

H.264/AVC Substitution Watermarking: A CAVLC Example

Dekun Zou¹ and Jeffrey A Bloom

Thomson Corporate Research, 2 Independence Way, Princeton, NJ 08540, USA

ABSTRACT

This work addresses the watermarking of an entropy coded H.264/AVC video stream. The phrase *Substitution Watermarking* is used to imply that the application of the watermark to the stream is accomplished by substituting an original block of bits in the entropy-encoded stream with an alternative block of bits. This substitution is done for many different blocks of bits to embed the watermark. This can be a particularly powerful technique for applications in which the embedder must be very simple (substitution is a very light operation) and a computationally complex, pre-embedding analysis is practical. The pre-embedding analysis can generate a substitution table and the embedder can simply select entries from the table based on the payload. This paper presents the framework along with an example for H.264/AVC streams that use CAVLC for entropy coding. A separate paper addresses the CABAC entropy coding case.

Keywords: digital watermarking, data hiding, video watermarking, CAVLC, H.264/AVC

1. INTRODUCTION

There are a number of video watermarking applications that have the following three properties: the watermark must be applied directly to an H.264/AVC stream by a very lightweight embedder; the stream to be marked can be pre-analyzed to generate some *embedding metadata* which is provided to the embedder along with the stream; and data recovery can be an informed process in which information from the original stream or from the pre-analysis can be provided to the detector. The first property is illustrated in Figure 1 below.

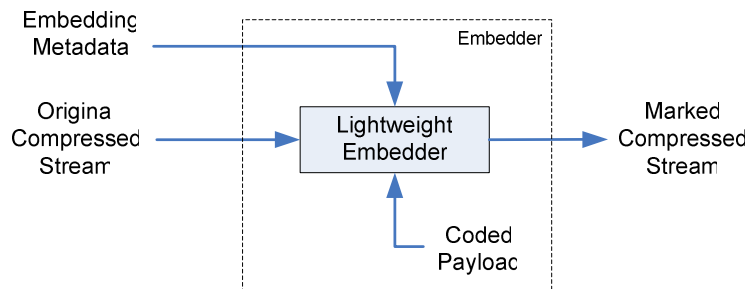


Figure 1. Lightweight Embedding Requirement

In this figure, the embedder has two inputs, the H.264/AVC video stream and some embedding metadata. The embedder then uses the embedding metadata to modify the compressed stream in order to represent the coded payload. A common approach to watermarking “compressed streams” is to perform a partial decompression, more specifically an entropy decode, to expose the syntax elements (coefficients, motion vectors, modes, etc.), perform the watermarking in the syntax domain, then apply an entropy encode to obtain the marked compressed stream (e.g., [3] for MPEG, [4] for MPEG2, and [5] for H.264/AVC). In this paper, we are interested in applications for which such partial decompression is too computationally expensive. Instead, the entropy-encoded stream must be directly manipulated in the embedding process.

A watermarking approach that is consistent with this lightweight embedding property is *substitution watermarking*. This phrase simply means that the watermark is applied by substituting an original block of bits in the entropy-encoded

* A shorter version of this paper was presented at ICASSP 2008 in Las Vegas, Nevada, USA [1].

¹ Dekun.Zou@thomson.net

stream with an alternative block of bits. This substitution is done for many different blocks of bits to embed the watermark. In this case, the embedding metadata is list of locations or offsets into the compressed stream and, for each, one or more alternative blocks of bits. For example, the list may have two alternatives for each offset; the first should be used to represent a '0' bit and the second used to represent a '1' bit. Then the embedder uses the coded payload to select the right combination of alternatives from the list and makes the substitution in the stream.

This approach clearly moves the difficult part of watermarking (insuring that the modified stream is compliant with the entropy coding and the compression standard, insuring that the decoded sequence has high enough fidelity compared to a decoded version of the unmarked stream, and insuring that the changes made allow the payload to be recovered from the decoded, and perhaps degraded or attacked sequence) to the generation of the embedding metadata. Thus, it is best used in applications that have the second property: the opportunity for the stream to be pre-analyzed to generate the embedding metadata. Typically, this pre-analysis is an off-line process and can be computationally more complex than the embedding. For example, the pre-analysis can typically afford to do an entropy decode to access the syntax values. In fact, for many applications, it could even do a full decompression; opening up the opportunity for a traditional, pixel-based fidelity analysis.

For applications that have the third property, a set of *detection metadata* can be made available to the detector. Each original stream will have a different set of detection metadata, but each watermarked copy of a single stream will use the same detection metadata. The need for this metadata is a result of the fact that the pre-analysis process generates a unique set of change locations for each original stream. A blind detector would not know where to look to find the changes. The concept here is that the pre-analysis process will generate the set of detection metadata as it generates the embedding metadata. Note that these two sets of data may be quite different. The embedding is performed on a compressed stream. Thus, the embedding metadata describes changes to be made to that stream. The detection, on the other hand, is typically performed in the baseband. Thus, the detection metadata must describe changes in terms of frames, blocks, and pixels.

The work described in this paper was motivated by an application that has these three properties. In this application, the sequence is compressed according to the H.264/AVC standard prior to the pre-analysis. The pre-analysis must be applied to the same compressed stream as that sent to the embedder. An additional constraint is that the size of each substitution block is very small; on the order of 10's of bits. The application supports H.264/AVC streams entropy coded with CAVLC or CABAC. This paper describes the framework and a CAVLC solution. The CABAC solution is presented in a separate paper.

2. REAL-TIME SUBSTITUTION WATERMARKING FRAMEWORK

As is depicted in Figure 2, the stream substitution watermark applies a watermark by directly changing the compressed video stream. The watermark embedding is as simple as replacing some bytes of data with alternative bytes of data. The computational burden lies in the entropy decoding and watermark generation, which might include a complex fidelity analysis to enhance the watermarking strength while suppressing the likelihood of visible artifacts. In many applications, we strip out the entropy decoding and watermark generation processes and put them in a pre-analysis step. The output of the pre-analysis step is the metadata needed for stream substitution watermark embedding. This framework is depicted in Figure 2.

If the application permits, the pre-analysis can be computationally complex. It can involve multiple passes and in some cases even human intervention. The output of the pre-analysis, the watermarking metadata, indicates the locations of changeable chunks of data in the bitstream and alternative values that can be used as substitutes. For each position, multiple alternative values can be used to replace the original data. The payload will determine which, of the multiple available alternative values, should be used for substitution. The challenge is in generating a set of embedding metadata that will maintain the compliance of the stream (compliance to both the entropy coding and the compression standards), is able to embed the payload with required robustness, and maintains the required level of fidelity. Once the watermarking metadata is prepared, the watermark embedding can be extremely fast.

This framework of real-time substitution watermarking can be applied to many distribution channels where the pre-analysis can be done at the server, stored with the content, and (securely) provided to all (secure) clients along with the

content.¹ The client obtains a local payload (e.g., the unique ID of the player) and uses this to select alternative values from the embedding metadata. These selected changes are then applied to the video stream via substitution. Some examples of applications where this might be appropriate include broadcast, multicast, and peer-to-peer distribution channels. Alternatively, if the content is being served to a single, known client, as in VOD or other unicast distribution, the substitution can be done at the server.

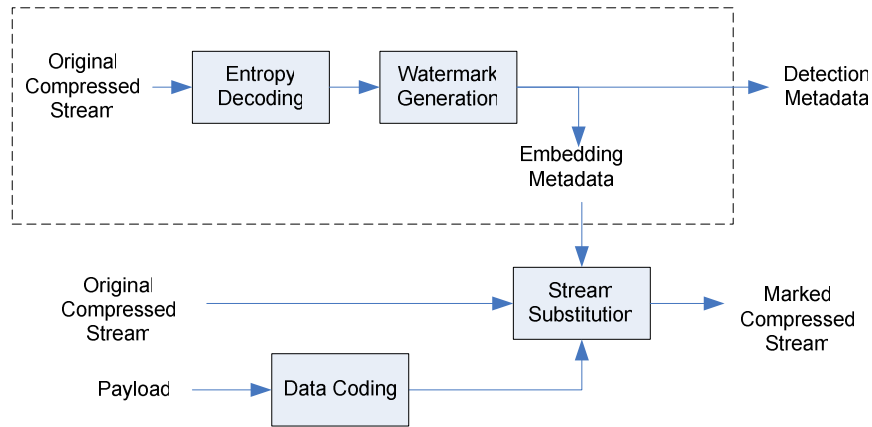


Figure 2. Real-time stream substitution watermarking

Note that the embedding metadata need not be distributed through the same channel as the content. In some applications, the metadata server has the payload information and can generate the metadata for a particular user by sending only one replacement value for each location; the one value that represents the corresponding coded bit. In this case, the metadata received by the client will already carry the payload and contain the information about this user.

This approach can be used with peer-to-peer distribution. A client requests the embedding metadata from a service provider and receives a customized metadata file. The client then joins the peer-to-peer downloading network to download this movie. Here we assume that the movie file received by the client is *damaged* in that the data in the locations to be watermarked has been removed and replaced with erroneous data. If done carefully, this will result in a compressed stream that is non-compliant and cannot be decoded. More careful design can insure that it is very difficult for an adversary to identify or fix these locations. Thus, the file distributed on the peer-to-peer network is not valuable without the embedding metadata. These ideas are represented in a 2003 white paper published by Cryptographic Research, Inc.[6].

A special player will replace the data bytes in the compressed movie stream with the data bytes in the metadata file. Note that the metadata is much smaller in size than the compressed movie stream. In our test system, the metadata can be on the order of several hundred kilobytes for a 2-hour HD movie while the compressed video is usually on the order of several hundred megabytes or even gigabytes. Therefore, it is still advantageous to distribute the compressed video via a peer-to-peer network and rely on the point-to-point client server architecture to deliver the metadata.

As an alternative to *damaging* the distributed video stream as described above, the original data bytes that are going to be replaced can be removed from the stream before transmission. Then the substitution will instead be an insertion. Again, the received video stream before insertion should not be a valid compressed video stream and should be very difficult to manually repair without the metadata.

¹ For the purposes of this discussion, we assume the existence of a “secure client” that can communicate with a “secure server” to obtain a protected copy of the embedding metadata, can securely obtain and code the local payload data, and can then securely apply the substitution watermark embedding before forwarding the compressed stream to the decoder.

3. INTRA-PREDICTION BASED WATERMARKING IN H.264/AVC

3.1 Intra-Prediction Mode Modification for Watermarking

The H.264/AVC video compression standard, like most video compression standards, achieves compression by predicting the values in a block of pixels from the values of one or more previously coded pixels. The difference between the prediction and the actual values, often called the residue, is then transform coded and quantized. The pixels used for the prediction, the reference, can be from the same frame or from different frames. Blocks whose reference comes from the same frame are called intracoded blocks or simply intra-blocks. In this case, the prediction is often called intra-prediction.

For the luminance samples, an entire 16×16 macroblock can be intra-predicted as a whole or can be divided into 8×8 sub-blocks or even 4×4 sub-blocks. Each sub-block will then be intra-predicted independently. For 16×16 luminance intra-prediction, four modes are defined in the standard. For 8×8 and 4×4 sub-blocks, nine modes are defined for each type of block. The encoder typically selects the prediction mode for each block that minimizes the difference between the predicted block and the actual pixel values.

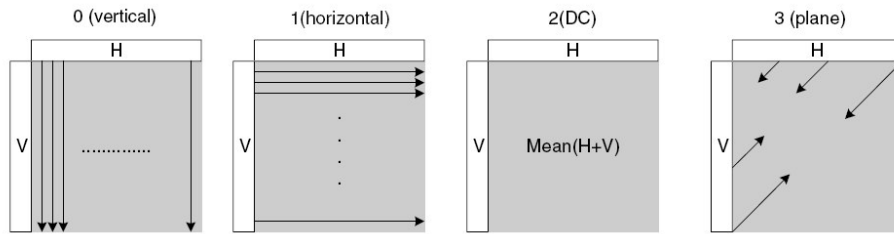


Figure 3. Four modes of 16x16 Intra-prediction. [8]

To simplify the discussion, we only present the 16×16 intra-prediction case in this paper. For 16×16 intra-prediction, the following four modes are defined as illustrated in **Error! Reference source not found.**:

Mode 0 (vertical) Extrapolation from upper samples (H).

Mode 1 (horizontal) Extrapolation from left samples (V).

Mode 2 (DC) Mean of upper and left-hand samples (H + V).

Mode 3 (Plane) A linear plane is fitted to the upper and left-hand samples H and V.

To decode a block, the predicted block is generated with the available pixels of previously decoded neighboring blocks as specified by the intra-prediction mode of the current block. Then, the decoded residue pixel values are added to the predicted block. The result is the final decoded pixel block,

$$B = P + R,$$

where B is the final decoded block of pixel values, P is the predicted block, and R is the block of decoded residues.

The basic idea of using intra-prediction for watermarking is to change the intra-prediction mode from one to another while keeping the residue data the same. The result will be a different predicted block. As a result, the final decoded pixel block will be different from what it should have been. The difference between the original block of pixel values and the watermarked block of pixel values, denoted ΔB , will be the same as the difference between the original predicted block of pixels and the watermarked predicted block of pixel values, denoted ΔP .

In our notation, $\Delta B = \Delta P$, where $\Delta B = B_w - B$, $\Delta P = P_w - P$, and the subscript 'w' indicates the watermarked version of each.

By changing the intra-prediction mode of a macroblock, the pixels of that block will change by ΔB . For this change to be appropriate for watermarking, we require the following:

1. Each change should be imperceptible in the reconstructed imagery.
2. Each change should be detectable from the reconstructed imagery.

3. The detection of each change should be robust to some predefined set of signal distortions.

To this end, we evaluate the suitability of each possible change and select only those changes that meet these requirements.

The first requirement, imperceptibility, can be interpreted in many ways. We can require that the reconstructed imagery have high visual quality or that it be indistinguishable from the original, unmarked imagery, or that the perceptibility of the changes fall below some threshold. There are many ways people judge the fidelity of a watermark. For the purposes of this paper, we allow any appropriate fidelity measure to be applied to judge whether or not, or to what extent, a proposed change meets the fidelity requirements of the application.

The second and third requirements, that the change be robustly detectable, requires establishing a good detection feature that can be reliably measured in the decoded imagery and can be modified by changing the intra-prediction mode of a macroblock. One detection feature we have investigated is the mean luminance of the macroblock. Each intra-prediction mode change will result in a change, ΔB , in the decoded pixel values. This ΔB may have a positive or negative average value and the magnitude of the average can vary. The sign of the change can be used to encode data and the magnitude can be used as an indication of the expected robustness of the change.

A second detection feature that we have investigated is the variance of the reconstructed block of pixels. DC mode (mode 2) is very different from other three intra-prediction modes in that all the 256 pixel values are predicted with a single value; the mean of the reference pixels. It is expected that an H.264/AVC encoder will use this mode when a block is smooth in nature. If we change the mode to one of the remaining three modes, the variance of the resultant block can be expected to increase. On the other hand, if an encoder chose modes 0, 1, or 3, it is expected that this block has higher fluctuation. By changing the intra-prediction mode to 2, the variance of this block should decrease. Again, the ΔB associated with a change of intra-prediction mode may result in an increase or decrease in the variance of the reconstructed block and the magnitude of the variance change can vary. The sign of the variance change can be used to encode data and the magnitude of the change can be used as an indication of the expected robustness of the change.

Clearly, these are two very simple detection features and more sophisticated features can be used. To demonstrate the ideas presented in this paper, the simplest detection feature, mean macroblock luminance, is used.

3.2 Changing Intra-prediction Mode Through mb-type

In H.264/AVC, the 16×16 intra-prediction mode of a macro-block is specified in the mb-type field. The mb-type field also specifies other parameters about this block such as the coded-block-pattern. Figure 4 is the list of mb-type values with their meanings. This table, taken directly from the standard [7], is used to find mb-type values that change the intra-prediction mode without changing the coded-block-pattern.

In order to change the intra-prediction mode for a 16×16 luminance macroblock, we change the mb-type, but that change is limited to the values in the table that differ only in intra-prediction mode. For example, an original mb-type of 11 indicates that the intra-prediction mode is 2 (DC) and that the coded-block-pattern for chroma and luma are 2 and 0, respectively. This mb-type can be changed to 9, 10, or 12 to change the intra-prediction mode without changing the coded-block-pattern.

The mb-type is entropy coded in the bitstream. If CAVLC entropy coding is used, the mb-type is encoded with the exp-Golomb code. The exp-Golomb code is a variable length coding scheme. Some special applications, such as the watermarking of authored DVD disks, require that the replacement bytes have exact the same length as the original data. In this case, only mb-types that result in the same number of bits can be used to replace the original mb-type. Figure 5 lists the bit string form and the corresponding value range of the Exp-Golomb code, again taken directly from the standard [7]. From this table, we find that, using the example from above, an mb-type of 9 would require 7 bits. This mb-type can only be replaced with another mb-type also requiring 7 bits. In this case, mb-types 9, 10, and 12 all fall in the same range (7-14) and all require 7 bits. In this way, we can combine the VLC bit lengths of Figure 5 with the macroblock types of Figure 4 to deduced which mb-types can be used to replace the original mb-type by enforcing the following rules:

1. The alternative mb-type should differ only in Intra-prediction mode.
2. The length of the bit string corresponding to the alternative mb-type should be the same as that corresponding to the original mb-type (if constant bit length is required).

Note that not all applications require this second, constant bit length, constraint.

mb_type	Name of mb_type	transform_size_8x8_flag	MbPartPredMode (mb_type, 0)	Intra16x16PredMode	CodedBlockPatternChroma	CodedBlockPatternLuma
0	I_NxN	0	Intra_4x4	na	Equation 7-33	Equation 7-33
0	I_NxN	1	Intra_8x8	na	Equation 7-33	Equation 7-33
1	I_16x16_0_0_0	na	Intra_16x16	0	0	0
2	I_16x16_1_0_0	na	Intra_16x16	1	0	0
3	I_16x16_2_0_0	na	Intra_16x16	2	0	0
4	I_16x16_3_0_0	na	Intra_16x16	3	0	0
5	I_16x16_0_1_0	na	Intra_16x16	0	1	0
6	I_16x16_1_1_0	na	Intra_16x16	1	1	0
7	I_16x16_2_1_0	na	Intra_16x16	2	1	0
8	I_16x16_3_1_0	na	Intra_16x16	3	1	0
9	I_16x16_0_2_0	na	Intra_16x16	0	2	0
10	I_16x16_1_2_0	na	Intra_16x16	1	2	0
11	I_16x16_2_2_0	na	Intra_16x16	2	2	0
12	I_16x16_3_2_0	na	Intra_16x16	3	2	0
13	I_16x16_0_0_1	na	Intra_16x16	0	0	15
14	I_16x16_1_0_1	na	Intra_16x16	1	0	15
15	I_16x16_2_0_1	na	Intra_16x16	2	0	15
16	I_16x16_3_0_1	na	Intra_16x16	3	0	15
17	I_16x16_0_1_1	na	Intra_16x16	0	1	15
18	I_16x16_1_1_1	na	Intra_16x16	1	1	15
19	I_16x16_2_1_1	na	Intra_16x16	2	1	15
20	I_16x16_3_1_1	na	Intra_16x16	3	1	15
21	I_16x16_0_2_1	na	Intra_16x16	0	2	15
22	I_16x16_1_2_1	na	Intra_16x16	1	2	15
23	I_16x16_2_2_1	na	Intra_16x16	2	2	15
24	I_16x16_3_2_1	na	Intra_16x16	3	2	15
25	I_PCM	na	na	na	na	na

Figure 4. Macroblock types for I slices [7].

Bit string form	Range of codeNum
1	0
0 1 x_0	1-2
0 0 1 $x_1 x_0$	3-6
0 0 0 1 $x_2 x_1 x_0$	7-14
0 0 0 0 1 $x_3 x_2 x_1 x_0$	15-30
0 0 0 0 0 1 $x_4 x_3 x_2 x_1 x_0$	31-62
...	...

Figure 5. Ex-Golomb coding table [7].

4. PRE-ANALYSIS AND COST ANALYSIS

In a previous section, we suggested that a potential change must be evaluated with respect to its impact on the fidelity of the reconstructed imagery and the robustness of its detectability. In this section, we describe a method for combining those two effects into a single cost value and the use of that cost value in selecting which changes to apply.

Referring back to Figure 2, the input of the pre-analysis is the H.264/AVC encoded bitstream. The output is the embedding metadata. The first step in processing the stream is to identify each intracoded block, the intra-prediction mode of which can potentially be changed. For each of these, we identify all alternative intra-prediction modes that satisfy the bit-length requirement (if the application requires constant bit length). Assuming a particular detection measure (such as the luminance mean or variance of a block as previously discussed), the next step is to gather the original detection value and all the alternative detection values. An alternative detection value is the value that would result using each of the alternative intra-prediction modes. Figure 6 illustrates this process when mean luminance is used as a detection measure.

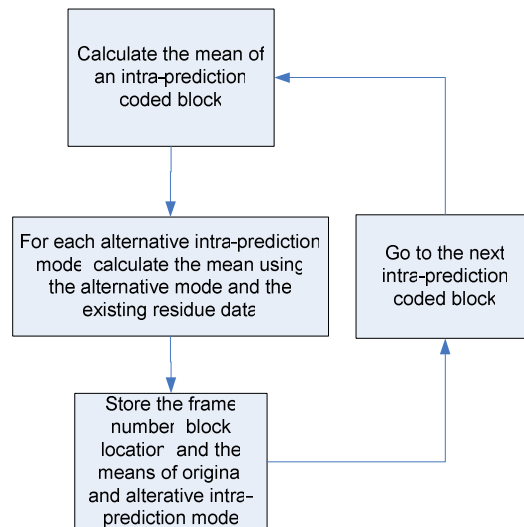


Figure 6. Block information gathering

For many applications, the key properties of the watermarking algorithm are the robustness and the fidelity. Often, these two properties work at counter purposes. We introduce an analytical method to specify the best balance between these two. Let C_F represent the fidelity cost. C_F is defined such that the more visible the watermark, the larger the value. We

would like to make changes that have very low fidelity cost. There are many methods for measuring fidelity [9][10][11], but they are beyond the scope of this paper. Let C_R represents the robustness cost. C_R is defined such that changes resulting in weaker robustness have a higher robustness cost. Again, we seek to make changes that have a lower robustness cost.

For each alternative change, we calculate both the fidelity cost and the robustness cost associated with making the change. We then combine these two costs to obtain a total cost for making the change. That total cost is calculated as follows:

$$C = \alpha C_F + \beta C_R$$

where it is often helpful to restrict $\alpha + \beta = 1$. In this case we can write

$$C = \alpha C_F + (1 - \alpha) C_R$$

The parameter α , restricted to the range of 0 to 1, is used to control the tradeoff between fidelity and robustness. A larger value of α places more emphasis on fidelity while a lower value of α places more emphasis on robustness.

An example can be demonstrated as follows. In a 16×16 intra-predicted macro-block, there are three available alternative intra-prediction modes and the original mode. For this example, we assume that all of the different intra-prediction mode values are valid alternatives in that we can change the mode to any one without changing the coded-block-patterns or the coded bit length (if constant bit length is required). In addition, we will assume that the detection measure is the mean luminance of the block.

To embed a data bit, two modes can be selected: one to represent a '1' and one to represent a '0'. Let's assume a mode resulting in an increase in mean luminance represents bit '0' and a mode resulting in decreased mean represents bit '1'. There are six possible pair combinations: $\{P_1 = (\text{mode0}, \text{mode1}), P_2 = (\text{mode0}, \text{mode2}), P_3 = (\text{mode0}, \text{mode3}), P_4 = (\text{mode1}, \text{mode2}), P_5 = (\text{mode1}, \text{mode3}), P_6 = (\text{mode2}, \text{mode3})\}$. For each combination, the C_F and C_R are evaluated as follows.

Let B denote the original block of pixels and let ΔB_0 and ΔB_1 denote the pixel difference between the watermarked block and the original block if bit '0' or bit '1' is embedded respectively. If the original intra-prediction mode is used to represent a '0' or '1' bit, the corresponding ΔB is simply all zeros. In this case, the fidelity cost is dependent only on the other mode that is selected. Assuming there are K available pairs, the fidelity cost of pair k can be obtained by

$$C_{FPk} = F(B, \Delta B_0, \Delta B_1)$$

where the function $F(\bullet)$, is any fidelity measure. Typically, this fidelity function will evaluate ΔB_0 and ΔB_1 independently and will return the larger of the two values.

Let L_0 and L_1 denote the luminance of the block if bit '0' or bit '1' is embedded, respectively. The robustness cost can be measured as a function of L_0 and L_1 .

$$C_{RPk} = G(L_0, L_1)$$

where the function $G(\bullet)$, is any robustness measure. Again, a typical robustness measure will evaluate the robustness of each change independently and return the larger of the two costs.

Finally, the final watermarking cost of this pair is

$$C_k = \alpha C_{FPk} + (1 - \alpha) C_{RPk}$$

Once the cost associated with each pair has been calculated, the pair with minimum cost can be selected for that block and the corresponding cost can be assigned to C , the cost of changing the block.

$$C = \min(C_k) \text{ where } k = 1 \dots K.$$

Once the watermarking cost of each block has been established, the block selection process can be controlled by a secret embedding key to pick blocks have small watermarking cost.

Referring again back to Figure 2, the embedding metadata must contain location information identifying where, in the stream, each substitution will take place and the values to be used in the case that the payload bit is a '0' or a '1'. Thus, when building the list of potential changes, we need to record the location in the entropy-coded input H.264/AVC stream

where the VLC code describing each mb-type can be found. We also need to record the specific bit string (from the table of Figure 5) that is used to represent each alternative value. Thus, after selection (based on minimizing the total cost), the system can write out the embedding metadata. For each entry in this file, there is location (e.g. a bit offset from the start of the stream) and two bit strings, one for each of the two possible payload symbols.

It was mentioned in the introduction, and will be further discussed in Section 5, that this method includes an informed detector. Synchronization aside, the information that the detector needs is a list of the pixel locations where changes are to be made as well as one or more reference detection values for each. In this case, the list includes the frame and block number of each change, an expected luminance if a ‘0’ bit is embedded, and an expected luminance if a ‘1’ bit is embedded. We refer to this list as the *detection metadata*.

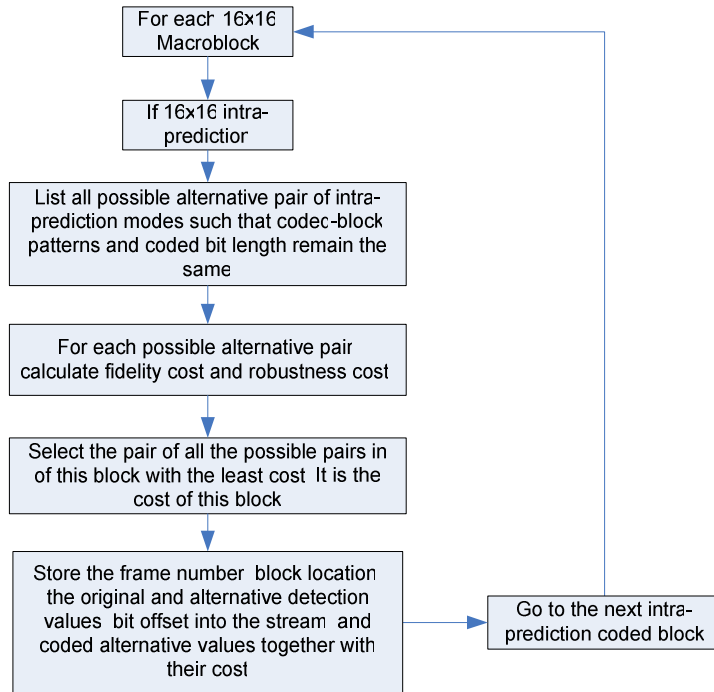


Figure 7. Pre-analysis System Flowchart

Figure 7 shows a block diagram of the bulk of a pre-analysis system. For each 16x16 macroblock that uses 16x16 intra-prediction, we consider all possible alternative intra-prediction modes such that the coded-block patterns and coded bit length remain the same. For each of these, we calculate the fidelity and robustness costs and use those to calculate a total cost. For each block, we select one pair (usually the pair with the minimum total cost). This pair is stored, along with the necessary metadata, and the pair cost is assigned to the block.

From the list of data generated by the system of Figure 7 we select a subset of blocks to change. This selection can be based on a secret embedding key as well as on the payload requirements of the application and is informed by the costs of each change. Usually, the selection will favor lower cost changes.

5. WATERMARK DATA RECOVERY

Figure 8 shows a block diagram of a system for recovering the watermark payload data. In this figure, we assume that the imagery arrives as pixel data (baseband imagery). If it instead arrives as compressed data, we can decode to obtain the baseband imagery. This baseband imagery may require temporal and/or geometric registration. This is necessary because the detection metadata describes each change by its frame number and block position within the frame. There are many approaches for obtaining this registration [12][13][14].

Each entry in the detection metadata file defines a change by its frame position and block within the frame. For each entry, the corresponding block of pixel data is extracted and from that, the detection feature is calculated. The calculated

feature is compared to the two values in the detection metadata file. The best match is selected and its corresponding symbol is output as a symbol stream. If a data coding process was applied in the embedder (Figure 7) then the corresponding data decoding process is applied to the symbol stream yielding the recovered payload, referred to in Figure 8 as the payload estimate.

When mean luminance is used as the detection feature, the system can become confused by global changes in brightness. In other words, a global increase in brightness can mean that the luminance extracted from the baseband imagery will always be closer to the brighter of the two values stored in the detection metadata. Thus, the symbol stream may consist of all '1' bits. To counter this, the embedder can add a number of static reference entries into the detection metadata. A static reference entry lists a frame number, block position, and original luminance value of the block for a block that is not affected by the watermarking process. The detector can read out the reference values and compare these to the corresponding values seen in the baseband imagery. The baseband imagery can then be adjusted (perhaps during registration) such that the measured luminance in the reference blocks matches that listed in the detection metadata file.

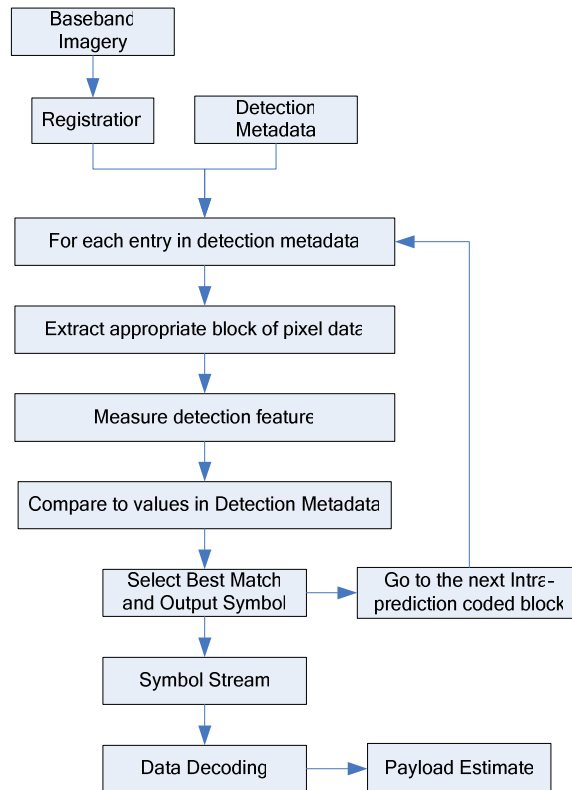


Figure 8. Watermark Extraction

6. PROOF-OF-CONCEPT EVALUATION

To validate the approach, we built a pre-analysis tool starting from the JM H.264/AVC decoder [15]. This tool is used to obtain the complete list of alternative intra-prediction modes of all 16x16 intra-predicted blocks. A simplified cost analysis model is built in which the fidelity for a block is evaluated by measuring the total absolute luminance change of the block when the alternative intra-prediction mode is used. The robustness is evaluated by the luminance change of the whole block under the alternative intra-prediction mode. Instead of using the cost method described in Section 4, we simply set thresholds for both the fidelity metrics and robustness metrics. By controlling the threshold, we can adjust the fidelity and robustness at a very coarse level. A 8640 frame clip from the movie "Independence Day" is used for testing. This source is HD resolution at 1920x1080 pixels per frame and the frame rate is 24p. It is compressed with H.264/AVC using CAVLC entropy coding. The compression bitrate 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 16×16 intra-predicted block. In the testing system, 18 HEX symbols are embedded representing a total of (18 symbols @ 4 bits per symbol) 72 bits of information in (18 symbols @ 300 changes per symbol) 4600 intra-predicted blocks.

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 downsizing and compression are tested. We use ffmpeg [16] for downsizing attacks and DIVX for compression. Table 1 shows some test results. The first column is the attack. The second column is the number of symbol errors out of 18 embedded HEX symbols. The third column lists the average correlation values associated with the selections.

Table 1. Test Result of Robustness on Downsizing and Recompression

Attack	# of Symbol Errors	Average Correlation
No Attack	0	0.9685
Downsize to 960x540	0	0.8133
Downsize to 480x270	0	0.2778
Downsize to CIF (352x288)	0	0.1804
Downsize to CIF & compressed to 1M bps DIVX	1	0.1148
Downsize to CIF & compressed to 780k DIVX	2	0.1181
Downsize to CIF & compressed to 300k DIVX	3	0.1026

From above table, we can see that the proposed watermarking algorithm can withstand downsizing to CIF. For recompression attacks, the symbol error rate increases as the compression bitrate decreases. In the meantime, the average correlation value decreases as well.

7. CONCLUSION AND NEXT STEPS

The substitution method for watermarking a compressed, CAVLC-encoded H.264/AVC stream works reasonably well. It achieves the goal of very fast and computationally inexpensive embedding using a small amount of embedding metadata. The fidelity, although not presented in the previous section, is as good as the fidelity model used in assigning a fidelity cost to each potential change. In our experiments, the changes were imperceptible to trained expert viewers when presented at real-time. Some changes could be noticed when a single frame was examined and the location on the frame was identified. Some material will be more challenging and it is expected that a more sophisticated fidelity model than the one presented will be needed. The robustness results presented are not horrible and may be appropriate for many applications. Again, it is expected that a stronger robustness model will further improve the robustness. Selection of a more robust detection feature will also improve the robustness.

REFERENCES

- [1] D. Zou and J. A. Bloom: H.264/AVC Stream Replacement Technique for Video Watermarking, IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2008.
- [2] I. Cox, M. Miller, and J. Bloom: Digital Watermarking: Principles & Practice, San Mateo, CA: Morgan Kaufman, 2001.
- [3] D. Simitopoulos, S.A. Tsaftaris, 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.
- [4] 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
- [5] 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.
- [6] P. Kocher, J. Jaffe, B. Jun, C. Laren, and N. Lawson: Self-Protecting Digital Content, Cryptographic Research Inc. White Paper, 2003. <http://www.cryptography.com/resources/whitepapers/SelfProtectingContent.pdf>
- [7] ITU-T Recommendation H.264 | ISO/IEC 14496-10 International Standard with Amendment 1.
- [8] Iain E. G. Richardson: H264 and MPEG-4 Video Compression -- Video Coding for Next-generation Multimedia, Wiley, 2003
- [9] A.B. Watson, G.Y. Yang, J.A. Solomon and J. Villasenor: Visual Thresholds for Wavelet Quantization Error, Human Vision and Electronic Imaging, SPIE-2657, pp381-392, 1996.
- [10] C.J. van den Branden Lambrecht and J.E. Farrell: Perceptual Quality Metric for Digitally Coded Color Images, Proceedings of EUSIPCO, oo.1175-1178, 1996.
- [11] J. Lubin: The Use of Psychophysical Data and Models in the Analysis of Display System Performance, in A.B. Watson, editor, Digital Images and Human Vision. Cambridge, MA: MIT Press, 1993.
- [12] L.G. Brown: A survey of Image Registration Techniques, ACM Computing Surveys, 24(4):325-376, 1992.
- [13] B.D. Lucas and T. Kanade: An Iterative Image Registration Technique with an Application to Stereo Vision, Proceedings of the International Joint Conference on Artificial Intelligence, pp.674-679, 1980.
- [14] H.S. Stone: Progressive Wavelet Correlation Using Fourier Methods, IEEE Transaction on Signal Processing, 47(1):97-107, Jan. 1999.
- [15] JM H.264/AVC Software: <http://iphome.hhi.de/suehring/tml/>
- [16] FFMPEG: <http://ffmpeg.mplayerhq.hu/>