A Scalable Video Coding Dataset and Toolchain for Dynamic Adaptive Streaming over HTTP

Christian Kreuzberger, Daniel Posch and Hermann Hellwagner
Institute of Information Technology (ITEC)
Alpen-Adria-Universität (AAU) Klagenfurt, Austria
firstname.lastname@itec.aau.at

ABSTRACT

With video streaming becoming more and more popular, the number of devices that are capable of streaming videos over the Internet is growing. This leads to a heterogeneous device landscape with varying demands. Dynamic Adaptive Streaming over HTTP (DASH) offers an elegant solution to these demands. Smart adaptation logics are able to adjust the clients' streaming quality according to several (local) parameters. Recent research indicated benefits of blending Scalable Video Coding (SVC) with DASH, especially considering Future Internet architectures. However, except for a DASH dataset with a single SVC encoded video, no other datasets are publicly available. The contribution of this paper is two-fold. First, a DASH/SVC dataset, containing multiple videos at varying bitrates and spatial resolutions including 1080p, is presented. Second, a toolchain for multiplexing SVC encoded videos is provided, therefore making our results reproducible and allowing researchers to generate their own datasets.

Categories and Subject Descriptors
H.5.1 [Multimedia Information System]: Video

General Terms
Algorithms, Measurement, Standardization

Keywords
DASH; Dataset; Toolchain; Scalable Video Coding

1. INTRODUCTION

Real-time entertainment platforms such as YouTube and Netflix cause over 50% of traffic on the Internet [17]. Traditional satellite or cable-TV with a fixed program schedule is slowly being supplemented by video-on-demand streaming solutions over the Internet [13]. The advantages are clear: consumers can watch the video anytime and anywhere, provided they have a broadband network connection; companies can use targeted advertisements to generate higher revenues.

Previously, streaming services were using push based protocols, e.g., Real-Time Transport Protocol (RTP), which often caused problems with firewall configurations, lossy connections and congested network paths. Contrary to RTP, Dynamic Adaptive Streaming over HTTP (MPEG-DASH, ISO/IEC 23009-1) uses an HTTP connection, therefore coping with most firewall and proxy settings (port 80, HTTP traffic). Furthermore, MPEG-DASH allows the distribution of content with varying encodings, therefore supporting heterogeneous end devices with different screen sizes, network connections and decoding capabilities.

However, the benefit of being able to satisfy many different devices and spatial resolutions, ranging from 180p up to 4k, comes at the cost of increased storage demands and network usage. Video platform providers have to use (external) Content Distribution Networks (CDNs) – such as Akamai or Amazon CloudFront – to handle the enormous amount of traffic. Furthermore, each video is stored several times at different spatial resolutions and various bitrates.

While real-time entertainment platforms are striving for maximizing the users' satisfaction, Internet Service Providers are looking at maintaining network stability. With the growing amount of multimedia traffic this is a non-trivial task. To deal with this problem the Future Internet community is looking at maintaining network stability. With the growing amount of multimedia traffic this is a non-trivial task. To deal with this problem the Future Internet community proposed architectures that implement inherent caching on network nodes. This would allow network nodes to store popular content closer to the customers, therefore reducing load on network links further away from the customers.

Caching popular content closer to the consumer provides a solution to reduce network load. However, DASH proposes to use many representations at various quality levels. Therefore, one consumer might request a video at basic quality, while another user might request a video at high quality, while yet another one might request a medium quality. A cache would have to store all three copies or store at least a very high quality and provide the lower qualities by transcoding. In the case of streaming multiple representations and many different videos, the caching node(s) would quickly become overloaded and ineffective.

To overcome this issue, the fusion of MPEG-DASH with Scalable Video Coding (SVC) [18] was proposed and evaluated by several papers [16, 7]. In particular, [7] stated that even though SVC does introduce some overhead, roughly 50% of storage space could be saved by replacing AVC encoded videos with SVC encoded videos.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or re-publish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.

http://dx.doi.org/10.1145/2713168.2713193.
In SVC, videos are split into several interdependent layers, with each layer subsequently increasing the video quality. The base layer (BL), which does not depend on any other layer, provides a basic quality (e.g., 360p, 24 fps). The remaining layers, also called enhancement layers (EL), increase the quality (e.g., 720p and 1080p at 24 fps, and 1080p at 48 fps), but depend on at least the base layer. Blending SVC with MPEG-DASH still allows to satisfy diverse consumer demands, while furthermore allowing CDNs and caches to be used more efficiently by prioritizing the base layer, and providing enhancement layers only when resources are available.

The remainder of this paper is structured as follows. Section 2 gives an overview of the related work in this area. We provide a Scalable Video Coding toolchain and dataset for Dynamic Adaptive Streaming over HTTP in Section 3. In Section 4 we present a short analysis of the generated content. We conclude the paper in Section 5, indicating future work using our dataset.

2. RELATED WORK

This section discusses related work in the context of MPEG-DASH and scalable content encodings. In terms of Scalable Video Coding, [18] provides a good overview of the coding concept, and [12] discusses Scalable High Efficiency Video Coding (SHVC) and its improvements compared to SVC.

In terms of MPEG-DASH and SVC, a lot of research already exists. [19] evaluates different DASH adaptation logic based on SVC. Graff et al. [7, 8, 9] analysed SVC encoder settings and argued that there is about 10% overhead per additional layer. Furthermore [8] proposed a hybrid SVC model, where content is split into multiple base layers at varying resolutions, e.g., 360p, 720p and 1080p, and then up to three quality layers are added for each resolution. This hybrid approach allows to combine the benefits of DASH and SVC, while keeping the overhead caused by the SVC encoding at a moderate percentage (roughly 40%).

While plenty of research exists for MPEG-DASH and SVC, to the best of our knowledge there are no MPEG-DASH datasets that provide content using Scalable Video Coding, except for the Tears of Steel (TOS) [5] video in [10]. However, while [10] provides many AVC encoded videos, they do not provide proper PSNR nor SSIM values for TOS, therefore inhibiting the ability to evaluate SVC-encoded videos for MPEG-DASH. [11] provides a dataset containing rather short SVC encoded sequences, though due to the encoder settings used and their brevity, those videos are not necessarily applicable for experimentations with MPEG-DASH.

3. DASH/SVC TOOLCHAIN

This section discusses the source videos and developed tools to generate the dataset. Furthermore, we briefly discuss the encoder settings used. We use the proprietary MainConcept [1] SVC/H.264 Encoder (Revision 1.5) in favour of the JSVM reference encoding software, as MainConcept has a significantly better performance in terms of encoding time (approx. 30 minutes compared to several days for encoding a 10 minute video at 720p). The toolchain and scripts provided cover both, the MainConcept and the JSVM encoder, allowing reproducibility of our results. Our open source toolchain is available at our github repository at http://tinyurl.com/DASHSVC.

3.1 Encoder Settings

A key problem of encoding content to be DASH-compliant is to force I-Frames to be at the beginning of every segment, and to make segments independently decodable. Assuming a segment size of two seconds, containing 48 frames, the following settings were necessary for the MainConcept encoder:

- \text{idr\_interval} = 48 guarantees that starting from frame 1, the next 48 frames only depend on each other, but not on any frames before or after, making segments independent from each other.
- \text{min\_idr\_interval} = 48 forces IDR frames every 48 frames, therefore ensuring I-Frames at the beginning of every segment.
- \text{reordering\_delay} = 4 and \text{use\_b\_slices} = 1 are required to enforce an IBBBP-like frame structure for each segment.

We omit the full configuration files due to space constraints and refer to our dataset, which includes the configuration files for the MainConcept encoder. Similar settings are necessary when using the JSVM reference encoding software (e.g., GOPSize 48, IDRPeriod 48). Furthermore, video specific parameters such as the video resolution, bitrate and quantization parameters (QP) need to be chosen per layer (see Section 3.4).

The BitStreamExtractorStatic tool of the JSVM reference encoding software shows the various layers produced. The tool lists all layers and indexes them based on their layer dependency (D), temporal layer (T) and quality layer (Q) [18]. An example using an encoded video from our dataset is provided in Listing 1.

3.2 De-Multiplexing the H.264/SVC Bitstream

The SVC encoder creates an H.264 compatible bitstream [15], which contains so called Access Units (AUs). Each AU represents a frame and contains one or more Network Abstraction Layer Units (NALUs). Each NALU then contains a type and some header information, describing several types of frames/slices: AVC-I, AVC-P, AVC-B frames.

```bash
$> BitStreamExtractorStatic BBB-I-360p.264 JSVM 9.19.15 BitStreamExtractor
```

**Contained Layers:**

<table>
<thead>
<tr>
<th>Res.</th>
<th>FPS</th>
<th>Bitrate</th>
<th>MinBitrate</th>
<th>(D,T,Q)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>640x360</td>
<td>6</td>
<td>162.5</td>
<td>162.50</td>
</tr>
<tr>
<td>1</td>
<td>640x360</td>
<td>12</td>
<td>325.0</td>
<td>325.00</td>
</tr>
<tr>
<td>2</td>
<td>640x360</td>
<td>24</td>
<td>650.0</td>
<td>650.00</td>
</tr>
<tr>
<td>3</td>
<td>640x360</td>
<td>6</td>
<td>275.0</td>
<td>275.00</td>
</tr>
<tr>
<td>4</td>
<td>640x360</td>
<td>12</td>
<td>550.0</td>
<td>550.00</td>
</tr>
<tr>
<td>5</td>
<td>640x360</td>
<td>24</td>
<td>1100.0</td>
<td>1100.00</td>
</tr>
<tr>
<td>6</td>
<td>640x360</td>
<td>6</td>
<td>412.5</td>
<td>412.50</td>
</tr>
<tr>
<td>7</td>
<td>640x360</td>
<td>12</td>
<td>825.0</td>
<td>825.00</td>
</tr>
<tr>
<td>8</td>
<td>640x360</td>
<td>24</td>
<td>1650.0</td>
<td>1650.00</td>
</tr>
<tr>
<td>9</td>
<td>640x360</td>
<td>6</td>
<td>550.0</td>
<td>550.00</td>
</tr>
<tr>
<td>10</td>
<td>640x360</td>
<td>12</td>
<td>1100.0</td>
<td>1100.00</td>
</tr>
<tr>
<td>11</td>
<td>640x360</td>
<td>24</td>
<td>2200.0</td>
<td>2200.00</td>
</tr>
</tbody>
</table>

Listing 1: SVC Layer Information
and SVC-I, SVC-P, SVC-B frames. Usually, an AU contains multiple NALUs, and each NALU describes the frame for a specific layer. Layer dependencies (D,T,Q) are included in each NALU’s header, allowing to identify the layer ID (e.g., BL, EL 1, EL 2, ...). An example of the frame ordering of SVC is provided in Table 1 for two enhancement layers.

To create a DASH-compliant structure, the SVC bitstream is de-multiplexed into several segments. The segment length has to be chosen in compliance with the encoder settings (see Section 3.1), i.e., \( n = 48 \) frames (or \( n = 96, 144, \ldots \) respectively). Except for the last segment, each segment contains exactly \( n \) frames, and every segment must start with an AVC-I Frame. Depending on the encoding process, each segment starts with an SVC header, describing the scalability structure (Sequence Parameter Set). However, said header contains the same information for each segment, therefore the header of the first segment can be re-used for all other segments. Similar to DASH/AVC, this header is removed from all segments and stored in an init-file. Now each segment contains frames for multiple layers, though for exploiting the full potential of SVC, each segment needs to be split into multiple files, one per layer, as depicted in Figure 1.

Multiplexing is done on a per-segment basis, by following the process depicted in Figure 1 in reverse order. If only a subset of the layers is present, e.g., BL and EL 1, only the subset is multiplexed. Finally the init-file is inserted at the beginning of the multiplexed segment. The decoder will notice any missing enhancement layers, and only decode the available subset. Based on [6], we provide enhanced tools, such as a bitstream analyzer, a fake video player (without video output) and new demultiplexer/multiplexer scripts for DASH/SVC at our github repository (see http://tinyurl.com/DASHSVC). Decoding of the generated dataset is possible for all encoded videos by using the provided scripts (i.e., `svc_merge.py`) in combination with the JSVM reference software on a per-segment basis.

### 3.3 Source Videos

As we wanted to create the possibility to evaluate DASH with SVC compared to the existing DASH dataset [10], we decided to encode the source files of Big Buck Bunny (BBB) [2], Elephants Dream (ED) [3], Tears of Steel (TOS) [5] and Sintel [4] (see Table 2). All four videos are available in the raw YUV format (YUV420p, 1080p, 24 fps), therefore PSNR and SSIM [21] values of the encoded videos can be calculated accordingly (see Section 4). In addition, we encoded a selection of short videos of derf’s collection (http://media.xiph.org): Blue Sky, Rush Hour, Pedestrian Area, Riverbed, Station2, Sunflower, Tractor, Factory.

### 3.4 Bitrates and Spatial Resolutions

Exploiting SVC with multiple enhancement layers in various dimensions (e.g., a mix of spatial and quality scalability) could result in a rather large overhead (roughly 10% per additional layer [7]). Furthermore, using too many enhancement layers would have a negative impact on the client (and potentially the server), as the client would have to issue a high number of HTTP requests for each segment. We encoded the content with a maximum of 4 ELs with MainConcept’s Variable Bitrate Encoding (VBR), using the following four different variants:

### Table 2: Videos included in the dataset

<table>
<thead>
<tr>
<th>Name</th>
<th>Length</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Big Buck Bunny [2]</td>
<td>14315 frames</td>
<td>Animation</td>
</tr>
<tr>
<td>Elephants Dream [3]</td>
<td>15691 frames</td>
<td>Animation</td>
</tr>
<tr>
<td>Tears of Steel [5]</td>
<td>17620 frames</td>
<td>Movie</td>
</tr>
<tr>
<td>Sintel [4]</td>
<td>21312 frames</td>
<td>Animation</td>
</tr>
<tr>
<td>Xiph.org Test Videos</td>
<td>(derf’s collection)</td>
<td></td>
</tr>
</tbody>
</table>
- Variant I uses the Hybrid SVC approach and bitrate recommendations of [7]. Content is split into several independent base layers at varying resolutions (e.g., 360p, 720p, 1080p), and then SNR scalability is used to enhance each resolution's quality successively (e.g., low, medium, high and very high quality). Each enhancement layer introduces 10% additional overhead, resulting in 10, 20 and 30% overhead, respectively – see Table 3.

- Variant II uses the same approach as Variant I, but the spatial resolutions and bitrates are chosen to match the existing dataset [10]. For simplicity, we limited the number of enhancement layers to two. The resolutions used are 480x360, 1280x720 and 1920x1080. See Table 4 for the bitrates used.

- Variant III is slightly different. It uses only a single base-layer at 640x360, with two spatial and one quality enhancement layer – see Table 5.

- Variant IV is based on Variant III, with the addition of one more EL at 1920x1080 – see Table 6.

Evaluations of PSNR/SSIM values and bitrates for each variant are given in Section 4.

3.5 Temporal Scalability
As the example in Section 3.1 shows, there are multiple temporal layers (at 6, 12 and 24 fps). In addition to the previously discussed layers at 24 fps, we provide all variants with temporal scalability at 6, 12 and 24 fps. The full benefit of temporal scalability can be exploited when content with higher framerates, e.g., 48 or 60 fps, is used.

3.6 Deployment
For each variant, the bitstream is de-multiplexed into multiple segments with multiple layers, resulting in the following structure: \$VideoName\$-SegNumber-L$LayerId$.svc. For Variant I a simplified example of the media presentation description (MPD) file is given in Listing 1. The full dataset for all videos including the MPD files is available at http://concert.itec.aau.at/SVCDataset/.

4. DATASET ANALYSIS
In this section we provide an analysis of the generated dataset with respect to Structural Similarity (SSIM) [21] and the respective video bitrates. Peak Signal-to-Noise Ratio (PSNR) values are provided at the dataset website.

As we used Variable Bitrate Encoding (VBR), the target bitrates as defined in Section 3.4 are only approximated. In addition, the encoder used does not support two-pass encoding, resulting in more variability within the effective video bitrates per layer. This results in different bitrates per video, though on average the target bitrates are approximated (see Tables 7 and 8).

4.1 Objective Quality Measurements
The SVC encoded videos were decoded and analyzed in terms of PSNR and SSIM values. The calculated PSNR and SSIM values are provided per frame and layer within the dataset. Furthermore, we calculated PSNR/SSIM values of BBB for the DASH/AVC dataset, and compared the SSIM

<table>
<thead>
<tr>
<th>LayerId</th>
<th>Resolution</th>
<th>AVC bitrate [kbps]</th>
<th>SVC bitrate [kbps]</th>
</tr>
</thead>
<tbody>
<tr>
<td>I.1.BL</td>
<td>640x360</td>
<td>600</td>
<td>600</td>
</tr>
<tr>
<td>I.1.EL1</td>
<td>640x360</td>
<td>900</td>
<td>990</td>
</tr>
<tr>
<td>I.1.EL2</td>
<td>640x360</td>
<td>1250</td>
<td>1500</td>
</tr>
<tr>
<td>I.1.EL3</td>
<td>640x360</td>
<td>1600</td>
<td>2075</td>
</tr>
<tr>
<td>I.2.BL</td>
<td>1280x720</td>
<td>1500</td>
<td>1500</td>
</tr>
<tr>
<td>I.2.EL1</td>
<td>1280x720</td>
<td>2500</td>
<td>2750</td>
</tr>
<tr>
<td>I.2.EL2</td>
<td>1280x720</td>
<td>4000</td>
<td>4800</td>
</tr>
<tr>
<td>I.2.EL3</td>
<td>1280x720</td>
<td>6000</td>
<td>7800</td>
</tr>
<tr>
<td>I.3.BL</td>
<td>1920x1080</td>
<td>4000</td>
<td>4000</td>
</tr>
<tr>
<td>I.3.EL1</td>
<td>1920x1080</td>
<td>5000</td>
<td>5500</td>
</tr>
<tr>
<td>I.3.EL2</td>
<td>1920x1080</td>
<td>6000</td>
<td>7200</td>
</tr>
<tr>
<td>I.3.EL3</td>
<td>1920x1080</td>
<td>8000</td>
<td>10400</td>
</tr>
</tbody>
</table>

Table 3: Variant I – Resolution and bitrates based on [7], with 10 % overhead per layer [7]

<table>
<thead>
<tr>
<th>Resolution</th>
<th>AVC bitrates [kbps]</th>
<th>SVC bitrates [kbps]</th>
</tr>
</thead>
<tbody>
<tr>
<td>480x360</td>
<td>180, 220, 370</td>
<td>180, 242, 444</td>
</tr>
<tr>
<td>1280x720</td>
<td>780, 1000, 1500</td>
<td>780, 1100, 1800</td>
</tr>
<tr>
<td>1920x1080</td>
<td>2000, 2900, 3190</td>
<td>2000, 3190, 5040</td>
</tr>
</tbody>
</table>

Table 4: Variant II – Resolution and bitrates based on [10], with 10 % overhead per layer [7]

<table>
<thead>
<tr>
<th>LayerId</th>
<th>Resolution</th>
<th>AVC bitrate [kbps]</th>
<th>SVC bitrate [kbps]</th>
</tr>
</thead>
<tbody>
<tr>
<td>III.BL</td>
<td>640x360</td>
<td>600</td>
<td>600</td>
</tr>
<tr>
<td>III.EL1</td>
<td>1280x720</td>
<td>2000</td>
<td>2200</td>
</tr>
<tr>
<td>III.EL2</td>
<td>1920x1080</td>
<td>4000</td>
<td>4800</td>
</tr>
<tr>
<td>III.EL3</td>
<td>1920x1080</td>
<td>8000</td>
<td>10400</td>
</tr>
</tbody>
</table>

Table 5: Variant III – Resolution and bitrates for spatial and quality scalability, with 10 % overhead per layer [7]

<table>
<thead>
<tr>
<th>LayerId</th>
<th>Resolution</th>
<th>AVC bitrate [kbps]</th>
<th>SVC bitrate [kbps]</th>
</tr>
</thead>
<tbody>
<tr>
<td>IV.BL</td>
<td>640x360</td>
<td>600</td>
<td>600</td>
</tr>
<tr>
<td>IV.EL1</td>
<td>1280x720</td>
<td>2000</td>
<td>2200</td>
</tr>
<tr>
<td>IV.EL2</td>
<td>1920x1080</td>
<td>4000</td>
<td>4800</td>
</tr>
<tr>
<td>IV.EL3</td>
<td>1920x1080</td>
<td>5200</td>
<td>6760</td>
</tr>
<tr>
<td>IV.EL4</td>
<td>1920x1080</td>
<td>8000</td>
<td>11200</td>
</tr>
</tbody>
</table>

Table 6: Variant IV – based on Variant III with one additional EL and 10 % overhead per layer [7].
values of the SVC and AVC encoded videos with respect to the spatial resolution and video bitrate.

While we consider both, PSNR and SSIM, for measuring the video quality, we focus on SSIM within this paper. Tables 7 and 8 provide a list of SSIM values per layer of the SVC encoded dataset, while Figure 2 shows the SSIM values of Big Buck Bunny for Variant I. The 360p and 720p encoded videos were decoded and then upsampled to 1080p. SSIM values were calculated by comparing the source 1080p YUV file with all decoded layers. ED and BBB achieve nearly identical quality at similar bitrates for sub-variants I.2 (720p) and I.3 (1080p), but not for I.1 (360p). The sub-variants (360p, 720p, 1080p) are annotated in Figure 2.

Figure 3 depicts the DASH/AVC SSIM values (red dotted line) for Big Buck Bunny, compared with Variant II of our dataset. The SVC Base Layers, marked as filled circles, are very close to the SSIM and bitrate values of the AVC content, as expected. However, the respective SVC Enhancement Layers always achieve a lower SSIM value, therefore worse video quality, than the AVC content at the same bitrate. For instance, Variant II at 4525 kbps (EL 2) has a calculated SSIM of 0.963, while the AVC counterpart achieves an SSIM value of 0.964 with only 3450 kbps. This results in a calculated overhead of roughly 30%. Even more overhead (roughly 78%) can be observed for the TOS content, though for the 1080p content the opposite (almost no overhead) is the case, as depicted in Figure 4. For Variant IV, Tables 7 and 8 show that the additional quality enhancement layer results in a roughly 1 Mbps higher bitrate (10%) for the last enhancement layer. While Graff et. al [7] suggests a 10% overhead per additional layer, our dataset suggests that an overhead of (up to) 15% per additional EL could be more accurate, though further investigation is needed to give a more accurate result for overhead. In addition, the accuracy of the VBR encoding of MainConcept (Rev. 1.5) is rather modest, and tweaking bitrate settings of the layers to meet the (calculated) overhead proved to be difficult.

Table 7: Bitrates and SSIM values for BBB, at their respective spatial resolutions

<table>
<thead>
<tr>
<th>Var.</th>
<th>Bitrates [kbps]</th>
<th>SSIM</th>
</tr>
</thead>
<tbody>
<tr>
<td>I.1</td>
<td>636, 975, 1401, 1808</td>
<td>0.962, 0.971, 0.977, 0.981</td>
</tr>
<tr>
<td>I.2</td>
<td>1524, 2655, 4454, 6716</td>
<td>0.952, 0.964, 0.974, 0.981</td>
</tr>
<tr>
<td>I.3</td>
<td>3936, 5484, 7004, 10757</td>
<td>0.969, 0.972, 0.975, 0.990</td>
</tr>
<tr>
<td>II.1</td>
<td>145, 275, 444</td>
<td>0.902, 0.911, 0.935</td>
</tr>
<tr>
<td>II.2</td>
<td>782, 1212, 1727</td>
<td>0.922, 0.931, 0.943</td>
</tr>
<tr>
<td>II.3</td>
<td>2071, 3013, 4525</td>
<td>0.95, 0.954, 0.963</td>
</tr>
<tr>
<td>III</td>
<td>613, 2079, 5003, 9680</td>
<td>0.885, 0.939, 0.966, 0.979</td>
</tr>
<tr>
<td>IV</td>
<td>610, 2073, 4944</td>
<td>0.886, 0.939, 0.966</td>
</tr>
</tbody>
</table>

Table 8: Bitrates and SSIM values for TOS, at their respective spatial resolutions

<table>
<thead>
<tr>
<th>Var.</th>
<th>Bitrates [kbps]</th>
<th>SSIM</th>
</tr>
</thead>
<tbody>
<tr>
<td>I.1</td>
<td>612, 962, 1361, 1807</td>
<td>0.945, 0.955, 0.962, 0.968</td>
</tr>
<tr>
<td>I.2</td>
<td>1472, 2541, 4332, 7105</td>
<td>0.944, 0.953, 0.961, 0.970</td>
</tr>
<tr>
<td>I.3</td>
<td>373, 5305, 6877, 10755</td>
<td>0.948, 0.951, 0.952, 0.964</td>
</tr>
<tr>
<td>II.1</td>
<td>205, 281, 460</td>
<td>0.869, 0.882, 0.906</td>
</tr>
<tr>
<td>II.2</td>
<td>824, 1266, 1765</td>
<td>0.916, 0.926, 0.934</td>
</tr>
<tr>
<td>II.3</td>
<td>2091, 2094, 4533</td>
<td>0.932, 0.937, 0.944</td>
</tr>
<tr>
<td>III</td>
<td>584, 1989, 4520, 9986</td>
<td>0.874, 0.925, 0.943, 0.956</td>
</tr>
<tr>
<td>IV</td>
<td>582, 1983, 4507</td>
<td>0.874, 0.925, 0.943</td>
</tr>
</tbody>
</table>

Figure 2: SSIM values (at 1080p) for BBB and ED (SVC Variant I)

Figure 3: SSIM values (at 1080p) for BBB for SVC Variant II and DASH with AVC

Figure 4: SSIM values (at 1080p) for TOS for SVC Variant I and DASH with AVC
5. CONCLUSION AND DATASET USAGE

This paper presented a DASH/SVC dataset, including four different variants, and evaluated the encoded videos compared to the DASH dataset [10] in terms of PSNR and SSIM. Furthermore, we provided a toolchain for converting SVC encoded videos into DASH-compliant streams. The dataset is available online, including configuration- and the generated MPD files, enabling researchers to immediately use the dataset with existing DASH systems and simulations – see http://concert.itec.aau.at/SVCDataset/.

This dataset enables researchers to do experimentations with SVC and DASH. Evaluations of existing research, such as the proposal of In-Network Adaptation in NDN/ICN by using SVC [14], could be extended to multiple videos and representations. In addition, advanced caching and routing strategies can be applied when using SVC content, e.g., prioritizing the base layer(s) to reduce video playback stalls.

Furthermore, finding an optimal number of DASH representations and video bitrates, as proposed by [20], could be applied to SVC encoded videos. Based on our dataset, several adaptation logics [19] for SVC can be re-evaluated with more content, and researchers are able to compare DASH/AVC with DASH/SVC in terms of video bitrate, PSNR and SSIM values.

Our toolchain also has the benefit of being open source and it will support – with minor modifications – the SHVC (Scalable High Efficiency Video Coding) extension of HEVC. The provided sources will also allow researchers to study and understand the structure of H.264/SVC encoded videos in more detail, as we provide plenty of debug output.

SVC has the potential to benefit both, the consumers in terms of QoE and the producers in terms of content dissemination, though advanced QoE and dissemination models are still to be researched for DASH/SVC.

6. ACKNOWLEDGMENTS

This work was partly funded by the Austrian Science Fund (FWF) under the CHIST-ERA project CONCERT (A Context-Adaptive Content Ecosystem Under Uncertainty), project number I1402.

7. REFERENCES