from the different sources. It stores all the synchronization information about the live event, so its stance is finished with the end of the videos and all data are saved to be used in a possible synchronized recorded version of the event. In our scope, however, the synchronization information is only necessary during the event, after it, there is no need to keep it.

Another aspect of the Coupler is the distribution of synchronization points (couplers) among pairs of video streams. When a user selects a pair of streams to watch, a message is sent from the MashUp to the Coupler, containing the required temporal relation. The Coupler answers it with the required information. If the relationship is unknown, it responds with a requisition to that viewer synchronize the pair and fill that gap.

The last functionality of the Coupler is to calculate the synchronization points between two videos streams (for example, A and B). Each relation ($\Delta_{A,B}$) in the Coupler may result from several contributions from multiple spectators that tried to synchronize the videos A and B. Thus, it is necessary to calculate an aggregated value of $\Delta_{A,B}$ based on those contributions. Here, multiple aggregation techniques may be used, such as Majority Decision, Expectation Maximization or Geometric Mean [Segundo et al. 2017; Hung et al. 2013].

All Coupler's functionalities are provided by the Dynamic Alignment List (DAL), an abstract datatype that is the core of the Coupler and is presented in the following section.

3. DYNAMIC ALIGNED LIST (DAL)

The Dynamic Alignment List (DAL) is an abstract datatype designed to manage the temporal relations (or simply couplers, according [Segundo and Santos 2015]) between videos as well as to provide the features required to store and process these relations. Each relation stored into a DAL instance corresponds to a coupler related with a pair A and B of video streams and the $\Delta_{A,B}$ between them.

The DAL is derived from a synchronization matrix MxM which can represent all possible relations between M videos, with each position representing the difference time ($\Delta_{A,B}$) calculated from the difference between the initial instants of a pair of videos A and B, considering a common clock. This matrix can be reduced to an upper triangular matrix, once that $\forall i, j < M, \Delta_{i,j} = -\Delta_{j,i}$ and the main diagonal is always equal to zero. The $\Delta i, j$ for each pair of videos is calculated by: $\Delta_{i,j} = begin(j) - begin(i)$, where $begin(X)$ returns starting time of the event transmission (uptime).

However, for our live scenario, $\Delta_{i,j}$ needs to be adapted once the clocks of i and j are not synchronized. Even if both transmissions started at the same time, their starting times may show different values due to the fact that their clocks are not synchronized. There is the need to find a common clock for both, so we can store only their real difference at a DAL, not the sources clock difference [Segundo and Santos 2015].

The common clock used is the client one. When synchronizing both videos, the cooperator (i.e., a worker in crowdsourcing jargon) finds the synchronization point that aligns both videos under his clock, getting the current event time being presented in his players. With the difference of the videos under the worker clock, we can calculate the difference between the uptime of both sources, mapping the local value with the uptime value, that gives us $\Delta_{i,j}$.

$\Delta_{i,j} = current(i) - current(j) + begin(i) - begin(j)$

In the other hand, a user that receives $\Delta_{i,j}$, can synchronize videos by making:

$current(i) - current(j) = \Delta_{i,j} - (begin(i) - begin(j))$

This is done by stopping one of the videos. With one of the videos stopped, the difference between them will change, until the equality becomes true and both videos are synchronized.

If $\Delta i, j > 0$, the video i starts before the video j; if $\Delta i, j < 0$, the video i starts after the video j; and when $\Delta i, j = 0$, both videos start at the same time. The values are represented in milliseconds, so $\Delta i, j = 30$ means that the video j should start 30ms after the video i starts in order to achieve synchronization. Additionally, in cases which are impossible to determine a relation between a pair of videos, the value registered is I, for indeterminate. Impossible cases are those where the workers couldn't find a synchronization point. An example when this can occur is two videos from different events that were incorrectly added to the same event. No spectator will be able to synchronize them as there is no synchronization point between the video contents.

Into a cooperative scenario where users would provide a value of $\Delta i, j$ for a pair of videos (i, j), it is important to store all contributions, as $\Delta i, j$ should be expressed considering the contributions made by all the workers for that pair. In this approach DAL stores a consensus value that is determined by processing these values. Moreover, the number of contributions for each pair can grow while the contribution process is active, and current $\Delta i, j$ will be updated whenever new contributions arrive.

A matrix, however, is not suitable to handle all these features. It is necessary to develop a datatype to both store data and implement functions that allow the analysis of these data. A DAL preserves the relational characteristics with an additional dimension related to the contributions for each pair of videos. Thus, it is implemented as a hash table, using linked lists hierarchically organized in three levels.

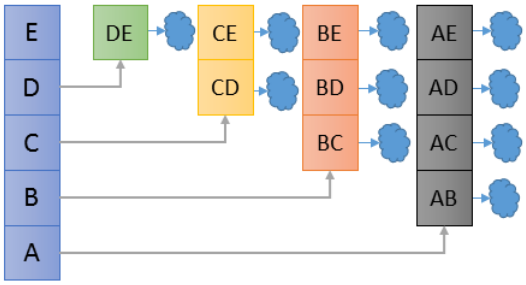

Fig. 2. DAL with 5 Videos (A,B,C,D,E)

—Level 1 - The first level is a list of video structures. Each structure has a unique identification label for the correspondent video;
—Level 2 - Each video structure points to a second level linked list composed by all possible relations that include its video. Thus, each node in a second level linked list corresponds to a relation between a video represented in its root and another video.
—Level 3 - Each relation in a level 2 list points to a third level list, in which each node represents a contribution for that relation.

Fig. 2 illustrates a DAL configuration with 5 video objects. The videos, labeled as A, B, C, D, and E, are in the first level linked list. Each video structure points to a second level list. It is possible to observe in Fig. 2 that each second level linked list has only the relations that would exist in a single line of an upper triangular matrix without the main diagonal. Moreover, in Figure 2 each relation, that is an element of the second level list, points to a cloud that represents a third level list of all contributions obtained in the cooperative process for that relation.

A complete DAL can provide all information needed to achieve synchronization for videos registered on it, although it is more than a data structure. A DAL is an abstract datatype that provides the data structure plus a set of features that allows access, manage and process of the information inside it, such as an infer synchronization function [Segundo et al. 2017]. Inferring synchronization, we can obtain additional relations by executing an inference algorithm over the current relations. This algorithm checks if it is possible to find an indirect path between two videos, by transitivity. For example, considering the videos A, B and C, if the relations AB and BC are known, by transitivity it is possible to infer the relation AC. This feature can reduce the number of contributions required to find all DAL relations [Segundo et al. 2017].

4. THE LIVESYNC TOOL

The LiveSync Tool is a Web implementation of LiveSync method. It provides all components required to proceed UGV live stream synchronization following this method. Moreover, LiveSync Tool also uses the DAL to deal with the user contributions related to the MashUp generation of synchronized video streams. The main features provided by the LiveSync Tool are:

Synchronized Live Video Player. the tool permits users to watch multiple videos from multiple sources in a synchronized presentation. The user selects from a list of sources the videos he wishes to watch and then they are synchronized using information provided by other users stored in the Coupler. The Live Video Player is able to receive synchronization information and present both videos synchronously. If during the presentation a video loses synchronization due buffering or loss of frames, the player can act and re-synchronize;

Video Synchronization. if a pair of videos does not have any information about their synchronization, users are invited to contribute and synchronize the videos;

Video Aggregation. although the focus of the LiveSync Tool is UGV live streams synchronization, it must allow users to add new stream sources to the application. Users only need to set the video source, and the video will be added to the DAL and to the list of available videos;

Multiple Platforms Support. one key point on this tool is to use other platforms as video sources. Videos presented to users, and synchronized by them, are provided by external live video stream platforms. To be compatible with this instance of the LiveSync Tool, two requisites are necessary:

(1) Remote Player: the platform must allow embeddability of their players on third-party pages, allowing the control of the player with its basic functionalities such as play, pause and stop by our MashUp player. It is used both to present the videos and synchronize them;
(2) Uptime Support: a second and fundamental requisite is an API that allows video uptime retrieval. Video uptime is the elapsed time since the beginning of the video presentation. This is fundamental to create and replicate the temporal couplers used to play the synchronized video streams.

Serverless Architecture. In serverless architecture approach, applications that significantly depend on third-party functions and services, putting much of the application behavior and logic on the front end. Such architectures remove dependencies of traditional server system sitting behind an application [Roberts 2016];

Multiplatform:. LiveSync Tool is a Web Based application designed and developed with the HTML5 standard as front-end (MashUp Player) component. It allows this application to be executed on multiple browsers, operational systems and devices;

Active vs Passive Contributions. it defines which pair of videos should be synchronized by each user. Two different roles can be configured: (i) active role allows users to navigate freely through the videos, synchronizing them when they wish to; (ii) passive role uses an automatic algorithm to ask users which pair of videos they should synchronize.

We successfully tested LiveSync's interoperability with three Video Source platforms: YouTube, Twitch and using WebSocket transmission (`https://goo.gl/KOIzV8`). This means that we could synchronize video streams being transmitted by these three video sources. They filled the two requirements that a video source should attend: Remote Players and Uptime Support. As each stream platform uses its own protocols, the embedded players were required for aggregation in the MashUp application. These players provided the play, pause and stop actions. Uptime Support was necessary to find the temporal couplers to video streams.

The Coupler provides a DAL instance and a storage service. The storage service is used to track all contributions made by the crowd, an important aspect of crowdsourcing support. The communication between Coupler and the MashUp application is made through WebSocket communication. The MashUp application creates a WebSocket channel to the Coupler, and requests the sync information or sends contributions from the crowd. A JSON based protocol is used for messages exchange after WebSocket connection: *action:value, data:object*. The first one contains the action to be performed and the second one, an object to complement this action.

The MashUp application was developed following HTML5 standards to facilitate cross-platform execution. Using the application, the user can either play or synchronize videos. Following the orientations on the screen, a user can select which videos he wants to watch, add new videos, and start the synchronization process for videos which he believes aren't synchronized. It is transparent to users where the videos are coming from. Figure 3 presents video mosaics examples created after the synchronization process applying the LiveSync. Both mosaics present multiple videos from different cameras synchronized after the synchronization process. In both cases, the videos were previously recorded and a multiple stream transmission has been simulated to test the synchronization process and presentation of the MashUp Player.

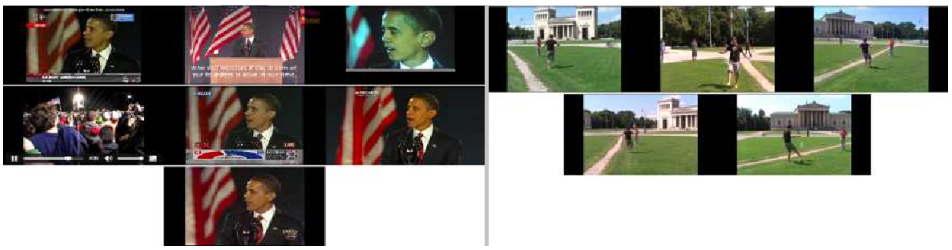

Fig. 3. Synchronized Mosaics: on the left a mosaic from Obamas' 2008 winning speech; and on the right a soccer video dataset from (http://www.lmt.ei.tum.de)

When a user chooses to synchronize a pair of videos, the MashUp display is reconfigured and the application enters in the synchronization mode. LiveSync Tool uses a Play and Pause approach to synchronize the videos [de Amorim et al. 2017]. In this approach, the user controls both players through play and pause options as can be seen in Fig. 4. The user is supposed to pause the video that is ahead of the other and to play it back when the second video reaches the paused one. The process continues until the lack of synchronization be not noticeable by the user.

Fig. 4 shows the interface of the MashUp during a test: two cameras live streaming a simulated television event to our MashUp application. Once the user decides that both videos are synchronized he clicks on the *DONE* button, so his contribution is committed to the Coupler that stores it into the DAL for further processing of the relation.

The demonstration of LiveSync tool can be seen in `https://goo.gl/bpC1V9`. This video shows how two live streaming videos from a lab experiment are synchronized: videos are added; synchronized; and presented to users.

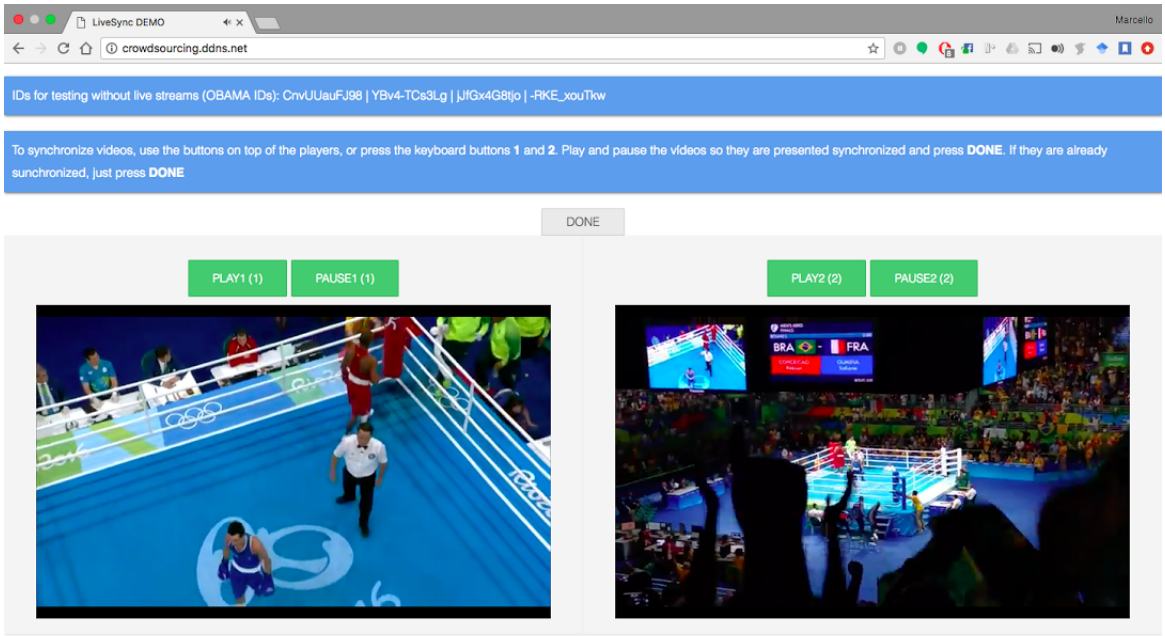

Fig. 4. Live Streams from Olympic Games Synchronized through two different cameras

5. FINAL REMARKS

LiveSync is a method that aims at achieving UGV live streams synchronization. The focus of this method is covering the scenarios where automatic techniques may face problems to work properly. The approach adopted in LiveSync is to use the work-power of a crowd of contributors to execute small tasks, that are used to determine the synchronization points among videos. This approach assumes that human perception is better than automatic techniques to synchronize heterogenous and non-standardized UGV. The LiveSync Tool is a Web implementation for LiveSync that allows users to contribute in UGV live streaming synchronization process, as well to watch synchronized UGV live streams from multiple sources.

The current work has limitations that shall be addressed in future works. It is unknown the impact of a greater number of videos on the platform, as no scalability tests were made [Segundo et al. 2017]. However, we believe that no major problems in terms of computation will arise. This is justified by the fact that the videos aren't streamed through the platform, but directly with the sources that already are scalable. However, more videos would probably raise the problem of recovering videos from different synchronized events, as the current interface doesn't implement the required features to an easy recovery of information.

As a future work we also plan to address usability issues of the implementation. The current work assessed only if it was possible the synchronization of real-time video streams from different sources, though, we didn't consider if the tools to perform the proposed task were easy to the users.

REFERENCES

Bohez, S., Daneels, G., Van Herzeele, L., Van Kets, N., Decrock, S., De Geyter, M., Van Wallendael, G., Lambert, P., Dhoedt, B., Simoens, P., et al. The crowd as a cameraman: on-stage display of crowdsourced mobile video at large-scale events. *Multimedia Tools and Applications* 1 (1): 1–33, 2016.

de Amorim, M. N., Segundo, R. M., and Santos, C. A. Livesync: a tool for real time video streaming synchronization from independent sources. *Workshop de Ferramentas e Aplicações - Simpósio Brasileiro de Sistemas Multimídia e Web* 1 (15): 112 – 115, 2017.

Hung, N. Q. V., Tam, N. T., Lam, N. T., and Aberer, K. An evaluation of aggregation techniques in crowdsourcing. In *Proceedings of the International Conference on Web Information Systems Engineering*. Berlin, Heidelberg, pp. 1–15, 2013.

Montagud, M., Boronat, F., Stokking, H., and van Brandenburg, R. Inter-destination multimedia synchronization: schemes, use cases and standardization. *Multimedia systems* 18 (6): 459–482, 2012.

Roberts, M. Serverless architectures. http://martinfowler.com/articles/serverless.html, 2016.

Schweiger, F. et al. Fully automatic and frame-accurate video synchronization using bitrate sequences. *Multimedia, IEEE Transactions on* 15 (1): 1–14, 2013.

Segundo, R. and Santos, C. Remote temporal couplers for multiple content synchronization. In *Computer and Information Technology, 2015 IEEE International Conference on*. Liverpool, UK, pp. 532–539, 2015.

Segundo, R. M. C., de Amorim, M. N., and Santos, C. A. S. Crowdsync: User generated videos synchronization using crowdsourcing. In *2017 International Conference on Systems, Signals and Image Processing (IWSSIP)*. Poznan, Poland, pp. 1–5, 2017.

Shrestha, P., de With, P. H., Weda, H., Barbieri, M., and Aarts, E. H. Automatic mashup generation from multiple-camera concert recordings. In *Proceedings of the 18th ACM International Conference on Multimedia*. New York, NY, USA, pp. 541–550, 2010.

Von Ahn, L. *Human Computation*. Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 2005.

Wang, O. et al. Videosnapping: Interactive synchronization of multiple videos. *ACM Transactions on Graphics (TOG)* 33 (4): 77, 2014.