SYNCHRONIZATION OF PROCESSED AUDIO-VIDEO SIGNALS USING TIME-STAMPS

Mohamed El-Helaly and Aishy Amer

Electrical and Computer Engineering, Concordia University, Montréal, Québec, Canada
Email: {elhel_m, amer}@ece.concordia.ca

ABSTRACT

This paper proposes a system for synchronizing audio and video signals after they have been captured and processed in separate modules. The captured audio signal is run through a speech recognizer while the video signal, captured using a single camera, is run through an object detector. The synchronization system uses time-stamps from both processes to produce a synchronized output containing the detected objects and their relative recognized speech. The processed audio and video streams are buffered and the paper proposes an integration scheme such that a synchronized output is realized.

Index Terms— Synchronization, Multimedia systems, Video signal processing

1. INTRODUCTION

In multimedia applications, there is a growing demand to integrate the various forms of media with one another. For example, the association of the object speech with its video images increases the effectiveness of audio-video systems such as video conferencing or video telephony. Especially when the media signals are captured and processed separately, such association requires synchronization of *processed* signals.

This paper proposes a system to synchronize separately *processed* audio and video signals. The proposed system is meant for online applications, where one would be able to observe moving objects while observing the recognized speech they uttered. The objective is to relate the detected video objects to their relevant speeches. Such relation is useful in interactive multimedia applications such as in interactive TV, video retrieval, or visual fine arts.

Lopes et al.[1] explain the use of MPEG-7 and synchronization in solving the data retrieval and management problem. Lopes at el. look at personalized TV services and use MPEG-7 descriptors to obtain the information on the scenes captured. More flexibility is achieved in [1] when communication with other devices because time is already embedded in the descriptor. One drawback for using the descriptors is that they have to be designed and therefore the success of the system will depend on the descriptor.

Kim et al.[2] explore the problem of intra-stream synchronization across computing platforms. Intra-media is the playback of the medium involved in continuous time while inter-media synchronization determines the scheduling of playback of the medium. The system in [2] contains methods to control events based on timing criteria. Kim et al. describe a target play time (TPT) which is the time of capture plus the streaming delay which governs playback of the frame. This approach uses a global time stamp method to achieve intra-synchronization, but does not give a solution for synchronizing different media streams.

Lienhart et al.[3] present a synchronization method for distributed audio-video capture devices where they insert system time-stamps into the audio data at A/D conversion time and process the audio data along with the time-stamp information. The timing information is used to convert the sampling rates such that they are synchronized with the video sampling rates. Note that the method in [3] does not perform any processing on the signals but rather uses sampling rate conversion to synchronize the signals.

Benslimane[4] analyzes multimedia synchronization issues present in telecommunication networks. A buffer is used to store the delayed media units (MUs). The receiver then informs the transmitter of the new delay in the network. The buffer size is dependent on the average delay of the network. The paper in [4] proposes a variable buffer size to overcome the varied jitter problem and provides a good approach by using a buffer system for integration and synchronization that is dependent on the delays of the network. However, [4] does not account for delays involving the processing of the MUs.

In this paper, we propose an algorithm to synchronize processed audio and video signals. In past literature, processing of the media streams was limited to simple operations such as sampling while in this paper the processing involves speech recognition and video object detection. The speech recognition module adds a delay to the audio signal information (i.e., the recognized text) and the video processing (i.e., detection of objects) adds its own delay. The proposed algorithm makes use of temporal information obtained using time-stamps to relate the media streams with one another. The contributions of the paper are extracting the audio and video time-stamps,

This work was supported, in part, by the Natural Sciences and Engineering Research Council (NSERC) of Canada. The authors would like to thank Rabih Mahzoub from the INRS in aiding to program the speech recognition module.

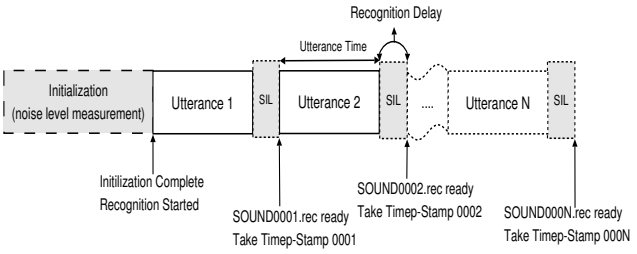


Fig. 1. Time-Stamps between Output Speech Files.

synchronizing the processed output using the time-stamp information and integrating the whole system such that proper use of the synchronization approach is utilized.

2. PROPOSED SYNCHRONIZATION ALGORITHM

The proposed synchronization system accounts for the processing delays incurred on the processed audio and video media streams. The audio processing delay is incurred from the time the audio is captured to the time the recognized speech (transcribed text) is available, while the video processing delay is the time between when the raw frame is captured to the time the segmented objects are available.

The processing delays are variable and therefore, a method is developed to extract these delay times. These delay times are obtained using time-stamps. Time-stamping is a method of obtaining the actual time at a specified point in a process. The proposed system extracts the time-stamps from the audio and video processes, respectively. If the time of processing of each video frame is calculated as well as extracting their time of arrival to the processing module, the actual time of the frame can be established. If the actual time of the audio is known via the time-stamps, then the time to insert the audio information (i.e., the recognized text in the case of this system) onto the video will be known.

2.1. Audio and Video Time-Stamps

Obtaining the audio time-stamps involves extracting the time of the utterance and the delay of the recognizer. The times of the utterances are obtained using time-stamps at the start and end of the recognition relative to a system clock. The time-stamps ($\tau_{s(i)}$) are taken once the text output file, i, is available to access. This file contains the recognized utterance (i.e., utterance i is in file i). The audio time-stamps are taken according to the availability of the recognizer's output file (Fig.1).

The delay, SIL, of the recognizer to recognize utterance i is represented by

$$SIL = \tau_{s(i)} - \tau_{s(i-1)} - \tau_{u(i)}, \tag{1}$$

where $\tau_{s(i)}$ represents the time-stamp of utterance i, $\tau_{s(i-1)}$ represents the time-stamp of the previous utterance $(i-1)$ and $\tau_{u(i)}$ represents the actual time of the utterance. $\tau_{u(i)}$ is then

$$\tau_{u(i)} = \tau_{s(i)} - \tau_{s(i-1)} - SIL. \tag{2}$$

The total time of the utterances, up to utterance i is represented by $\Delta A(i)$, and is calculated by

$$\Delta A(i) = \Delta A(i-1) + \tau_{u(i)}, \tag{3}$$

where $\Delta A(i-1)$ is the previously accumulated times up to utterance $i-1$. If the time of the utterance, $\tau_{u(i)}$ is not known, and the delay SIL, is known, $\Delta A(i)$ can be obtained by

$$\Delta A(i) = \Delta A(i-1) + \tau_{s(i)} - \tau_{s(i-1)} - SIL. \tag{4}$$

$\Delta A(i)$ is the output of extracting the audio time-stamps and is used in the synchronization process.

The processing of the captured video frames uses [5] to detect objects and their contours. For each frame, a time-stamp is taken at the moment the raw frame is available from capture, and subsequently another time-stamp is taken after all the processing is complete. With this information, the delay of the video processing is obtained and the actual output frame rate, F_R can be calculated using

$$F_R = \frac{\tau_{frames}}{F_{Count}}, \tag{5}$$

where τ_{frames} is the processing time of F_{Count} frames. In fact, F_{Count}, is a sliding index such that the F_R is representative of the recent timing parameters of the video sequence.

2.2. Synchronization Approach

The synchronization module is responsible for using the timing information obtained from the video and audio processing modules. The objective is to calculate the frame number that the recognized text relates to, using the audio time-stamps. Using the total time of the utterance, the frame number at which to insert the recognized text is calculated by using the frame rate, F_R, of the processed video frames . Therefore, the synchronized frame, $SyncFrames$, which is the frame at which at the recognized text is to be inserted, is calculated using

$$SyncFrames = \triangle A(i) \cdot F_R, \tag{6}$$

where $\Delta A(i)$, as in Eq. 4 is the accumulated time of the utterances up to utterance i and is calculated using the total time of audio processing minus the recognizer's delay. F_R is calculated in Eq. 5.

The synchronization module uses the information from Eq. 6 to synchronize the processed output of the audio and video modules. The video frames are buffered because the speech recognition is not real-time and its processing time is not uniform. The delay of the video processing may also be non-uniform and without a buffer system, it would not be possible to synchronize the processed streams as each streams'

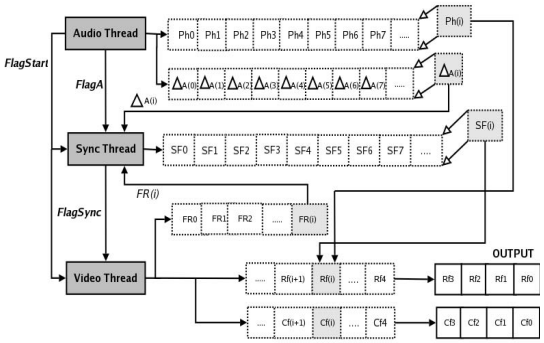


Fig. 2. Buffer system and Communication.

delay is independent of one another. Each recognized utterance is buffered along with the $SyncFrames$ associated with it. The synchronization module maintains the indexing of the buffers as well as their associated outputs.

In the system implementation there are 6 buffers: the real frame buffer, the object contour frame buffer, the phoneme (recognized text) buffer, the audio time-stamp buffer, the video time-stamp buffer, and the $SyncFrames$ buffer. The flag system to control the buffers is illustrated in Fig. 2.

The first thread to start processing is the audio thread as it sends the initialization flag to all the other threads *FlagStart*. The audio thread sets *FlagStart* and in turn, the video thread starts capturing the frames and filling its respective buffer with real and contour frames. A flag is triggered by the audio thread, called *FlagA*, which indicates to the synchronization thread that a new output file is ready to be synchronized. The timing information from the recognizer's output file is passed to the synchronization thread as well as the frame rate, such that it can calculate the $SyncFrames$ as indicated in Eq. 6. *FlagSync* is set when $SyncFrames$ is available which in turn signals the video thread to read the sync frame number, *SF(i)* (the frame at which to insert the text) and the text, *Ph(i)*. If the frame number has not been outputted yet, it waits until it reaches the frame number. If the frame count is larger, the video thread then outputs the text immediately, similar to the approach in [2]. It is impossible to exactly catch the correct frame to superimpose the text. After the text is inserted the process restarts by collecting the next output file.

2.3. System Adaptability

The proposed system uses the timing information of the processed signals for the synchronization process. The system is therefore, adaptable to synchronize systems where the audio processing delay is larger or smaller than the video processing delay. The timing information is used as the basis of the proposed synchronization approach, which allows the system to be adaptable to varying processing delays. The time-stamp information does not rely on the number of detected objects or the number of speakers in the frame. The only limitation is that the speed of the speech recognition system. The recognizer is only trained for the first author's voice but that can be changed, if required.

3. SIMULATION RESULTS

Quantitative measures are produced using an experimental setup: the camera captures the video and the speakers utter predetermined sentences. The video buffer size depends on the delay of the recognizer which depends on the utterance length. Therefore, the length of the utterance is controlled as a high delay in recognition may cause the system to fail.

Several experimental trials were performed and are aimed to test the systems' limit in terms of synchronization success rate and buffer capacity. The time length of each utterance and the total time of speech are measured by the aid of a stopwatch. The time of each utterance from the stop watch is compared to the time that the recognized text is inserted onto the frame. From the system itself, the frame rate F_R, the $SyncFrames$ (Eq. 6), and the difference in frames $DiffFrames$ (DF), between the $SyncFrames$, (SF) and the actual frame number at which the text was inserted, are measured.

In this paper, a sample trial evaluates the performance of the synchronization system and its relative enhancement caused by increasing the video buffer from 20 frames to 40 frames. If the frame buffer is increased, then it allows the system more time to evaluate the SF and decrease the synchronization error. However, an increase in buffer size increases the overall output delay of the system. The output delay of the system due to the buffer size of 20 frames is 2.75s and 4.44s for the 40 frame buffer. Table 1 and 2 displays the results of the trials using 20 and 40 frames for the size of the video buffer. Note that the first percentage errors were not given because the error calculated with the data available was not reflective of the actual events, as the time is very small and a one frame error would result in large percentage error.

The %error is the differnece between the measured time and the calculated time from the SF and the F_R. The maximum error is not over 15% (20 frame: 1.4s, 40 frame: 0.67s). The objective is to test the speed of the system and its accuracy when large amounts of processing is required over a short period of time.

The trial involving the 20 frame buffer shows that the first synchronization error was present at the 4th iteration of the first sentence. The maximum error at the last iteration, was at a 15% error. In the trial involving the 40 frame buffer, the synchrnoization error was present at the 6th iteration. Therefore, it is eveident that the 40 frame buffer is less prone to error and proves that the size of the buffer effects the synchrnoization success rate. Also if the total frames of synchroniuzation error is noted (the last row in Table 1 and 2) we will notice the change in percentage error of the DF.

There is a significant reduction in error caused by the increase of the buffer size, from 20 frames to 40 frames. The

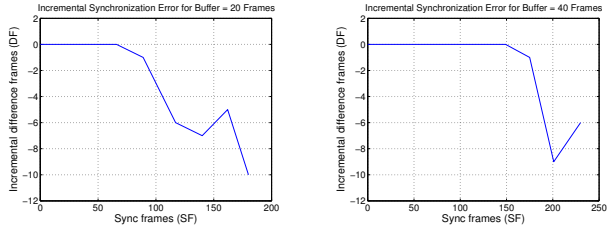
Time(sec)	SF	F_R	***DF***	%error
1.71	22	7.6	**0**	-
5.53	43	7.4	**0**	5.0
9.36	66	7.3	**0**	3.4
13.29	89	7.3	**-1**	8.3
16.86	117	7.3	**-6**	4.9
21.37	140	7.3	**-7**	10.3
25.56	162	7.3	**-5**	13.2
29.02	180	7.3	**-10**	15.0

Table 1. Results of 20 Frame Buffer Trial

Time(sec)	SF	F_R	***DF***	%error
1.59	31	9.7	**0**	-
5.61	61	9.3	**0**	16
9.67	90	9.1	**0**	2.3
13.76	119	9.0	**0**	3.9
17.68	149	9.0	**0**	6.4
21.66	175	9.0	**-1**	10.2
25.71	201	9.0	**-9**	13.0
29.76	230	8.9	**-6**	13.0

Table 2. Results of 40 Frame Buffer Trial

frame rate, F_R, is also increased in the system with the larger buffer. This is due to the fact that more time is dedicated to processing rather than outputting the frames because of the buffer size. The graphs in Figs. 3(a) and 3(b) show that the system with a 40 frame buffer gives a better performance. The graphs are intended to show that there is overall a lower error in the system with a larger buffer.



(a) 20 frame Buffer (b) 40 frame Buffer

Fig. 3. Graph of Incremental Synchronization Error.

A sample screen shot of the system output is shown in Figs. 4(a) and 4(b). As can be seen, the phonemes are superimposed onto the image, under the faces boxes. Figures 4(a) and 4(b) also display the exchange of speech.

(a) Real frame (b) Contour frame

Fig. 4. Visual System output.

Table 3 summarizes the actions of the proposed synchronization system relative to the difference frames and an error threshold of 20%. In summary, the system has a better performance when the utterances are kept short which causes the recognizer delay to be low. There is also a tradeoff where an increase in buffer size reduces the synchronization error but increases the output delay. The output of the system is delayed from the real event by an amount equal to the size of the buffer.

$DF < 0$	Do Nothing
$DF = 0$	Insert Text
$0 < DF < (0.2 \cdot TotalFrames)$	Insert Text
$DF \geq (0.2 \cdot TotalFrames)$	Restart System

Table 3. System Actions based on Difference Frame (DF).

4. CONCLUSION

This paper proposes a multimedia system that use raw media streams and produces a different output through their processing. The system achieved in extracting audio and video time-stamps from the speech recognition and video processing modules, respectively. Also, the system proposed a buffer system that stored and indexed the recognized speech and processed video frames. The synchronization system uses the time-stamps of each media stream along with the information in the buffer to produce a synchronized output of recognized text superimposed onto contour frames. The integration of the system was established using a flag system such that communication between the processing threads is possible.

5. REFERENCES

[1] R.J. Lopes, A.T. Lindsay, and D. Hutchison, "The utility of MPEG-7 systems in audio-visual applications with multiple streams," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 1, pp. 16–25, Jan 2003.

[2] K.H. Kim, S. Liu, M.H. Kim, , and D.H. Kim, "A global-time-based approach for high-quality real-time video streaming services," in *Seventh IEEE International Symposium on Multimedia*, 2005, p. 9pp.

[3] R. Lienhart, I. Kozintsev, and S. Wehr, "Universal synchronization scheme for distributed audio-video capture on heterogeneous computing platforms," Berkeley, CA, USA, Nov 2003, ACM Annual Conference on Multimedia, pp. 263 – 266.

[4] A. Benslimane, "A multimedia synchronization protocol for multicast groups," in *Proceedings of the 26th Euromicro Conference*, Sept 2000, vol. 1, pp. 456 – 463.

[5] A. Amer, "Memory-based spatio-temporal real-time object segmentation," in *in Proc. SPIE Int. Symposium on Electronic Imaging, Conf. on Real-Time Imaging (RTI)*, Santa Carla, USA, 2003, vol. 5012, pp. 10–21.