Abstract – It is well known that when the input video comes upscaled, the effectiveness of its transcoding and delivery may suffer. With adaptive streaming, the top stream may use more pixels and bits than necessary, and then a ladder of streams may be created to support gradual switching to such maximum resolution and bitrate. The result is a significant waste of storage and bandwidth resources. In this paper, we explain the origins of this problem, why upscaled videos exist in modern practice, and survey some methods suggested in the past for addressing it. We then propose our solution. Our proposed design incorporates a novel "true resolution" detection technique and a traditional CAE (context-aware encoding) encoding ladder generator. The CAE generator receives the detected "true resolution" of content as a limit for resolutions to include in the ladder. Such a limit enables all subsequent savings. We describe the details of our proposed resolution detection method, bring examples explaining how it works, and then study the performance of our proposed system in practice. Our study, performed using 500 video assets representing 120 hours of real-world production material, confirms the effectiveness of this technique. It shows that in many practical cases, the incoming content is, in fact, upscaled and that adding a "true resolution" detector to CAE brings very appreciable savings in bandwidth, storage, and computing costs.

Introduction

In the modern world, we are witnessing continued evolution and increasingly hybrid operation of traditional/broadcast and OTT/streaming systems. Such co-existence often leads to complex distribution flows with many video transcoding and format conversion operations [1].

For example, consider a hybrid broadcast + OTT distribution system presented in Figure 1. As typical for broadcast systems, the incoming video feeds originate from remote and field production. A contribution encoder is employed to produce such a feed. It converts video from camera-native format to one required on ingest by the broadcast system. Then, once the content reaches the master control/playout system, it undergoes additional transformations. The playout system may add channel bugs, lower thirds, ad avails, etc. It may also mix content from different sources and formats. Then, another encoder is employed to transmit streams from the broadcast system to an OTT delivery workflow. And then, within the OTT delivery system, another encoder produces outputs for DASH/HLS streaming distribution [2,3]. As easily observed, there are several transcoding operations involved.

Format Conversion Operations

The described chain of transcoding or editing operations may introduce changes in video formats. Furthermore, in some cases, such conversions may lower the effective "spatial density" of the content. This happens with upscaling, SAR/DAR conversions, removal of black bars, etc. Conversions between interlaced and progressive formats may also include upscaling, as in SD to HD format conversions.
Table 1 lists several commonly used video formats, along with examples of conversion operations increasing the "declared resolutions" of the mezzanines.

<table>
<thead>
<tr>
<th>Video format</th>
<th>Width</th>
<th>Height</th>
<th>DAR4:3</th>
<th>SAR4:3</th>
<th>DAR</th>
<th>SAR</th>
<th>Interlaced</th>
<th>Up-conversions</th>
</tr>
</thead>
<tbody>
<tr>
<td>SD/480i</td>
<td>352</td>
<td>480i</td>
<td>4:3</td>
<td>20:11</td>
<td>16:9</td>
<td>80:33</td>
<td>BFF</td>
<td>720p, 1080i, 1080p</td>
</tr>
<tr>
<td></td>
<td>480</td>
<td>480i</td>
<td>4:3</td>
<td>4:3</td>
<td>16:9</td>
<td>16:9</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>528</td>
<td>480i</td>
<td>4:3</td>
<td>40:33</td>
<td>16:9</td>
<td>160:99</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>544</td>
<td>480i</td>
<td>4:3</td>
<td>40:33</td>
<td>16:9</td>
<td>160:99</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>640</td>
<td>480i</td>
<td>4:3</td>
<td>1:1</td>
<td>16:9</td>
<td>4:3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>704</td>
<td>480i</td>
<td>4:3</td>
<td>10:11</td>
<td>16:9</td>
<td>40:33</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>720</td>
<td>480i</td>
<td>4:3</td>
<td>10:11</td>
<td>16:9</td>
<td>40:33</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SD/576i</td>
<td>352</td>
<td>576i</td>
<td>4:3</td>
<td>24:11</td>
<td>16:9</td>
<td>32:11</td>
<td>BFF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>480</td>
<td>576i</td>
<td>4:3</td>
<td>8:5</td>
<td>16:9</td>
<td>32:15</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>544</td>
<td>576i</td>
<td>4:3</td>
<td>11:12</td>
<td>16:9</td>
<td>64:33</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>704</td>
<td>576i</td>
<td>4:3</td>
<td>12:11</td>
<td>16:9</td>
<td>16:11</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>720</td>
<td>576i</td>
<td>4:3</td>
<td>12:11</td>
<td>16:9</td>
<td>16:11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DVCPRO/HD</td>
<td>960</td>
<td>720</td>
<td>4:3</td>
<td>16:9</td>
<td>4:3</td>
<td>Progressive</td>
<td>720p, 1080p</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1280</td>
<td>1080i</td>
<td>16:9</td>
<td>3:2</td>
<td>1:1</td>
<td>TFF</td>
<td>1080i, 1080p</td>
<td></td>
</tr>
<tr>
<td>HD/720p</td>
<td>1280</td>
<td>720</td>
<td>16:9</td>
<td>1:1</td>
<td></td>
<td>Progressive</td>
<td>1080p, 4K</td>
<td></td>
</tr>
<tr>
<td>HD/1080i</td>
<td>1440</td>
<td>1080i</td>
<td>16:9</td>
<td>4:3</td>
<td>1:1</td>
<td>TFF</td>
<td>1080p, 4K</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1920</td>
<td>1080i</td>
<td>16:9</td>
<td>1:1</td>
<td></td>
<td>Progressive</td>
<td>4K</td>
<td></td>
</tr>
<tr>
<td>HD/1080p</td>
<td>1920</td>
<td>1080</td>
<td>16:9</td>
<td>1:1</td>
<td></td>
<td>Progressive</td>
<td>1080p, 4K</td>
<td></td>
</tr>
<tr>
<td>2K</td>
<td>1920</td>
<td>800</td>
<td>2.4:1</td>
<td>1:1</td>
<td></td>
<td>Progressive</td>
<td>1080p, 4K</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1920</td>
<td>816</td>
<td>2.35:1</td>
<td>1:1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2048</td>
<td>858</td>
<td>2.39:1</td>
<td>1:1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2048</td>
<td>864</td>
<td>2.4:1</td>
<td>1:1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2048</td>
<td>1080</td>
<td>1.9:1</td>
<td>1:1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Formats including horizontal "overscan" pixels

**TABLE 1: SOME STANDARD VIDEO FORMATS AND COMMON UP-CONVERSION OPERATIONS.**
Problems Posed by Upscaled Mezzanine Content

When the final OTT/streaming transcoder in the system in Figure 1 receives the content, it is generally unaware of any earlier conversion operations performed with the content. It only sees the resolution as declared in the mezzanine metadata. Hence, if input content is upscaled, it becomes transcoded for delivery as is, producing outputs that may be suboptimal from a quality and efficiency standpoint.

For example, if a 1080p asset becomes up-converted to 4K earlier in the workflow, it likely will be transcoded as 4K content for delivery. Furthermore, with HLS/DASH streaming requirements, this will result in a ladder of 9-12 streams with intermediate resolutions to 4K. We show an example of such an encoding ladder in Table 2. It shows that encoding video as 4K may easily double or even triple bandwidth and storage costs. But quality-wise, if this was a 1080p stream initially, it won't look much better. The same experience can be delivered by a more compact 1080p ladder using much fewer bits.

The described problem, unfortunately, is recurrent in modern practice. As shown in Table 1, many standard format conversion operations produce upscaled outputs. With increasingly more complex media delivery workflows and additional encoding and format conversion operations introduced in practical systems, this problem becomes even more significant.

Related Approaches and Prior Work

Among related prior work, we must recognize several categories of techniques that may be helpful.

The first category comprises "per-title," "content-aware," and "context-aware" encoding (CAE) techniques [5-10]. Such techniques first analyze each incoming video asset and then decide how many bits to use to encode it most efficiently. In other words, instead of using a fixed ladder, as shown in Table 2, they generate a custom ladder for each input video. If the video is "simple" to encode, it receives fewer bits. If the video is "complex," more bits and possibly more streams may be generated. In the case of upscaled content, it is reasonable to expect at least the top few renditions (the ones with the highest resolutions) to become more compressible. Hence CAE could save some bits. But it won't trim the encoding profile or reduce the maximum resolution automatically. In other words, while CAE could lessen the inefficiency introduced by upscaling, it can't eliminate it.

The second category of techniques comprises video encoder-level optimizations, dynamically changing resolutions within the encoded streams. Such functionality is allowed in the latest codecs, such as VVC [11]. With older codecs, such as H.264 [12] and HEVC [13] it is also possible with HLS and appropriate support from HLS clients and decoders [14]. However, dynamic resolution changes are not always safe. For example, they may alter the artistic appearance of film grain, background textures, or

---

**Table 2: Example HLS Encoding Ladder [4]**

<table>
<thead>
<tr>
<th>#</th>
<th>16:9 aspect ratio</th>
<th>HEVC Bitrate [kbps]</th>
<th>Frame rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>640 x 360</td>
<td>160</td>
<td>≤ 30 fps</td>
</tr>
<tr>
<td>2</td>
<td>768 x 432</td>
<td>360</td>
<td>≤ 30 fps</td>
</tr>
<tr>
<td>3</td>
<td>960 x 540</td>
<td>730</td>
<td>≤ 30 fps</td>
</tr>
<tr>
<td>4</td>
<td>960 x 540</td>
<td>1090</td>
<td>≤ 30 fps</td>
</tr>
<tr>
<td>5</td>
<td>960 x 540</td>
<td>1930</td>
<td>Same as source</td>
</tr>
<tr>
<td>6</td>
<td>1280 x 720</td>
<td>2900</td>
<td>Same as source</td>
</tr>
<tr>
<td>7</td>
<td>1280 x 720</td>
<td>4080</td>
<td>Same as source</td>
</tr>
<tr>
<td>8</td>
<td>1920 x 1080</td>
<td>5400</td>
<td>Same as source</td>
</tr>
<tr>
<td>9</td>
<td>1920 x 1080</td>
<td>7000</td>
<td>Same as source</td>
</tr>
<tr>
<td>10</td>
<td>2560 x 1440</td>
<td>9700</td>
<td>Same as source</td>
</tr>
<tr>
<td>11</td>
<td>3840 x 2160</td>
<td>13900</td>
<td>Same as source</td>
</tr>
<tr>
<td>12</td>
<td>3840 x 2160</td>
<td>20000</td>
<td>Same as source</td>
</tr>
</tbody>
</table>
other fine details. Dynamic resolution changes may also introduce an inconsistency in video appearance throughout playback. When working with previously upscaled content, such techniques could also help, but there is no guarantee that the resolutions they select dynamically on a segment-to-segment or frame-by-frame basis would match the original or "true" resolution of the content. They also will not affect the number of streams in the ABR encoding ladder. In other words, this class of techniques could also help, but only partially.

Finally, the last category of relevant techniques includes "original resolution" or "true resolution" detectors [15-18]. These algorithms detect if a given image or video is upscaled. Their traditional uses include forensic video analysis, restoration, and other applications [17]. Some of these techniques are highly effective but only work for specific upscaling filters. For example, the well-known A. C. Gallagher's method [15] only works well for cubic interpolation [16]. The normalized energy density technique [17] is also limited to classic reconstruction filters. The method utilizing the ratio of the low- and high-frequency energy densities, proposed in [18], appears to be more general. However, it is most effective as bound on the range of likely original resolutions. It does not strongly indicate that a particular sampling frequency is the best candidate. The method of detecting a "sharp decline" in the accumulated log-amplitude spectrum [19] is also more general. The authors in [19] report success in its application to modern super-resolution upscaling techniques [20-24]. However, as we observed in our experiments, none of these methods is perfect. They work in many cases but may also fail in some. Many are sensitive to noise and compression artifacts introduced by prior-generation encoding.

But in principle, we believe that techniques for detecting "original" or "true" resolution provide the right tools for addressing the described problem. We will utilize several of these techniques in our proposed solution.

**Proposed Solution**

We show the overall block diagram of our proposed system in Figure 2. The design incorporates a novel "true resolution" detector and a CAE encoding ladder generator [6,8]. In this work, we use the CAE tool provided by the Brightcove VideoCloud system [25,26].

The proposed system detects the "true resolution" of the mezzanine, checks if this resolution is different, and if job configuration parameters allow limiting, it passes the detected resolution as a limit
for the maximum resolution in the profile. Otherwise, the declared resolution of the mezzanine passed as such a limit. The system also produces a notification to the system operator, advising him that the supplied input mezzanine was upscaled.

**Resolution Detection Algorithm**

As shown in Figure 2, the input mezzanine file is first decoded and analyzed for the level of codec-introduced distortions as present in this file. For this purpose, we implement the codec noise-level estimation method proposed in [27,28]. This algorithm works well with most DCT-based codecs (H.264, HEVC, PRORES, etc.). We will use the mezzanine noise level estimate in our resolution detector.

Figure 3 shows the flow diagram of operations within our detector. The detection of horizontal and vertical resolutions is performed separately, utilizing row and column data in each frame. In both cases, we turn data in the DFT domain [29], extract spectral features, and perform an initial selection of candidate resolutions in both directions. The final block finds the best joint (horizontal, vertical) resolution pair and reports it as the detected "true resolution" of the content.

Our system also employs a cascade of safety checks preventing it from selecting unrealistically low resolutions or ones that may result in non-conformant SARs or DARs. When the system finds no compelling candidate resolutions or cues that the video is upscaled, it reports the mezzanine resolution as "true resolution."

**Frequency-Domain Processing**

Once we turn frame row/column data in the DFT domain, we compute several metrics. We explain their meaning in Figure 4. Parameter $f_N$ denotes the Nyquist frequency of the mezzanine sampled data. Parameter $f_c$ denotes the "true resolution" frequency under the test. The shaded regions show
The dotted line moving in the opposite direction relative to the spectrum envelope shows an overlap of the adjacent spectral image [29], appearing if the signal was sampled at $2f_c$. This overlap is the cause of classic aliasing artifacts [29]. The so-called "post-aliasing" artifacts [30] also relate to the propagation of the conjugate-symmetric spectral components coming from the adjacent spectral image.

In theory, the ideal filter designs must eliminate both types of aliasing artifacts from signals. But none of the practical filters are perfect. Furthermore, as explained in [30], some minor aliasing and post-aliasing artifacts are normal and acceptable in practice. We will use the presence of such artifacts as indicators of "true resolution" in our detector.

To quantify the presence of aliasing or post-aliasing in the vicinity of $f_c$, we use normalized correlation metric $\rho^*_f$:

$$\rho^*_f = \frac{\sum_u x[f_c - u] \cdot x[f_c + u]}{\sqrt{\sum_u x[f_c - u]^2 \cdot \sum_u x[f_c + u]^2}}$$

(1)

where $x[.]$ are imaginary parts of DFT spectral components. A Gaussian-smoothed window of +/−32 spectral lines around $f_c$ is used for this analysis. We first compute such metrics for each line or column in a frame. We then aggregate the results for this frame and for the entire sequence.

The other "true resolution" cue that we employ is a "sharp decline" effect discovered in [19]. In Figure 4, we illustrate it by gap measured as $\delta_{f_c}$. We compute it as follows:

$$\delta_{f_c} = \frac{\lambda[f_c]}{\text{median}(\lambda[f_c - m], ..., \lambda[f_c + m])}$$

(2)

where $\lambda[.]$ denote logarithm-domain amplitudes of spectral components averaged across the entire sequence [19]. The $\text{median}(.)$ denotes the median filter. A +/−32 point-window around $f_c$ is used to compute this criterion.

In most cases, both aliasing $\rho^*_f$ and sharp decline $\delta_{f_c}$ criteria will point to the same candidate frequency. We show it in an example presented in Figure 5. But, in some instances, these predictors may disagree or show many candidates or fail to detect any.
Hence, to improve the robustness of our detector, we must apply several additional checks. For this purpose, we use normalized versions of signal energies $E_{f < f_c}$ and $E_{f \geq f_c}$ in each band:

$$e_{f < f_c} = \frac{E_{f < f_c}}{f_c}, \quad e_{f \geq f_c} = \frac{E_{f \geq f_c}}{f_N - f_c}.$$  \hfill (3)

The first check is a comparison of $e_{f \geq f_c}$ against the normalized energy of codec noise in the mezzanine:

$$e_{f \geq f_c} < C_1 \cdot e_{N, Mezz}.$$  \hfill (4)

This condition prevents our classifier from choosing frequencies that may remove any significant (above mezzanine codec noise-level) signal components in the rejection band.

The other criterion that we apply is similar to the energy ratio criterion suggested in [18]:

$$E_{f \geq f_c} < C_2 \cdot E_{f < f_c}. $$  \hfill (5)

This check ensures that the energy in the rejection band stays small relative to the signal’s energy preserved by the choice of $f_c$. Constants $C_1$ and $C_2$ are learned thresholds. We used a large dataset of videos with various types of conversions and transcoding operations present to train this classifier.

**Final Checks and 2D Resolution Selection**

With all described checks applied, horizontal and vertical detectors produce candidate detected frequencies, arriving as input to our final 2D resolution selection block, as shown in Figure 3. This block applies a few additional rules selecting the most likely combination of horizontal and vertical results. This logic looks at the resulting frame aspect ratios, SARs, and their match to mezzanine DAR and disqualifies highly improbable or non-conformant variants.

The pair of horizontal + vertical resolution values that passes all checks and has the highest combined confidence is finally selected. If none of the candidates passes, the detector reports full mezzanine resolution as a default choice.
In our full practical implementation, the system also considers the presence of black bars and uses extra deinterlacing artifact detectors to confirm 480i, 576i, and 1080i origins. With interlaced formats on input, the processing is also somewhat different. However, the core resolution detection logic stays as described above.

Example of Operation

To show how the proposed design work, we will use a "Tears of Steel" sequence [31] converted to 16:9 DAR, as explained in Figure 6. The original content comes in a wide-screen 1920x800 format, so the conversion to 1080p results in upscaling the mid-section. Therefore, the true resolution of this video is only 1422.2 x 800 pixels.

For encoding of this video, we will use three variants of ingest profiles:
- standard HLS encoding ladder for H.264 and 16:9 content, as recommended by HLS encoding guidelines [4],
- using Brightcove CAE [25], treating the input as 1080p content, and
- using Brightcove CAE with the proposed "true resolution" detection operation enabled.
Tables 3-5 show the results of our encodings and summary playback statistics for each case. First columns in these tables list encoding ladder parameters: codec, profile, and resolution of each stream as encoded. Then we list "true width" and "true height," presenting rendition resolution parameters clipped by true resolution limits. They indicate true resolution as delivered by each stream. Then we list encoding bitrates and SSIM quality values [32]. The final column lists the load probabilities of each rendition, retrieved by the Brightcove analytics system [26] after the playback.

The average values listed in Tables 3-5 are computed as follows:

\[ X_{\text{average}} = \frac{1}{\sum_{i=1}^{n} \Pr[i]} \sum_{i=1}^{n} X[i] \cdot \Pr[i]. \]

They report the average performance characteristics achieved while streaming each profile. We report average bitrates, average SSIM scores, and also average resolution as delivered to viewers during the playback. The "storage" values present the sums of bitrates of all renditions used in each profile.

Looking at the numbers in Tables 3 and 4, we first note the significant improvements achieved by the CAE vs. HLS reference profile. We observe that average bitrates went down to 3386 kbps from 5705 kbps, a 40% saving in bandwidth. It also uses much less storage: 7912 kbps vs. 25640 kbps, an over 3x reduction. And there is also a significant reduction in the number of streams: 6 vs. 9. The average 0.981 SSIM delivered by CAE indicates a sufficiently high level of encoding quality. The average resolution of 1392x783 delivered to viewers using CAE is higher than the 1316x740 delivered using the reference profile. Hence CAE already provides significant improvements in all domains.

However, if we next look at Table 5, corresponding to the case when we enabled "true resolution" detection, we see additional improvements. We observe that the average bitrate is now 2502 kbps vs. 3386 kbps, an extra 26.1% saving in bandwidth, and that overall storage is now 6159 kbps vs. 7912 kbps, a saving of 22.1%. The SSIM statistics are the same. And yet, we also note that the average effective resolution as delivered is now 1394x784 vs. 1392x783 – another slight improvement. And we also notice that the "true resolution" detection, in this case, has worked – the top rendition resolution is now 1440x810 (a rounded-up 1422x800), enabling all the improvements noted.

**Experimental Study**

Finally, in this section, we present the results of an experimental study assessing the effects of CAE and CAE with "true resolution" detection on the efficiency of streaming systems.

To perform this study, we used a corpus of 500 video assets, with a combined duration of over 120 hours, representing 33 different content categories, such as action movies, sports, documentaries, etc. All these assets were real-world 1080p and 720p mezzanines sampled from existing OTT distribution workflows. Each video was subsequently encoded using three encoding profiles (HLS reference [4], CAE, and CAE+TR = CAE with enabled "true resolution" detection). Then we instrumented players to play the content and collected the playback statistics.

In Table 6, we present the results. The top rows present performance statistics as observed for each content category. The last three rows show the overall statistics across all categories, the relative savings delivered by CAE vs. reference profile, and the CAE + true resolution vs. standard CAE.

As can be observed, CAE savings are very significant. Overall, we note an almost 40% savings in bandwidth and about 65% savings in storage compared to the reference HLS profile over this test set. However, with "true resolution" detection, CAE+TR savings are even higher. We observe that, on average, "true resolution" detection brings about 9.38% extra savings in bandwidth relative to the
We have discussed the problems posed by up-converted media content for video streaming applications. We have explained the origins of this problem, surveyed several existing tools and techniques that may be useful for addressing it, and proposed a method integrating them into a practical and easily deployable solution.

Table 6: Performance of Reference, CAE, and CAE + True Resolution Encodings.

By analyzing the detection results and visually inspecting files in our data set, we have confirmed that some of our 1080p and 720p mezzanines were, in fact, deinterlaced and upcaled 576i or 480i originals, some used 3:4 SAR pixels, and some originated from wide-screen formats. In some cases, the achieved per-asset bandwidth savings were quite dramatic. We show the distribution of per-asset bandwidth savings as observed in our dataset in Figure 7.

Conclusions

In terms of storage, the additional savings are 11.25%. There is also a considerable reduction in the number of encoded streams. On a per-category basis, we observe even higher savings. For example, we notice 26.97% savings in bandwidth and 28.41% in storage for basketball content.

By analyzing the detection results and visually inspecting files in our data set, we have confirmed that some of our 1080p and 720p mezzanines were, in fact, deinterlaced and upcaled 576i or 480i originals, some used 3:4 SAR pixels, and some originated from wide-screen formats. In some cases, the achieved per-asset bandwidth savings were quite dramatic. We show the distribution of per-asset bandwidth savings as observed in our dataset in Figure 7.

Conclusions

We have discussed the problems posed by up-converted media content for video streaming applications. We have explained the origins of this problem, surveyed several existing tools and techniques that may be useful for addressing it, and proposed a method integrating them into a practical and easily deployable solution.
The presented experimental results indicate that our proposed solution is effective. With a test corpus containing real-world mezzanines, we’ve observed average bandwidth savings of approximately 9.38% and storage savings of 11.25%. Across different content categories, we noted that the savings could be much higher, approaching 26.97% and 28.41% in bandwidth and storage usage, respectively.

We find these results both encouraging and alarming. On the one hand, they show that our proposed tool works and is effective. But on the other, they also indicate that a significant percentage of videos as distributed OTT today are, in fact, upscaled.

References


[31] "Tears of steel" video, Blender project. https://mango.blender.org/download/