Detect Audio-Video Temporal Synchronization Errors in Advertisements (Ads)

Zongyi (Joe) Liu*, Devin Chen, Yarong Feng, Yuan Ling, Shunyan Luo, Shujing Dong, Bruce Ferry
Customer Experience and Business Trends
Amazon.com
Seattle, Washington, 98121
Email: joeliu*, devichen, yarongf, yualing, shunyl, shujdong, bferry@amazon.com

***Abstract*—Detecting audio-video (A/V) synchronization error is important to measure end user experience. Today, researches in this domain are mainly focused on contents such as movies or sports. The state of art algorithms usually first detect a specific type of events and then correlate the A/V data within during these events, e.g., find the human chatting events and then correlate the vocals with the lip shapes. Detecting A/V sync errors during Ads, on the other hand, has not received a lot of attentions. Compared with contents, an Ads section do not contain a particular type of events that can be used to detect A/V sync error. For example, many vocals in Ads are either from background narrators or have a very short period of time, so that the popular lip-sync based algorithms won't work accurately. In this paper, we present a novel algorithm that uses the scene change time features: we first segment out individual Ad from a playback. Then for each pair of temporal adjacent Ads, we compute the scene change time for the video data and the audio data separately, and then build their time difference histogram. Next, we aggregate the histograms from all Ads pairs within one Ads section. Finally, we combine the aggregated histogram to compute the A/V off-sync time values. We show that compared with the traditional lip-sync based algorithms, the new algorithm not only significantly improves the prediction rate, but also increases the prediction accuracy.**

I. Introduction

Most video streams today are companied by audio streams. It has been well-known that A/V synchronization errors can severely impair user-experiences. Recently, J. Zhao *et al.* performed a subjective experiment to evaluate the impact of four aspects of quality indexes: video quality, audio quality, temporal A/V synchronization and spatial A/V synchronization, on the end user experiences [1]. Their results showed that the spatial and temporal A/V synchronization indexes are more detrimental than the video and audio indexes, so that it is crucial for streaming service providers such as Prime Video and IMDB to build automatic A/V off-sync detector to catch this type of errors quickly.

Building binary classifiers is one direction in this domain. For example, R. Arandjelovic and A. Zisserman [2] researched the video-audio correspondence by constructing a vision subnetwork and an audio subnetwork, and then fusing the features and then predicting whether the input A/V pair corresponds. Bruno *et al.* presented a cooperative learning of A/V model [3] that computes contrastive loss from the output pairs of the video subnetworks and the audio subnetworks, and reported that it improved the activity recognition accuracy. F. Xiao *et al.* also proposed a similar method in their AV slowfast network [4] that fuses video-audio streams from different clips or from different time stamp of the same clips in order to boost the accuracy of activity recognition. However, these binary classifiers are usually designed to detect spatial A/V sync errors or temporal A/V sync errors with a precision of ~ 2 seconds or longer. But for online streaming services, the primary goal is to detect temporal A/V synchronization errors that are human perceivable. According to the International Telecommunication Union study [5], humans can tell A/V are off-sync if an audio stream is +50 milliseconds (ms) ahead or -125 ms behind its video stream.

Researchers have also studied on algorithms to detect temporal A/V sync errors with high precision. For example, J. Chung and A. Zisserman [6] presented a lip-sync based algorithm. In the video pathway, it takes in a video clip with face region cropped out. In the audio pathway, it computes the Mel-frequency cepstral coefficient (MFCC) [7], [8]. Then it correlates the features between the video data and the audio data to estimate the A/V off-sync value. More recently, J. Ebenezer *et al.* presented an algorithm to detect the A/V sync error [9] in online streamed tennis games. Briefly speaking, this algorithm first detects the tennis hit sound events from the audio data. Then for each detection, it looks into the video data using the detected time stamp to verify if it is really a tennis hit. The authors showed that this algorithm is able to detect A/V sync errors with audio signals +80 ms ahead or -240 ms behind video signals.

Today, A/V off-sync detection researches are mainly focused on contents such as movies or live sports. Detecting this type of errors during Ads time, on the other hand, has received little attentions. Compared with a content video, an Ad video has the following characteristics: first, its vocals are either from background narrators or have a very short period of time, so that lip-sync based algorithms won't work well; second, it does not contain a specific event, such as a ball hits in a tennis game, that can be used to correlate the sound and video signals. So the literature algorithms described above won't be reliable when applying into Ads videos. In this paper, we present a novel algorithm to tackle this problem. The main contributions of our work include: (i) we present an algorithm that only uses audio data to segment out individual Ad from a video playback, and (ii) we present a novel heat map algorithm

TABLE I
THE ACRONYMS USED IN THIS PAPER.

Acronym	Meaning
SS	Silent Segment
CP	Change Point: Content $\leftrightarrow$ Ad or $Ad_i \leftrightarrow Ad_j$
LMS	Log Mel Spectrogram
CL_{CP}	Change Point Classifier
AS	Ads section
CT_V/CT_A	Video/Audio scene change time
HM	Scene change time heat map
$Off_{SyncNet}$	A/V off-sync time computed from SyncNet model
$CF_{SyncNet}$	confidence score from SyncNet model

that is able to capture the A/V off sync time from the Ad transitions. By combining the heat map model with a lip-sync based SyncNet model [6], we show that the algorithm has significantly improved the performance.

The following of the paper is organized as follows: in Sec. II- IV we describe the algorithm in detail, in Sec V we evaluate the performance of our algorithm, and in Sec. VI we will conclude the paper and discuss the future studies. To improve readability, we listed the acronyms in Table. I.

II. Audio Only Ad Segmentation Algorithm

To compute the A/V sync errors, we need first segment out individual Ad. Our segmentation algorithm is shown in Fig. 1. It starts with searching silent short segments (SS) from the audio data, based on our observation that *there is usually a short transition time between a content and an Ad or a pair of Ads*. Here an SS is defined to be a time period such that the maximum volume of the audio signals within it is less than a threshold. In our algorithm, we set the threshold empirically: the minimum duration of SS is set to 10 milliseconds and the volume threshold is set to 4 after we normalize the wav data into $int16$ type and then take the absolute values. To clean up the noise, we apply a band pass filter of $[300HZ, 6000HZ]$ [10] to the input audio data.

Then, we check whether the SS is a CP. Here we first extract the audio clips of length win: wav_1 and wav_2 that are before and after SS. Note that the value of win is important for the downstream classifier because if it is too small, we don't have enough information; but if it is too big, then it may contain another scene changes that will confuse the classifier. We will study win value in Sec. V. Next, we compute the Log Mel Spectrogram (LMS) [7], [8] for both wav_1 and wav_2 and get $2D$ audio cues as LMS_1 and LMS_2, respectively. Next, we extract the a 512 dimensional feature vectors from LMS_1 and LMS_2, respectively. Here we picked the modified ResNet-34 presented by University of Oxford Robotic Lab's SpeakerID framework. Briefly speaking, this model starts with a 7×7 convolution layer, followed by 3×3 max pooling layer, which is then followed by four residual net blocks. Then it performed a temporal Self-Attentive Pooling [11] and a fully connected layer with output dimension 512. For details of this network, please refer to A. Nagrani *et. al*'s paper [12], [13], [14]. Then we compute the Eulidean distance between

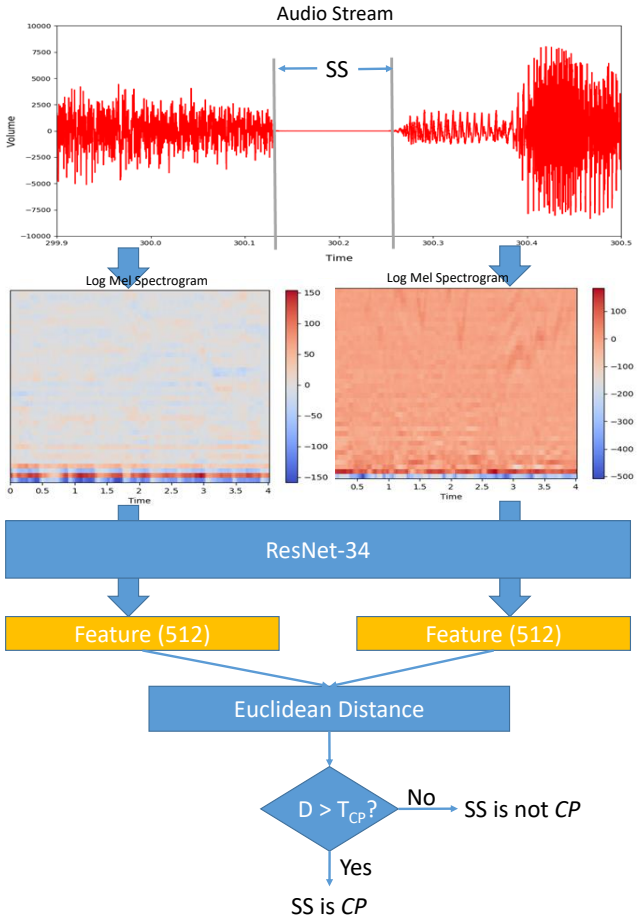

Fig. 1. The flow chart of our audio only playback segmentation algorithm that finds a *silent segment* (SS) first and then classifies whether it is a *change point* (CP).

the feature vectors of LMS_1 and LMS_2, and label SS as a CP if its value is below a pre-defined threshold T_{CP}.

We picked the Siamese network architecture for the CP classifier (CL_{CP}) so that we can easily add new training videos without having to change the network structure. Given a training audio clip i, we first find all SS inside it, then randomly select one to create a non-CP (positive) pair $(LMS_1(i), LMS_2(i))$. If no SS is found, then we randomly select a time stamp to create a positive pair. The CP (negative) pairs can be obtained by sampling audio clips from different playbacks: $(LMS_1(i), LMS_2(j))$. Since a transition can happen from content to Ad and between individual Ads, we included both Ad and content clips in the training stage. For the loss function, we studied both the triplet loss [15] as defined in Eq. 1 and the contrastive loss [16], [17] as defined in Eq. 2. The result as listed in Sec. V showed that triplet loss has better performance. Note that for a training batch of size NB, we have NB non-CP (positive) pairs and $(NB-1)^2$ CP (negative) pairs. So we also performed hard data mining as described in [12] during training stage to achieve better performance. Transfer learning is employed where we picked

the weights pre-trained on the VoxCeleb2 dataset [13] that contains a million utterances from $6,112$ different people. The pre-trained model we used is listed in [18], [19].

$$\begin{aligned} L(A,P,N) &= max(L(P) - L(N) + \alpha, 0) \\ L(P) &= ||f(A) - f(P)||^2 \\ L(N) &= ||f(A) - f(N)||^2 \end{aligned} \quad (1)$$

The triplet loss function where A is an anchor input, P is its non-CP pair (from the same video), N is its CP pair (from a different video), α is a margin between CP and non-CP pairs that is usually set to 1, and f is the feature extractor (ResNet-34 network).

$$\begin{aligned} L(X_1, X_2) &= L(1) + L(2) \\ L(1) &= (1-Y)\frac{1}{2}(||f(X_1) - f(X_2)||^2) \\ L(2) &= Y\frac{1}{2}max(0, \alpha - ||f(X_1) - f(X_2)||^2) \end{aligned} \quad (2)$$

The contrastive loss function where X_1 and X_2 are the input pairs of CP classifier, $alpha$ is the margin between CP and non-CP pairs that is usually set to 1, and f is the feature extractor (ResNet-34 network in our algorithm). $Y = 0$ if X_1 and X_2 are a non-CP pair (from the same playback), and $Y = 1$ if they are a CP pair (from different playbacks).

After SS detection and CP classification, our algorithm outputs a list of CP: $[CP_1, CP_2, CP_3, ..., CP_n]$. These CP split the input playback into $n+1$ segments: $\{SG_0 = [0, t_{11}], SG_1 = [t_{12}, t_{21}], SG_2 = [t_{22}, t_{31}], ..., SG_n = [t_{n2}, t_{end}]\}$. Here t_{i1}, t_{i2} are the starting time and ending time of CP_i, and t_{end} is the ending time of the playback.

Next, we label SG as an Ad segment if its duration is less than 85 seconds: an empirically picked value as we observed that most Ads are shorter than 60 seconds. Then we group a set of temporally adjacent Ad segments into one Ad section (AS), which is the base unit in our algorithm to compute the A/V sync error metric. Note that only using duration feature to classify Ad may produce a few false positives. But such error does not affect our downstream computation as we assume the A/V off-sync time within one AS is constant.

III. A/V Scene Change Time Heat Map

There is a scene change between every pair of temporal neighboring Ads. If the A/V is in-sync, then the state transition time from the video data and the audio data should have the same value, otherwise they will be different. In our algorithm, we assume that the A/V off-sync time within an AS are constant, so that the state transition time difference between audio data and video data should have similar values across all Ads pairs inside one AS.

For the video data, we run open source algorithm available at [20] that looks at HSV color feature changes to compute the scene change time $CT_V = [t_1(V), t_2(V), ..., t_n(V)]$. For the audio data, we simply use the time of the detected SS points: $CT_A = [t_1(A), t_2(A), ..., t_m(A)]$ where $t_i(A)$ is the time when a silent segment SS_i as described in Sec. II is detected. Then for each $t_i(A)$, we find all $t_j(V)$ using a search window T_{CT} centered at $t_i(A)$, and construct a heat map (HM) as

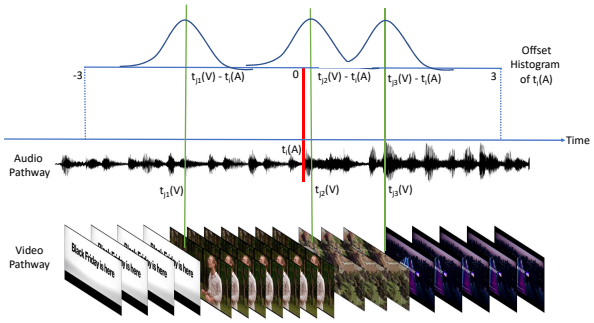

Fig. 2. An example of building a heat map for $t_i(A)$. For each $t_j(V)$ within the searching range, we use the Gaussian function to increment the histogram. The sample images listed here are publicly available online.

Fig. 2 showed. Note that for each $t_j(V)$ inside T_{CT}, we used the Gaussian function centered at $t_i(A) - t_j(V)$ to increment HM. If t_k in HM is covered by multiple Gaussian functions, e.g., in Fig. 2 there is an overlapping area between $t_{j2}(V)$ and $t_{j3}(V)$'s Gaussian functions, then we will take the maximum value in the overlapping area. In our algorithm, the T_{CT} is set to 6 seconds because (i) most A/V sync errors in reality are within ± 3 seconds, and (ii) the detected CT_A and CT_V may contain scene changes other than state transition between Ads pairs, so that setting a large window will include more noises.

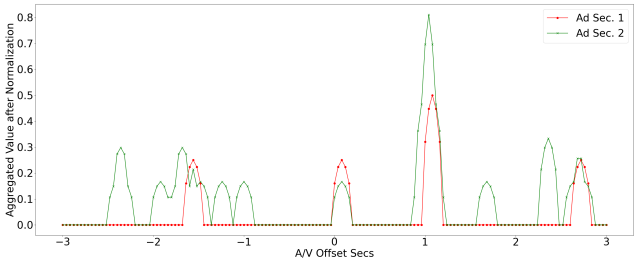

Fig. 3. A sample plot of the aggregated heat map for two Ad sections that both have A/V sync error of one second. For better comparison, both heat maps have been normalized by the number of Ads in its Ad section.

After we built HM for each Ad section, we aggregated all HM within one AS. Ideally, if the A/V are off-sync by t_n seconds ($t_n = 0$ if they are in-sync), then all HM from Ad transitions should have largest heat value at t_n. As a result, the aggregated HM should have a peak at t_n with value k where k is the number of Ads in the AS. On the other hand, HM from non Ads transition will have randomly distributed peaks. So that the aggregated HM should have a peak at t_n with much higher value than the other local maxima. However, in reality an aggregated HM may contain multiple local maxima that have close values because (i) the computed $t_i(A)$ and $t_j(V)$ contain noises so that the HM from different neighboring Ads pairs do not have similar heat values, and (ii) HM from non Ads transition can also have alike heats. For example,

Fig. 3 listed the aggregated HM of two AS sampled from our dataset that both have A/V off-sync by one second. For better comparison, we normalized the HM by the number of Ads in each AS. We can see that AS_2 (green line) showed a good case: it has a strong heat at 1 sec. offset time with value $> 80\%$ and its other values are much less heat: at or below 30%. AS_1 (red line), on the other hand, showed a bad case because its heat value at 1 sec offset time is only 50% and it also contains several other heat values around 30% at 0, -2.5 and 2.5 offset seconds. This suggested that we need to be careful in selecting the value from HM.

IV. A/V Offset Value Estimation

To help select peak from an aggregated HM described above, in our algorithm we employed the SyncNet model [6], a popular lip-sync based A/V sync detector. Since this model takes in a face video clip, we built a pre-process step that first detects face regions from each image in a video using the MTCNN model [21], then tracks each face by grouping the bound boxes with the maximum spatial overlaps over time. Next, for each tracked face, it creates one video clip by cropping out the face region and then adding the audio data of the same time period. For every AS, we run the pre-process step to create a list of face video clips, and then run the SyncNet model to process each of them so that we get a list of A/V off-sync secs ($Off_{SyncNet}$) and their confidence scores ($CF_{SyncNet}$). Then we pick the $Off_{SyncNet}$ value with the largest confidence score and return it as the SyncNet model prediction for this AS.

Next, we build a rule engine to combine the aggregated HM with the $Off_{SyncNet}$ and its $CF_{SyncNet}$ score. The steps are listed in Alg. 1. We can see that if the $CF_{SyncNet}$ is high enough, we directly report the $Off_{SyncNet}$ value as the predicted A/V off-sync time. If the $CF_{SyncNet}$ is moderate, then we search the HM to find a local maxima near $Off_{SyncNet}$ and then report their mean value as the A/V offset time. If the $CF_{SyncNet}$ is low, then we only use the peaks in the HM. Note that the algorithm generates three types of output: (i) the A/V off-sync seconds when we have at least one high confidence value from the SyncNet and HM; (ii) binary output: A/V in-sync or off-sync, when we don't have a high confidence value but have at least one moderate confidence value, (iii) *unknown* when we only have low confidence values from both of them.

Algorithm 1 Compute A/V off-sync value from the scene change time heat map and the SyncNet model output.

procedure CompOffSync($HM, Off_{SyncNet}, CF_{SyncNet}$)
  // Initialize the thresholds
  $T_{Sync}(H) \leftarrow 3.0$, $T_{Sync}(L) \leftarrow 2.1$
  $T_{Off}(H) \leftarrow$(# of audio scene changes in HM) / 2
  $T_{Off}(M) \leftarrow max(HM) \times 0.9$
  $T_{Off}(L) \leftarrow max(HM) \times 0.5$
  $T_{insync}(W) \leftarrow 0.12$ secs
  if $CF_{SyncNet} > T_{Sync}(H)$ **then**
    return $Off_{SyncNet}$
  if $CF_{SyncNet} > T_{Sync}(L)$ **then**
    $W \leftarrow [Off_{SyncNet} - 0.12secs, Off_{SyncNet} + 0.12secs]$
    // Find the peak of HM within search window
    $t \leftarrow argmax_{t \in W}(HM(t))$
    if $HM(t) > T_{off}(L)$ **then**
      return $(Off_{SyncNet} + t)/2$
  $S \leftarrow [t] \forall_t \in HM(t) > T_{offset}(M)$
  if S size = 1 **then**
    // Single peak in HM
    $t \leftarrow S[0]$
    if $HM(t) > T_{Off}(H)$ **then**
      return t
    else if $HM(t) > T_{Off}(L)$ **then**
      // We do not have enough information to compute the offset time
      if $t > T_{insync}(W)$ **then**
        Return Video is ahead of Audio
      else if (**then**$t < -T_{insync}(W)$)
        Return Video is behind Audio
  else if S size > 1 **then**
    // Multiple peaks in HM
    if $t < -T_{insync}(W) \forall t \in S$ **then**
      // All peaks are negative
      Return Video is behind Audio
    else if $t > T_{insync}(W) \forall t \in S$ **then**
      // All peaks are positive
      Return Video is ahead of Audio
  Return Unknown

V. Experiments and Performance Evaluation

A. Dataset Description

As we are not able to find a literature dataset for Ad detection, we created our own dataset for performance evaluation. It consists of two parts: (i) short video playbacks with 1 to 5 minutes long each. They were collected from the IMDB lab, the Prime Video library and so on. These playbacks are further broken into sub-playbacks of $10-30$ seconds for better scene coverage in the training and testing process. (ii) Long video playbacks with each 20 to 120 minutes long. They are provided by the IMDB and PV library catalog that consist of 32 movies of different titles and 16 sport events.

B. Training Process Description

We used the short video dataset (part 1) to perform training CL_{CP}. Specifically, we selected 70% of the sub-playbacks as the training data and 15% as the validation data. To have a balanced data points across classes, we performed down-sampling so that the ratio between samples of Ad and Content is within 0.9 to 1.1. During the training process, we iterate for 196 epochs and pick the one with the best validation performance. Transferred learning is employed. Specifically,

TABLE II

THE NUMBER OF VIDEO SUB-PLAYBACKS IN THE TRAINING, TESTING AND VALIDATION SET FOR CL_{CP}. THESE SUB-PLAYBACKS WERE CREATED FROM THE SHORT VIDEO PLAYBACKS (PART 1) WITH 1 TO 5 MINUTES LONG EACH.

	Ad Count	Content Count
Train	5609	5367
Test	1038	1154
Validate	1040	1154

we initialized the CL_{CP} shown in Fig. 1 using the pre-trained model provided in [18].

C. Scene Change Classifier Performance Evaluation

We evaluated the performance of CL_{CP} using the remaining 15% of sub-playbacks of the short video dataset (part 1). Similar to the training process, we also performed data down-sampling to have a balanced dataset. Specifically, we have 1038 content clips and 1154 Ad clips from 121 unique playbacks.

To evaluate CL_{CP}, we sequentially selected one sub-playback from each playback, randomly sampled a pair of temporally adjacent video clips from it and then computed their feature vectors. By iterating all testing playbacks, we got 121 pairs of feature vectors. Then we computed the distance matrix of 121×121, where the diagonal 121 values are non-CP distances and the remaining off-diagonal values are CP distances. Since one playback may have up to 38 sub playback, we repeated this process 38 times to iterate over these sub playbacks, and then concatenate all the distances. So in total we got $121 \times 121 \times 38 = 556358$ distances with $121 \times 38 = 4598$ of them are non-CP type.

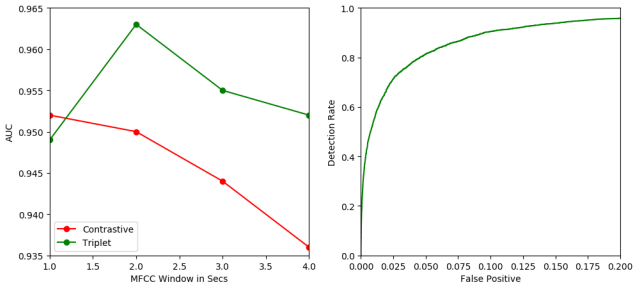

Fig. 4. The performance study of CL_{CP} on the testing dataset with a total of 556358 distances. The left figure is the Area Under Curve (AUC) for models trained using different values of win and loss functions L_{tr} and L_{ct}, and the right figure is the ROC curve up to 0.2 false positive rate for the model trained using the triplet loss and window size of 2 seconds.

Fig. 4 lists the performance of LC_{CP}. The left sub-figure lists the impacts in terms parameter values of win, L_{tr} as defined in Eq. 1 and L_{ct} as defined in Eq. 2. We can see that L_{tr} has better performance than L_{ct} overall. We also see that CL_{CP} has best accuracy when win is set to 2 seconds. This confirms our hypothesis proposed in Sec. II. The right sub-figure plots the ROC curve when L_{tr} is used and win is set to 2 seconds. We can see that the $\sim 90\%$ detection rate is achieved at 10% false positive rate.

TABLE III

THE PERFORMANCE OF THE MODEL IN TERMS OF PREDICTION RATES FOR THE SYNCNET MODEL AND OUR ALGORITHM ON DIFFERENT A/V OFF SYNC TIME (t_{off}). THE TOTAL NUMBER OF AD SECTIONS IN THE DATASET IS 128.

t_{off} (Secs)	SyncNet Prediction Count	SyncNet Prediction Rate	Ours Prediction Count	Ours Prediction Rate
-1	65	50.80%	**85**	**66.40%**
-0.8	76	59.40%	**95**	**74.20%**
-0.6	77	60.20%	**95**	**74.20%**
-0.4	79	61.70%	**95**	**74.20%**
-0.2	76	59.40%	**97**	**75.80%**
0	70	54.70%	**99**	**77.30%**
0.2	75	58.60%	**94**	**73.40%**
0.4	80	62.50%	**95**	**74.20%**
0.6	76	59.40%	**97**	**75.80%**
0.8	77	60.20%	**90**	**70.30%**
1	70	54.70%	**85**	**66.40%**
Total	821	58.30%	**1027**	**72.90%**

D. End-end algorithm Performance Evaluation

We also performed end-end test on the 48 long playbacks (part 2 of our database) that consists of 128 Ad sections in total. These playbacks include both movie contents and live sports contents. We first manually verified that all these playbacks are A/V in-sync. Next, we artificially shifted the audio signals in order to generate the testing data with A/V synchronization errors. To have a better understanding on our algorithm's performance on a specific A/V off-sync value, we shifted the full dataset's audio signals using a fixed time t_{off} instead of a random time for each video playback, and then we used t_{off} as the ground truth to validate the algorithm on the synthesized data. Starting from -1 second (audio is head of video by 1 second), we hop t_{off} by 0.2 seconds each time, till it reaches 1 second (audio is behind video by 1 second).

Our first study is the model prediction rate: the percentage of input data that the model makes a prediction instead of output *Unknown* due to low confidence. Table. III compares performances between the SyncNet model and our algorithm. We can see that by using our algorithm, the prediction rate increased by 14.6% in average. Particularly, when the A/V is in-sync (the row when t_{off} is zero in Table. III), we saw the prediction rate increased by 23%.

Our next study is the model accuracy in terms of the binary level classification. That is, for the algorithm outputs, we excluded the *unknown* ones and relabeled the others as *in-sync* or *off-sync* based on their values. Table. IV summaries the results for different A/V off-sync time (t_{off}). Note that the ground truth is *in-sync* for the row where t_{off} is zero and *off-sync* for the other rows. We can see that both models achieved over 95% or above accuracy for most t_{off} except for it has a small value: -0.2 and 0.2 seconds, respectively. We also see that our algorithm not only substantially boosts up the prediction rate, but also achieves similar accuracy with the SyncNet model for most t_{off} values except when t_{off}

TABLE IV

THE PERFORMANCE OF THE MODEL ACCURACY IN TERMS OF THE BINARY LEVEL CLASSIFICATION: OFF-SYNC VS IN-SYNC, FOR THE SYNCNET MODEL AND OUR ALGORITHM ON DIFFERENT A/V OFF SYNC TIME (t_{off}). THE TOTAL NUMBER OF AD SECTIONS IN THE DATASET IS 128.

t_{off} (Secs)	SyncNet Prediction Accuracy	Ours Prediction Accuracy
-1	50.80%	**66.4%**
-0.8	59.4%	**74.2%**
-0.6	60.2%	**74.2%**
-0.4	61.7%	**74.2%**
-0.2	**45.3%**	44.5%
0	51.6%	**75%**
0.2	42.2%	**52.3%**
0.4	62.5%	**72.7%**
0.6	59.4%	**75%**
0.8	60.2%	**69.5%**
1	54.7%	**66.4%**

is 0.2. We studied the videos with in-accurate prediction and found that some videos have longer the Ad transition time, so that when the audio data are slightly off from the video data, their A/V scene change times still align so that the algorithm falsely outputs the in-sync results. Fig. 5 lists one example of such error: when the SS is 0.5 seconds, the A/V scene change time differences are still zero after we shifted the video data forward or backward 0.2 seconds.

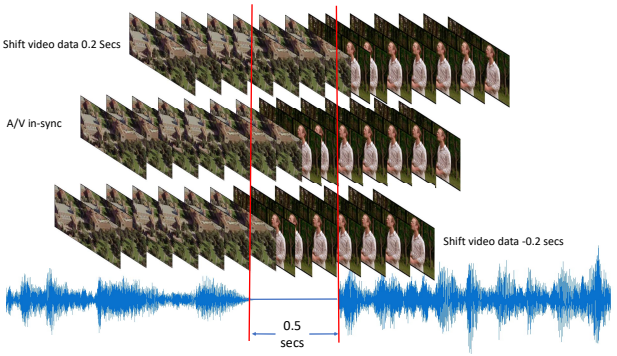

Fig. 5. An example of an playback clip with audio transition time (SS) 0.5 secs long, so that the A/V transition time are still aligned after we shifted the video data forward and backward 0.2 secs. The sample images listed here are publicly available online.

Our third study is the model accuracy in terms of the difference between the predicted off-sync time and the ground truth off-sync time. Table. V lists the results for different t_{off}. It suggests that our algorithm has improved accuracy. Specifically, the percentage of error within 0.2 secs has increased from 94.2% to 95.9%, and the percentage of error over 0.5 secs dropped from 4.3% to 1.9%.

VI. CONCLUSION AND FUTURE STUDY

In this paper, we presented an algorithm that captures the A/V off-sync errors during Ads time using the scene change time difference between the video data and the audio

TABLE V

THE PERFORMANCE OF THE MODEL ACCURACY IN TERMS OF THE ERROR IN SECS BETWEEN THE PREDICTED OFF-SYNC TIME AND THE GROUND TRUTH OFF-SYNC TIME FOR THE SYNCNET MODEL AND OUR ALGORITHM ON DIFFERENT A/V OFF SYNC TIME (t_{off}). THE TOTAL NUMBER OF AD SECTIONS IN THE DATASET IS 128.

t_{off} (Secs)	SyncNet Algo.		Our Algo.	
	Error < 0.2	Error $>= 0.5$	Error < 0.2	Error $>= 0.5$
-1	86.2%	12.3%	**88.8%**	**5.0%**
-0.8	93.4%	5.3%	**93.7%**	**2.1%**
-0.6	**96.1%**	3.9%	94.7%	**1.1%**
-0.4	92.4%	6.3%	**94.7%**	**4.2%**
-0.2	96.1%	3.9%	**99.0%**	**1.0%**
0	100.0%	0.0%	100.0%	0.0%
0.2	97.3%	2.7%	**100.0%**	**0.0%**
0.4	92.5%	3.8%	**95.7%**	**0.0%**
0.6	**97.4%**	**1.3%**	96.8%	3.2%
0.8	93.5%	2.6%	**94.3%**	**2.3%**
1	90.0%	5.7%	**95.1%**	**2.5%**
Avg.	94.2%	4.3%	**95.9%**	**1.9%**

data. The algorithm starts with segmenting Ads out from an video playback using the audio signal only. Then it groups temporally adjacent Ads into Ad sections. Next, it computes the scene change time for the video data and audio data separately and builds a histogram of time difference between these two input sources. After that, it combines the histogram with a popular lip sync based model (SyncNet) to compute the A/V off-sync time for a video playback.

We tested the algorithm using two datasets: a short video clips dataset for the audio based change point classifier CL_{CP}, and a long video playback dataset for end-end testing. The results showed that our CL_{CP} achieved 90% detection rate at 10% false positive by only using audio data. It also showed that compared to the SyncNet model, our algorithm not only increases the predication rate 23%, but also increases the prediction off-sync value by 1.7% as well as reducing the large prediction error ($> 0.5secs$) by 2.5%. Since the SyncNet model generally has confidences and descent accuracy on vocals of long period, we can conclude that the A/V scene change time heat map has improved the detection on the video clips with very short vocals or background narrators.

In terms of the future study, there are several directions: (i) improve the precision of our algorithm. Today, we saw that the combined algorithm gets higher errors when the A/V off-sync time is small (± 0.2 seconds). So we will investigate how to improve the HM computation to better handle videos with long scene change time as shown in Fig. 5. (i) Our algorithm assumes that the A/V off-sync values are constant within one AS, so another direction will be how to handle incremental A/V off-sync errors. (iii) The A/V offset value estimation part as listed in Alg. 1 is rule based and it is worth replacing it with an algorithm that is trainable. And (iv) we are working on releasing a long video large Ads dataset so that the research community can re-produce the results of this work and dive deeper in this domain.

REFERENCES

[1] J. Zhao, B. Zhang, Z. Yan, J. Wang, and Z. Fei, "A study on the factors affecting audio-video subjective experience in virtual reality environments," in *2017 International Conference on Virtual Reality and Visualization (ICVRV)*, 2017, pp. 303–306.

[2] R. Arandjelovic and A. Zisserman, "Look, listen and learn," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 609–617.

[3] B. Korbar, D. Tran, and L. Torresani, "Cooperative learning of audio and video models from self-supervised synchronization," *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, p. 7774–7785, 2018.

[4] F. Xiao, Y. J. Lee, K. Grauman, J. Malik, and C. Feichtenhofer, "Audiovisual slowfast networks for video recognition," 2020.

[5] "Iinternational telecommunications union," https://www.itu.int/rec/T-REC-J.248/en, 2008.

[6] J. S. Chung and A. Zisserman, "Out of time: automated lip sync in the wild," in *Workshop on Multi-view Lip-reading, ACCV*, 2016.

[7] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 28, no. 4, pp. 357–366, 1980.

[8] P. MERMELSTEIN, "Distance measures for speech recognition, psychological and instrumental," *Pattern Recognition and Artificial Intelligence*, pp. 374–388, 1976. [Online]. Available: https://ci.nii.ac.jp/naid/10026808024/en/

[9] J. P. Ebenezer, Y. Wu, H. Wei, S. Sethuraman, and Z. Liu, "Detection of audio-video synchronization errors via event detection," 2021.

[10] "Voice frequency," https://en.wikipedia.org/wiki/Voice_frequency.

[11] W. Cai, J. Chen, and M. Li, "Exploring the encoding layer and loss function in end-to-end speaker and language recognition system," 04 2018.

[12] A. Nagrani, J. S. Chung, W. Xie, and A. Zisserman, "Voxceleb: Large-scale speaker verification in the wild," *Computer Science and Language*, 2019.

[13] J. S. Chung, A. Nagrani, and A. Zisserman, "Voxceleb2: Deep speaker recognition," in *INTERSPEECH*, 2018.

[14] A. Nagrani, J. S. Chung, and A. Zisserman, "Voxceleb: a large-scale speaker identification dataset," in *INTERSPEECH*, 2017.

[15] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 815–823.

[16] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, 2005, pp. 539–546 vol. 1.

[17] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2, 2006, pp. 1735–1742.

[18] J. S. Chung, J. Huh, S. Mun, M. Lee, H. S. Heo, S. Choe, C. Ham, S. Jung, B.-J. Lee, and I. Han, "In defence of metric learning for speaker recognition," in *Interspeech*, 2020.

[19] http://www.robots.ox.ac.uk/∼joon/data/baseline_lite_ap.model.

[20] https://pyscenedetect.readthedocs.io/en/latest/.

[21] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks," *IEEE Signal Processing Letters*, vol. 23, no. 10, pp. 1499–1503, 2016.