Fast Simultaneous Video Encoder for Adaptive Streaming

Johan De Praeter #1, Antonio Jesús Díaz-Honrubia 2, Niels Van Kets 1
Glenn Van Wallendael 1, Jan De Cock 1, Peter Lambert 1, Rik Van de Walle 1

1 Ghent University - iMinds, ELIS - Multimedia Lab
Gaston Crommenlaan 8 bus 201, B-9050 Ledeberg-Ghent, Belgium
# johan.depraeter@ugent.be
2 Albacete Research Institute of Informatics (I3A), University of Castilla-La Mancha
Campus Universitario s/n, 02071 Albacete, Spain

Abstract—Content providers create different versions of a video to accommodate different end-user devices and network conditions. However, each of these versions requires a resource intensive encoding process. To reduce the computational complexity of the encodings, this paper proposes a fast simultaneous encoder. This encoder takes a single video as input and creates a number of bit streams encoded with different parameters. Only one version of the video is created with a full encode, whereas encoding of the other versions is accelerated by exploiting the correlation with the fully encoded version using machine learning techniques. In a practical scenario, the fast simultaneous encoder achieves a complexity reduction of 67.3% with a bit rate increase of 5.2% compared to performing a full encode of each version.

I. INTRODUCTION

Traditionally, digital linear television is delivered to consumers through dedicated broadcast facilities. In their simplest form, these facilities consist of a dedicated multicast network and set-up boxes in the homes of consumers. However, with the advent of other video-enabled devices such as tablets or smartphones, and the need to consume video outside a home environment, a shift towards other delivery platforms was needed. To achieve distribution of video to these devices, video content providers have moved to a model of delivering video over the internet.

Since this new video distribution landscape contains many different devices and networks with varying capabilities, content providers need to adapt their video to these circumstances. This video adaptation can be achieved by using scalable video or adaptive streaming. Scalable video creates a bit stream with a base layer, which represents a version of the video with the highest compression, and enhancement layers, which provide less compression for higher quality [1]. However, each enhancement layer results in a bit rate overhead, i.e. when aiming for a certain quality, a single layer encoding will always have a higher compression efficiency than a base layer with an enhancement layer. Therefore, video streaming services make use of single-layer adaptive streaming [2]. In this approach, several bit streams encoded with different parameters are stored on a server. Depending on the current capabilities of the network and client device, the viewer receives the appropriate version of the video.

With adaptive streaming, the provider must encode several versions of the same video. Since such an encoding operation is computationally very demanding, the provider might limit the number of supported versions in the case of a live broadcast. Even in a non-live scenario, the number of versions might be limited due to the sheer amount of videos that the provider needs to encode.

The different versions of a video in adaptive streaming are all encoded with different parameters. However, since decisions in an encoder such as partitioning a frame into block structures are influenced by the original video content, encoder decisions for different encodes of the same video might show some correlations.

In this paper, we propose a fast simultaneous encoder that uses this correlation to reduce the amount of necessary processing power to encode different versions of the same video. This encoder performs a full encode of one version of the video, while the encoding of other versions is accelerated by exploiting correlation with the full encode. To provide good compression, the recent video standard High Efficiency Video Coding (HEVC) is used [3]. However, the proposed simultaneous encoder can be applied similarly for any other video compression standard, or even for a mix of output video versions encoded with different standards.

The rest of the paper first describes the relation with other work in Section II. Then, the fast simultaneous encoder architecture is proposed in Section III, followed by the results in Section IV. Finally, the conclusion is given in Section V.

II. RELATED WORK

Since encoding is a computationally complex operation, many efforts have been made to reduce this complexity. This is typically achieved by using models to skip encoding decisions. This approach has been applied for fast encoding, transcoding, and scalable video coding. Each of these scenarios mainly differs in the source of input information of the models to skip the decisions.
In fast encoding, the encoder complexity is reduced by exploiting information obtained by previous decisions made by the encoder [4]–[7]. Examples of this information include Rate-Distortion (RD) cost calculations and mode decisions of spatially and temporally neighbouring blocks. On the other hand, transcoding creates a new version of the same video with encoding parameters that differ from the old version [8]. Therefore, fast transcoding algorithms can exploit information about the encoding decisions in the old version to predict the decisions in the new version [9]–[12].

Acceleration of scalable video coding focuses on reducing the complexity of coding the enhancement layer [13]–[16]. To achieve this, information from the base layer is exploited to predict encoding decisions in the enhancement layer. This scenario might be considered conceptually similar to fast simultaneous encoding. However, in scalable video the base layer always needs to be encoded first, meaning that it is not possible to use a layer with higher quality to predict a layer with lower quality. Moreover, scalable video uses inter-layer prediction, which increases the encoding complexity of the enhancement layers.

Simultaneous encoding could be considered a fourth scenario besides fast encoding, transcoding, and scalable video and has been studied to provide a parallel encode of a H.264/AVC and VP8 bitstream [17]. However, the correlation between different quality and resolution versions of the same video codec has not been exploited.

As its main novelty, this paper proposes the idea to specifically speed up the encoding of different versions of the same video in an adaptive streaming scenario by using machine learning techniques. In this scenario, several versions of the same video are encoded in parallel by exploiting the correlation between decisions of the different versions. Contrary to scalable video, it is possible to use a higher quality version of the video to accelerate encoding of lower quality versions. However, a lower quality version can also be used as a basis for acceleration. As such, choosing the optimal version for a full encode which shows the most correlation with all provided versions is one of the main challenges in this new field.

III. SIMULTANEOUS ENCODER ARCHITECTURE

A. Fast simultaneous encoding

In an adaptive streaming scenario, several versions of the same input video are encoded at different bit rates. This can be accomplished by techniques such as modifying the quantisation parameter (QP) and/or spatial resolution in the encoder. However, since the same input video is used, the decisions taken by the encoder for one version may be related to decisions taken for a version encoded with different parameters. A fast simultaneous encoder should exploit this relation between versions to reduce the overall encoding complexity for encoding the different versions.

Fig. 1 shows the proposed simultaneous encoder architecture. In this example, three versions of the video are generated for adaptive streaming. The simultaneous encoder first starts a full encode of the version with the highest quality. This fully encoded version will be referred to as the master version in the rest of the paper. Once enough encoding information is known, the alternative versions can use this information to skip steps during their own encoding process.

By reducing the encoding complexity of the alternative versions of the video, the overall remaining encoding complexity of the simultaneous encoder is reduced to a factor $C(n)$ in function of the number of versions $n$ that need to be generated. $C(n)$ can be represented by the total encoding time $T_{fast}(n)$ of $n$ versions encoded with a fast simultaneous encoder divided by the total encoding time $T_{full}(n)$ of a simultaneous encoder that uses a full encode for all versions. Consequently, $C(n)$ can be written as Eq. 1, with $T_{0,full}$ being the encoding time of the master version (represented by the full encode seen in Fig. 1), $T_{k,full}$ being the full encoding time for alternative version $k$, and $\alpha_k$ being the fraction of remaining encoding complexity during fast encoding of alternative version $k$.

$$C(n) = \frac{T_{fast}(n)}{T_{full}(n)} = \frac{T_{0,full} + \sum_{k=1}^{n} \alpha_k T_{k,full}}{T_{0,full} + \sum_{k=1}^{n} T_{k,full}}$$

(1)

By calculating $C(n)$, it is possible to compare the overall complexity reduction of the simultaneous encoder for using a different master version as an input to predict information for the alternative versions.

B. Prediction of information

To achieve a fast simultaneous encoder, encoding decisions are predicted based on the relation between different versions of the video. To determine which encoding information should be predicted, the characteristics of the compression standard should be taken into account. In this paper, HEVC is used for compression. However, if some master or alternative versions are encoded with a different standard, the appropriate encoding decisions should be determined for that standard.

In HEVC, the video is compressed by using flexible block structures [3]. Each frame is first divided into Coding Tree Units (CTU) (in this paper chosen as the maximum supported size of 64x64 pixels). These CTUs can be recursively partitioned into Coding Units (CUs) down to a size of 8x8 pixels. For each of these CUs, one of eight possible Prediction Unit (PU) modes needs to be selected. Moreover, CUs are also further partitioned into Transform Units (TUs) for coding of the residual information.
Determining the optimal block structure results in the high encoding complexity of HEVC, since motion estimation and evaluation of intra-prediction modes is performed for all possible structures. Therefore, in this paper the complexity is reduced by predicting the CU structure of the alternative versions, limiting the number of blocks that need to be evaluated.

To predict CU structures, a machine learning model is trained on the first $M$ frames of each alternative version by using the Random Forest algorithm [18]. This algorithm generates an ensemble of decision trees for which a random set of candidate features is used in each tree. The tree is built by determining rules in each node with the goal of maximizing entropy reduction for the given samples [19]. In this paper, a node is not split any further if a split would result in a node containing less than 1% of the total number of samples used in the tree. An example of such a tree is shown in Fig. 2. Given an input sample with $\text{max}_T \text{Usize} \geq 32 \times 32$ and $\text{var}_D \text{CTvar} > 0.0005$, the tree would predict that the sample is split, since the class distribution shows that 36 training samples were not split and 99 were split.

A single tree can be sensitive to noise and outliers in the data. On the other hand, a random forest has a higher noise robustness due to averaging over the different trees. Moreover, other machine learning algorithms may have a high computational complexity, such as SVM [20] with a training complexity of the order $O(n^3 \cdot m)$, with $n$ the number of samples and $m$ the number of features. On the other hand, random forests have fast training times of the order $O(p \cdot m \cdot n^1 \cdot \log(n^1))$, with $p$ the number of trees. Note that for random forests $n^1$ is used since each tree only uses a random subset of the total number of samples.

For each output CU size, the random forest model determines whether the block should be split based on information from the co-located blocks in the master version (as shown in Fig. 3). However, when a co-located block is fetched from a version with a different resolution, it is possible that only part of an input block is co-located with the output block. In this case, the information is weighted. For example, if only half of an input block is co-located with the output block, the information will only carry half the weight of a block that is completely co-located with the output block.

The input feature information of the machine learning algorithm consists of the variance of the transform coefficients, motion vector variance, and information about the block structure [12]. The transform coefficient variance and motion vector variance are a good indicator for the motion activity and general complexity in a scene. Similarly, block structure information (the mean, variance, maximum and minimum values of the co-located CU, PU, and TU block sizes) is also a good indicator for scene complexity, since smaller blocks are used for more complex sections of the picture.

IV. RESULTS

The fast simultaneous encoder was evaluated by modifying version 12 of the HEVC reference software [21]. The sequences BasketballDrive, BQTerrace, Cactus, Kimono, and ParkScene were used as input videos for the simultaneous encoder. These sequences have a duration of ten seconds and a respective frame rate of 50, 60, 50, 24, and 24 frames per second. For each of these videos, 48 different versions were generated with different spatial resolutions and QPs. The resolution $R$ was chosen as $R \in \{1920 \times 1080, 1280 \times 720, 960 \times 536\}$ and the QP ranged from 14 to 44 with a step of 2.

Each version was encoded using a random access configuration. Since adaptive streaming typically uses segment sizes between one and ten seconds, a segment size of five seconds is chosen in this paper. Consequently, a new intra-frame is inserted every five seconds. Additionally, the frame structure is configured to have every set of eight frames end with a P-frame, while the other seven frames are B-frames. The first nine frames (one I-frame, seven B-frames, and one P-frame) are used to train the machine learning model for each of the alternative versions of the video. Additionally, the QP of each frame is fixed to match the constant QP of the video. Finally, the random forest algorithm used 50 trees.

In the following subsections, the prediction accuracy, compression efficiency, and complexity reduction are evaluated for the fast-encoded alternative versions. In these subsections, the observations are illustrated with only the sequence BasketballDrive, since the other sequences follow similar trends. At the end of this section, the choice of the master version for full encode is also investigated in a practical scenario.

A. Evaluation of fast encodes

1) Prediction accuracy: To measure the prediction accuracy of the machine learning models, a score is calculated for each block size. This score equals the percentage of correct predictions made for that block size.
As seen in Table I, if the master and alternative version are the same, the prediction accuracy is 100% because the model learns that it should copy the original CU decisions. However, the accuracy decreases when the target QP deviates more from the original QP, since alternative versions that differ more from the master version are harder to predict.

Additionally, the alternative versions with the highest and lowest QP-values are easier to predict, since at a high QP, the majority of the blocks is never split, whereas at a low QP, most blocks are split. In extreme cases, such as when predicting the structure of a version with QP 14 and a resolution of 1920×1080 pixels, a trivial classifier that always forces a split can attain an accuracy of more than 90% for BasketballDrive and BQSquare, and more than 75% for the other sequences.

Furthermore, as is also illustrated in Table I, using a higher quality version to predict a lower quality version yields a better result than the other way around, likely because a low quality version would have lost details that are present in the high quality version. E.g. if the master version has QP 32 and the alternative version has QP 22, the accuracy is 77%, whereas the accuracy is 84% if the master version has QP 22 and the alternative version has QP 32.

Finally, it appears that information in a version with a low resolution correlates better with a higher resolution if the master version has a high quality. However, to predict an alternative version with a lower resolution, the quality of the master version should also be lower. As an example, in Table II the highest accuracy for predicting an alternative version with a resolution of 1920×1080 is achieved with QP 18, whereas the best result for a resolution of 960×536 is achieved with QP 24. This means that a low-resolution master version needs to compensate by choosing a lower QP to approximate the amount of details present in a higher resolution. On the other hand, a lower resolution of the alternative version requires a less detailed master version to achieve good correlation.

2) Compression efficiency: The compression efficiency is determined by comparing the Peak Signal-to-Noise Ratio (PSNR) and bit rate of a fast encode of an alternative version with the PSNR and bit rate of a full encode of the same alternative version. The Bjøntegaard Delta rate (BD-rate) is used to express the relative bit rate overhead introduced by fast encoding compared to full encoding for the same PSNR [22]. To calculate the BD-rate, four RD-points are needed. Therefore, four subsequent QP-values are used to calculate the BD-rate. E.g. the BD-rate 2.4% for QP 14 for the alternative version in Table III means that this BD-rate was calculated by using the rate-points of QP 14, 16, 18, and 20.

The compression efficiency does not appear to follow the observations of the prediction accuracy. If the master and alternative version are the same, the resulting BD-rate is not 0% (Table III), despite having an accuracy of 100%. This behaviour is caused by a fast encoding setting in the HM reference software which is enabled by default. With this setting the encoder bases some decisions on previous information such as the PU mode of the parent CU. Since this information was never calculated because CU decisions are skipped, the encoder might make a less optimal decision during the remaining encoding decisions. If this setting is disabled, the BD-rates are reduced to 0%. However, the encoding time of the HM software increases, which results in a higher complexity reduction of the fast encoder. To provide a fair comparison in terms of complexity reduction, the setting was left enabled.

Despite extreme QP-values being easier to predict, a higher QP value results in a higher BD-rate (Table III). Since a higher QP value produces a lower bit rate, a small absolute increase in bit rate results in a higher relative increase.

Finally, similar to the observations of the prediction accuracy, the BD-rate performance is better if a higher quality master version is used. However, the difference in average
TABLE III
BD-RATE INCREASE (%) FOR THE SEQUENCE BasketballDrive. BOTH THE MASTER VERSION AND ALTERNATIVE VERSION HAVE A RESOLUTION OF 1280×720 PIXELS. IN EACH ROW, A DIFFERENT MASTER VERSION IS USED, WHEREAS IN EACH COLUMN ENCODING DECISIONS FOR A DIFFERENT ALTERNATIVE VERSION ARE PREDICTED. COMBINATIONS WITH A BETTER COMPRESSION EFFICIENCY ARE INDICATED WITH A LIGHTER COLOUR.

<table>
<thead>
<tr>
<th>QP</th>
<th>14</th>
<th>16</th>
<th>18</th>
<th>20</th>
<th>22</th>
<th>24</th>
<th>26</th>
<th>28</th>
<th>30</th>
<th>32</th>
<th>34</th>
<th>36</th>
<th>38</th>
</tr>
</thead>
<tbody>
<tr>
<td>QP of alternative version</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>2.4</td>
<td>2.5</td>
<td>2.9</td>
<td>3.4</td>
<td>4.0</td>
<td>4.6</td>
<td>5.6</td>
<td>6.5</td>
<td>7.6</td>
<td>8.4</td>
<td>9.3</td>
<td>10.2</td>
<td>11.0</td>
</tr>
<tr>
<td>16</td>
<td>2.0</td>
<td>2.4</td>
<td>2.6</td>
<td>3.1</td>
<td>3.7</td>
<td>4.4</td>
<td>5.2</td>
<td>6.3</td>
<td>7.3</td>
<td>8.3</td>
<td>9.2</td>
<td>10.2</td>
<td>10.9</td>
</tr>
<tr>
<td>18</td>
<td>2.2</td>
<td>2.2</td>
<td>2.4</td>
<td>2.7</td>
<td>3.4</td>
<td>4.1</td>
<td>5.0</td>
<td>6.0</td>
<td>7.2</td>
<td>8.1</td>
<td>9.0</td>
<td>9.6</td>
<td>11.0</td>
</tr>
<tr>
<td>20</td>
<td>2.3</td>
<td>2.5</td>
<td>2.4</td>
<td>2.5</td>
<td>3.0</td>
<td>3.7</td>
<td>4.6</td>
<td>5.7</td>
<td>6.9</td>
<td>8.0</td>
<td>8.7</td>
<td>9.8</td>
<td>10.4</td>
</tr>
<tr>
<td>22</td>
<td>2.6</td>
<td>2.8</td>
<td>2.8</td>
<td>2.7</td>
<td>2.7</td>
<td>3.3</td>
<td>4.2</td>
<td>5.3</td>
<td>6.5</td>
<td>7.6</td>
<td>8.6</td>
<td>9.4</td>
<td>10.4</td>
</tr>
<tr>
<td>24</td>
<td>2.8</td>
<td>3.2</td>
<td>3.2</td>
<td>3.2</td>
<td>3.0</td>
<td>3.0</td>
<td>3.7</td>
<td>4.8</td>
<td>6.0</td>
<td>7.2</td>
<td>8.2</td>
<td>9.2</td>
<td>10.0</td>
</tr>
<tr>
<td>26</td>
<td>3.1</td>
<td>3.5</td>
<td>3.8</td>
<td>3.7</td>
<td>3.6</td>
<td>3.3</td>
<td>3.3</td>
<td>4.2</td>
<td>5.4</td>
<td>6.6</td>
<td>7.9</td>
<td>8.8</td>
<td>9.8</td>
</tr>
<tr>
<td>28</td>
<td>3.4</td>
<td>4.0</td>
<td>4.2</td>
<td>4.4</td>
<td>4.3</td>
<td>4.0</td>
<td>3.7</td>
<td>4.8</td>
<td>6.0</td>
<td>7.2</td>
<td>8.2</td>
<td>9.2</td>
<td>10.0</td>
</tr>
<tr>
<td>30</td>
<td>3.6</td>
<td>4.4</td>
<td>4.8</td>
<td>5.0</td>
<td>5.1</td>
<td>4.9</td>
<td>4.6</td>
<td>4.1</td>
<td>4.1</td>
<td>5.1</td>
<td>6.4</td>
<td>7.7</td>
<td>9.2</td>
</tr>
<tr>
<td>32</td>
<td>3.8</td>
<td>4.7</td>
<td>5.5</td>
<td>5.8</td>
<td>5.8</td>
<td>5.7</td>
<td>5.7</td>
<td>5.3</td>
<td>4.7</td>
<td>4.4</td>
<td>5.6</td>
<td>7.0</td>
<td>8.2</td>
</tr>
<tr>
<td>34</td>
<td>4.0</td>
<td>5.3</td>
<td>6.3</td>
<td>6.8</td>
<td>6.7</td>
<td>6.5</td>
<td>6.4</td>
<td>5.8</td>
<td>5.0</td>
<td>4.9</td>
<td>6.1</td>
<td>7.5</td>
<td>7.5</td>
</tr>
<tr>
<td>36</td>
<td>4.3</td>
<td>5.5</td>
<td>7.3</td>
<td>8.1</td>
<td>8.1</td>
<td>7.6</td>
<td>7.4</td>
<td>7.1</td>
<td>6.9</td>
<td>6.1</td>
<td>5.4</td>
<td>5.4</td>
<td>6.6</td>
</tr>
<tr>
<td>38</td>
<td>4.5</td>
<td>5.6</td>
<td>7.9</td>
<td>9.7</td>
<td>9.8</td>
<td>9.3</td>
<td>8.8</td>
<td>8.3</td>
<td>7.9</td>
<td>7.3</td>
<td>6.3</td>
<td>5.7</td>
<td>5.8</td>
</tr>
</tbody>
</table>

TABLE IV
TIME SAVING (%) (UPPER HALF) AND AVERAGE CU DEPTH (BOTTOM HALF)

<table>
<thead>
<tr>
<th>Res. of alt.</th>
<th>14</th>
<th>16</th>
<th>18</th>
<th>20</th>
<th>22</th>
<th>24</th>
<th>26</th>
<th>28</th>
<th>30</th>
<th>32</th>
<th>34</th>
<th>36</th>
<th>38</th>
<th>40</th>
<th>42</th>
<th>44</th>
</tr>
</thead>
<tbody>
<tr>
<td>QP of alt.</td>
<td>1080p</td>
<td>65</td>
<td>64</td>
<td>65</td>
<td>69</td>
<td>71</td>
<td>73</td>
<td>74</td>
<td>74</td>
<td>75</td>
<td>75</td>
<td>76</td>
<td>77</td>
<td>77</td>
<td>77</td>
<td>77</td>
</tr>
<tr>
<td>720p</td>
<td>67</td>
<td>69</td>
<td>71</td>
<td>72</td>
<td>73</td>
<td>73</td>
<td>74</td>
<td>74</td>
<td>75</td>
<td>75</td>
<td>75</td>
<td>76</td>
<td>77</td>
<td>77</td>
<td>77</td>
<td>77</td>
</tr>
<tr>
<td>536p</td>
<td>68</td>
<td>70</td>
<td>69</td>
<td>70</td>
<td>71</td>
<td>72</td>
<td>72</td>
<td>73</td>
<td>74</td>
<td>74</td>
<td>75</td>
<td>75</td>
<td>76</td>
<td>76</td>
<td>76</td>
<td>76</td>
</tr>
</tbody>
</table>

smaller CUs mean that the total amount of CUs that need to be evaluated by the encoder is larger than when many large CUs are used, more complexity is reduced by forcing only larger CUs to be tested.

B. Practical application

In a practical scenario, the complexity reduction of a simultaneous encoder should not be measured by averaging the individual complexity reduction of all alternative versions. Instead, the overall complexity of the simultaneous encoder should be reduced. This means that \( C(n) \) in Eq. 1 needs to be minimized, or that the complexity reduction \( C(n)^* = 1 - C(n) \) needs to be maximized. Besides the choice of \( n \), this value is also affected by the selection of the master version.

To investigate the best choice of a master version in an adaptive streaming scenario, such a scenario was simulated by selecting different encoded versions of the video based on target bit rates and resolutions. Each of the provided versions is used once as the master version to predict the CU decisions of the lower version when selecting a different master version.

The results of the simulation are shown in Table V with \( n = 9 \) for all sequences except BasketballDrive with \( n = 8 \). For the tested sequences, the choice of the master sequence makes a difference of at most 3.7% in average BD-rate, which is seen for Cactus between a master version at 479 kbps and a master version at 6186 kbps. Since this difference can be considered relatively small in practical scenarios, and since using the lowest resolution as the master version results in the highest complexity reduction, the versions at the lowest resolution appear to be the best candidates for being the master version. An example of this is seen with ParkScene where a master version at 690 kbps has a BD-rate of 5.2% with a complexity reduction of 67.3%.

V. CONCLUSION AND FUTURE WORK

In this paper, a fast simultaneous encoder architecture was proposed. This encoder exploits the correlation between different versions of the same video encoded for adaptive streaming. By using this correlation, only one video needs to be fully
for Innovation by Science & Technology (IWT), the Fund for Scientific Research (FWO Flanders), and the European Union, and were carried out using the Stevin Supercomputer Infrastructure at Ghent University.

### References


