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

Ricardo M. C. Segundo[1], Marcello N. de Amorim[2], Celso A. S. Santos[3]

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

Abstract. This work presents a method that allows users to synchronize live video streams from multiple sources, such as YouTube and several other video streaming sources, as well as a web application for ease its use. The proposed method for pursuing multi-camera video streaming synchronization is based on a crowdsourcing approach, using the human processing-power of a crowd of contributors to synchronize videos, requiring each user to find only one synchronization point for a single couple of video streams. Additional synchronization is inferred from the processed contributions, using transitivity properties and an appropriate structure for this inference, the Dynamic Alignment List.

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

Keywords: live video, synchronization, crowdsourcing

1. INTRODUCTION

This paper extends the previous work with same title [de Amorim et al.] presented in the XXII Brazilian Symposium on Multimedia and Web Systems (WebMedia 2016) where it was honoured with the Best Paper Award at XVI Workshop of Tools and Applications (WFA).

Multiple camera synchronization is an ubiquitous theme within the multimedia area, commonly related to automatic video synchronization (AVS) methods [Wang et al. 2014]. Although, these automatic methods generally demands vast example databases as well controlled conditions, characteristics and marks to work properly. These characteristics make AVS suitable to synchronize well-structured video productions, professional coverage for sport events and other modalities of planed videos, but they tends to face challenges attempting to synchronize heterogenous videos in situations where have to deal with wide baselines, camera motion, dynamic backgrounds and occlusion [Schweiger et al. 2013].

Into a scenario in which user generated videos (UGV) live streaming continuously grows in number end relevance boosted by platforms such as Facebook, Youtube and Vimeo, it is relevant find out methods capable to synchronize them. UGV are generated over the user's point of view, so different users covering a same event tends to result in heterogeneous video streams, in different angles, qualities, audio content and other characteristics that make AVS unappropriated to synchronize these sources.

To achieve UGV live streaming synchronization this work introduces LiveSync, a method that explore the human ability of associate heterogeneous videos, to synchronize UGV live streams from different sources. LiveSync in based on the Human Computation (HC) paragidm [Von Ahn 2005], using human intelligence to execute tasks usually hard to machines but easy to ordinary people.

The authors would like to thank the "Fundação de Amparo à Pesquisa e Inovação do Espírito Santo" (FAPES) and "Coordenação de Aperfeiçoamento de Pessoal de Nível Superior" (CAPES) for financial support.

Additionaly HC can improve performance by division of labor because it helps to define tasks that can be executed in parallel [Rohwer 2010]. This characteristic is potentialized in LiveSync by adopting a crowdsourcing [Howe 2006] approach to using efficiently the processing power of a crowd of collaborators, collecting their contributions in parallel. Each collaborator can contribute by simply finding a synchronization point between a pair of video streams, following a crowdsourcing strategy that consists in to offer workers tasks that can be completed easily and quickly [Difallah et al. 2015].

This paper is organized as follows: section 2 describes LiveSync method, section 3 details LiveSync Tool, section 4 presents an practical experiment to validate LiveSync and LiveSync Tool. Finally, section 5 concludes with some remarks about this work and further research.

2. LIVESYNC

LiveSync is an method to achieve video synchronization for live streams from multiply sources. The application scenario involves an event registered for multiply users that transmit the event through live video streams. These users can be amateurs, generating videos spontaneously, without a plan or previous definitions. In other words, they potentially transmit UGV live streams. The different UGV live streams can be asynchronous, so they must be synchronized to be consistently presented, or to generate a coherent composition on them. Although UGVs use to be heterogeneous, with different audio information, backgrounds, camera motion among other issues that make this scenario unsuitable for automatic methods [Schweiger et al. 2013].

Automatic methods usually present good efficiency and accuracy using image or video marks, as well objects detection as approach to synchronize videos [Goswami et al. 2014]. However, this class of methods commonly uses a machine learning approach, that demands well structured videos and a large example library to work properly [Karpathy et al. 2014a]. These techniques has obtained very promising results for video description and classification in sport games, advertisements or TV series, which offers similar audiovisual content (in terms of structure, concept, objects, features and others elements) for successive transmissions of this kind of streaming [Karpathy et al. 2014b]. Although, automatic methods have problems relating heterogeneous videos, human being usually can deal with this kind of content. In this way is correct affirm that synchronize a pair of UGV tends to be a task difficult for automatic methods but easy for humans. Based on this, LiveSync was designed over a method that uses human contributions to synchronize pairs of UGVs, and process these contributions to synchronize a set of UGV live streams.

The user contributions are collected by a mash-up application in which all UGV live streams are grouped, and which provides a player with specific functionalities to allows the user to find out a synchronization point for a pair of videos and submit it. Moreover the same application provides a different view that aligns the synchronized UGVs and displays them to users.

Video live streams are associated to real-time delivery, so it is an important characteristic of LiveSync allow users to play the streams while attempt to find synchronization points between then. In this way, users can consume this live content while performing their contribution tasks.

When performing a synchronizing task, the user don't need analyses entire videos seeking for synchronization point. An user can simply use the buttons Play and Pause on the video player until both videos are playing synchronously. Once the user achieved a local synchronism to a pair of videos, the application measure the delay between them and registers it as a $\Delta time$ contribution. All contributions are registered, and used as input to calculate the delay between each pair of videos that will likely satisfy users.

As soon as a pair of videos are synchronized, the application registers this synchronization and align the videos in a visualization interface with the other synchronized videos. So, anyone may access this view and watch the synchronized videos in an aligned view.

3. LIVESYNC TOOL

The LiveSync Tool is a Web implementation for the LiveSync, it provides all components required to proceed a UGV live stream synchronization fallowing the LiveSync method. Moreover, the LiveSync Tool also provides an implementation for Dynamic Aligned List (DAL), the abstract datatype used to register and manage the user contributions.

3.1 DAL

To achieve video streaming synchronization was used the Remote Temporal Couplers (RTC) technique [Segundo and Santos 2015] that allows to align videos from different sources. This technique collects synchronization points for pair of videos, and uses an inference algorithm to find out more relations between videos by transitivity over known relations.

The RTC technique was operationally represented as the Dynamic Alignment List (DAL), an abstract type designed to organize the relations between videos as well provide the functionalists required to store and process the relations.

The start model for the storage structure inside the DAL was a relational matrix MxM which can represent all possible relations between M videos, each position representing the $\Delta time$ between the initial point of a pair of videos. This matrix was reduced to a upper triangular matrix because $\forall i, j < M, \Delta_{i,j} = -\Delta_{j,i}$ and the main diagonal was eliminated because $\Delta_{i,j}$ is always equal to zero. The $\Delta time$ for each couple of videos is calculated for $\Delta i, j = start(j) - start(i)$ where $start(X)$ returns the offset of video X from the start of the event's timeline.

These values are used to synchronize the M videos. 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 the video j should starts 30 ms after the video i starts in order to achieve their synchronization. Additionally, in cases which is impossible to determine a relation between a pair of videos, the value registered is I that means impossible.

Into a collaborative scenario where users can contribute providing a $\Delta i, j$ for a pair of videos (i, j), it is important to store all contributions, because the $\Delta time$ value should be calculated considering the contributions collected for that pair. In that approach the formula $\Delta i, j = start(j) - start(i)$ is used to calculate the $\Delta time$ for each contribution, and the value stored in DAL is determined processing all contributions for each pair. Moreover, the number of contributions for each pair can grow while the contribution process is active, and current $\Delta time$ of a pair can change while contributions are incoming.

In order to represent this model it was needed to define a structure more sophisticated then a matrix. This structure preserves the relational characteristics of a upper triangular matrix without the main diagonal, with an additional dimension related to the contributions for each relation between video pairs. However, it is implemented as a hash table, using linked lists hierarchically organized in three levels.

—Level 1 - The first level is a list of video structures. Each structure has a unique identification label for a correspondent video.

—Level 2 - Each video structure points to a second level linked list composed by all possible relations that include its video, considering the relations in a upper triangular matrix without the main diagonal. Thus, each node in a second level linked list corresponds to a relation between it's 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 represent a contribution for that relation.

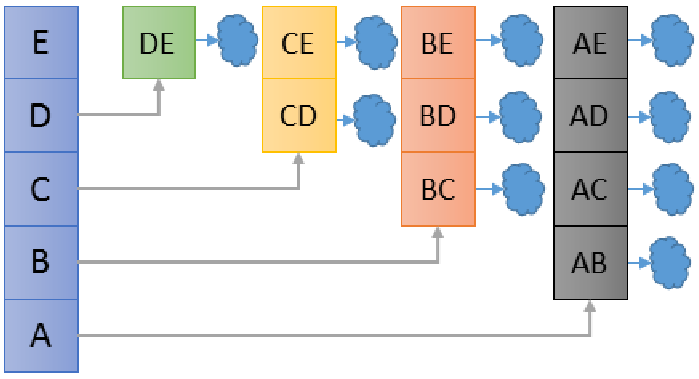

Fig. 1. DAL with 5 Videos (A,B,C,D,E)

Figure 1 exemplify a data structure into a DAL with 5 videos. The videos, labeled as A, B, C, D and E, are in the fist level linked list. Each video points to a second level list, it is possible to observe in Figure 1 that each second level linked list have only the relations that would exist in a single line of a upper triangular matrix without the main diagonal. Moreover, in Figure 1 each relation, that is an element of second level list, points to a cloud that represents a third level list with all contributions for that relation.

A completed DAL can provide all information needed to achieve synchronization for videos registered on it, although it is more than a data structure. DAL is an abstract datatype that provides the data structure plus a set of features that allows access, manage and process the information inside it, such as the function *Infer Synchronization*.

Infer Synchronization can obtain additional relations by executing an inference algorithm over the current relations. This algorithm checks if 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 is possible to infer the relation AC. This feature can reduce the number of contributions required to complete a DAL.

3.2 Main Funcionalities

The main functionalities provided by the LiveSync Tool are:

Synchronized Live Video Player -. the tool permits users to watch multiple videos synchronized. He selects from a list of sources the videos he wish to watch and them they are synchronized using information provided by other users.

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 to allow users to add new stream sources to the application. He only needs to set the video source, and the video will be added to the DAL and list of videos. However, videos added are not filtered, this means that the user can add any video to the application, even ones that contains none relation with the other videos. In future versions will be added an functionality to users mark which video as not related, and then remove them.

Multiple Platforms Support -. One key-point on this tool is to use other platforms as video sources. The videos presented to users, and synchronize by them, are provided by external live video stream platforms. To be compatible with LiveSync Tool are required two requisites:

(1) Remote Player: the platform must allows embeddability into player on third pages, allowing us to control the player with its basic functionalities such as: play, pause and stop;
(2) Uptime Support: a second and fundamental requisite is an API that allows video uptime retrievement. Video uptime is the time since the beginning of the video that is presented on the video player. This is fundamental to create and replicate the couplers generated in synchronization process.

Serverless Architecture -. Serverless architectures refer to applications that significantly depend on third-party services and putting much of the application behavior and logic on the front end. Such architectures remove the need for the traditional server system sitting behind an application [Roberts 2016].

Multiplatform -. LiveSync is a Web Based application designed and developed in compatibility with HTML5 standard to its front-end (Mash-up Player) component. It allows this application to be executed on multiple browsers, operational systems and devices.

Active vs Passive Contributions -. Currently exist two versions for LiveSync Tool that only differ in what pair of videos should be synchronized by each user. The active version allow users to navigate freely through the videos, synchronizing them when they wish to. The Passive Contribution version uses an automatic algorithm to ask users which pair of videos they should synchronize.

3.3 Architecture

The LiveSync Tool has three main components (Figure 2): the Content Providers (Video Sources), the Coupler and the Mash-up Player.

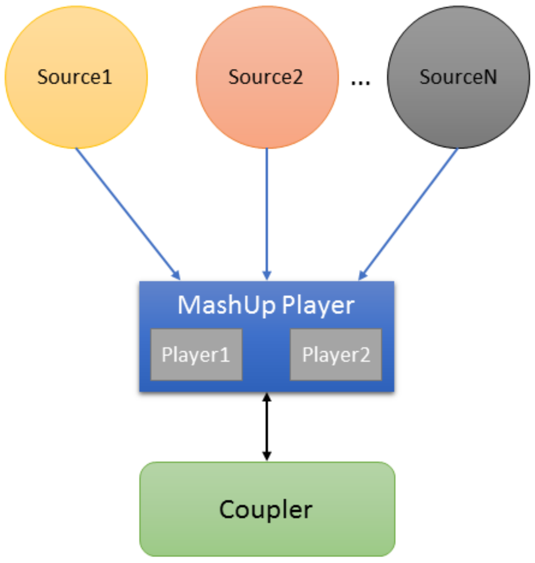

Fig. 2. LiveSync Tool's components

3.3.1 *Content Providers.* Content Providers are third-parties videos streamers platforms. There are multiple Live Video Stream applications in the market, such as YouTube Live, LiveStream, Twit-Cast, Twitch and Ustream. One of our objective was to allow the use of different platforms as video sources, so we maximize the number of videos for an event and allow the use of already in market platforms. A Content Provider needs two requisites to be compatible with LiveSync: a Remote Player and Uptime Support. As each stream platform uses their own protocols, we opt to use their embeddable players into a mash-up application. These players must allow us to play, pause and stop the video stream. The second requisite, Uptime Support, is necessary to find the couplers among the videos. Uptime is the time passed since the beginning of the live stream until the video part being presented in the player at the moment of the call.

3.3.2 *Coupler.* The Coupler is responsible for storage, distribution and calculation of synchronization points among video streams from the Content Providers.

An instance of coupler is composed of a DAL instance and Log files. This goes in direction of the Serverless Architecture. We wanted an architecture that needed low resources (another justification for using third party stream services) and easy deployment. All that is necessary do execute the coupler is a NODE.JS (https://nodejs.org/en/) server instance. This is possible because the coupler is fully developed in JavaScript and compatible with the HTML5 standards. To deploy the coupler, we use a Backend as a Service or ”BaaS” platform, more specifically we use the Heroku (www.heroku.com) one, that permits free use of NODE.JS instances.

It stores synchronization information only during the duration of the event, so its stance is finished with the end of the videos and all data is lost. In the current scope, the sync info is only necessary during the event, after it, there is no need to store the information. For reasons of testing and using the filmed videos from YouTube we create log files that contains all contributions made by the crowd. If it is important to maintain all contributions and data for post analyses and further use, unstable version of the LiveSync is being configured to use a fully transactional database. We use a fully transactional database because we want to maintain track of all contributions made by the crowd, an important aspect for crowdsourcing support and that is also supported by the DAL.

Other aspect of the coupler is that it is responsible for the distribution of synchronization couplers. When a user chooses two videos, a message is sent from the mash-up to the coupler, containing the required relation. The coupler then answer with the required information. If the relations is unknown, it answer soliciting the user to synchronize and contribute with those two videos.

The last function of the coupler, is to calculate the synchronization points among Videos. Each relation ($\Delta_{A,B}$) may contain several contributions, then it is necessary to calculate a value to that relation based on the contributions. In the current version we calculate a Geometric Mean of the contributions to find an ideal value. This however may not be the best value, because the more accurate the sync is, the better the results are, so older contributions must have a lighter weight, something that does not happen in current version. The other calculation made by the Coupler, is to infer unknown relations based on the afore mentioned transitive relation.

The communication between coupler and mash-up is made through Websocket communication. The mash-up creates a WebSocket channel with the Coupler, and requests the sync information or sends contributions from the crowd. A simple protocol is used in JSON messages: act:value, data:object. The act field contains the action to be made and the data contains an object to complement the action. As example we have an act to send a new contribution ("contribution") that is complemented with a new relation that contains the videos involved, the value and an id to that contribution.

3.3.3 *Mash-up Player.* Mash-ups are applications generated by combining content, presentation or other applications functionalities from disparate sources. They aim to combine these sources to create useful new applications or services (the offer and consumption of data between two devices) to users. The LiveSync Tool combines videos from different sources with synchronization information from the Coupler to reproduce a synchronous presentation for these videos. The Mash-up Player is used to both presenting video synchronously and collecting the synchronization values.

At the top of the interface all information necessary to the user as may be observed on Figure 3. Fallowing the orientations on the screen an user can select which videos want to watch, add a new videos, as well start the synchronization process for videos that he believes are not synchronized.

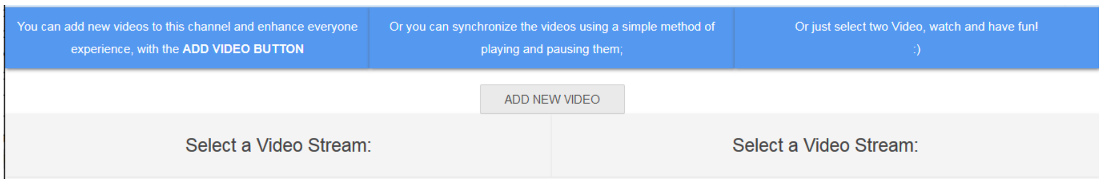
You can add new videos to this channel and enhance everyone experience, with the ADD VIDEO BUTTON
Or you can synchronize the videos using a simple method of playing and pausing them;
Or just select two Video, watch and have fun! :)
ADD NEW VIDEO
Select a Video Stream:
Select a Video Stream:

Fig. 3. Action menu at the top of interface

When an user adds a video, an input text is shown to him, so he can add the video URI (WebSocket) or video ID YouTube). The page reloads and the new video is listed in the video list for everyone that connects to the application. Also, when the video is added by the user, an message is sent to the coupler, containing the action to add a new video to the DAL, and the specification of it, such as label and URI.

When just playing two selected videos from the videos list, each for video player is created an instance that is compatible with that source (YouTube or WebSocket). Moreover, it is invisible to users where the video is coming from.

The last functionality of the mash-up is to synchronize the videos. When an user chooses to synchronizing a pair of videos, the mash-up display is reconfigured and the application enter in synchronization mode. LiveSync Tool uses a Play 'n Pause approach to synchronize the videos. Figure 4 represents the interface of the mash-up during a test: two cameras live streaming (content providers) a simulated television event to our mash-up application. Once the user decided the videos are synchronized he clicks on the done button, so his contribution committed to the coupler that stores it into the DAL for further processing of the relation.

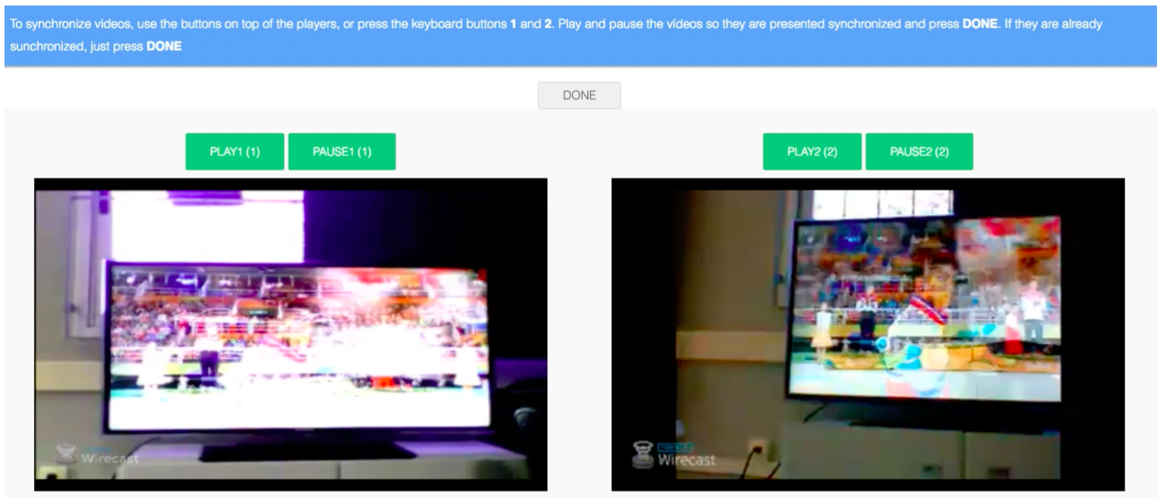
To synchronize videos, use the buttons on top of the players, or press the keyboard buttons **1** and **2**. Play and pause the videos so they are presented synchronized and press **DONE**. If they are already sunchronized, just press **DONE**

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

4. EXPERIMENT

The experiment consists in...

5. FINAL REMARKS

LiveSync is a method that aims achieve UGV live streams synchronization. The focus of this method is cover the scenarios where automatic techniques facing problems to work properly. The approach adopted in LiveSync is to use do work-power of a crowd of contributors to executing small tasks generally easy for humans but hard to machines, registering these contributions and using it to determine the synchronization points between videos. This approach assume 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 contribute in UGV live streaming synchronization processes, as well to watch synchronized UGV live streams from multiple sources. However, this tool presents some limitations that shall in the futures be suppressed, such as the number of events that we can follow. Now, each instance of Coupler and mash-ups can handle only one event, in other words, we can not cover two independent live events at the same time with one instance, for that porpoise we need more than on instance of each service. Also, new stream services must be added to increase our compatibilities.

All code for LiveSync Tool may be freely downloaded from Github `https://github.com/rmcs87/liveSync`.

REFERENCES

de Amorim, M. N., Segundo, R. M., and Santos, C. A. Livesync: a tool for real time video streaming synchronization from independent sources.

Difallah, D. E., Catasta, M., Demartini, G., Ipeirotis, P. G., and Cudré-Mauroux, P. The dynamics of micro-task crowdsourcing: The case of amazon mturk. In *Proceedings of the 24th International Conference on World Wide Web*. WWW '15. ACM, New York, NY, USA, pp. 238–247, 2015.

Goswami, A. K., Gakhar, S., and Kaur, H. Article: Automatic object recognition from satellite images using artificial neural network. *International Journal of Computer Applications* 95 (10): 33–39, June, 2014. Full text available.

Howe, J. The rise of crowdsourcing. *Wired magazine* 14 (6): 1–4, 2006.

Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., and Fei-Fei, L. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014a.

Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., and Fei-Fei, L. Large-scale video classification with convolutional neural networks. In *2014 IEEE Conference on Computer Vision and Pattern Recognition.* pp. 1725–1732, 2014b.

Roberts, M. Serverless architectures, 2016.

Rohwer, P. A note on human computation limits. In *Proceedings of the ACM SIGKDD Workshop on Human Computation.* HCOMP '10. ACM, New York, NY, USA, pp. 38–40, 2010.

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.* IEEE, IEEE, pp. 532–539, 2015.

Von Ahn, L. *Human Computation.* Ph.D. thesis, Pittsburgh, PA, USA, 2005. AAI3205378.

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