ASSESSMENT OF AUDIO/VIDEO SYNCHRONISATION IN STREAMING MEDIA

François Pitié[1], *Damien Kelly*[2], *Thierry Foucu*[2], *Naomi Harte*[1] *and Anil Kokaram*[2]

[1]Trinity College Dublin
Ireland

[2]Google/YouTube
Mountain View, CA

ABSTRACT

Quality assessment in the streaming media industry has matured to the stage that it encompasses not only traditional notions of pixel and audio sample integrity but also file format consistency and the media consumption experience itself. Audio/Video synchronisation has already been established as an associated measure of media quality and is well known in video conferencing and movie streaming applications. This paper presents a new system for the assessment of audio and video synchronisation in a media file. The system incorporates the idea of learning features which are robust to coding artefacts to establish robust fingerprints for A/V Sync measurement. Results from large scale testing of 30,000 clips from YouTube show why measurement of A/V Sync is important for file based video repositories and highlights issues that can now be addressed quantitatively.

Index Terms— Lip sync, audio/video synchronisation, LDA, h.264, vp9

1. INTRODUCTION

Quality control for large scale ingest of media/movie titles is a key component of the activities of any media publishing house. The rise of streaming media distribution direct to the consumer has meant that this step in ingest now extends beyond DVD/BluRay publishing to the large scale streaming businesses of Hulu, Netflix, GooglePlay, YouTube, Vimeo and Amazon. On ingest, it has become typical to verify container integrity and compressed file formats because studios are not reliable in creating the production media packages which must be supplied to the distributors. A litany of problems are typical e.g. missing or incorrect audio streams, wrongly formatted picture framing, frame dropouts, missing/incorrect scenes/captions. A small industry has sprung up to supply automated QC checks for the vast amount of data that has to be handled on ingest e.g. Interra, Tektronix all supply QC equipment of various types. One problem that frequently occurs not only on ingest but also after the many layers of transcoding applied by a distributor is the loss of synchronisation between the audio and visual streams of the same media (A/V-sync). This is also an issue in live streaming. There has been little published work on reliably detecting and measuring this phenomenon.

This paper presents a system for measuring the amount of change in A/V-sync between a reference media record and some processed record. The processed record might come from the client in a live streaming application or from a video-on-demand service or simply be at the end of one of the internal transcoding/processing steps in a distribution chain. We first present and overview of the process and then the audio and video features used, and finally the algorithm for matching. We test our system in a control group of 80 videos, randomly sampled from YouTube, and subjected to varying distortions. We show that our algorithm can detect at any time A/V-sync changes of less than 40ms with up to 99.77% precision, and this in the most extreme distortion scenarios. We also discuss the types of distortion which pose significant challenge to our methods.

Finally, we apply this quality metric in a more interesting environment. We report on large scale testing of about 30,000 videos from YouTube. We measured the A/V-sync between videos in the h264-720p/aac format and other format transcoded by YouTube (Vorbis/VP9-360p, AAC-HE/H264-144p and AAC-LC/H264-480p). Results shows why measurement of A/V-sync is important and highlights issues that can now be addressed quantitatively.

2. OVERVIEW

In this paper we are concerned with monitoring the audio to video synchronisation (also called A/V sync or lip sync) of a video that goes through a transcoding system. Consider what happens when uploading a video to a social media platform: the video must be converted to different formats that target multiple hardware. Due to different interpretations of codec standards between encoders and decoders, delays may be introduced in either the audio or the visual streams of the video, and thus generating lip-sync errors.

What is proposed here, is a signal processing solution that generates fingerprint streams out of the reference audio and visual streams (eg. as uploaded by the artist) and com-

This work was supported by Enterprise Ireland Project IP-2013-0232 "Advanced Metrics for Audio-Visual Signal Quality in Internet Communications" and Google/YouTube Research@Development Targeted Research Funding. Email: fpitie@mee.tcd.ie

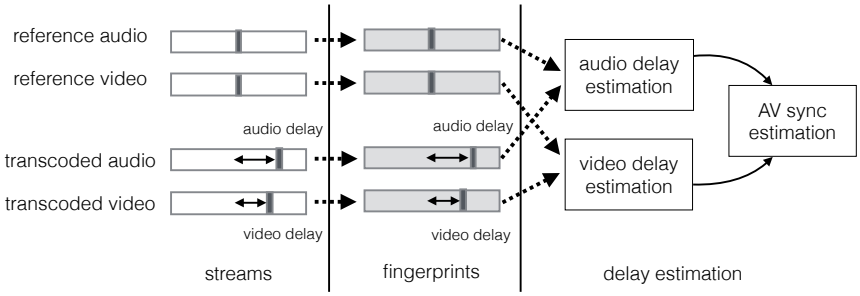

Fig. 1. System Overview.

pares them against the fingerprints of the transcoded medium (see Fig. 1). A fingerprint stream consists of a sequence of time stamps (called PTS in MPEG streams) paired with feature vectors that represent the audio/visual frame at that time stamp. By comparing fingerprints between the reference and transcoded audio/visual streams, delays in each stream can be estimated, and an overall audio/visual sync delay measured.

Similar approaches have been presented in the literature. See for instance the work of Kudrle et al. [1] and Radhakrishnan et al. [2] in the context of broadcast production. The novelty of our system resides in the choice of audio/visual features and in the delay estimation mechanism, which combined together can cope with the wide range of audio/visual degradations observed in social media platforms.

3. FEATURES

Prior art publications have shown multiple implementations for the generation of multimedia fingerprints. As part of multimedia source identification systems, several works have also presented mention of fingerprinting (or robust hash) audio and visual streams. Visual fingerprints can use brightness information, frame to frame differences as in Carrires et al. [3] and Radhakrishnan et al. [4], motion or color histograms. For audio fingerprints, some techniques look at points of interests in the spectrogram (e.g. maximums and minimums). These techniques are however more suited to identification purposes as they provide discrete signatures, which are not practical when dealing with continuous measurements. More relevant to the delay estimation application are the techniques that are based on mapping the audio signal into a lower dimensional space, such as in works of Radhakrishnan et al. [4] and Srinivasan et al. [5], where projections and moments for the audio spectrogram are considered.

3.1. Audio Fingerprinting

In our system all audio inputs are normalised to 44.1KHz, mono. The audio feature vector (or fingerprint) consists of linear combinations of the power spectrum density (PSD) coefficients over overlapping windows of 1024 samples. The PSD coefficients are obtained by taking the norm of the Short-Time Fourier-Transform coefficients over a window. PSD coefficients that are higher than 11KHz are discarded as they are proned to be altered by codecs.

To find linear combinations of PSD coefficients that are robust to common coding artefacts, a Linear Discriminant Analysis (LDA) was used in this case. Similar approach was made in Burges et al. [6] in the context of Audio identification. Here the noise profile is obtained by transcoding original audio files to different codec formats.

We sampled $N = 70000$ audio feature vectors across 80 videos. Each feature vector contains the P bands of the audio PSD. Each of the samples had $K = 4$ versions: $k = 1$ for original and $k = 2, 3, 4$ for Vorbis 64kbps/AAC 64kpbs and MP3 64kpbs.

In the following f_i^k refers to the ith feature vector for the kth version. e.g. f_i^1 is a feature vector of dimension P, corresponding to the original version of sample i .

We want to find a linear mapping that maximises the class separation between the "signal" and the "compression noise". With LDA, this done by maximising

$$S = w^t C w / w^t R w \tag{1}$$

where C is the covariance matrix of the clean signal class and R the covariance matrix of the compression noise.

The covariance matrix for the "compression noise" is computed as follows.

$$R = \frac{1}{N(K-1)} \sum_{i=1}^{N} \sum_{k=2}^{K} (f_i^k - f_i^1)(f_i^k - f_i^1)^t \tag{2}$$

It is assumed here that the noise has zero mean.

The covariance matrix for the "signal" is simply the covariance matrix of the original features. If we define X as the $N \times P$ data matrix, i.e. $X = [f_1^1, f_2^1, \cdots, f_N^1]^t$, then

$$C = \mathrm{cov}(X) \tag{3}$$

The LDA solution is to resolve the generalised eigenvalue problem stated as

$$\lambda C v = \lambda R v \tag{4}$$

The transform matrix V is given by the last generalised eigenvectors. Here we take the last 32 features so that

$$\tilde{F}_i^k = f_i^k V \tag{5}$$

3.2. Video Fingerprinting

The video feature vector are obtained for every frame by converting the image to grayscale on computing its vertical and horizontal integral projections. The vertical image projection is the sum of the image intensities along columns. Similarly the horizontal projection is the same along rows. The Integral projections are well-known to be a powerful tool for video and motion analysis (see work of Crawford et al. [7] in context of dominant motion estimation).

To reduce further the size of the fingerprint, only the low frequency information of the Integral projections are considered. The Integral projections are thus windowed (Hamming window) to mitigate border effects and then transformed via the Fast Fourier Transform. Only the first coefficients of the FFT of the normalized projections are kept (in this case the first 32 complex coefficients).

A notable property of this fingerprint is that it is invariant to a change of scale of the image.

4. DELAY ESTIMATION

For each input and reference Audio/Video streams, the feature extraction returns a fingerprint stream, which consists of a sequence of feature vectors, each of them being associated with a time stamp, or PTS values (Presentation Time Stamp in a MPEG transport stream).

For each of the PTSs of the input sequence, the delay can be computed by correlating the fingerprint stream of the input/transcoded video to the fingerprint stream of the reference/uploaded stream.

Maximum Likelihood Delay Estimation. The correlation is performed as follows. For a given PTS value, a window of neighbouring feature vectors (e.g. 5) is considered and matched against the reference stream. Finding the best match can be done by testing a range of possible delays (here every 5 milliseconds). The reference fingerprint is linearly interpolated for these delays and the match is evaluated between the interpolated reference and input fingerprints by means of the L1-norm (sum of absolute differences) over the window of neighbouring feature vectors. The delay that yields the least error is the maximum likelihood delay (ML delay).

If the audio or video signals are stationary (eg. white noise for audio and still frame for video), then there should be no clear best match. Hence, are discarded the PTS values, for which the best match yields a cost that is only 1.1 times less than all other possible delays that are at least 20 milliseconds away. Such matches are considered as ambiguous and no delay can be estimated from those.

Aggregated Delay Estimation. Now that a delay has been estimated for each PTS/feature vector of the input signal, delay estimates can be aggregated further to infer more robust delay estimation. Indeed, although the described features are very robust, near stationary signals may still be susceptible to erroneous matches. For an instantaneous measurement, we collect the ML delay estimates over a window of a few seconds around the considered time stamp (eg. +/- 4 seconds). For a global measurement, we would collect the ML delay estimates over the entire video.

Then, using robust techniques such as RANSAC, it is possible to fit a line though all these pairs of delay estimates/PTS and discard outlier measurements. The aim is to find an affine delay model that fits a maximum of PTS/ML delays pairs (ie. for which the delay error is less than 20ms to the model). The intercept of the line corresponds to the global delay and the intercept to the drift over time of this delay.

Combined A/V Delay Estimation. At this stage, the delay is only measured for a single stream: ie. the audio or the visual stream. To obtain the A/V sync delay, the delay must be estimated for both the audio and visual stream. The delays can be both measured for the same instantaneous time location or for the entire video. The A/V sync delay and drift (ie. variation of the delay per second) can be estimated as the difference between the video and audio estimates.

5. PERFORMANCE TESTING

5.1. Database

For measuring the performance of the system, we randomly sampled 80 HD videos from YouTube. Different techniques can be used for sampling videos from YouTube via the YouTube search API. We employ here the technique proposed by Zhou et al. [8]. They developed a simple prefix query strategy that allowed them to estimate the number of videos in YouTube to be roughly 500 million in May 2011.

5.2. Choice of Container/Codec

In this experiment, we desire to test the performance of the system against Ground-Truth data. The key issue here is that we need a guarantee that the decoder used for extracting the audio/visual features does not introduce any further A/V delays. This is a problem as we cannot know for sure that the decoder is correctly implemented.

However, we note that most of the problems arise when the input and reference streams have different containers/codec. Thus most of the problems could be removed if both input and reference streams are encoded by the same software to the same format/codec. It is reasonable to assume that both input an reference streams would be then decoded in a consistent way. There might be some errors, but the errors will be cancelled out when measuring the difference between both streams.

We chose WebM as the reference container, VP8 as the video codec and Vorbis as the audio codec.

5.3. Comparisons and Degradation Scenarios

Codec Comparisons. We tested the system for 4 types of codec/bitrate/video size combinations as follows: 1) VP8,720p,400kbps / Vorbis 128kbps, 2) VP8,480p,300kbps / Vorbis 128kbps, 3) VP8,320p,200kbps / Vorbis 64kbps, 4) VP8,144p,100kbps / Vorbis 64kbps. We chose to consider (1) as a reference stream and compare it against the 3 other settings. Thus our comparisons will include (2) against (1), (3) against (1) and (4) against (1).

Silence Delay. The first degradation considered is to insert a silence at the beginning of the audio stream. We propose 3 scenarios: no silence, a 52ms silence and a 322ms silence. This kind of padding at the beginning of the stream is typical of some codec such as AAC and can be easily mixed up by the transcoder.

PTS Warping. The second type of degradation is the manipulation the PTS values for the input stream. The first manipulation is an affine warp of the PTS values t as follows.

$$t' = t + \text{drift} \times t + \text{offset} \tag{6}$$

In our test scenario, only the audio stream was affected and the we set the audio audio drift to 0.005 ms/ms and the audio offset to -3000ms.

In the second scenario, we consider a triangular wave delay distortion as follows.

$$t' = t + A\left(4\left|\frac{t}{T} - \left\lfloor\frac{t}{T} + \frac{1}{2}\right\rfloor\right| - 1\right) \tag{7}$$

The triangular wave distortion is applied to both the audio and visual streams. The value of the maximum delay A is set to 3000ms for the audio and 1000ms to the visual stream. The value of the period T is set to 90s for the audio and 35s for the visual stream. This choice of triangle wave time warp ensures that most of the delay range is tested. Note that it is more important to test many different input videos than it is to test many different delays as there is no reason to believe that a particular delay value is intrinsically harder than others.

In total we have thus 3 codec pairs comparisons, 3 scenarios of silence insertion (none/52ms and 311ms), and 3 types of PTS warping (none/affine/triangle wave). With a database of 80 videos, the number of file comparison is thus 2160.

5.4. Results

We report here on the error measurements for the ground-truth comparisons described previously. For each comparison, we set the search window to $\pm 7s$, aggregated A/V sync delay estimates (see section4) are evaluated every second of the videos and we use a window of $8s$ for the estimation.

	Error Median	$P(Err < 40ms)$	Measure Rate
Audio	3.849 ms	99.95%	85.32%
Visual	4.140 ms	99.75%	80.01%
A/V	5.567 ms	99.77%	71.49%

Table 1. System Performance on the 80 test videos. On the first column, the median error measurement in ms. The error are compensated for the ground-truth distortions. On the second column, the percentage of measurements that have an delay error which is less than 40ms. On the third column, the percentage of measurements considered as reliable.

Unreliable Cases. Delay Estimates, for which at least 30% of corresponding ML delays are outliers, are marked as *unreliable* and removed from the global statistics. Unreliable measurements occur when the audio or the video streams are near stationary. Typically, unreliable measurements occur for music videos that have static visual content or when the audio track only contains background noise (eg. wind, traffic, etc.).

Performance. The statistics of error delays for all measurements are presented in Table 5.4. On the first column, we report the median of the error in milliseconds. Note that the errors are compensated for the ground-truth distortions. On the second column, we report the percentage of measurements that have an delay error which is less than 40ms. The 40ms threshold is chosen to match the EBU recommandation R37 [9]. According to EBU-R37, the end-to-end audio/video sync should be in the range +40ms (sound before picture) and -60ms (sound after picture) to be undetectable. On a 25Hz video, 40ms corresponds to 1 video frame. On the third column we report the percentage of reliable measurements.

From 5.4 we can see that median error for our system is around 6ms for the AV delay estimates. Also, less than 0.3% of measurements are miss-estimated by more than 40ms. Across our 80 videos, a measurement was possible for more than 70% of the time.

Since the A/V delay estimates are obtained by combining both audio and video estimates, it is expected that the overall performance of A/V delay measurement is worse than both audio and video estimates. In particular, we note that the percentage of reliable measurements is indeed less for the A/V estimates.

6. LARGE-SCALE TESTING

As a further test in a more realistic environment, we experimented our system on large scale testing files transcoded to different formats.

Setup. We looked at 30K videos from YouTube. We sampled videos uploaded to YouTube in March 2014 that are less than 5 minutes in length and originally encoded in H264 720p/AAC. The video database is split into 3 set of 10K videos. For each of the splits, we compared the 720p ver-

sion against one of the other versions proposed by YouTube. The three other transcoded formats are Vorbis/VP9-360p, AAC-HE/H264-144p and AAC-LC/H264-480p.

YouTube transcodes files by first normalising the input file to 30 fps and 44.1 KHz sampling using `ffmpeg` and then transcoding to the output format in question as a second step (see [10]).

Results. We used our system to estimate A/V sync delays caused by transcoding. The statistics for a selection of file comparisons are reported on Figure 2. We see that the distribution of A/V Sync delays shows a much more complicated picture than shown in our ground-truth tests.

Some of the peaks in the distribution can be explained by expected behaviour of codec implementations. For example, the peak observed in (a1) could be explained by 3280 samples of silence introduced by AAC-HE. This delay is added by the AAC encoder and is compensated by decoders that comply with the standard. The peak in (b1) could also be explained by the 1600 samples of silence introduced by AAC-LC. After investigation, the main peak at 200ms in (a2) for H264 AAC-HE turned out to be real and introduced by the following framerate conversion command:

```
ffmpeg -i input -v sync 1 -r 15 output
```

Surprisingly, the 200ms delay was not perceivable for the videos we investigated. This might be due to the low resolution (144p) and the low framerate (15fps).

Correctness. We use YouTube's patched version of `ffmpeg v2.2.1` to extract the audio/visual features. The correctness of our results is thus dependent on the correctness of this particular version of `ffmpeg`. Using different different combinations of decoders/transcoders might result in different delay histograms. This uncertainty means that estimated delays may not correspond in all cases to *observable* delays. However, since many multimedia players are based on `ffmpeg`, the reported delays depict a reality for some of YouTube's viewers. Thus, if there is still no way of knowing whether a transcoder/decoder is correct with respect to A/V sync, at least we are now able to identify issues between different transcoder/decoders.

7. CONCLUSION

We presented a system for quantitatively measuring A/V sync in social media platforms. The system builds on the idea of learning features which are robust to coding artefacts to establish robust fingerprints for A/V Sync measurement. We can detect at any time A/V-sync changes of less than 40ms with up to 99.77% precision.

We also sampled YouTube videos on a larger-scale and measured the A/V Sync between various transcoded versions of the videos. The resulting delay histograms show some observations that agree with the standard behaviour of codecs, and some of the observed offset would be normally compensated in a decoder. However it is unclear what percentage of the observed offsets are due to the probe itself (`ffmpeg` in this case), or the normal compensable behaviour of a decoder, or due to errors in container syntax. These issues are currently under investigation.

8. REFERENCES

[1] S. Kudrle, M. Proulx, P. Carrires, and M. Lopez, "Fingerprinting for solving a/v synchronization issues within broadcast environments," *SMPTE Mot. Imag J.*, vol. 120, no. 5, pp. 36–46, July-August 2011.

[2] R. Radhakrishnan, K. Terry, and C. Bauer, "Audio and video signatures for synchronization," in *Multimedia and Expo, 2008 IEEE International Conference on*, June 2008, pp. 1549–1552.

[3] P. Carrières, "Method and apparatus for providing signatures of audio/video signals and for making use thereof," June 2 2011, US Patent App. 12/627,728.

[4] R. Radhakrishnan, C. Bauer, C. Cheng, and K. Terry, "Audio signature extraction based on projections of spectrograms," in *Multimedia and Expo, 2007 IEEE International Conference on*, July 2007, pp. 2110–2113.

[5] V. Srinivasan, K. Deng, and D. Lu, "Audio signature extraction and correlation," Mar. 2 2010, US Patent 7,672,843.

[6] Christopher J C Burges, John C. Platt, and Soumya Jana, "Distortion discriminant analysis for audio fingerprinting," *Speech and Audio Processing, IEEE Transactions on*, vol. 11, no. 3, pp. 165–174, May 2003.

[7] A. J. Crawford, H. Denman, F. Kelly, F. Pitie, and A.C. Kokaram, "Gradient based dominant motion estimation with integral projections for real time video stabilisation," in *Image Processing, 2004. ICIP '04. 2004 International Conference on*, Oct 2004, vol. 5, pp. 3371–3374 Vol. 5.

[8] Jia Zhou, Yanhua Li, Vijay Kumar Adhikari, and Zhi-Li Zhang, "Counting youtube videos via random prefix sampling," in *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, New York, NY, USA, 2011, IMC '11, pp. 371–380, ACM.

[9] EBU Recommendation R37-2007, "The relative timing of the sound and vision components of a television signal," 2007.

[10] Jason Gaedtke, "Optimizing QoE in Large-Scale Video Networks," Packet Video Workshop, Dec 2013, http://pv2013.itec.aau.at/workshop-program/.

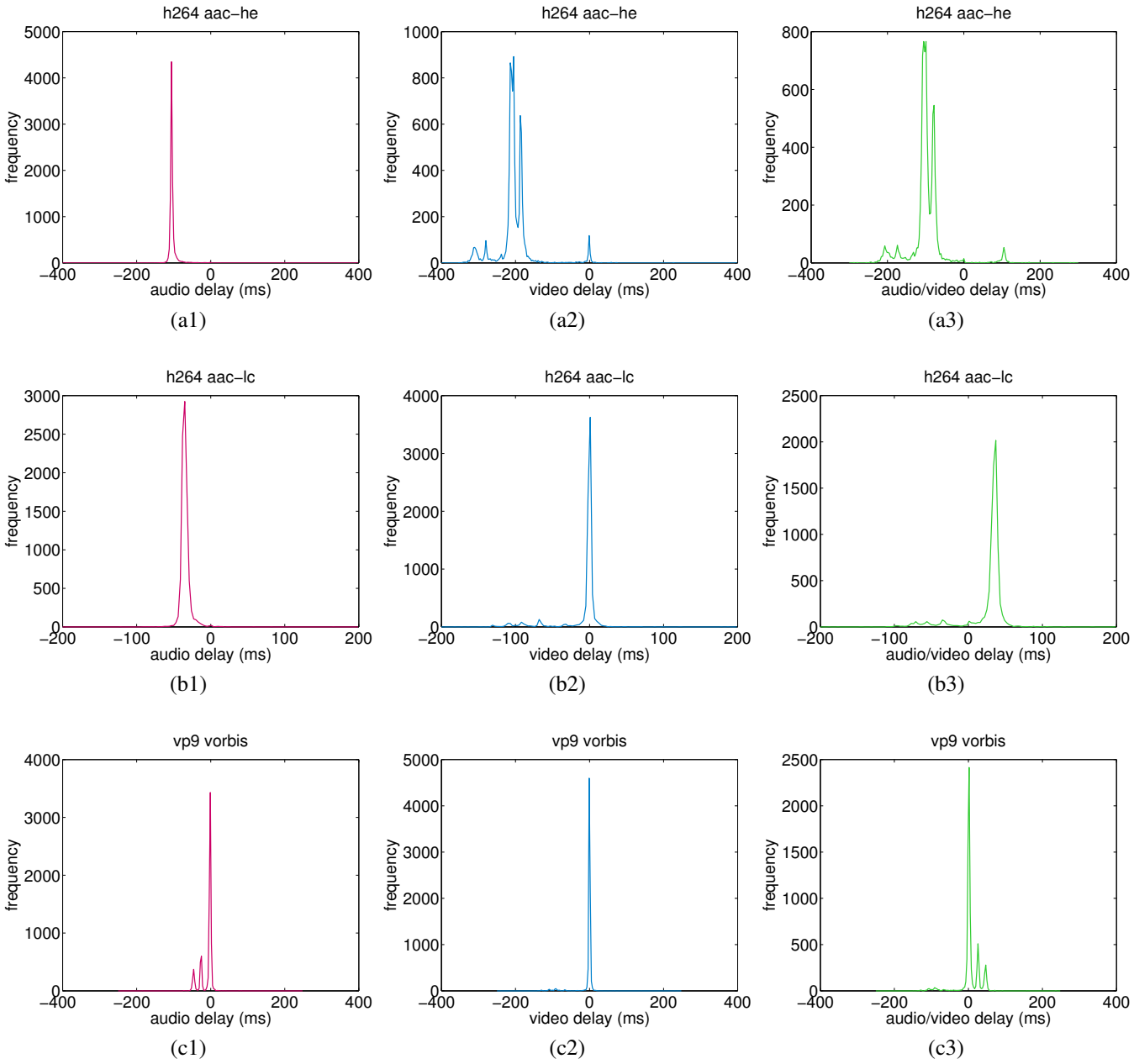

Fig. 2. Delay Histograms on YouTube videos. On each row, histogram of audio/visual and A/V delays between 10K videos in the h264 720p/aac format and their transcoded versions, as transcoded by YouTube. Videos were sampled from videos uploaded in March 2014 with h264/aac 720p format and less than 5 minutes in length. Features were extracted using a patched version of `ffmpeg v2.2.1` as a decoder. On top row, the target format is h264/aac-he, on the middle row h264/aac-lc and on bottom row vp9/vorbis. Some Peaks in the audio delay histograms can be explained by the AAC codec specifications. However the other modes show why measurement of A/V-sync is important and highlights issues that can now be addressed quantitatively.