Abstract — This paper describes in detail an algorithm proposed for improving the quality of video when there is a quality drop in a streaming environment. Based on the Dynamic Adaptive Streaming over HTTP (DASH) standard and using the High Efficiency Video Coding (HEVC) codec this solution aims to make the decoder improve the quality of the frames following a quality drop that may occur while streaming a video. This solution will allow the decoder to continue decoding a HEVC compliant stream while improving the quality.

Index Terms— MPEG-HEVC, MPEG-DASH, decoder, video processing, video coding.

I. INTRODUCTION

This paper will focus on the development of an algorithm that aims to increase video quality in a streaming environment based on existing technologies.

In the past years there has been a lot of progress in the compression of video. Several codecs have been released obtaining increasing compression ratios, the most recent being High Efficiency Video Coding (HEVC). To achieve this, complexity has mainly been added to the encoder while the decoder continues with a much lower complexity. Given the progress in hardware that has occurred the limitations of making the decoder work more, originally due to the fact that it had to be easily distributable and cheap, have started to fade giving way for solutions that require decoder complexity to be more interesting.

On the other hand the internet has become a video content provider for millions of people. Streaming video has made the demand for high performing codecs increase to save on as much bandwidth as possible so as not to overload network capacity. In the complex networks that are now in use there are times when a user may not have enough bandwidth to guarantee that the video can be loaded in time. One solution to this problem is to adapt the bandwidth a video needs by coding in various qualities, done by a protocol called Dynamic Adaptive Streaming over HTTP (DASH). Lower qualities will occupy less memory and therefore need less bandwidth and vice versa with higher quality segments. On one hand this leads to a more seamless video viewing experience but may lead to a quality drop while streaming.

With this in mind the objective of this work is to develop a solution that while increasing video quality in the case of a quality drop by exploiting video redundancy and that adds complexity only to the decoder. The solution should be compliant with a HEVC bitstream, only altering the decoder while, above all else, improving the quality of the lower quality segment.

In the following section an overview of the HEVC and DASH standard will be presented. Section III will discuss in detail the algorithm developed with an architecture and explanation of each module. In Section IV the methodology, conditions and metrics will be discussed followed by the results obtained for the algorithm.

II. REVIEW OF RELEVANT BACKGROUND TECHNOLOGIES

In this Section the 2 relevant technologies mentioned before, DASH and HEVC will be discussed highlighting the most important features of these technologies.

A. DASH Overview

DASH was created out of the need of having an adaptive streaming protocol that could be used by all therefore overcoming the previous ones that existed from companies like Microsoft and Apple. The reason that these protocols exist has to do with the available bandwidth a user has while streaming that may vary depending on the network. What DASH does is regulate the quality of the video the server is sending to the client in real time therefore adapting to the conditions.

![DASH Diagram](image)

Fig. 1. An example of how DASH can adapt depending on available bandwidth.

To achieve the versatility needed DASH has 2 key elements: coded segments and the Media Presentation Description (MPD):

1) Video and audio is coded at the server offline in time intervals previously decided, ranging anywhere between a couple of seconds to minutes. Each of these intervals is coded in different qualities, the more qualities the more adaptive DASH becomes.

2) The MPD regards metadata about the content. It will contain the Uniform Resource Locators (URLs) of the...
segments, byte range of the segments, segment availability, presentation start time and the segment duration.

The DASH architecture (see Fig. 2) relies on the MPD to tell the DASH client where the next segment they should get is located. The DASH client makes this decision based on information available such as Internet Service Providers’ bandwidth information.

Fig. 2. DASH architecture [1]

A typical session will play out as follows:

1) The client begins by requesting the MPD file for information about the available audio-video content, thus obtaining the available rates, segment lengths, etc.
2) Once informed, the client may request the segments chosen in accordance with the bitrate available; from this point on, the client will be in charge of requesting the best segments to fit the network conditions.
3) When the client requests a change, the server complies by transiting to a lower or higher quality with the change occurring at the next Switch Access Point (SAPs), positions in the coded stream where random access and thus switching is possible.

B. HEVC Overview

The general architecture of the HEVC encoder is similar to previous standards’ encoders and is presented in Fig. 3.

Fig. 3. HEVC architecture [2]

The input video signal is available in PCM format and each frame is initially divided into so-called Coding Tree Units (CTUs), one of the new tools in HEVC. When Inter coding is used, the difference between a so-called Coding Unit (CU) and some equal-size part of a reference (decoded) frame or frames (available at both the encoder and decoder) is then computed, transformed, scaled and quantized. These processes are controlled by the encoder control which takes the necessary decisions and creates the required flags, modes and parameters to be sent to and control the decoder. As usual, temporal predictions are made more effective by means of motion estimation which strongly increases the encoder complexity. Filters are also applied to reduce the visual artefacts, notably the block effect. All the created symbols, notably motion vectors and transform coefficients, are after entropy coded using context-adaptive binary arithmetic coding (CABAC), a form of lossless entropy coding that eliminates statistical redundancies, thus contributing to a better compression factor. Some of the more relevant tools will be explained in more detail.

1) In contrast to previous video codecs, HEVC does not use the concept of macroblock (16x16 luminance samples) anymore but instead uses CTUs, which are used to initially divide the frames and may take the sizes 64x64, 32x32 or 16x16, see Fig. 4. A CTU contains a luminance block and two chrominance blocks, called Coding Tree Blocks (CTBs). For coding purposes, the CTUs are dynamically divided into CUs; CUs can range from 8x8 to 64x64 (they are always square and have dimensions that are powers of 2). When there are large areas with little detail, the CUs are made larger. There are also the transform units (TUs) and transform blocks (TBs) which may go from 4x4 to 32x32 pixels in size. The TUs are associated to the size of the transform applied to the residual error within the CU. When using the Intra or Inter prediction, the encoder uses the so-called Prediction Units (PUs) and Prediction Blocks (PBs). PUs designate the area where prediction will happen and the PBs are the associated blocks, from between 4x4 to 64x64 pixels.

Fig. 4. Example of an image being divided into CTUs [2]

2) Slices are a sequential set of CTUs that can, in most instances, be independently decoded although their main purpose is re-synchronization. The slice size may be determined by the maximum number of bits in a CTU, typically leading to a variable number of CTUs in each slice. Slices can be divided into 3 categories depending on the associated coding modes: in I-slices, all CUs are Intra coded; in P-slices, only Intra coding, and Inter coding with a single reference for each CU may happen; finally, B-slices may use the Intra and Inter coding modes without limitations. Tiles, another organizational division, are more or less equal divisions of a frame which are also each independently decodable. The main purpose is to take advantage of multi-core processors.

3) The HEVC Intra coding innovations have led to an average 22% bitrate reduction regarding H.264/AVC for the same video content and quality [3]. The main improvement
regards the set of possible angular predictions, a process by which samples values in a CU are first predicted using the values of their spatial neighbours. This tool is very helpful to reduce the Intra prediction error and HEVC has increased the number of Intra prediction modes to 33 (see Fig. 5) different angular positions (from at most 9).

Fig. 5. Angular predictions available for intra-coding [2]

4) The main HEVC transform is again the integer DCT. However, the iDCT transform can now be applied to several block sizes, naturally associated to the most efficient TU size. A new transform has also been introduced in HEVC, the discrete sine transform (DST), which shows better results for Intra coded 4x4 luma blocks.

5) There are various ways to divide and code a PU, thus allowing for greater freedom at the encoder. A CU with size 2Nx2N can be PU partitioned into 4 equal squares, two divisions (vertical or horizontal), two asymmetrical divisions and may also have dimension 2Nx2N. For luma samples, HEVC “uses a single consistent separable interpolation process to generate all fractional positions without intermediate rounding operations, which improves precision and simplifies the architecture of the fractional sample interpolation” [2]. The motion vectors have quarter pixel accuracy which leads to better predictions, naturally at the cost of increased complexity.

6) Quantization is similar to the previous standard with a quantization step that can be altered at the CU level and varies between 0 and 51 with a bitrate reduction of approximately 12.5% per unit increment (the scale is logarithmic).

7) Entropy coding is similar to the previous standard, notably CABAC has been selected. Transform coefficient scanning is performed in 4x4 sub-blocks for all TU sizes (i.e. using only one coefficient region for the 4x4 TB size, and using multiple 4x4 coefficient regions for larger transform blocks). Three coefficient scanning methods are available: diagonal up-right, horizontal and vertical scans. Usual scanning is diagonal up-right, except for the Intra 4x4 and 8x8 modes which use the horizontal and vertical scanning modes.

8) There are two filters used in the HEVC standard to remove visual artefacts: the de-blocking filter (DBF) and the new sample adaptive offset (SAO) filter that are applied in this order to the frames. The DBF has the objective of reducing the block effect associated with block based coding and is only applicable to the 8x8 block edges. DBF starts by filtering the horizontal and after the vertical edges. The filter can have three filtering strengths (between 0 and 2). The SAO aims to increase the image sharpness and to reach better gradient reconstruction. There are three choices for the encoder: to deactivate it, to use the band offset mode and to use the edge offset mode. The band offset mode divides the pixel range into 32 different bands; then there are four offset values that are applied to pixels. The edge offset mode compares the pixel in question to its neighbours and then decides whether this pixel is at a minimum, maximum, edge or none of them, applying after an appropriate offset so as to smooth the image.

The new tools reviewed above allow reaching higher compression factors, reducing the rate from the previous standard in about 50% for the same perceptual quality [4]. The fact that HEVC can achieve such good compression performance in comparison with its predecessors is largely due to its increased complexity. Increasing the number of coding options available at the encoder makes it more complex, thus more clever and efficient [5].

The solutions presented though do not account for an eventual quality drop that may occur. Despite there being decoder based solutions for this kind of problem ( [6] , [7] ) they are either not for this specific case or are not HEVC compliant. The solution proposed is HEVC complaint and works with the DASH model.

III. ALGORITHM: ARCHITECTURE AND TOOLS

This section describes in detail the framework designed and implemented with the objective of improving the quality of the first frame after a quality drop resulting from the adjustment of the coding rate to the available network rate. This first frame improvement should also allow the following frames to be improved as coding process means they rely on this first reconstructed frame after the drop.

A. System Architecture

Before discussing the algorithm architecture it is important to understand the frames available: Previous Good Frame (GF-1), Good Frame (GF) and Start Frame (SF). The first 2 are in a higher quality while SF is the first frame in a lower quality segment. The improved SF or New Start Frame (NSF) is what the algorithm will ultimately generate (see Fig. 6).

Fig. 6. Illustration of the frames available at the decoder (green and red) and the output (blue)

With this in mind the general architecture can be seen in Fig. 7. The algorithm can be divided into 3 main parts: Static Block Detection, Motion Block Detections and NSF creation. The SF is divided into 4x4 blocks and each of these blocks is classified either as Static, or New Content.
Fig. 7. Architecture of the proposed solution.

Each of these modules will be explained in detail below.

B. Static Block Detection

This part of the Decoder-based Quality Improvement solution exploits the three decoded frames available and mentioned above to determine whether a block is stationary or not. The basic idea is that the static blocks, parts of the image that have not moved, should be copied from the GF to the NSF as the GF has better quality than the SF. This module includes the following steps:

1) Motion Estimation GF to GF-1

The idea behind this step is that most static areas in the GF continue static in the SF (since in most cases motion does not change abruptly) and it would be useful to detect the static (or the areas without significant motion) areas in the SF. As motion estimation works better between frames coded with the similar QPs, it is proposed here to perform motion estimation between the GF and GF-1 frames using 4x4 blocks.

To more reliably detect the motion, it is proposed to perform first integer motion estimation and apply a refinement process with half pixel motion estimation for the blocks whose motion vectors have amplitude values less or equal than 1 in both the x and y coordinates. This serves to detect the blocks with a motion vector as close as possible to (0, 0) and (½, ½). The largest motion vector allowed to still classify a block as static is (½, ½), i.e. any block with a motion vector with one of the components larger than ½/2 is considered as a non-static block and will be processed as a non-static block. Since the GF and GF-1 frames are coded with the same QP, the static regions should be accurately identified and the motion vector for static blocks should be (0, 0) in most cases.

To design the static blocks detection algorithm, an experiment was set up to determine the best block size. All experiments in this section were performed with typical YUV test sequences coded with a QP of 30 for the first 32 frames (the higher quality segment) and a QP of 35 for the remaining frames (the lower quality segment). The prediction structure used was IBBP, with only one intra frame in each segment. There were QP offsets of 1 and 2 that applied to the B and P frames that have a QP value that is 2 above and 1 above the I frame’s QP value respectively. The remaining encoding choices follow the conditions defined for the low_delay_scenario defined by the JCT-VC group [8]. The spatial resolutions for the test sequences vary from 832x480 to 1280x720 for the luminance component.

In this experiment the blocks classified as static were copied from the GF and the remaining ones from the SF, thus creating a new start frame (NSF) where only the static blocks are improved. Despite its well-known limitations, the PSNR was used to assess the quality of the NSF. As shown in Fig. 8, it was found that using larger blocks (8x8 or 16x16 for example) did not provide the best results, especially for sequences with high motion, such as Chinaspeed and Fl owervase. 4x4 blocks were chosen.

2) Static Block Filtering

At this stage, there are several blocks which have been pre-classified as static. However, there is always the possibility that the motion estimation may have failed for some blocks or the assumption that blocks classified as static for the GF are also static for the SF does not hold. To detect these blocks and avoid having them wrongly classified as static two filtering tools have been developed.

- High Quality Frame Similarity Filtering
  This tool aims to check if the block match between GF and GF-1 is correct or not, i.e. if between these two frames the static block candidate is really static. In fact, there are cases where the best match may have been (0, 0) but in reality the blocks are not as similar as they should be and thus copying them to the NSF will not be a good idea for quality improvement. With this purpose in mind, the MSE between the 4x4 block in GF and collocated 4x4 blocks in GF-1 is measured; if this MSE is above 5, then the static block candidate is finally not classified as static.

- High to Low Quality Frame Similarity Filtering
  This tool aims to check if the motion computed between GF-1 and GF is similar to the motion between GF and SF, i.e. if a significant motion change occurred in the SF and the assumption that the block remains static is not valid. With this purpose in mind, the metric, $T_{ABD}$, corresponding to the average intensity difference between the pixels in the SF block and the collocated GF block, is calculated as shown in (1).

$$T_{ABD} = \frac{\sum_{i=1}^{4} \sum_{j=1}^{4} |SF(i, j) - GF(i, j)|}{16}$$  (1)

If $T_{ABD}$ is above 10, then the block is not classified as static anymore. As shown by the NSF performance gains in Fig. 9, the gains seem to stabilize when $T_{ABD}$ is greater than 10, thus justifying the threshold mentioned above. To obtain these results, the previous technique calculated the
worst 30% of the blocks based on their MSE value and then this technique was applied. Later these techniques were combined in their final forms.

![Graph showing NSF PSNR gain (dB) vs. ThADB] Fig. 9. NSF PSNR when not classifying as static the blocks with ThADB greater than a specific value and the worst 30% of the previous MSE based threshold.

The static block candidates that do not fulfil the two filtering criteria mentioned above are not classified as static and thus proceed in the processing flow to be classified as motion or new content blocks as described in the next section.

C. Motion Block Detection

1) Hierarchical Motion Estimation SF to GF

In this module, hierarchical motion estimation is performed between the SF and GF. The hierarchical coarse-to-fine motion estimation solution allows performing fast motion tracking and is less sensitive to the quantization noise, a problem very much present when comparing two frames with different QPs. This procedure starts by performing motion estimation with a large block size where a coarse motion field is calculated; then the motion vectors are refined using progressively smaller blocks. To begin the SF is divided into 16x16 blocks and motion estimation is performed using the GF as the reference frame. The (4x4) blocks that were considered static may be included in this motion estimation process (see Table I).

<table>
<thead>
<tr>
<th>Sequence</th>
<th>NSF PSNR gain using static blocks</th>
<th>NSF PSNR gain using previous MV and static blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Johnny</td>
<td>0.001129</td>
<td>0.00782</td>
</tr>
<tr>
<td>Chinaspeed</td>
<td>0.011203</td>
<td>0.148308</td>
</tr>
<tr>
<td>Flowervase</td>
<td>-0.001343</td>
<td>0.005794</td>
</tr>
</tbody>
</table>

After the 16x16 motion estimation, the SF is further divided into 8x8 blocks and the same motion estimation procedure is reapplied. The motion vectors calculated for a certain coarser scale (e.g. 16x16) are reused as the starting position for the blocks at a finer scale (e.g. 8x8). After this, the SF is divided once more into 4x4 blocks following the same procedure as for the previous splitting.

As usual in motion estimation, the search range defines a square area on the reference frame that is used to look for similar blocks. Thus, it is very important to determine the search range to be used at each level of the hierarchical motion estimation procedure. A set of experiments was performed to determine the best search range value for each motion estimation level. In Fig. 10, the search range value is doubled and halved starting from a 16x16 pixel range as a smaller block size is used; it may be concluded that doubling the search range for each smaller block size yields the best NSF PSNR gains in relation to the SF. For sequences where there is global motion, the gains get better, with the exception of Basketballdrill, due to the capability to get better matches that might be further away due to the motion. The initial 16x16 motion vector serves as a “pointer” in the direction where that block may be and as the size of the block goes down, the smaller blocks search larger areas with better starting locations.

![Graph showing NSF PSNR gain (dB) vs. search range] Fig. 10. NSF PSNR gain when the ME search range is doubled or halved each estimation with respect to a fixed search range value of 16 pixels.

Half pixel precision is done but only used in the last ME iteration because it only makes sense to use it when the motion vector will remain unchanged afterwards.

2) Motion Vector Smoothing

After the motion vectors have been estimated, a weighted spatial motion vector smoothing algorithm is used to regularize the motion vectors, thus removing false or erroneous estimated motion vectors. For example, if all the motion vectors surrounding a 4x4 block are similar between them and very different to the centre block’s motion vector then it is quite likely that the motion estimation process is not reliable for that block. By analysing the neighbouring motion vectors, the central block motion vector may be improved, thus obtaining a smoother motion vector field.

The motion vector smoothing finds the minimum weighted L2-norm between each central block (block under processing) motion vector and the motion vectors for each of the 8 neighbouring blocks. The weight computed for each vector is proportional to how well the specific motion vector being evaluated characterizes the block motion using MSE as the ME reliability metric. The adopted MV smoothing algorithm is described in more detail in [9]. The NSF PSNR gains of using this type of filtering are shown in Fig. 11.
3) Feature Extraction and SVM

The final two steps regard the extraction of three features and their use in a support vector machine to determine which blocks should be classified as new content blocks and which blocks should be classified as new content. To select the most discriminative features for the SVM, many global and local features were evaluated, both at block level (4x4 blocks) and frame level (such as affine model parameters). A good feature should allow to easily distinguish a motion block from a new content block. The three features used are explained below.

The first feature, \( SE_\alpha \) (2), is the logarithm of the square error between the SF block under processing and the collocated GF block (see Fig. 12). This feature serves as a metric of how similar the two frames (SF and GF) are.

\[
SE_\alpha = \log_{10} \sum_{i=1}^{4} \sum_{j=1}^{4} (SF(i,j) - GF(i,j))^2
\]  

(2)

The second feature, \( SE_\beta \) (3), is the logarithm of the square error between the motion compensated GF (MCGF) using the estimated motion vector field and the GF block collocated with the SF block under processing (see Fig. 12). By using this feature, the best GF motion estimation, MCGF, can be compared to the GF high quality collocated with SF to help detect motion blocks that might have not been detected in the static section.

\[
SE_\beta = \log_{10} \sum_{i=1}^{4} \sum_{j=1}^{4} (GF(i,j) - MCGF(i,j))^2
\]  

(3)

The final feature, \( \Delta_{\text{chroma}} \), is defined using the SF block and the MCGF block chrominances. If the chroma samples are very different between the SF and MCGF, then the blocks are most likely from different areas in the frame and thus may not have a strong enough similarity to justify being classified as a motion block. In this feature, both the U and V absolute differences between each of the chroma samples available are added with \( \Delta_{\text{chroma}} \) coming out as follows:

\[
\Delta_{\text{chroma}} = \log_{10} \sum_{i,j} |SF_u(i,j) - MCGF_u(i,j)| + \sum_{i,j} |SF_v(i,j) - MCGF_v(i,j)|
\]  

(4)

All these features have logarithms applied as it was verified that performing this action helped separate the motion and new content blocks.

D. NSF Creation

Finally, after classifying each SF block the NSF frame is created by applying the appropriate processing to each type of block, notably:

- Static block processing – All blocks considered static are copied from the collocated GF block. These blocks improve the quality of NSF.
- Motion block processing – Blocks considered motion are obtained from the MCGF block. These blocks, if well classified by the SVM, should also improve the quality of the NSF.
- New content block processing – These blocks are the only ones obtained from the SF and therefore do not lead to an increase or decrease of quality in the NSF.

A visual representation of this part of the algorithm can be seen in Fig. 12.

E. Remarks

In summary, the proposed algorithm attempts to classify the image blocks into three typical cases that appear in video: static, motion and new content regions. By dividing the problem into three smaller problems and tackling each individually, the resulting NSF should be reach quality than the SF and also allow improving the quality of the subsequent frames.

The strong point of this algorithm is that it can be applied at any type decoder (HEVC or not) without having to change the coding process. Despite being implemented for HEVC, it can theoretically work for most codecs used nowadays with similar success.

The weak point of this algorithm is the SVM as it is very hard to train well enough that it can classify with great precision the motion and the new content blocks.

IV. PERFORMANCE EVALUATION

A. Test Methodology

To test the performance of the proposed algorithm the following methodology was designed (see Fig. 13):
B. Test Conditions

Three different quality drops were tested:

- Quality Drop 1 – QP increases from 30 to 35
- Quality Drop 2 – QP increases from 30 to 40
- Quality Drop 3 – QP increases from 35 to 40

In HEVC the QP can vary from 0 to 51 with higher values corresponding to lower qualities. With these 3 quality drops 2 different jump sizes are tested, 5 and 10, as well as 2 different quality ranges for SF and GF. The frames were coded using an IBBP structure where only the first frame was an I-frame. There were QP offsets of 1 for the P frames (meaning the QP value was one higher than stated for these frames) and of 2 for the B frames. Despite DASH segments being the same size the sequences used contain different jump sizes are tested, 5 and 10, as well as 2 different size ranges for SF and GF. The frames were coded using an IBBP structure where only the first frame was an I-frame. There were QP offsets of 1 for the P frames (meaning the QP value was one higher than stated for these frames) and of 2 for the B frames. Despite DASH segments being the same size the sequences used were divided into 2 different sizes: the high quality segment was 32 frames long while the remaining frames of the sequences used constituted the lower segment. This was done to see the effect over a long period of time. The sequences used and their details can be seen in Table II.

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Spatial res.</th>
<th>Frame rate</th>
<th>Nº frames</th>
<th>Type of motion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basketballdrill</td>
<td>832x480</td>
<td>50</td>
<td>300</td>
<td>High Motion</td>
</tr>
<tr>
<td>BQMail</td>
<td>832x480</td>
<td>60</td>
<td>600</td>
<td>Panning</td>
</tr>
<tr>
<td>Chinaspeed</td>
<td>1024x768</td>
<td>30</td>
<td>500</td>
<td>High Motion</td>
</tr>
<tr>
<td>Johnny</td>
<td>1280x720</td>
<td>60</td>
<td>600</td>
<td>Low Motion</td>
</tr>
<tr>
<td>Flowervase</td>
<td>832x480</td>
<td>30</td>
<td>300</td>
<td>Zoom In</td>
</tr>
<tr>
<td>4 People</td>
<td>1280x720</td>
<td>60</td>
<td>600</td>
<td>Low Motion</td>
</tr>
<tr>
<td>Partyscene</td>
<td>832x480</td>
<td>50</td>
<td>500</td>
<td>Zoom In</td>
</tr>
<tr>
<td>Slide Editing</td>
<td>1280x720</td>
<td>30</td>
<td>300</td>
<td>Low Motion</td>
</tr>
<tr>
<td>Vidyo1</td>
<td>1280x720</td>
<td>60</td>
<td>600</td>
<td>Low Motion</td>
</tr>
</tbody>
</table>

C. Performance Metrics

The objective of this work is not to only improve the quality of the first frame SF, but also improve the remaining frames in the low quality segment. With this in mind the following metrics were chosen to analyze the performance of the algorithm:

1) PSNR/SSIM NSF Gain

This metric aims to measure how much the NSF has improved in regards to the SF. The PSNR ((5) and (6)) and SSIM (7) values are calculated for both the SF and NSF using the raw frame as a comparison and then the difference is calculated.

\[
MSE = \frac{1}{mn} \sum_{i=0}^{m} \sum_{j=0}^{n} (A(i,j) - B(i,j))^2 \quad (5)
\]

\[
PSNR = 10 \log_{10} \left( \frac{MAX^2}{MSE} \right) \quad (6)
\]

\[
SSIM(x,y) = \frac{(2 \mu_x \mu_y + c_1) + (2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \quad (7)
\]

In (5) \(m\) and \(n\) represent the dimensions of the frame and in (6) \(MAX\) corresponds to 255, the maximum value for luma samples. In (7) \(\mu_x\) and \(\mu_y\) correspond to the averages for \(x\) and \(y\), \(\sigma_x^2\) and \(\sigma_y^2\) the variances for \(x\) and \(y\), \(\sigma_{xy}\) the covariance for \(x\) and \(y\) and \(c_1\) and \(c_2\) are two variables to stabilize the division (\(c_1 = 6.025\) and \(c_2 = 58.5225\)).

2) Average Gains in the First Seconds

This metric targets to measure how much the insertion of the NSF influences, positively or negatively, the remaining frames in the lower quality segment in the first (AG1) and second (AG2) seconds, this means in the first 25/30 or 50/60 frames depending on the sequence frame rate. This metric is the average of the PSNR for this period of time.

3) Number of Frames with Gain Above \(\delta\)

This metric targets to measure for how long, this means for how many frames, the quality gain obtained for the NSF propagates in time by detecting when the quality gain regarding the SF based decoding drops below a certain threshold \(\delta\); in the experiments, \(\delta\) was fixed at 0.1 dB.

D. Results and Analysis

1) PSNR/SSIM NSF Gain

To better understand how the algorithm behaved the results for this metric were obtained only for static blocks (see Table III) where all other blocks were treated as new content and then the final results were obtained (see Table IV).
These results obtained for the static module allow to derive the following conclusions:

- There are positive average PSNR and SSIM NSF gains for the three quality drops tested.
- Flowervase, a zoom in sequence, is the only sequence which actually dropped in quality, both for SSIM and PSNR, although only for Quality Drop 1 (with a quality loss which is almost negligible). Partyscene, another zoom in sequence, has almost negligible gains. As zoom in sequences have few static blocks, these results are expected.
- On the other hand low motion sequences such as Johnny, Vidyo1 and 4 People had large quality gains as could be expected. High motion sequences were also able to improve the SF as they also contain static areas that can benefit from the proposed algorithm.
- The static block processing module is capable of identifying static areas with great success both in high and low motion sequences as shown by the averages for all sequences which in PSNR are all above 0.5 dB and in SSIM all positive.

Table IV shows the final PSNR and SSIM NSF gains and allows deriving the following conclusions:

- Motion block detection is more successful for the Quality Drop 2 with an average PSNR gain of around 0.35 dB while for Quality Drop 1 the quality gain is much smaller, around 0.05 dB. The Quality Drop 2 case shows higher gains because the quality difference between segments is larger and thus is easier to obtain PSNR NSF gains than when the quality drop is smaller. Slide Editing has the largest quality gain, on average 2.695 dB, most likely because it is a computer generated sequence. From all the sequences with high motion, Chinaspread is the one gaining most for the 3 quality drops, on average, 0.625 dB for the static blocks and 0.35 dB for the motion blocks, showing that the proposed algorithm behaves better for artificial content where the motion is more clearly defined.
- Naturally, low motion sequences gain less with the motion processing part of the algorithm as there are few blocks that can benefit.
- It is important to note that there are 2 sequences with negative gains, Flowervase and Partyscene, both corresponding to zoom ins and only for the Quality Drop 1; these are the only 2 cases with negative NSF quality gains. This may happen due to the nature of a zoom in and the inability to find good matches in terms of motion estimation when a translation motion model is adopted; moreover, there is naturally a low gain coming from the static blocks as there are not that many.

2) Average Gains in the First Seconds

As quality gains in a single frame may not be that relevant from the subjective quality point of view, it is very important to understand how the NSF improvements will affect the remaining frames in the lower quality segment as the objective is to improve not only the SF but as many subsequent frames as possible. The results can be seen in Table V.

Table V

AVERAGE PSNR GAIN [dB] IN THE FIRST 2 SECONDS AFTER THE SF
The results in Table V allow deriving the following conclusions:

- For the sequences with more motion and zoom in, Basketballdrill, BQMall, Flowervase and Partyscene, the gains achieved in the first frame are lost in the following 2 seconds although, excluding Flowervase, most losses are negligible.
- For the more static and artificial sequences, the quality improvement seems to hold even after 2 seconds. This is to be expected as the static blocks’ gain is still valid throughout the remainder of the sequence. Once again, Slide Editing maintains its gains the best with average PSNR gains above 1 dB in the first 2 seconds.
- Quality Drop 2 has the best AG1 value but surprisingly Quality Drop 3 has the best AG2 value although the difference is small.

3) Number of Frames with Gain Above $\delta$

This metric serves to assess how long the quality improvement is non-negligible by counting the number of frames where the gain is above a certain threshold. It is important to note that even if the PSNR gain drops below $\delta$ (in this case 0.1 dB), this does not mean that the sequence is worse than the SF decoded sequence (see Table VI).

### Table VI

<table>
<thead>
<tr>
<th></th>
<th>Quality Drop 1</th>
<th>Quality Drop 2</th>
<th>Quality Drop 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basketballdrill</td>
<td>8</td>
<td>2</td>
<td>13</td>
</tr>
<tr>
<td>BQMall</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Chinaspeed</td>
<td>239</td>
<td>21</td>
<td>212</td>
</tr>
<tr>
<td>Johnny</td>
<td>90</td>
<td>270</td>
<td>568</td>
</tr>
<tr>
<td>Flowervase</td>
<td>0</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>4 People</td>
<td>568</td>
<td>568</td>
<td>360</td>
</tr>
<tr>
<td>Partyscene</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Slide Editing</td>
<td>268</td>
<td>268</td>
<td>268</td>
</tr>
<tr>
<td>Vidyo1</td>
<td>292</td>
<td>278</td>
<td>299</td>
</tr>
<tr>
<td>Average</td>
<td>162.888</td>
<td>157.222</td>
<td>191.444</td>
</tr>
</tbody>
</table>

The following conclusion can be drawn:

- High motion sequences quickly lose the quality gains achieved in the first frames after the SF while static frames take benefit for much longer, sometimes until the end of the lower quality segment, still having PSNR gains above $\delta$. The sequence 4 People has the best results followed by Vidyo1 and Slide Editing. The worst sequences are the zoom in sequences, Flowervase and Partyscene, which either have gains below $\delta$ to begin with or lose them in the first few frames after SF.
- The Quality Drop 2 is the worse on average, most likely due to the Chinaspeed performance. Despite Chinaspeed having good averages for the first 2 seconds, the PSNR dips below 0.1 dB quite early in comparison with the other cases, perhaps due to the larger difference in QP values. Quality Drop 3 has the most frames with benefits on average due to the Johnny sequence performing very well in comparison to the other two quality drop cases.

### E. Examples

Presented here are 2 examples of NSF frames that were improved. In the first (Fig. 14) is Johnny, zoomed in, in Quality Drop 1. On the left is the SF and on the right is NSF.

In Fig. 15 Slide Editing is shown with its improvement in Quality Drop 2. The improvement is most noticeable in the bullet points which are dots in SF and arrows in NSF.

### F. Final Remarks

The performance results are positive and show that the proposed algorithm works very well especially with static sequences, consistently having high long lasting quality gains. These gains are verified by the PSNR and SSIM metrics for the SF frame and the PSNR averages in the first two seconds. The gains can also remain for a significant period after the SF frame and the PSNR averages in the first two seconds. Sequences with high motion easily lose the quality gains obtained in the SF frame as static blocks are most likely covered up my moving ones further on in time. Despite this effect, there are still relevant initial gains.

Zoom ins have shown to be a more complicated type of content as static blocks almost don’t exist and the motion is not translational while the used motion model it is. This results into rather low gains and also losing them quickly when they are initially positive. Despite this effect, it was possible to obtain positive gains for the SF frame in both zoom sequences for the Quality Drops 2 and 3.

Finally, artificial content showed to have a lot to gain from the proposed algorithm as both high and low motion sequences have good long lasting quality gains. This is most
likely due to motion estimation that works much better as there is much less noise than for other types of content.

V. CONCLUSION

This paper has discussed two of the most recent technologies used in streaming, the codec HEVC and the streaming protocol DASH. Varying bandwidth availability may lead to a quality drop while streaming a video and this creates an undesirable user experience. Taking into account less hardware limitations the old paradigm of concentrating codec complexity on encoders has passed and the objective of creating a HEVC bitstream compliant solution where the decoder would work more was introduced.

An algorithm to achieve this was developed adding complexity solely to the decoder and exploiting the relations in video to combat the quality drops that may occur. Only the decoder was altered meaning all complexity added was decoder-side as initially intended. The algorithm essentially divides the frame into 4x4 blocks at the time of the quality drop into 3 different categories: static blocks, motion blocks and new content blocks. Each of these is treated in a different manner and a new frame is created and inserted at the beginning of the lower quality segment resulting in a propagation of the higher quality frame in the subsequent frames.

The results achieved are positive in general. In almost all tested cases the first frame gained in regards to the original one. Motion patterns like zoom ins proved themselves to be hard to treat not responding very well. Low motion sequences responded very well as well as artificial sequences having high initial gains as well as long lasting gains.

Further studies in this field may want to investigate over motion characterization of the sequences to detect zoom ins as well as improving the motion and new content classification. Another direction may be exploiting more high quality frames or still using information contained in the HEVC bitstream.

VI. BIBLIOGRAPHY


