Elecard StreamEye 4.x

Application Use Cases

Assessment of Efficient Usage of Encoding Tools for H.264/AVC Standard
Contents

Introduction ........................................................................................................................................... 3
  Application ........................................................................................................................................... 3
  Preconditions ......................................................................................................................................... 3
Understanding of GOP Structure ........................................................................................................... 5
  Utilizing I-Frames ................................................................................................................................... 5
  Utilizing P- and B-Frames ....................................................................................................................... 6
Video profile and level ............................................................................................................................. 6
Entropy Encoding: CABAC and CAVLC ............................................................................................... 8
Deblock Filtering ..................................................................................................................................... 9
Modes of Macroblock Subdivision and Prediction ................................................................................ 13
Analysis of block predictions ................................................................................................................ 16
  Reduction of Spatial Redundancy (Intra Prediction) ........................................................................... 16
  Reduction of Interframe Redundancy (Motion Estimation) ................................................................. 17
Adaptive Quantization ............................................................................................................................ 23
Introduction

Application

The document describes essential ways of utilizing Elecard StreamEye v.4 to assess effectiveness of compression of video sequences encoded with H.264/AVC codecs. The document also briefly describes the software functions.

About StreamEye

Elecard StreamEye is a powerful software tool designed for professionals and prosumers in video compression field. Elecard StreamEye enables the user to perform an effective in-depth analysis of video sequences.

Elecard StreamEye provides a visual representation of the encoded video features and a stream structure analysis of MPEG-1/2 or AVC/H.264 Video Elementary Streams (VES), HEVC/H.265 (ISO/IEC 23008-2 MPEG-H Part 2) Video Elementary Stream (VES), MPEG-1 System Streams (SS), MPEG-2 Program Streams (PS) and MPEG-2 Transport Streams (TS).

Automated analysis and standard compliance check is available with the command line tool of the StreamEye program. Configuration via XML-file provides an easy and flexible way to pick specific information from the processed stream and save it into CVS-files.

Initial Setup

To perform the scenario, you must have a video sequence in the supported format (the demo video sequence is available via this link) and Elecard StreamEye v.4, installed (click to request the demo for Windows or Mac).

Run Elecard StreamEye v.4 and open the source video file by selecting the File → Open command. As a result, the first frame of the opened video sequence appears in the main program window (Fig.1). In the described usage scenario, the following elements are utilized: BarChart (number 1 in Fig.1) and the window, which displays decoded video frames (number 2 in Fig.1).
Fig. 1: The main StreamEye 4 window with an opened video file
Understanding of GOP Structure

To make a basic analysis of the video sequence, use the BarChart element (number 1 in Fig.1). BarChart displays vertical rectangles colored red, blue, or green. Each rectangle corresponds to an individual frame of the video sequence encoded as I-frame (Intra predicted, red), P-frame (Predicted, blue), and B-frame (Bidirectional, green). The height of each rectangle represents a size of the encoded frame in bytes. In contrast to other frames of a video sequence, I-frames (red rectangles) are usually largest because they are encoded apart from other frames of the video sequence. P-frames and B-frames are usually smaller because they contain only encoded data about changes occurred in previous frames. For more detailed description of navigation elements, see Section 4.2.2 of User Manual.

BarChart allows you to assess frame size in bytes and frequency of usage of described frame types in encoded sequence. A group of pictures (GOP) means sequence of frames of different types (I, P, and B) between two consecutive I-frames.

Utilizing I-Frames

I-frames or key frames are encoded independently from other frames of the video sequence. Decoding of the whole video sequence, including scenarios at positioning, starts with key frames (also referred to as IDR frames; all the other frames positioned in between IDR frames cannot make reference or be dependent to/on the frames outside of this interval). For instance, video starts playing only after receiving a key frame at network broadcasting.

Hence, frequency of I-frames usage determines:
- positioning time (in offline systems);
- latency: the length of time it takes to receive video data till the moment video playback starts (in broadcasting systems).

Depending on the video system intended use, one key frame is used per each 1–10 seconds of a video sequence. The larger GOP is, the less frequently I-frames are used, thus video bitrate is smaller (or quality is better at the same bitrate). Another scenario of key frames usage is response to video scene change, for example shot type change. Since the main purpose of P- and B-frames is to encode changes in the current frame in comparison with preceding frames then on scene change, the difference between the current frame and previous ones becomes considerable; therefore, it is better to use I-frames instead of P- or B-frame at the beginning of a new scene. You can save or adjust the frequency of key frames usage depending on the practical use of your video system.
Utilizing P- and B-Frames

The most universal and widely used structure of P- and B-frames sequence is 2-3 B-frames per one P-frame. B-frames must be times smaller than P-frames if used properly. Each B-frame adds extra delay because buffering and re-ordering frames is required. From 1 to 15 preceding frames can be used to predict video data of the current frame within H.264/AVC standard.

The BarChart window shows all frames that the current frame depends on, as well as decoding and frame reordering sequence.

Video profile and level

Any video encoded in compliance with H.264/AVC standard contains headers, which determine general video properties such as resolution, frame frequency, bitrate, etc. Another piece of important data is compliance of the encoded video sequence to parameters of a definite encoding profile and level. Decoders use this data to pre-calculate their capabilities of decoding video sequences and allocate required storage space before video sequence frames start to be received and decoded.

Information about compliance to parameters of a definite profile and level is presented in the syntactic element called Sequence Parameter Set (SPS). There are two ways to learn about the selected profile and level with the help of Elecard StreamEye v.4. The first way is to open a window providing general information about the video (Fig.2). To open the window, select the View→Info→Stream command. The second way is to view corresponding items of the SPS syntactic element, which opens via the command View→Info→Headers.
The profile selected for encoding provides the decoder with primary information about requirements for supporting definite optional encode tools within H.264/AVC standard. For example, profiles, which differ from Baseline, admit the possibility to apply adaptive binary arithmetical encoding. As a result, the decoder must also support this tool to be able to restore the encoded video sequence. In most cases the decoder will not playback the video sequence if the decoder does not support this type of entropy encoding.

The encoding level specified in the headers of the video sequence applies specific requirements to the decoder performance. For example, supported level in the profile will indicate the maximum image resolution, frame rate and bitrate, thus, for example, a hardware decoder must provide the required bus capacity, network interface, etc. A decoder complying with the level must decode all bit streams, which are encoded for this level and for all lower levels.
Entropy Encoding: CABAC and CAVLC

H.264/AVC standard describes two modes of entropy encoding of stream syntactic elements:

- CAVLC – Context-Based Adaptive Variable-Length Coding;
- CABAC – Context-Based Adaptive Binary Arithmetic Coding.

The video encoder selects an entropy encoder used for video stream compression. The only limitation of H.264/AVC standard is that CABAC is supported only in Main Profile and higher profiles, and is not supported in Baseline Profile and Constrained Baseline Profile (see Section A.2 of the ITU-T H.264/AVC specification).

Use of CABAC provides compression, which is more efficient than CAVLC, and allows you to get a bitrate that is 10% lower with the same noise/distortion level. CABAC takes more time for calculations when encoding and decoding data. Plain compression systems support only CAVLC to reduce costs for hardware and video encoder development. However, it is ineffective to use CAVLC in most of video systems.

To define an entropy encoder mode used for encoding a video segment, in Elecard StreamEye v.4 open the Stream element containing general information about the video sequence. The “coding mode” parameter displays CABAC or CAVLC (Fig.3).

![Fig.3: The Stream element showing data about profile and entropy encoding mode](image-url)
Pay attention to the “profile” parameter (Fig.3). If the parameter displays a value that differs from Baseline, and CAVLC is used at the same time, this means that the video encoder is not configured properly.

**Deblock Filtering**

H.264/AVC standard allows including deblock filtering into the encoding/decoding process. While encoding, each frame of the video sequence is divided into blocks. Deblock filtering is designed to reduce blocking artefacts which occurs on borders of these non-crossing blocks. The higher quantization level used during encoding is, the more blocking artefacts occur on block borders. In other words, data loss increases during compression if the bitrate decreases. The data loss intensifies pixel brightness difference on block borders and reduces the image quality.

Deblock filtering is optional to use during compression and usually it is enabled in the encoder configuration. In general, deblock filtering improves the visual quality of video frames and improves effectiveness of P- and B-frame compression. At the same time, if deblock filtering is used for encoding, then deblock filtering must be used for decoding. It results in more calculations and performance of encoder/decoder.

It is recommended to use deblock filtering regardless of slight increase in data calculation.

In StreamEye v.4, there are two ways to identify if deblock filtering is used.

The most effective way is to check corresponding PPS (pic_parameter_set) and SPS (seq_parameter_set) header items in the *headers* element (Fig.4). The `deblocking_filter_control_present_flag` item shows if there are any additional items in slice headers, which enable/disable deblock filtering. If `deblocking_filter_control_present_flag=0`, these commands are absent in the stream and deblock filtering is enabled by default. If `deblocking_filter_control_present_flag=1` (Fig.4), the `disable_deblocking_filter_idc` item must be present in each slice header (Fig. 5). If value \(1\) is assigned to the item, deblock filtering is not used in the corresponding slice.

Therefore, if `deblocking_filter_control_present_flag=1`, you can make sure that deblock filtering is used only by checking all frames or at least several frames at random. In such cases, it is better to use the other way, the visual one.
Fig. 4: Headers element with opened pic_parameter_set items
Fig. 5: Headers element with opened slice_header items
Use the *Difference* tool to visually identify if deblock filtering is utilized. The tool allows you to compare two random frames of the video sequence using one of the metrics on several stages of decoding.

To open settings, click the *Difference* tool placed on the toolbar (Fig. 6). Then in the *Compare* drop-down list, select the *Compare* metric, which shows the slightest differences between two pictures. In the *Decoded* drop-down list, select the *Decoded* value which means that the video frame is presented in a decoded view. In the *Reference* drop-down list, select the *Unfiltered* value which means that the frame is presented as the original one before filtering. The final view of the settings pane is shown in Fig. 7.

![Fig. 6: Difference tool arranged on the toolbar](image)

In the area of video frame display, go to the *Difference* panel, which is used to show metrics. When deblock filtering is disabled, decoded and unfiltered view of frames will not have any differences, and the picture will be visually homogeneous on the *Difference* pane. If the deblock filtering is used, you can easily notice differences between two views of the video frame (Fig. 8).
Fig. 8: Pane for visual display of difference between two frames

By default, the difference between frames is displayed for full color YUV presentation. StreamEye v.4 allows selecting a required color component from the drop-down list of the toolbar and examine differences for each color component separately (Fig. 9).

Fig. 9: Pane for selecting a color display of video data

Modes of Macroblock Subdivision and Prediction

StreamEye v.4 allows displaying information about prediction type and block subdivision. Control elements are placed on the Visualization toolbar in the upper part of the window (Fig. 10).
To make the compression efficient, the encoder must select proper prediction and block subdivision type. Let’s see P-frame №26 of the demo video sequence (Fig. 11). Blue segments indicate inter prediction of blocks from video sequence frames encoded earlier. To restore such blocks, it is necessary to copy a required field of the preceding frame and add the difference between blocks information. Transparent blocks illustrate cases of interframe predictions when remaining data has not been encoded. That means that to restore a block, it is necessary just to copy a required field of the previous frame.

Red segments indicate blocks of intra prediction. Intra blocks are required on P- and B-frames only in cases when new objects appear in the scene, or when object move is accompanied with deformation. In general, these are cases when prediction from previous frames differs significantly from data contained in the predicted block. Fig. 11 shows a biker moving uniformly in parallel to with the frame. Such movement can be successfully predicted from frames encoded before, by saving only the difference information. However, the encoder selected intra prediction. It means that the compression might be ineffective. For more details, see the section about the removal of interframe redundancy.
Besides, frame regions belonging to static background must be encoded in the mode of transformation coefficient skip. These regions are displayed with transparent blocks. The background of the frame №26 is covered with blue blocks in the majority of cases, which means the encoder took ineffective decision. If you have a look at №29 (Fig. 12), you can notice a great number of transparent blocks, which means more effective encoding.
Analysis of block predictions

The next two sections describe ways of effectiveness analysis of intra and inter block predictions.

Reduction of Spatial Redundancy (Intra Prediction)

H.264/AVC standard defines 10 methods of spatial macroblock prediction. Possible ways of spatial prediction for macroblocks of different size are provided in Table 1. Spatial prediction of blocks with the size of 8x8 pixels is available starting from High Profile, and is unavailable in Baseline and Main profiles.

The encoder must select the most effective mode of spatial block prediction. To reduce complexity of compression algorithm calculation of, the encoder can skip some of the available prediction modes, and this fact may greatly affect the compression effectiveness.

Table 1. Acceptable modes of spatial prediction for blocks of different sizes

<table>
<thead>
<tr>
<th>Prediction Mode</th>
<th>4x4</th>
<th>8x8</th>
<th>16x16</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLANE</td>
<td>✗</td>
<td>✗</td>
<td>✔</td>
</tr>
<tr>
<td>VERTICAL</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>HORIZONTAL</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>DC</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>DIAGONAL DOWN LEFT</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>DIAGONAL DOWN RIGHT</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>VERTICAL RIGHT</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>HORIZONTAL DOWN</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>VERTICAL LEFT</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>HORIZONTAL UP</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
</tr>
</tbody>
</table>

StreamEye v.4 allows you to easily identify what modes of spatial prediction have been used for video stream compression. For this, open the Prediction pane by selecting the View → Info → Prediction command. The “picture” and “stream %” tabs display statistics for using modes of spatial prediction for an individual frame in absolute number and for the whole video stream in percentage ratio. This information helps assess usage of modes of spatial prediction. For example, Fig. 13 displays information for a demo video file. It’s evident from the figures that intra prediction has not been used for 4x4 blocks. Besides, the VERTICAL RIGHT, HORIZONTAL DOWN, VERTICAL LEFT and HORIZONTAL UP modes have not been used for spatial prediction of 8x8 blocks. Thus, usage of the mentioned modes of spatial prediction can...
increase compression effectiveness.

![Fig.13: Modes of spatial prediction](image)

**Reduction of Interframe Redundancy (Motion Estimation)**

Interframe prediction allows removing superfluous frame data more efficiently if using data from other frames. As a result, efficiency of interframe prediction greatly impacts compression efficiency. One of the key factors affecting efficiency of the prediction is for example, a size of motion search area where the encoder selects frame region that looks similar.

For example, let's position №26 frame that follows the second I-frame in the demo video. Then open the pane of statistics by executing the **View → Info → Statistics** command. Select “picture”, “pred type inter” and “by area” from configuration settings as shown in Fig.14. The window also provides data about maximum values of motion vectors in a video sequence. The values are given with quarter pixel accuracy i.e. the difference of conditional unit \(1\) corresponds to the difference of a quarter pixel. In this case, maximal horizontal and vertical values and are 32, i.e. the maximum motion reduction occurs from a distance of 8 pixels. The conclusion is that when compressing data, the encoder looked for prediction in the area of +/- 8 pixels from the block position, and the value is quite low.
You can see some room for compression improvement after assessment of prediction success. Let’s see the difference between two frames: the reference I-frame and the current P-frame. Position to I-frame №25 and add its decoded view into the program memory. For this, select the To Memory value from the drop-down list on the toolbar. Select the Decoded value from the drop-down list to add a decoded view of the active frame into the program memory (Fig. 15).

Position back to P-frame №26. Select the Difference tool on the toolbar. Select the Temperature value as a metric. For comparing frames, select the decoded view of active P-frame and a frame saved earlier in the program memory (Fig. 16).

Select the Difference tab containing a temperature chart showing the difference between the current and reference frames (Fig. 17). There is a black color over the major portion of the
frame, which means very little difference between these areas. The area can be predicted by areas in the same position of the previous frame i.e. there is no need to estimate motion.

Frame regions colored blue and brighter red and green, contain significant differences, which have to be encoded into the stream without effective interframe prediction and increase the bitrate of the encoded video.

![Image: Temperature difference between reference and current decoded frames]

*Fig. 17: Temperature difference between reference and current decoded frames*

The decoded view of the area with a biker (Fig. 18 ‘a’, decoded tab) and its prediction (Fig. 18 ‘b’, predicted tab) shows that the biker motion has been predicted inefficiently because of a small area of motion search. Fig.19 ‘a’ displays the temperature difference between the predicted and encoded pictures for frame #26. This difference will be encoded in the video stream to restore the frame.

Here is another example, now of efficient motion estimation and encoding. The difference between decoded view of B-frame №27 and its prediction from I-frame №25 (Fig. 19 ‘b’) shows that there is much less data remaining. The B-frame is located closer to the I-frame in the decoding sequence, the biker displacement between these two frames is located within 8 pixels, and this fact resulted in successful redundancy removal on the B-frame. Thus, expansion of the area for movement catch will increase efficiency of demo video file compression.
Information about utilized reference frames is also of great interest, and this data is provided by statistics about usage of inter prediction throughout the stream. To view statistics, execute the View→Info→Prediction command and go to the “stream %” tab (Fig. 20). The table displays usage frequency of reference frames for blocks of different size. H.264/AVC standard describes the methods of interframe prediction by blocks of 16×16, 16×8, 8×16, 8×8, 4×8, 8×4, 4×4 size. In StreamEye v.4 statistics, nonsquare sizes are rounded to upward. For example, blocks of 16×8, 8×16 will be included into statistics of 16×16 blocks of, etc. As you can see from the table, only 16×16 blocks are used in the demo video. Thus, the encoder has not used all available tools for interframe block prediction.
Another aspect of interframe prediction is a number of reference frames, regions of which can be used for prediction of the current frame block. Depending on the motion intensity, increasing in number of reference frames can result in more efficient compression.

The number of reference frames is limited by the syntax element max_num_ref_frames in SPS. To view a value of the element, use the headers pane (View→Info→Headers) by opening seq_parameter_set as it is shown in the figure 21. The 2 value is used in the demo video. This means that no more than two frames can be used for interframe prediction. These reference frames must be buffered by the decoder into so-called Decoded Picture Buffer. To view a content of the buffer at any time defined by the decoded frame, use the DPB tool (View→Info→DPB Info). As you can see in figure 22, the buffer contains two frames, which are displayed in the left column. The other two right columns RefList0 and RefList1 show frames included into reference lists 0 and 1 respectively. In the demo video, the motion is smooth; therefore, increase in the number of reference frames will impact compression efficiency less than expansion of the search area.
Subpixel prediction is another important parameter of the encoder. H.264/HEVC standard allows for interframe prediction with quarter pixel accuracy. During such prediction, intensity values are interpolated between entire countings of pixels. It considerably increases compression efficiency. Prediction of this type can be very efficient when compressing video with smooth object motion. In some cases, bitrate gain can be 50-60%. As efficiency of interframe prediction impacts the level of distortions appeared after quantization of difference signal, then it further impacts the effectiveness of prediction of succeeding frames. At the
same time, many encoders (especially hardware ones) do not use subpixel predictions for decreasing processing load.

The block info tool allows defining if subpixel prediction has been used or not. As it is shown in figure 23, you can view values of motion vector by placing a mouse cursor over separate blocks. If quarter pixel prediction is used, the motion vector value will be odd, for example 1, 3, 7, 31, etc. If only semipixel prediction is used, vector values are always times 2. For example, 4, 6, 8, etc. If integral prediction is used, vector values are always times 4. For example, 0, 4, 16, etc. Thus, if you cannot find a motion vector which is not multiple of 4, subpixel prediction has not been used for video encoding.

![block info tool](image)

**Fig. 23: block info tool**

**Adaptive Quantization**

Adaptive block quantization allows increasing efficiency of compression by skipping less important information on some frame areas. Proper use of adaptive quantization helps increase quality of video sequence compression.

The bar chart tool allows defining if adaptive quantization has been used or not. To define this, just place the mouse cursor over a frame. The appeared hint displays an average value of quantizer for a frame. If the value is integral, all blocks of the frame are likely encoded
with equal quantizer (see Fig. 24 below).

![Fig. 24: Average value of quantization for a selected frame](image)

On a separate frame, you can see what values of the quantization parameter were used during data compression. For this, select the Extended tool arranged on the toolbar. Then select Quant and Quant text checkboxes (see Fig. 25 below).

![Fig. 25: Extended toolbar](image)

An additional layer with information about used quantization values will be displayed as a result of frame display (Fig. 26). When adaptive quantization is used, values for blocks will be different. Note that blocks with no quantization parameter specified do not contain a remaining signal, which is to be encoded; therefore, they are encoded only with information about
required prediction variant.

Fig. 26: Decoded frame display with additional layer of information about block quantization parameters