LiveSync: a method 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 features LiveSync, a method that allows users to synchronize live video-streams from multiple sources, such as YouTube, Vimeo and various other video streaming platforms.The proposed method aims to cover a 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. LiveSync is based on the humans' ability on associating heterogeneous videos using their intelligence. This method adopt a hybrid strategy, synchronizing a set of video streams by aggregating user contributions that consist in identify a temporal delay between a pair of videos. The outcome from this process is the temporal alignment of the video streams, allowing to generate synchronized mosaics from subsets of these videos. Also introduced it is the Web application LiveSync Tool to support its use, as well is presented the Dynamic Alignment List, an abstract datatype defined to store and to manege the contributions, as well to perform relevant operations over these data, such as to infer additional synchronization points by using transitivity properties. The usage of this tool in real video synchronization scenarios is also presented in the paper.

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

User generated videos (UGV) live streaming continuously grows in number end relevance boosted by platforms such as Facebook, Youtube and Vimeo. Into this scenario is relevant find out methods capable to synchronize them.

Automatic methods are widely used to reach video synchronization. Although, they generally demand vast example databases as well controlled conditions, standardized structure and marks to work properly [Wang et al. 2014]. These methods usually present good results synchronizing well-structured video productions, professional coverage for sport events and other modalities of planed videos. However, 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].

UGV are produced 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 automatic methods unappropriated to synchronize these sources.

To achieve synchronization for UGV live streamings from different sources, this work introduces LiveSync, a method that explore the human ability of associate heterogeneous videos. LiveSync is based on the Human Computation paragidm [Von Ahn 2005], using human intelligence to execute

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.

tasks usually hard to machines but easy to ordinary people using its senses (vision and hearing). Additionally, human computation 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 approach, to use efficiently the processing power of a crowd of collaborators, collecting their contributions in parallel [Howe 2006] .

Crowndourcing is an approach in which a problem is divided in tasks, and the execution of these tasks is delegated to the crowd composed by individuals engaged in the solution process. The partial result delivered by each contribution are registered, and a final outcome is generated by processing them. A very popular crowdsourcing strategy consists in to distribute tasks that can be completed easily and quickly [Difallah et al. 2015].

The application scenario for LiveSync contains a set of UGV live streams from different sources, that must be synchronized. These synchronization issues are related to time delays between videos streamed from different sources. The occurrence of time delay between a pair of videos is manifested as misalignment, in other words, there is a time difference between scenes that should be displayed at the same time. In this work the time delay between a pair of videos if refereed as $\Delta time$.

In the proposed method is created a list with all possible pairs of video streams, and each collaborator is asked to provide a synchronization point for one pair. The delay between each pair of streams is determined by processing the contributions, and the streams are aligned based on these delays, making possible to align them in a synchronized presentation.

This paper extends the previous work "LiveSync: a tool for Real Time Video Streaming Synchronization from Independent Sources" [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).

This remain of this work is organized as follows: section 2 describes method LiveSync, section 3 details LiveSync Tool, section 4 presents an study case to evaluate the proposed method. Finally, section 5 concludes with some remarks about this work and further research.

2. LIVESYNC

LiveSync is a 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 stream plattforms. These users can be amateurs generating videos spontaneously, without a plan or previous definitions. In other words, they 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, audio or video marks, as well object 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, while automatic methods have problems relating heterogeneous videos, human being usually can deal with this kind of content. In this way is correct to affirm that to synchronize a pair of UGV tends to be a difficult task for automatic methods but easy for humans. Based on this, LiveSync was designed to use human contributions to synchronize pairs of UGVs, and process these contributions to synchronize a set of UGV live streams.

Initially, this method is to determine all possible pairs of streams, because its objective is to temporally align them based on the $\Delta times$ between the streams. This approach is based on the previous method Remote Temporal Couplers [Segundo and Santos 2015], used to align a main content with additional content from different sources. The major difference between both of methods is: while Remote Temporal Couplers aims to align additional content over the timeline associated to the main content, LiveSync align pairs of streams determining the $\Delta time$ between them, and to aggregate this information to align all streams on a new timeline.

The strategy adopted to LiveSync, in order to obtain the delay between all possible pairs of videos, is to send a pair of streams to each collaborator and to ask him if it is possible determine the delay between the streams, and what is this delay. All contributions are stored, and all contributions for each pair is processed to calculate the most probably delay between the streams in it. As the synchronization information is determined, the relations between the videos are created, so that in a situation that knows the delay between all pairs, the whole set of streams will be related. Streams that could not be related to others are left out of alignment.

3. LIVESYNC TOOL

The LiveSync Tool is a Web implementation for the LiveSync, it provides all components required to proceed UGV live stream synchronization following 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 Remote Temporal Couplers (RTC)

LiveSync is based on the previous technique Remote Temporal Couplers [Segundo and Santos 2015] that intents to synchronize multiple contents related to the same main-content (MC). These contents may come from different Content Suppliers (CS), and range from video objects to extra content (EC), such as subtitles, additional information boards, labels, audio objects, or any other media artifacts that can me aligned over the same timeline.

Remote Temporal Couplers is based on three components: Main Content Provider, Content Suppliers, and User Devices (UD). Figure 1 [Segundo and Santos 2015] shows the communication between these three components.

—*MC Provider* is responsible for streaming the MC, that usually is a video stream transmitted over accessible channels to UD and CS.
—*Content Suppliers* provide EC related to a MC requested by UD.
—*User Devices* are responsible to control one or more services provided by CS and playing the MC.

Remote Temporal Couplers consider that MC Providers usually offers no explicit timing information in its content to support the synchronization. Although, CS can provide synchronized content without depending on a explicit time specification sent by the MC Provider. The way found for reach synchronization into this scenario is delegate to CS the responsibility for generate temporal couplers. A temporal coupler is an structure that relates a scene to the local timestamp, associated with it on a CS. In other words, for each scene Sx presented by a CS, it provides a temporal coupler *[Sx, Timestamp]* that identifies the local time when the scene occurred in its local timeline.

Finally, the Remote Temporal Couplers technique uses the timestamp into each temporal coupler to synchronize the contents from multiple Content Suppliers, as well to compose a coherent presentation in a mash-up application, in which users can choose what contents should be included.

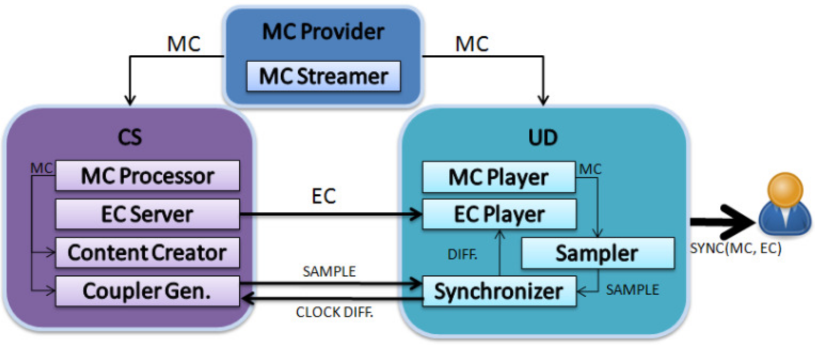

Fig. 1. Remote Temporal Couplers - Entities Composition for Content Synchronization

Although LiveSync approach to synchronize video streams is based on Remote Temporal Couplers, it uses relational couplers, instead the temporal couplers used by Remote Temporal Couplers. A relational coupler relates pairs of videos and the $\Delta time$ between them. In this approach, alignment is constructed by relating a set of video streams, pair to pair, to generate coverage on the event timeline, rather than taking on a main-stream and to align each other stream with it. Moreover temporal couplers still are used in LiveSync to correctly play the synchronized streams.

3.2 Dynamic Aligned List (DAL)

The Dynamic Alignment List (DAL) is an abstract datatype designed to organize the relations (relational couplers) between videos as well to provide the features required to store and process the relations. Each relation stored into a DAL instance corresponds to a relational coupler related with a pair of video streams and the $\Delta time$ between them.

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

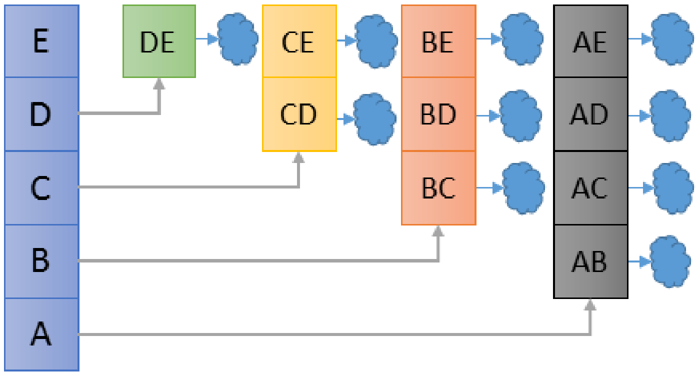

Fig. 2. DAL with 5 Videos (A,B,C,D,E)

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.

Figure 2 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 2 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 2 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.

3.3 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 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 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 allow 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 temporal couplers used to play the synchronized video streams.

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.

3.4 Architecture

The LiveSync Tool has three main components (Figure 3): the Content Providers (Video Sources), the Coupler Service and the Mash-up Player.

3.4.1 *Content Providers.*

Content Providers are third-parties videos streamers platforms, such as YouTube Live, LiveStream, TwitCast, Twitch and Ustream. LiveSync Tool supports different platforms as video sources, maximizing the number of videos streams that can be related to an event.

There are two requirements that a Content Provider must attend to be compatible with LiveSync: Remote Player and Uptime Support.

As each flow platform uses its own protocols, the embedded players are required for aggregation in the mash-up application. These players must provide buttons to play, pause and stop actions.

Uptime Support, is necessary to find the temporal couplers to video streams. 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.

3.4.2 *Coupler Service.*

The Coupler Service is responsible for storage, distribution and calculation of synchronization points between video streams from the Content Providers. Coupler Service provides a DAL instance and Log files.

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

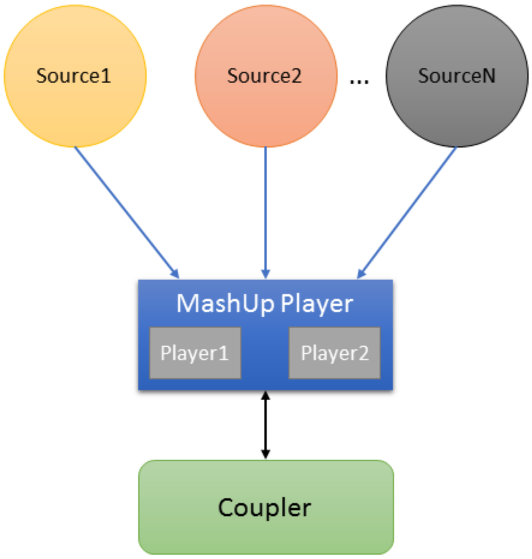

Fig. 3. LiveSync Tool's components

the filmed videos from YouTube was generated 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. A fully transactional database is used to track all contributions made by the crowd, an important aspect for crowdsourcing support and that is also supported by the DAL.

Other aspect of the CS is that it is responsible for the distribution of relational couplers, containing synchronization information between pairs of video streams. When an user selects a pair of streams, a message is sent from the mash-up to Coupler Service, containing the required relation. CS then answers it with required information. If the relation is unknown, it responds with a prompt for the user to synchronize them.

The last functionality on Coupler Service, is to calculate the synchronization points between videos streams. 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, this delta is calculated by the geometric mean of the contributions to find an satisfying value. Coupler Service also uses the Inference functionality by DAL to find additional relations over these calculated values.

The communication between Coupler Service and the mash-up application is made through Websocket communication. The mash-up application creates a WebSocket channel to Coupler Service, and requests the sync information or sends contributions from the crowd. A simple protocol is used for JSON messages: act:value, data:object. The *act* field contains the action to be made and the *data* field contains an object to complement the action. As example, the action used to send a new contribution, transmit as value for its *data* field a relational coupler representation that contains the videos involved, as so the contribution value for the Δ between them.

3.4.3 *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 Service 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 4. 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.

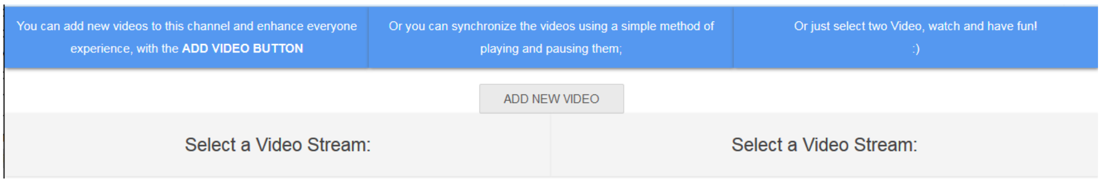

Fig. 4. Action menu at the top of interface

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 Service, 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 5 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 Service that stores it into the DAL for further processing of the relation.

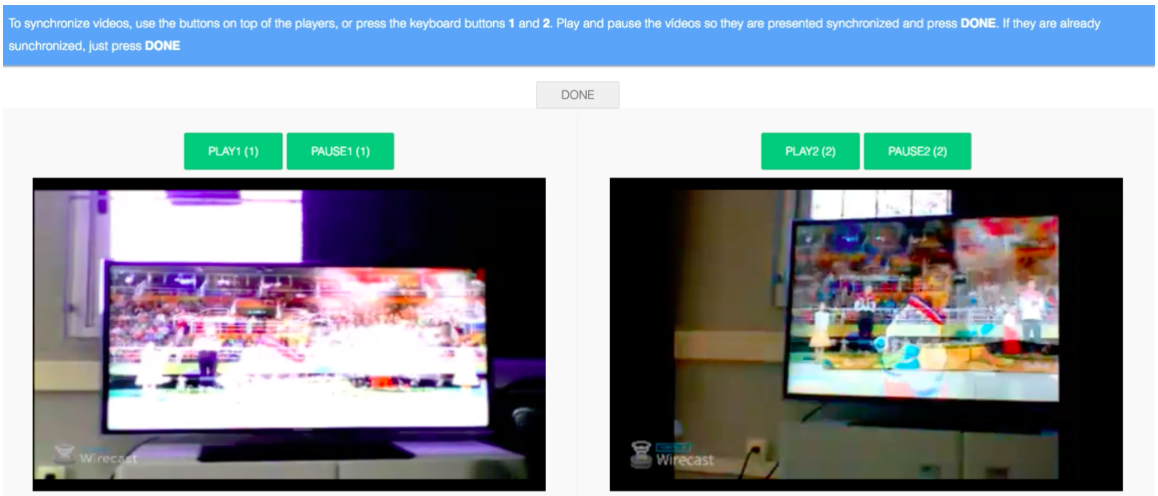

Fig. 5. Live Streams from Olympic Games Synchronized through two different cameras

4. CASE STUDY

The case study chosen to validate LiveSync, as well LiveSyncTool, was an internal event that could be streamed by volunteers through their smartphones. This event consisted in [XXXXXXXXXXXXXXXXXXXXXXX].

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.