

## H.264 STREAM REPLACEMENT WATERMARKING WITH CABAC ENCODING

Dekun Zou\* and Jeffrey A Bloom\*\*

\* Technicolor Corporate Research  
[dekun.zou@technicolor.com](mailto:dekun.zou@technicolor.com)

\*\* Dialogic Media Labs  
[bloom@ieee.org](mailto:bloom@ieee.org)

### ABSTRACT

This paper describes a watermarking method to directly embed information into a CABAC entropy coded H.264/AVC stream. In theory, modification of a part of an arithmetically coded stream will cause the rest of the stream to be misinterpreted and thus become undecodable. However, the method presented in this paper exploits a feature in the fixed-point integer implementation of the arithmetic coding to allow modifications without introducing such errors. Most of the hard work in this method goes into looking for watermarkable locations and building an embedding table during an analysis stage. This table identifies small segments of data in the encoded stream that can be replaced and identifies one or more replacement values. The embedding process is then very fast. It involves replacing each identified segment with one of the alternative values from the table. The choice of alternative is informed by the payload to be embedded. The resulting bitstream is guaranteed to be compliant with the H.264/AVC standard.

**Keywords**— digital watermarking, data hiding, video watermarking, CABAC, H.264/AVC

### 1. INTRODUCTION

There have been many papers published in recent years discussing methods for watermarking motion imagery in the pixel domain. A general summary of such *baseband* digital watermarking techniques can be found in the book *Digital Watermarking* [1]. For almost as long as researchers have examined baseband watermarking, they have also considered compressed domain watermarking techniques. These techniques are useful for applications in which the original unmarked works are already compressed and the watermarking process is time and resource constrained such that reconstruction, watermarking, and recompression (presumably using all of the motion estimates, quantization scale factors, and mode decisions from the original compressed file) would be too expensive.

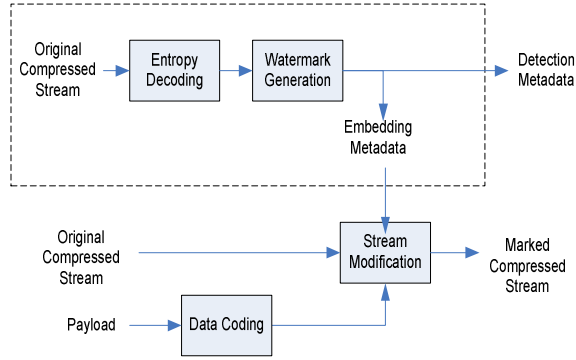
Most published works describing a compressed-domain watermarking process do not, in fact, operate on the compressed bitstream. Instead, they apply entropy decoding and parsing of the bitstream to recover the syntax elements (coefficients, motion vectors, modes, etc.). The watermark

is typically applied in this syntax domain and the result entropy coded. These methods are often called *partial decompression*. Examples of approaches that target MPEG [2], MPEG-2 [3], and H.264/AVC [4] have been developed.

A new class of applications requires the watermarking to be applied directly to an entropy encoded bitstream without the prior partial decompression and subsequent entropy re-encode steps. In this case, the bitstream will be modified directly where necessary to embed the watermark. This modification should only happen at limited locations. In other words, most bytes of the stream are passed unchanged, but some of the bytes are replaced with alternative values. The challenge is to determine where and how the bitstream can be changed so that the modified bitstream is still a valid, compliant compressed bitstream. The resulting video should meet two basic requirements: compliance and fidelity. To be more specific, the modified bitstream must be both CABAC compliant and H.264/AVC compliant. When decoded, it should be perceptually identical to the original video. In order for the watermark to be effective, the modification (or information represented by the modification) must be recovered from the decoded, baseband pixel data. In some applications, robustness is an additional requirement.

The computational burden lies in the entropy decoding and watermark generation steps which might include a complex perceptual analysis to control the tradeoff between robustness and fidelity. In many applications, including those that motivate our work, the original compressed stream can be preprocessed off-line. Thus, we present the entropy decoding and watermark generation processes as an analysis step. The output of the analysis step is the metadata needed for stream modification watermarking. This framework is illustrated in Fig. 1 with the dashed box representing the analysis step.

The analysis can be computationally complex. It can involve multiple passes and, if necessary, may even involve human intervention. The output of the analysis, the embedding metadata, indicates the location of changeable bytes in the bitstream and the values to which these bytes can be changed. For each position listed in the embedding metadata, a number of different alternative values may be specified. At embed time, the coded payload will determine which of the multiple available values should be used for replacement. Note that in many applications, including



**Fig 1.** Real-time stream replacement watermarking

those of interest to us, the alternative values must occupy the same number of bits in the entropy coded stream as the original values being replaced.

The work described in this paper is motivated by an application of watermarking a motion image sequence represented by the H.264/AVC (AVC) standard [5]. AVC is supported by Blu-ray, the high definition DVD standard, and it is expected to become more widely used in the future. An earlier paper [6] described a method for modifying the H.264/AVC stream entropy coded with CAVLC. The method presented in this paper, follows the same basic framework, but addresses the much more difficult challenge of watermarking a CABAC encoded H.264/AVC stream.

It is expected that the watermarking method presented in this paper will have wide applicability. As H.264/AVC becomes more widely deployed and the use of watermarking becomes more accepted in the market, the method presented in this paper will find many uses.

## 2. STREAM REPLACEMENT WATERMARKING

Before presenting the contribution of this paper, it is helpful to review the framework in which this method is applied. That framework was presented in [6] where it was applied to H.264/AVC streams entropy coded with CAVLC.

Prior to watermarking, the content is analyzed. The stream is first entropy decoded, or parsed, to reveal the syntax elements. During this parsing process, the link between each syntax element and its location in the encoded bitstream is recorded. Then, for each occurrence of the target syntax element, a list of current value/alternative value pairs is built. In the CAVLC case, the target syntax element was the intra-prediction mode represented by the macro-block type. The alternative values are values that meet the compliance constraints. Replacement of a target syntax element value by one of the alternative values will result in a compliant stream. For each value, original and alternatives, a detection measure is calculated. In addition, the bit string that would result from entropy coding each alternative is calculated.

The list of possible original/alternative pairs is then pruned by a number of different filters. The first filter

identifies those target syntax elements for which there are multiple compliant alternatives. Entries in the list that do not meet this requirement are removed from the list.

A fidelity filter removes all alternatives that would result in an unacceptable perceptual distortion, thus guaranteeing a minimum fidelity. A robustness filter removes all alternatives that would be undetectable after a target distortion, thus guaranteeing a minimum robustness. The goal of the robustness filter is to select pairs for which the detection measures resulting from the use of one or the other alternative value are significantly different. Of the remaining pairs, a random selection can be made to provide an additional level of security.

Two types of metadata are then created. In the embedding metadata a list of all changes is created. Each entry represents one change. For each change, the location in the bitstream is recorded along with the entropy-coded bit string representing one alternative value and the entropy-coded bit string representing the second alternative value. Use of one of these alternatives will increase the detection measure and the other will decrease it. The embedder uses this list, along with the watermark payload, to replace the original bit-strings in the compressed original file with one of the two alternatives. (For example, use the first alternative if the corresponding watermark bit is a 0 and use the second alternative if the watermark bit is a 1.)

The second type of metadata created is the detection metadata. Again each entry in the list represents one change. For each change, the location of the change in the image sequence is recorded (frame number, block number) along with a detection measure that is half way between that which will result from use of one alternative value and that which will result from use of the other. With this information, the detector can calculate the detection measure at the specified block and determine which alternative was selected in the embedder.

The embedding metadata is made available to the embedder and the detection metadata is made available to the detector. Note that the detection metadata is not embedded as a watermark. It is sent via some external channel to the detector.

For a given work, the analysis creates one embedding metadata file and one detection metadata file. The embedder (or embedders) can create many copies of the video, each with a different payload. The detector (or detectors) only needs the single detection metadata file for that original work to extract the payload from any watermarked version of that work.

The difficulty in applying this method to a CABAC encoded H.264/AVC stream comes in the identification of alternative values that meet the compliance constraints. As will be discussed in the next section, theoretically there are no arithmetically-coded syntax elements that can be changed to alternative values while maintaining CABAC compliance. In theory, this means that this stream replacement method cannot be applied to CABAC-encoded

bitstreams. The main contribution of this paper is a solution to this apparent restriction.

### 3. CABAC STREAM REPLACEMENT WATERMARKING

Context-Adaptive Binary Arithmetic Coding (CABAC) is the entropy coding method widely used in the AVC standard. Watermarking a CABAC stream involves changing a coded syntax element in the CABAC stream. This is a hard problem because the value of each element in a CABAC stream affects the interpretation of the values that follow. In general, changing a coded syntax element will cause all future coded elements to be misinterpreted by the decoder. This section describes a method to exploit an implementation feature of CABAC encoding to allow changes of syntax elements, thus enabling watermarking.

#### 3.1. CABAC Background

The final step in many video compression methods is the lossless entropy coding of the compressed data. In AVC, an arithmetic coding system, called Context-based Adaptive Binary Arithmetic Coding or CABAC, can be used for entropy coding. CABAC is an arithmetic coding scheme that achieves improved compression performance by maintaining individual contexts for each syntax element type and by adapting these contexts with each coded element. This represents a number of extensions of typical arithmetic coding schemes. First, CABAC is designed to arithmetically encode binary data. Syntax element values that are not inherently binary, are first binarized prior to arithmetic coding. This means that there is only the probability of a 0 and the probability of a 1. The specific method of binarization is intended to optimize the effectiveness of the subsequent arithmetic coding. Each binarized syntax element is called a bin string. A second extension is that the coding is adaptive. As each bit of bin string is processed, the decoding variables that control the coding are allowed to change. Finally, this adaptation is allowed to be context-based. Associated with each bin of each syntax element is a context index which points to some *Coding Variables*. Coding of one syntax element causes the associated contexts

to adapt, but does not affect other contexts. AVC defines over 300 separate contexts that are maintained during encoding and decoding.

The term “Coding Variables” used above refers to two sets of variables. The first set is called *Context Variables* which contains two variables that represent the probability of a 0 or a 1 (represented by the identification of which of these symbols is least probable, *LPS*, and an index, *I*, into a table of probabilities indicating the probability of this *LPS* appearing next in the bitstream). Each context is associated with a context index which maintains a set of context variables. The second set is called *State Variables* which constitute two variables to define a range (represented by

the bottom of the range, *L*, and the width of the range, *R*). The *L* and *R* variables are shared among all syntax elements while separate *LPS* and *I* values are maintained for each context.

In general, there are no two bin strings that result in the same changes to the state variables (specifically the *L* and *R*) and this would imply that it is not possible to modify an arithmetically-encoded bitstream without causing the decoder to misinterpret all subsequent coded elements. However, CABAC requires that *L* and *R* be represented with fixed length binary numbers. This is accomplished through a process known as renormalization. Specific details of the renormalization of the *L* and *R*, can be found in the H.264 standard [5].

One result of this renormalization process is that it becomes possible for a number of different bin strings to induce the same changes to the *L* and *R*. The current invention exploits this fact to identify syntax element values that can be replaced with different values while maintaining the original *L* and *R* values in the CABAC decoder.

In the special case of syntax elements for which the two symbols have equal probability, the *LPS* and state index would be updated constantly without introducing any gain in terms of coding efficiency. The probability represented by the index would be essentially 50% meaning that the range, *R*, would experience the same predictable change from one bit to the next.

In order to speed up the encoding/decoding process without compromising coding efficiency, syntax elements for which the symbols are expected to have equal probability are identified and treated in a special way called *bypass mode*. Here, no context is used and the probability of both symbols is fixed to be 0.5. The result is that the range, *R*, will be half of the value after each decoded bin regardless of its value. The renormalization process will double the *R* so that the final fixed-length representation of *R* will remain unchanged in bypass mode decoding. Only the bottom of the range, *L*, adapts as new symbols are coded. We exploit this property to embed a watermark.

#### 3.2. Data Embedding in AVC with CABAC Encoding

Our goal is to replace a string of bits in a CABAC encoded AVC stream with a different string of bits such that the modifications affect only a targeted syntax element and do not interfere with the correct interpretation of any other coded syntax elements. We require that the replacement string of bits be the same length as the original string of bits.

Given a CABAC encoded AVC bitstream, the first step in the analysis is to build a mapping from syntax elements to coded bits in the bitstream. In practice, this is done by decoding the bitstream and keeping track of which bitstream bits produced which syntax elements. In the second step, the syntax element to be modified, which we will refer to as

the *Target Element*, is identified and the contiguous block of bits in the bitstream representing that element are identified.

The Target Element has an original value which is binarized to an original bin string. The original bin string is CABAC coded, along with other syntax elements to yield the original Coded Block of bits. In the CABAC coding process, the original bin string modifies state variables and sets of the context variables that are associated with the syntax from their original states to their modified states.

The third step in our analysis is to search for an alternative value for the target element. This alternative value will be binarized to an alternative bin string. The alternative bin string is CABAC coded, along with other syntax elements to yield the alternative Coded Block of bits. In the CABAC coding process, the alternative bin string modifies state variables and sets of the context variables. Those coding variables will be examined. If all of the coding variables resulting from the encoding of the alternative bin string are identical to those resulting from the encoding of the original string, this alternative value for the syntax element will be considered *changeable*. In other words, the change is acceptable if the state variables and the sets of the context variables that are affected are in the same states that they would have been if the change had not been made.

### 3.3. Alternative MVD

As has been pointed out in previous section, any syntax element can be used to carry watermark data as long as it is *changeable*. However, this is extremely hard to achieve since there are hundreds of contexts the encoder maintains. A practical approach is to use syntax element that are encoded with bypass mode in which no context is used. In the H.264 standard, only three types of syntax element use bypass mode. They are the sign of a coefficient, the suffix of a coefficient, and the suffix of a motion vector differential. The most promising one according to our investigation is the motion vector differential. Motion vector differential values are encoded as a prefix and a suffix. We only explain the suffix encoding process here to illustrate our algorithm.

In CABAC, all motion vector differentials (mvds) with magnitude of 9 or higher will require an mvd suffix to be coded. This coding is performed in bypass mode (Refer to Table 9-25 of H.264 Specification [5]). The binarization of the suffix is summarized in Table 1, where xxx represents the standard binary representation which can be from 000 to 111.

For each suffix value, we perform a search for other suffix values that can be substituted without interfering with the CABAC or AVC syntax. The goal of this part of the watermarking process is to generate a list of all possible changes that remain CABAC/AVC compliant. Later this list can be filtered by other criteria.

Table 1. Binarization of the mvd suffix

abs(mvd) range	suffix range	Binarization
9 – 16	0 – 7	0 xxx
17 – 32	8 – 23	10 xxxx
33 – 64	24 – 55	110 xxxxx
65 – 128	56 – 119	1110 xxxxxx

We will use an example to explain the idea. From Table 1, if an mvd has a magnitude in the range from 17-32, the suffix has 6 symbols. But the first two symbols are fixed to be 1 0 for the entire set in the range. Only the last 4 symbols can differ. The fixed part of the suffix is defined as the fore-suffix and the variable part is defined as the rear-suffix. For the sixteen mvds magnitudes in the range 17-32, the rear-suffix is 4 symbols long. We try to encode every value in the 17-32 range and observe the coding variables. As has been pointed out, the context variable will be the same since there is no context used. The state variables  $R$  will remain fixed as well. The only variable we need to watch is  $L$ . If encoding of an mvd other than the original mvd results in the same  $L$ , this value will be accepted as a valid alternative. Use of this mvd instead of the original will result in a CABAC/AVC compliant stream. It will be saved to a list containing all the valid alternative values.

Once we have the list, we evaluate each alternative value in terms of fidelity and robustness in the same way as in [6]. The changes that satisfy all three criteria will be saved as embedding metadata. Finally, we consider the effect of a potential change on a detection feature (see next section). We identify syntax values for which there are two alternatives that yield significantly different detection feature values. These will be used to embed a ‘0’ bit or a ‘1’ bit into the video stream.

In the meantime, information about the changes is also saved as *detection metadata*. This data includes the frame and block position of the change along with a mid-point value of the detection feature. After this analysis, the watermark embedding process is just a stream replacement process as shown in Fig.1.

### 3.4. Watermark Extraction

The watermark extraction is *informed* by the detection metadata generated during the embedding stage. For each entry, the corresponding block of pixel data is extracted and from that, a detection feature is calculated. For example, consider using the mean luminance of a block as the detection feature. The calculated feature is compared to the value in the detection metadata file. The sign of the difference is calculated and the corresponding symbol is output as a symbol stream. This is very similar to the method used in [6]. Please refer to [6] for further details.

#### 4. SIMULATIONS

To validate the approach, we built an analysis tool starting from a professional H.264 encoder. This tool is used to obtain the complete list of alternative mvds. Only mvds of  $16 \times 16$  macro-blocks are selected which means we only watermark macro-blocks using  $16 \times 16$  inter-prediction mode. The fidelity for a block is evaluated by measuring the total absolute luminance change of the block when the alternative mvd is selected. The robustness is evaluated by the luminance change of the whole macro-block. A five minute clip from the movie "Independence Day" is used for testing. This source is  $1920 \times 1080$  pixels per frame and the frame rate is 24p. The compression bit-rate is 15 Mbps.

In our testing, the basic unit of the payload is a HEX symbol. We apply a simple spread spectrum channel coding. Sixteen 300-bit long, orthogonal, binary sequences are predefined. Each sequence is assigned to a HEX symbol. To embed a symbol we use its associated 300-bit long binary sequence and each bit is embedded into a  $16 \times 16$  intra-predicted block. In the testing system, 38 HEX symbols are embedded representing a total of (38 symbols @ 4 bits per symbol) 152 bits of information in a five minute clip. This required 11,400 changes to the entropy-coded bitstream (300 changes per symbol) or approximately 38 changes per second.

We use a correlation-based detector. The extracted bit sequence is divided into chunks of 300 bits. Each chunk is correlated against each of the predefined 300-bit patterns. The pattern that gives the biggest correlation is determined to be the detected HEX symbol.

Robustness to shifting, downsizing, and compression are tested. The shifting test simulates registration errors. The downsizing included a crop rather than a change in aspect ratio and the smallest size ( $352 \times 288$ ) was then compressed using DIVX down to 300kbps.

In all resize and compression tests, all symbols were correctly recovered. Although it is expected that further downsizing and/or compression will eventually introduce errors, the requirements of our application do not require further robustness. However, we did find that shifting started to introduce errors at about 3 pixels vertically or horizontally when combined with 2 or 3 pixel shifts in the other dimension. This is because the detector relies on the accurate location of the macroblock to extract pixel luminance information and therefore, the embedded information. In some application, the original picture is available to be used for registration. With accurate geometric registration, this problem can be addressed.

#### 5. CONCLUSION AND APPLICATIONS

In this paper, we have described a new stream replacement method for video watermarking on an H.264 CABAC encoded bitstream. This method is suitable in applications in which entropy-coded, compressed video data must be modified on the fly without entropy decoding, but when pre-

analysis is possible. A real-time, data byte replacement method is discussed which uses embedding metadata to guide the payload embedding. The experimental results showed that this method can survive compression and downsizing distortions. It is suitable to be used for tracing the piracy originating from online content distribution channels as those described in [7].

The method is sensitive to synchronization errors in time and space. For many applications, these can be addressed by registering the watermarked content to the original (or individual key frames from the original).

We did not present results of fidelity testing as the fidelity level is pre-established during the analysis process. Higher fidelity can be achieved by incorporating a more sophisticated, full-reference fidelity model during analysis. The cost of increased fidelity will be a decrease in the number of available changes which, in turn, will be seen as a reduction in payload capacity or a reduction in robustness depending on the channel coding.

#### 6. REFERENCES

- [1] Cox, M. Miller, and J. Bloom: Digital Watermarking: Principles & Practice, San Mateo, CA: Morgan Kaufman, 2001.
- [2] D. Simitopoulos, S.A. Tsafaris, N.V. Boulgouris, M.G. Strintzis: Fast MPEG watermarking for copyright protection, 9th International Conference on Electronics, Circuits and Systems, 15-18 Sept. 2002 Page(s):1027 - 1030 vol.3.
- [3] T. Chung, M. Hong, Y. Oh, D. Shin, S. Park: Digital watermarking for copyright protection of MPEG2 compressed video, IEEE Transactions on Consumer Electronics, Volume 44, Issue 3, Aug. 1998 Page(s):895 - 901
- [4] M. Noorkami, R.M. Mersereau,: Compressed-domain video watermarking for H.264, IEEE ICIP 2005. 11-14 Sept. 2005, Volume: 2, On page(s): II- 890-3.
- [5] ITU-T Recommendation H.264 | ISO/IEC 14496-10 International Standard with Amendment 1.
- [6] D. Zou and J. A. Bloom; "H.264/AVC Substitution Watermarking: A CAVLC Example"; Media Forensics and Security XI; E. Delp, J. Dittmann, N. Memon, and P. Wong, Editors; Proceedings of SPIE Vol. 7254, 2009.
- [7] D. Zou, N. Prigent, J. Bloom, "Compressed Video Stream Watermarking for Peer-To-Peer Based Content Distribution Network" 2009 IEEE International Conference on Multimedia and Expo (ICME09), New York, June 28 - July 3, 2009