Noname manuscript No.
(will be inserted by the editor)

An Industry-Level Blu-ray Watermarking Framework

Jan De Cock · Heinz Hofbauer · Thomas Stütz · Andreas Uhl · Andreas Unterweger

the date of receipt and acceptance should be inserted later

Abstract In this paper, we present our H.264 Blu-ray watermarking framework which operates at bit stream level and preserves the length of the underlying bit stream. Apart from a description of our watermark embedding and detection (and synchronisation) approaches, we discuss the embedding capacity for different exemplary Blu-ray disks based on their bit stream characteristics as well as the robustness of our watermark to H.264 transcoding and resizing. Furthermore, we assess the parallelizability of our embedding approach and the impact of different hard drive configurations on the overall embedding speed, showing that low access times are as relevant as high transfer rates when maximum speedup through parallelization is desired. Lastly, this paper provides a discussion on a variety of design choices and practical issues which arise when designing an industry-level watermarking framework.

Keywords framework, watermarking, H.264, length-preserving, parallelization

Acknowledgements Special thanks to SONY DADC Austria AG, in particular Reinhard Blaukovitsch, for the cooperation in the project and the insights into industry requirements.
This work has been supported by the FFG bridge project 834165.

1 Introduction

As more and more movies are released on Blu-ray disk, the number of illegitimate copies which make it onto a variety of platforms throughout the Internet **before** the official release date increases, resulting in significant financial losses. Usually, sales[1] for the second week are about $60 - 80\%$ lower than the first week. This makes the first week the most influential for financial success of a release. A leak prior to the release can thus reduce the revenue of the financially most rewarding sales period. While DRM following release can also be an issue, it is not usually solved by means of a watermark but rather by copy protection mechanisms [10]. It should also be noted that the goal of the watermarking system presented in this paper is not to prevent but to reveal leaks. The security on-site, i.e., in the production plants, and the employed process security is responsible for preventing leaks.

We explicitly only deal with the leakage of content prior to the release date. The goal of the watermarking

Jan De Cock
Ghent University – iMinds, Gaston Crommenlaan 8 bus 201, B-9050 Ledeberg-Ghent, Belgium
E-mail: jan.decock@ugent.be

Heinz Hofbauer · Andreas Uhl · Andreas Unterweger
University of Salzburg, Jakob Haringer Str. 2, 5020 Salzburg, Austria
E-mail: {hhofbaue, uhl, aunterweg}@cosy.sbg.ac.at

Thomas Stütz
FH Salzburg, Urstein Süd 1, 5412 Puch bei Hallein, Austria
E-mail: thomas.stuetz@fh-salzburg.ac.at

[1] Sales information is taken from `http://www.the-numbers.com`

scheme is twofold. On the one hand, it allows to ascertain whether the content was leaked from a specific site, or conversely to plausibly deny that a leak has occurred. And, on the other hand, when a leak has happened, it allows to identify the source of the leak and aid in improving the local security arrangements to prevent further leaks. Content can be leaked in different production stages of a Blu-ray disk, making it necessary to identify the stage in which the leak occurred in order to eliminate it. One way to do so is by adding a watermark after the completion of each production stage. If content leaks, the existence of the watermarks from previous production steps identifies the production step in which the leak occurred.

A number of constraints are imposed on such a watermarking system intended for industrial application. In conjunction with our industrial partner, SONY DADC Austria AG, we identified the following list of constraints for both, practical and economical reasons.

Firstly, the watermark has to be robust against transcoding. The leaked video could be altered in terms of format, bitrate or aspect ratio, e.g., by reencoding to another format. In order to identify the source of the leak, the watermark has to be robust against such changes in order to be detected reliably after a leak.
Secondly, the watermark has to be invisible to the human eye. Any change in quality is a problem for a content provider since it would displease consumers and content creators alike. This in turn can impact sales and the reputation of the content provider.
Thirdly, the size of the watermarked content has to be equal to the size of the original content, i.e., the watermarking process has to be length-preserving. This is a practical restriction originating from a concurrent workflow. On the one side the video content is handled and on the other side the accompanying content, e.g., menus and chapter lists, are handled. On the menu side, the jump-in points to the video content are offset based. As such, they would have to be adjusted whenever the length of the video content changes. This would introduce a higher cost in the production process since the concurrency in the workflow would be inhibited.
Finally, Blu-ray watermarking has to be fast. While "fast" does not necessarily mean real-time processing, it means that undue delays in the production should not occur. This, again, would influence the production cost and is not acceptable. This implies that bitstream-based watermarking is more feasible than other watermarking techniques which require format-compliant reencoding and subsequent compliancy checks.

All other constraints which are usually assumed when dealing with a modern watermarking system, e.g., the requirement of blind watermarking or further robustness issues, are second to these primary concerns.

In this paper, we present a watermarking framework which fulfills all of the aforementioned criteria by watermarking a user-defined selection of the Blu-ray disk's video tracks. As nearly two thirds of the Blu-ray disks released to date contain video streams which are H.264-compliant[2] [7], most of which use context adaptive binary arithmetic coding (CABAC) [9] entropy coding, our approach is targeted at H.264 with CABAC.

Although full watermarking frameworks like ours have not been described in the literature, bit-stream-based and length-preserving watermarking approaches for H.264 have been proposed before. Our watermarking framework uses a variation of the approaches proposed in [11, 12] and [13], which both embed watermarks by changing motion vector differences in the bitstream. Although we do so as well, our modification allows for a significantly higher embedding capacity than the approach described in [13]. This is due to the greater set of modifications allowed by our approach as described in detail in Section 2. Although the capacity of our approach is slightly smaller than the one described in [11], the latter is limited to context adaptive variable length coding (CAVLC) entropy coding, which is rarely used on H.264-compliant Blu-rays.

CAVLC and CABAC are the two ways in which H.264 bit streams are entropy-coded. We apply the watermarking approach of Stütz et al. [11], which performs CAVLC watermarking, to CABAC entropy coded bit streams. Since entropy coding is inherently lossless, the actual changes we make to the visual data are entirely identical to the changes of the approach by Stütz et al. Therefore, both our approaches share the same properties with respect to rate distortion performance, subjective quality degradation and robustness, and security which are therefore not discussed in detail herein (they are described in detail in [11, 12]).

CABAC approaches come at the expense of an additional entropy reencoding step, which is not required by [11, 12] as they aim at finding substitutable code word parts which do not require entropy reencoding. While our CABAC approach employs only one entropy reencoding step for the entire bitstream, numerous fine grain entropy reencodings step are applied in the approach of [13]. The advantage of the approach of [13] is that actual watermark embedding can be implemented by simple bit substitutions. However, in our targeted application scenario this feature (substitution watermarking) is not required.

[2] `http://www.blu-raystats.com/Stats/TechStats.php` as of February 18, 2013

Since our watermarking framework is similar to the CAVLC framework proposed by Stütz et al. [11] as described above, we do not aim at reinvestigating their results, but instead focus on the industry-level implementation of our framework as well as on practical considerations thereby complementing results in [11, 12]. Thus, the contributions of this paper are as follows: First, we detail the technical approach to conduct H.264-CABAC bitstream-based embedding of the CAVLC technique in [11, 12] and explain the corresponding differences to [13]. Second, we discuss questions of detection and (re-)synchronisation in manipulated (i.e. scaled, cropped, transcoded) video. Finally, highly practical questions like computational embedding issues (runtime and storage aspects) as well as embedding capacity are covered.

This paper is structured as follows: In Section 2, we describe our watermarking framework, including the details of our H.264-CABAC-based watermarking algorithm w.r.t. embedding and detection. Subsequently, in Section 3, we outline practical considerations that evolved during the development of our framework (quality control, synchronisation and actual transcoding). Finally, in Section 4, we evaluate our watermarking approach as well as our framework in terms of speed and embedding capacity before concluding this paper in Section 5.

2 Framework Overview

Our watermarking framework consists of two major parts – one for watermark embedding and one for watermark detection. Figure 1 shows the components of the watermark embedding process as well as their interdependencies. The dotted line indicates the interfaces between our framework (on the right) and pre- or post-processing steps which are out of scope.

The watermarking process involves the following steps and components: Firstly, the demuxed H.264 stream is split into the smallest possible groups of pictures (GOPs) to allow parallelized watermarking. Secondly, each GOP is analyzed for possible watermark locations using a modified version of the H.264 reference software (JM). Thirdly, a quality control loop eliminates watermark locations which cause spatial drift as described in detail in Section 2.1.

Finally, the remaining watermarks are embedded using a transcoder as described in Section 2.2 before the watermarked GOPs are merged back together to form the watermarked output stream. Note that the watermark embedding framework additionally outputs detection information for the watermarks, i.e., the precise locations of the watermarks so that they can be found again during detection, the process of which is described in Section 2.3.

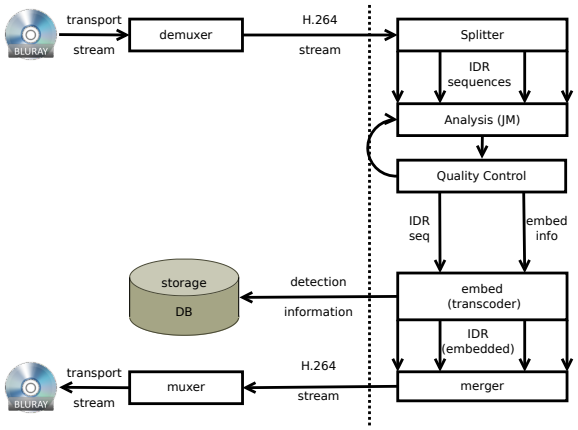


Fig. 1 Watermark Embedding Overview

2.1 Watermarking Approach

The basic principle for robust watermarking is to embed the watermark in coefficients of known robustness [4]. For a real world application this requires a feature which is robust against transcoding as well as spatial transformation, i.e., scaling and cropping. Since current video coding standards, e.g., H.264 [7], H.265 [3] and MPEG-4 Part 2 [6], rely on DCT based encoding the DC coefficient, i.e., the average luminance over a macroblock, is a good choice. Furthermore, if the spatial transformation applied to a video can be inverted, the same average luminance can be regained for a given macroblock. The utilization of the average luminance in the DC coefficient for watermarking is further affirmed by literature, see Hartung and Kuttner [5] for an overview or Chen et al. [1] for the use of DC coefficients for H.264. Overall, the known robustness characteristics regarding transcoding and spatial transformations render the DC coefficients the optimal choice for our application scenario.

It is possible to change the luminance of a macroblock by changing the motion vector differences in order to predict from another macroblock. If the new macroblock is brighter or darker, then the macroblock originally used for prediction, the predicted macroblock in the frame will also be brighter or darker. In this way we can adjust the average luminance with a minimal change in the bitstream. To find suitable blocks for watermark embedding we modify the MVD of a macroblock (i.e., we scan every macroblock in the reference frame in a given search radius for brighter and darker macroblocks which do not introduce a too large distortion). A macroblock can be watermarked if we find a

brighter (embedding a 1-Bit) and a darker (embedding a 0-Bit) macroblock to predict from.

Only a subset of the candidate MVD changes preserves the length of the bitstream. In H.264/CABAC MVDs are binarized (MVDs larger than 9 are encoded using exponential Golomb codes). Exponential Golomb codes consist of a prefix and a suffix. The bits of the suffix are encoded in bypass mode, i.e., all bits are assumed to have equal probability. In a perfect arithmetic encoder equal probabilities would result in a same length bitstream, in the case of the H.264 arithmetic encoder length-preservation is at least very likely.

While our approach employs all these candidate MVD (with same prefix, but different suffix) the approach of Zou and Bloom [13] further reduces the candidate MVD changes dramatically. Zou and Bloom consider only MVD changes that preserve the exact arithmetic encoder / decoder state. No probability states are updated in bypass mode and the range variable R (codIRange in [7, see clause 9.3.1.2]) is also preserved [13]. However, it has to be checked whether the encoding of the different suffix results in the same offset L (codIOffset in [7, see clause 9.3.1.2]). Therefore the suffix bits need be to arithmetically encoded (for all candidate changes) and checked against the offset from encoding the original suffix. The variable codIOffset is in 16 bit register precision and requires a minimum precision of 10 bits [7, see clause 9.3.1.2]. Thus only one of 1024 candidate changes will not be rejected (using the conservative assumption of a uniform distribution on the values of codIOffset). The significant reduction of candidate changes reduces the capacity and / or requires to analyze more candidate changes. Furthermore, while the approach of Zou and Bloom requires a significant amount of entropy encoding in the analysis step, our approach completely avoids any entropy encoding in the analysis stage and performs only one entropy encoding pass in the embedding stage.

A change in a macroblock can introduce further bit errors through the prediction modes utilized by the H.264. In order to prevent inter-frame propagation of errors we watermark only non-reference frames, i.e., we utilize non-reference B-frames or if the GOP structure is of the form IP* we only change macroblocks from the trailing P-frame in the GOP. There is still the problem of intra frame predictions which can lead to spatial drift in the same frame. In order to deal with this we employ a quality assurance (QA) loop, described in Section 3.1, which detects drift in the decoded frame and reverts the macroblock changes which introduce the drift.

The drift is only removed if a given error is exceeded in non-watermarked macroblocks (for the used threshold see Section 3.1). In order to prevent the drift we remove the embedding from possible prediction sources. Since the intra prediction predicts from macroblocks to the left and above of the current macroblock, only embeddings in this region are removed. By removing all possible prediction ancestors, the QA-loop does not impact the performance of the system unduly, but the embedding capacity is reduced more than strictly necessary. However, since the capacity is still high enough, see Section 4.2, this faster way of removing drift sources is preferable to a slower but more precise method.

2.2 Embedding Approach

When changing the MVDs of a CABAC bit stream by changing the corresponding CABAC code words, the state of the arithmetic coder is very likely to change, resulting in invalid bit streams if the code words are only replaced. Hence, a bit stream transcoder is required which performs the CABAC reencoding so that the rest of the bit stream remains valid. Note that no actual pixel-level decoding or reencoding is necessary as all required changes only involve the entropy coding layer.

As regular transcoders are not capable of performing entropy-only-reencoding with the additional ability to change MVDs, we used a special bitstream transcoder developed at Ghent University which is capable of performing the required changes [2]. The transcoder provides an interface which allows locating and changing the desired MVDs for each frame and outputs the modified, i.e., watermarked, bit stream.

In the transcoder, a cascade of a decoder and an encoder, which is typically used in video stream adaptation, is avoided. Not only will such a cascaded approach lead to a higher complexity (since it combines a decoder and encoder loop), it will also introduce a quality loss, even at identical quantization settings (caused by rounding). To avoid these drawbacks, an open-loop mechanism is used in our transcoder [2]. First, the bit stream is entropy decoded, resulting in the syntax elements listed in the H.264 specification (such as macroblock types, MVDs, and residual coefficients). Then, the MVDs are modified where needed, while all other elements remain identical, hereby avoiding changes which are not related to the watermarking process. Subsequently, the syntax elements are again entropy coded with the updated state of the arithmetic coder.

2.3 Synchronization and Detection

Watermark detection is non-blind and relies on a detection info file containing temporal and spatial water-

mark location as well as the embedded bit along with the original feature value. In order to extract the watermark from the video under test we have to synchronize the video under test and the original video by reconstructing the original spatial and temporal dimensions. In the spatial domain, eventual scaling and cropping needs to be reversed. For the temporal dimension we only deal with cut or added frames at the beginning of the video.

In essence we only need to determine the crop (left, right, top and bottom) of the original video to the video under test. Given the crop (c_l, c_r, c_t and c_b) we can calculate the inverse aspect ratio and scale since the original video size, $o_w \times o_h$ and the size of the video under test, $t_w \times t_h$, is known from the detection file and actual bitstream respectively. To invert the scaling and cropping by linearly transforming the video from $t_w \times t_h \;\mapsto\; (o_w - c_l - c_r) \times (o_h - c_t - c_b)$ and pad with black border according to c_l, c_r, c_t and c_b. See Section 3.2 for strategies how to actually determine crop parameters.

Since the spatial dimensions are aligned we can now utilize the watermark information from the detection file to do a scan for temporal alignment. Utilizing N watermark bits we can scan the first F frames of the video under test and calculate the correlation C, as given below. Under the assumption that the video is watermarked the scan should yield a unique frame offset where the correlation reaches maximum. If the highest correlation is not unique a rescan of the prospective offsets with an increased N should reduce the number of equal correlations until only one remains. This offset is taken as temporal shift and used in the actual watermark detection with the whole watermark sequence. This approach differs from traditional temporal synchronization approaches which utilize redundancy in the watermark, e.g. [8]. However, since we utilize a non-blind watermarking scheme we do not require redundancy since the whole watermark information is available during detection.

Given a synchronized video under test we then have two binary sequences, one is the original watermark sequence, $\mathrm{wm}, \forall i : \mathrm{wm}_i \in \{0,1\}$, from the detection file which consists of the bits embedded in the original video. The other sequence is the extracted watermark sequence, $\mathrm{ex}, \forall i : \mathrm{ex}_i \in \{0,1\}$ which is extracted from the synchronized video under test. The extracted watermark sequence is calculated by extracting the relevant feature from the given location and comparing it with the original feature as given in the detection info, this process is illustrated in figure 2.

The detection is based on the probability of false positive, i.e., the probability that a watermark is detected in a non-watermarked video. The watermark bits wm_i are drawn from a uniform random distribution in $\{0,1\}$ and we assume that the extracted bits ex_i are also uniformly distributed in $\{0,1\}$. We calculate the correlation between wm and ex in the following manner

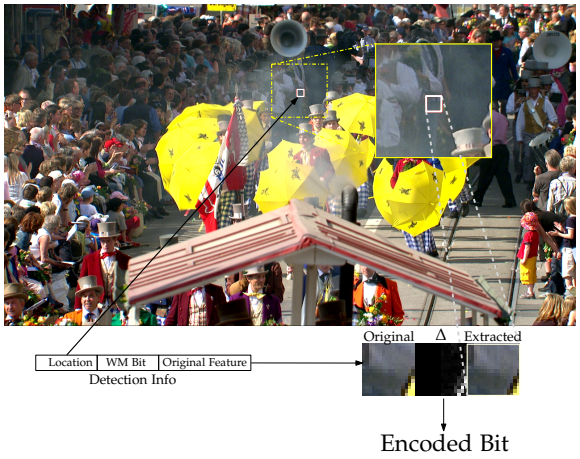


Fig. 2 Extraction of a single watermark bit from a video under test.

$$C = \frac{1}{n}\sum_{i=1}^{n}(2\,\mathrm{wm}_i - 1)(2\,\mathrm{ex}_i - 1),$$

where n is the number of bits of wm and ex. The probability of false positive is then the probability that two random sequences have at least correlation C. We can easily see that each member of the sum, $(2\,\mathrm{wm}_i - 1) \cdot (2\,\mathrm{ex}_i - 1)$, is a Bernoulli trial with $p = q = \frac{1}{2}$. Thus C has a binomial distribution $B(n,p)$ and the probability of false positive is consequently

$$\mathrm{pfp}(C) = \sum_{k=k_C}^{n}\binom{n}{k}p^k q^{(n-k)} = \\ = \sum_{k=k_C}^{n}\binom{n}{k}\left(\frac{1}{2}\right)^n = \frac{1}{2^n}\sum_{k=k_C}^{n}\binom{n}{k},$$

where $k_C = \frac{(C+1)n}{2}$.

We assume video under test is a leaked video if the probability of false positive is lower than a threshold, i.e., $\mathrm{pfp}(C) < T_C$, T_C defaults to 10^{-12} but can be freely chosen.

Figure 3 gives an overview over the probability of false positives (pfp) under different scaling and quantization parameters. An original video (a sample of Band MF), which is encoded in H.264 with HD1080 resolution, was watermarked, 1839 bits were embedded in 1644 frames. The pfp is given in logarithmic scale and capped at 10^{-100}, the default threshold (10^{-10}) for watermark detection is also given. As can be seen the wa-

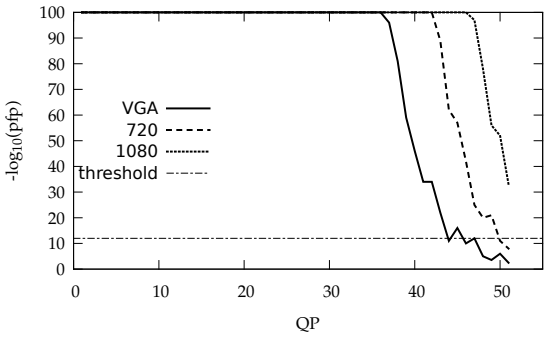


Fig. 3 Probability of false positive for different quantization parameters and resolutions. Plot is capped at pfp = 10^{-100} and the default threshold is also given at 10^{-12}.

termark detection is robust against scaling, bit rate reduction and the transformation to a different aspect ratio, i.e., $16:9 \mapsto 4:3$. Figure 4 shows the sample part of a frame from the original and for the rescaled versions. The samples from the rescaled version were taken from the sequence with a QP for which the pfp $< T_C$, which is QP 44 and 50 for VGA and HD720 respectively, compare fig. 3. For more information about watermark correlation under different embedding strength and quality parameters see [11].

3 Practical Considerations

In this section we provide information about effects and circumstances which in practice impacted the design and decision making regarding the framework. The topics presented here were selected because they have a huge impact on either the design of the framework, like the decoder, or are important to consider for practical application, i.e., quality assurance and length preservation.

We look at the quality assurance and show how, and why, the current embedding strength was chosen. We explain the practical considerations behind the process of dealing with the situation when a GOP changes length and finally, we explain why the use of a transcoder is necessary and what problems can arise from using a transcoder.

3.1 Quality Assurance

For quality assurance the need to utilize a fast and reliable metric on a basic level lead to the use of the MSE for watermark embedding and quality assurance. In [11, 12] a subjective experiment is presented, which suggests that an embedding strength of 100 in terms of MSE is sufficiently low to be imperceivable. Since our approach and the one from [11, 12] share the same properties as explained in Section 1, we use an embedding strength of 100 as well. On the one hand we found that using MSE 100 as a limit for the macroblock change allows for a sufficient number of watermark bits. Statistics about the possible number of embedded watermark bits, depending on source material, are given in Section 4.2. This high embedding strength results in a good detection response and low probability of false positives, even for highly impaired images, as detailed in Section 2.3, fig. 3. However, a error of higher than 100 MSE can still occur through prediction from a modified macroblock and drift of the error.

We can preclude temporal drift by systematically avoiding embedding in frames which are a source of temporal prediction. This leaves non-reference B-frames or, in the case of GOPs with IP* structure, trailing P-frames for embedding. However, spatial drift of the error can still occur for such frames.

As the targeted application scenario requires reliably high quality, we introduce a quality assurance stage to eliminate spatial drift. In order to prevent a higher than allowed distortion the quality assurance loop checks the whole frame for errors that surpass our MSE 100 limit. If such errors are found the QA loop traces the source of the predictions which introduces these errors and reverts any changes to the responsible macroblocks. A given macroblock is used for prediction only by macroblocks to the right and downwards of the current block. Conversely, the source of an error for a given macroblock is located left or upwards of the current macroblock. The QA loop searches for potential sources of error drift and removes the embedding from them. While this lowers the embedding capacity, the resulting capacity is still high enough for all practical purposes, see Section 4.2.

3.2 Synchronization Method

In the final framework we chose a semiautomatic method for watermark synchronization to improve detection. The main reason was to increase the stability of the detection. The drawback of the semiautomatic method is that human intervention is needed to measure crop, if present. While this is more costly, in terms of personnel cost and time, it also increases the detection rate by providing exact crop detection. However, the time consumed by exact crop measures is refunded by the fast scanning for synchronization which can be done when crop is known.

The other option would be a fully automated synchronization by detecting both crop and synchronization algorithmically. The problem with a fully auto-

(a) HD1080 (b) HD720, QP 50 (c) VGA, QP 44

Fig. 4 Side by side comparison of a sample of frame 111 from the test sequence. Shown are the original and scaled version (for failed watermark detection).

mated approach is that the fast scanning for synchronization requires a known crop while the automatic detection of the crop requires two known matching frames. Thus, we have to switch to a different synchronization method.

3.2.1 Automated Detection of Temporal Displacement

For synchronization without known crop we utilize a scale-invariant feature based synchronization, by extracting scale-invariant features from the original video and the video under test and searching for matching frames. The advantage of this approach is the fact that the potential scale and crop of the video under test do not have to be known in order find temporal synchronization. While this works well it also has certain drawbacks. The scale-invariant feature extraction, and the subsequent matching, is slow and computationally expensive. Furthermore, it requires the original video, as opposed to the current approach, since only the original feature values are stored in the detection info file (see fig. 2). This further increases the computational demand for this method since the original video also has to be decoded. Additionally, care has to be taken to not wrongly synchronize with repeating sequences. An example of this would be a transition sequence which appears multiple times in the video under test and can match with the same transition sequence at another point in time in the video. While these problems can be handled, the resulting synchronization attempt is more complicated and time consuming than the one currently employed.

The method used for the detection of the temporal offset is based on outlier detection. Using scale-invariant-features we can calculate the difference between matching feature points. While this difference hardly ever becomes zero, due to changes in quality during re-compression of the video under test, we expect matching frames to produce a significantly lower feature distance than non-matching frames. In order to find the offset, a frame from the video under test is compared to frames in a search window of the original video. On this search window we perform outlier detection and find the best matching frame. If no outliers are found, the search windows is advanced in the original video and the process is repeated. If an outlier is detected we apply another detection with the next frame of the video under test. This has to be done in order to ascertain whether the outlier was a set of matching frames as opposed to a random statistic outlier. We assume a true match if the following N consecutive frames from the video under test also match the following N consecutive frames from the original video. This process is illustrated in fig 5.

Since not all scale-invariant feature detection approaches exhibit the same performance, a number of tests were conducted to find the most likely candidate. In order to find the best feature detector and feature point extractor pair we conducted an experiment using the SURF, ORB, FAST, STAR, HARRIS and MSER detectors and SURF, ORB and BRIEF extractors provided by the OpenCV. A test set was generated based on four short sequences to be used as original video as well as a number of expected changes, i.e. temporal crop, combined with scaling and quality reductions. The videos under test exhibit offsets of 10 or 25 frames, down-sampling to HD720 and VGA resolution (from an HD1080 original video) combined with a bit rate cap of 1024kbps and 200kbps. The experiments using the above algorithm (with a search window of 21 and

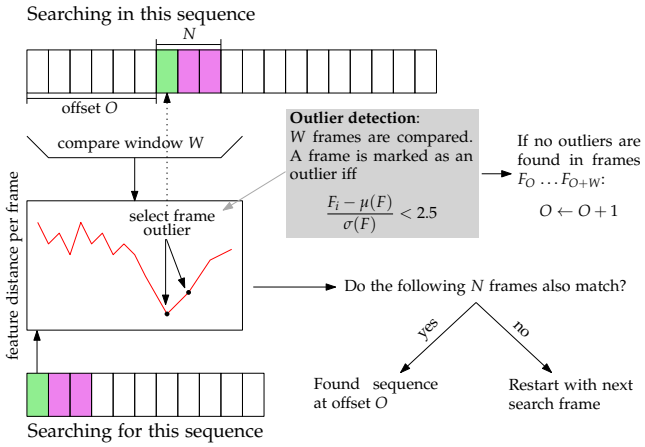


Fig. 5 Overview over the temporal shift detection using scale-invariant-based features.

Table 1 Temporal offset detection rates for various combinations of feature detectors and extractors.

[%]	Extractor		
Detector	SURF	ORB	BRIEF
SURF	100.000	84.375	84.375
ORB	75.000	68.750	59.375
FAST	84.375	87.500	90.625
STAR	56.250	56.250	50.000
HARRIS	78.125	81.250	78.125
MSER	81.250	81.250	81.250

5 required consecutive matches) produces the detection rates as shown in table 1. SURF is clearly best choice among those tested. However, for all of the detectors under test the introduction of crop, especially under low quality conditions produces faulty synchronization.

In addition to finding the correct offset in low quality, scaled and cropped videos under test there is also a systematic error which is introduced by repeating or similar sequences which can lead to a faulty offset detection. Typical examples of similar sequences are cross fades, fades to black and scene change sequences. There is no clear way to exclude these sequence except by increasing the number of necessary consecutive matches (N in the above algorithm). However, increasing the number of consecutive matches also leads to an overall lower performance when detecting temporal shift in low quality sequences.

3.2.2 Automated Detection of Spatial Displacement

The automatic detection of spatial displacement assumes a temporal alignment and tries to find the crop and scale which leads to the spatial displacement. The only other influencing factor, besides spatial changes, is the quality of the video under test.

While there is the option of using the feature points extracted for temporal synchronization to find the projection of one video into the other, experimental results showed that this is unreliable. There are instance where the number of feature points are insufficient to find a projection. Another problem is avoidance of features points which can not be matched, while this is required for some sequences it will introduce errors into others. Overall the use of extracted feature points for spatial synchronization did not consistently perform well enough.

Thus, in order to detect crop an approach based on template matching is the obvious solution. The template matching approach uses the video under test as a template and tries to find it in the original video. A direct search however is bound to produce a mismatch if scaling also affects the video under test. In order to compensate for scaling we have to do a template match with different scale factors. A list of possible scale factors, with visual examples, are given in fig. 6.

This exemplifies that we have to consider different scales when performing template matching. The scale space is hardly limited besides very one-sided scaling options, like stretching along one axis and shortening along the other. What further complicates the matter is the fact that template matching, under these transforms with the template error as distance measure does not create a convex space. This is illustrated in fig. 7 where for each scaling factor the value of the best match is given as a heat map. If the space were convex, we could perform a gradient descent search for the optimal match. However, since the space is not convex, we have to do a more complex, and consequently computationally more expensive search.

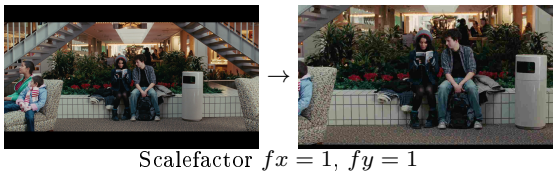
Assuming, based on the examples from fig. 6, no more than half of a picture is cut and upscaled and at most a down-sampling to VGA from HD1080 the scale space is in the range $\mathcal{S} = [0.5, 3] \times [0.5, 3]$. Assuming we utilize a search step of δ_s we can calculate the maximum number of pixels by which we will miss the correct resolution. This can be done by down-sampling with the maximum scale which is also at the largest distance from the chosen search step. The pixel difference δ_p will then be

$$\delta_p = \frac{1920}{3 - \frac{\delta_s}{2}} 3 - 1920.$$

Conversely, we can calculate δ_s for a given δ_p by

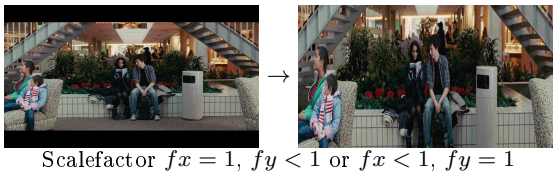
$$\delta_s = 6 - 6\frac{1920}{\delta_p + 1920}.$$

For a negligible pixel difference, i.e. $\delta_p^N < 0.5$ such that rounding to integer produces the correct resolution, the



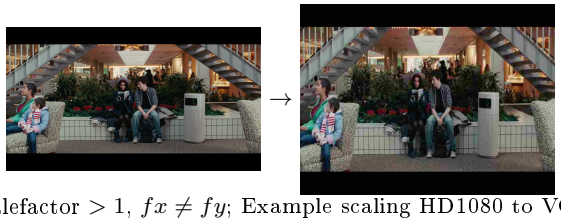
Scalefactor $fx = 1$, $fy = 1$

(a) Crop



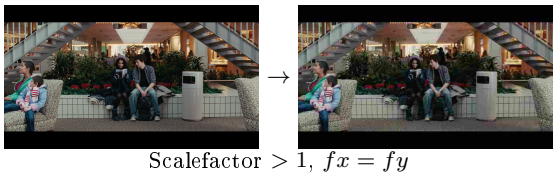
Scalefactor $fx = 1$, $fy < 1$ or $fx < 1$, $fy = 1$

(b) Crop and upscale



Scalefactor > 1, $fx \neq fy$; Example scaling HD1080 to VGA
$fx = 1920/640 = 3$, $fy = 1080/480 = 2.25$

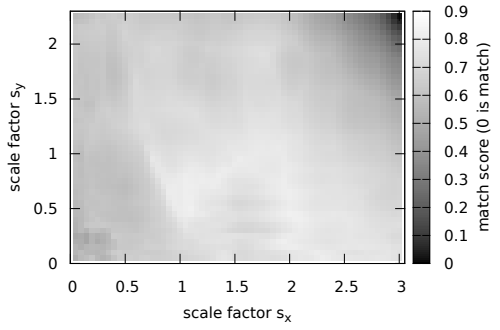
(c) Scale



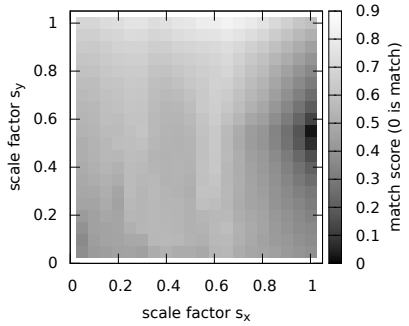
Scalefactor > 1, $fx = fy$

(d) Scale, keep aspect ratio

Fig. 6 Examples of different scale factors based on various possible spatial distortions.



(a) Scale



(b) Crop and upscale

Fig. 7 The heat map shows the matching score (lower is better) for different scales, separate for the x- and y-axis. The heat maps are for the examples Scale (fig. 6c) and Crop and upscale (fig. 6b).

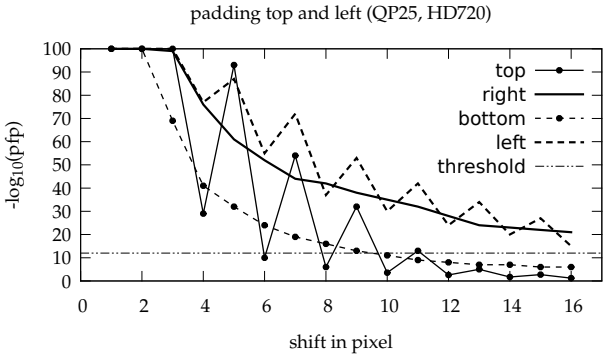


Fig. 8 Detection rate when the spatial de-synchronization in the video under test could not be correctly compensated.

resulting search step size is $\delta_s^N = 0.00156$. Searching in $\mathcal{S}$ with δ_s^N would result in over $2.5 \cdot 10^6$ template matches. For a rough comparison $2.5 \cdot 10^3$ matches take about 10 minutes. Thus, clearly this approach is not feasible.

The question is then what influence δ_p has on the detection rate, since an increase in δ_p significantly increases δ_s. We did a test with a medium quality sequence and simply shifted the video under test in the range from 1 to 16 pixels.

Figure 8 shows the result for the detection. Note that the y-axis is capped at 10^{-100}. A 6 pixel shift is the first offset where detection fails, as such the pixel error has to be significantly lower. Note that the ffmpeg library used for rescaling treats even picture sizes differently due to alignment-related optimizations, exhibiting the depicted fluctuations in the detection rate for the top and left curves.

Let us assume $\delta_p^2 = 2$, since for each of the cases the detection rate for a two pixel shift is well above the threshold. This would result in $\delta_s^2 = 0.006$ with $160 \cdot 10^3$ required matching steps, which would take almost 11 hours.

Overall, using a semiautomatic method is faster and more accurate than the fully automatic method.

3.3 Transcoding

As described in Section 2.2, the change of the state of the arithmetic coder requires reencoding. Due to these introduced changes, the positions in the bit stream where the arithmetic coder performs its renormalization may change, thus potentially changing the length of the bit stream. As the arithmetic coder is reinitialized at slice boundaries, these length changes cannot influence subsequent slices, unless they are watermarked as well.

As changes in length are not allowed, watermarked GOPs are replaced by their original, i.e., unwatermarked, versions during the merging process at the end. This way, all watermarked GOPs whose length remains unchanged are kept and the GOPs whose length changed are not watermarked. Note that this is easy to do, but lowers the embedding capacity, influencing detection later. We discuss this in detail in Section 4.2. It is also possible to preserve length at NALU level using a similar process which replaces all watermarked NALUs whose lengths differ with their original versions.

Another practical issue that has to be considered during the watermarking process involves open GOPs. An open GOP references pictures which are not contained in that GOP, as opposed to a closed GOP in which each picture can be decoded independently of pictures from other GOPs. Although open GOPs can be easily detected, they cannot be watermarked unless they are grouped together with preceding and/or subsequent closed GOPs. For the sake of simplicity, we detect and omit open GOPs from the watermarking process. Note that this potentially reduces the embedding capacity depending on the number of open GOPs. We analyze and discuss this in detail in Section 4.2.

4 Statistics and Evaluation

In this section we will evaluate two important properties discussed in previous sections.

First, the framework was designed with separate splitting and merging steps in order to utilize the context separate GOP structure for parallelization. We will show how parallelization influences the embedding process and illustrate where the bottlenecks for parallelization are.

Second, in previous sections we argued that the chosen embedding strength is sufficient to embed a high number of watermark bits even with the possible loss of potential watermarking locations due to length changes. We will give statistics about the actual occurrence of length changes and open GOPs as well as occurrence and distribution of watermark bits in an embedded stream.

4.1 Parallelization and Runtime

The QA loop performs a number of decodings of the original bit stream in order to find suitable watermarkable macroblocks. Consequently, the QA loop has high computational requirements and is slow. An example of this is given in table 2 where watermarking a 30 minute sequence takes a total of almost 12 hours. This is unsuitable for a practical application and the time requirement has to be reduced. If parallelization is possible the watermarking time can be split among a number of cores or machines and consequently reduce the overall watermarking time greatly (at the cost of computational power).

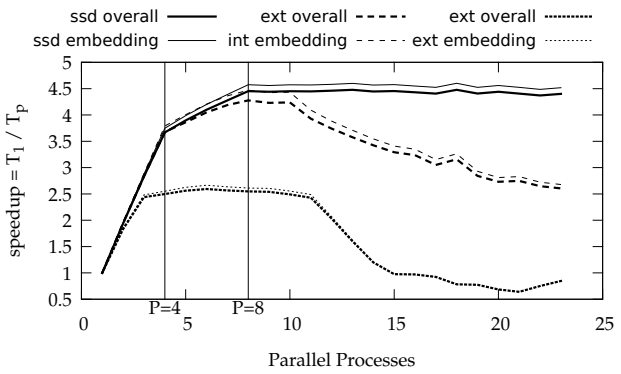
Our framework splits the H.264 bitstream into separate GOPs, performs analysis and embedding per GOP and, after sanity checks, merges the GOPs together to create the watermarked bitstream. The important part is that GOPs do not share a context, i.e., we can handle GOPs separately without interdependence on the bitstream side. Since we embed a random sequence based on a key the same concept of independent context holds for the embedded bits. Thus we can parallelize the analysis and embedding steps, which accounts for the major part of the watermarking time.

For the figures and tables in this section we used a 30 minute full HD (HD1080) subsequence of the Hancock movie. Parallelization was done on a machine with an INTEL core i7-3770 with four physical cores and eight logical cores via hyper-threading, all cores share a common L3 cache and a separate L2 and L1 cache is available per core. In order to distinguish between cache effects and tertiary storage effects on parallelization we ran the experiments twice on the same PC but with different hard disks.

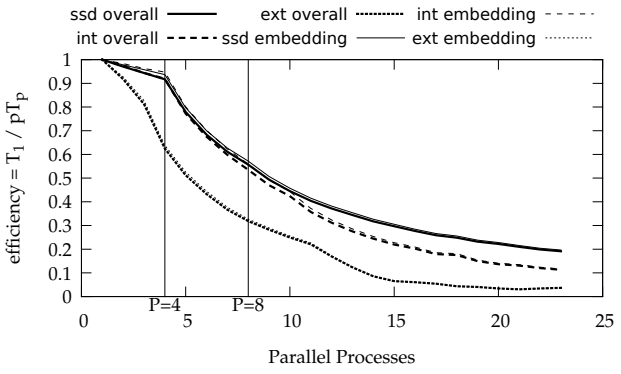
For the cache test run we used a SSD disk (denoted *ssd* where applicable), a Liteon solid state disk (LCT-256M3S) with 256 GB capacity, an average transfer rate of 324.9 MB/sec and 0.1ms average access time. Another test is done with a regular internal disk (denoted *int*), a Western Digital Caviar Blue (WD10EALX) with 1 TB capacity, an average transfer rate of 101.7 MB/sec and 16.8ms average access time. In order to show the impact of a slower disk we used a external hard disk (denoted *ext* where applicable), a Western Digital Caviar Green (WD20EARX) with 2 TB capacity, an average transfer rate of 37.2 MB/sec and 14 ms average access time. The limiting factor for the transfer rate of the external disk was the transfer speed over the USB port rather than the actual hard disk transfer rate. Throughput and access time measurements were performed with HD Tune 2.55.

Table 2 Time distribution for watermarking a 30 minute full HD sequence.

task	*ssd*			*int*			*ext*		
	time [s]	% of total	time	time [s]	% of total	time	time [s]	% of total	time
splitting	262	0.61	4:22	325	0.75	5:25	454	0.87	7:34
embedding	42849	99.30	11:54:09	43009	99.16	11:56:49	51365	99.03	14:16:05
merging	39	0.09	39	40	0.09	40	46	0.09	46
total	43151	100.00	11:59:11	43375	100.00	12:02:55	51866	100.00	14:24:26
parallel 4x	11762	27.26	3:16:02	11798	27.20	3:16:38	20765	40.03	5:46:05
parallel 8x	9691	22.46	2:41:31	10141	23.38	2:49:01	20354	39.24	5:39:14



(a) Parallelization Speedup



(b) Parallelization Efficiency

Fig. 9 Speedup and efficiency plot for parallelization with p processes on an Intel i7-3770 CPU with 4 cores and 8 logical cores (through hyper-threading).

Table 2 shows the time required for a full watermarking run and how the required time is distributed among splitting, embedding and merging. The table also shows the total time required for embedding under 4× and 8× parallelization, i.e., four or eight analysis/embedding steps are started simultaneously, the overall splitting and mering time for the parallelization processes is the same.

A more detailed overview is given in fig. 9 were the speedup and efficiency are given for a different number of parallel processes. Given are the overall time, i.e., splitting, embedding and merging combined, as well as embedding only.

If we disregard HDD limitations, i.e., the *ssd* case, it can clearly be seen that parallelization up to the number of physical cores is almost linear (efficiency > 0.9). Further parallelization up to the number of logical cores still improves overall speedup but at a lower rate, this is due to cache conflicts in the shared L2 and L1 cache between two logical cores. Parallelizing with a number of processes higher than the number of logical cores does not improve speedup.

From the *int* and *ext* cases we can see that access time is not a limiting factor for initial speedup. Even though the *int* HDD has the slowest access time it shows the same basic speedup pattern as the *ssd* case while the *ext* HDD has a tremendous impact on speedup. There is still a speedup and the overall process benefits from parallelization but when the HDD transfer limit is reached, at $P = 3$ in the figure, further parallelization does not improve overall computation speed.

Furthermore the parallelization should never use more cores than are actually present, counting logical cores. While we see in fig. 9 that utilizing more threads is not detrimental as long as the HDD is able to handle the seek time, which is easily the case for SSD disks. However, when looking at the speedup for the *ext* case it is clear that a high number of threads, and associated reads and writes, can cause a slowdown due to seek time. In the figure at $P = 11$ for the *ext* case and $P = 10$ for the *int* case showcase this stalls. Since the *int* case shows a slowdown earlier than the *ext* case this behaviour cannot be due to transfer rate. However, when looking at the average access time of the *int* and *ext* case, 16.8ms and 14ms respectively, it is clear that this slowdown is due to seek stalls during reads and writes. These seek stalls prevent the required data from reaching the worker threads leading to an overall drop in speedup, in extreme cases, e.g., $P = 21$, the speedup can drop below 1. This is a hard limit of the HDD, meaning the tertiary storage transfer rate as well as average access time limits the parallelization.

Overall, it is clear that the parallelization works well, almost a linear speedup with the number of processes used, but is limited by sharing primary memory as well as the access time and throughput of tertiary memory.

4.2 Embedding Capacity

To evaluate the embedding capacity of our watermarking approach, we used the main movies of nine different Blu-ray disks. All movies were watermarked completely, i.e., from beginning to end. The results are summarized in table 3.

We distinguish two different capacities: On the one hand, applications which require length-preservation at NALU level enforce that NALUs whose length changed during the watermarking process are replaced by their unmodified versions, i.e., the unwatermarked NALUs. This replacement reduces the number of embedded bits, leaving a total capacity denoted as "Capacity (N)". On the other hand, applications which require length-preservation at GOP level tolerate NALU-level length changes as long as the GOP length remains the same. Similar to the NALU-level length preservation, GOP-level length preservation enforces GOPs whose length changed during the watermarking process to be replaced by their unmodified versions. This replacement reduces the number of embedded bits on a GOP level, leaving a total capacity denoted as "Capacity (G)".

As NALU-level length preservation only required replacing single NALUs whose length changed during the watermarking process, it generally allows for a higher capacity than the GOP-level length preservation. The latter has to discard all bits in a GOP when its length changed, reducing the capacity significantly if the number of GOPs is low, i.e., the number of frames and therefore NALUs per GOP is high. In the examples listed in table 3 the capacity of the NALU-length preservation watermarking approach is between about 1.5 and 3 times higher than the capacity of the GOP-length preserving approach.

It is clear that the overall embedding capacity varies strongly, although several conclusions can be drawn: Firstly, movies with lots of motion, e.g., Resident Evil: Extinction, tend to have a higher capacity, whereas the opposite is true for movies with little motion, e.g., Enemy at the Gates. Secondly, movies which are longer, e.g., Gandhi with more than 270,000 frames, tend to have a higher capacity, whereas the opposite is true for short movies, e.g., Maya with less than 132,000 frames (which was to be expected). Thirdly, movies with a high percentage of non-reference B frames (denoted as b frames), e.g., 1492, tend to have a higher capacity than movies with a low percentage of b frames, e.g., Maya.

Furthermore, the distribution of watermark bits as given by the capacity in table 3 is not uniform but also depends on the structure of the video. Figure 10 illustrates this on a high capacity video (1492) and a low capacity video (Enemy at the Gates). The figure gives the average number of bits per frame calculated on a GOP basis and is plotted over the frame number, which represents the location of the capacity in the video.

However, there is another important factor which influences the embedding capacity: the existence of open GOPs. As open GOPs cannot be watermarked (see section 3.3), the potential watermarking capacity is reduced by each open GOP, therefore being lower when there is a high percentage of open GOPs. Although movies with little motion and a significant number of b frames, e.g., Enemy at the Gates, have a significantly lower capacity compared to the other movies in table 3, the number of embedded bits is still very high and allows for easy detection.

Note that the relative number of open GOPs seems to be very low, although a larger test set would be necessary in order to evaluate this in more detail. In our small test set, most movies have either no or only one open GOP, which is located at either the very beginning or the very end of the corresponding movie. Note that open GOPs at the end of a movie do not necessarily reduce the capacity as linearly scrolling credits lead to MVDs which are mostly zero and can therefore not be watermarked using our approach.

5 Conclusion

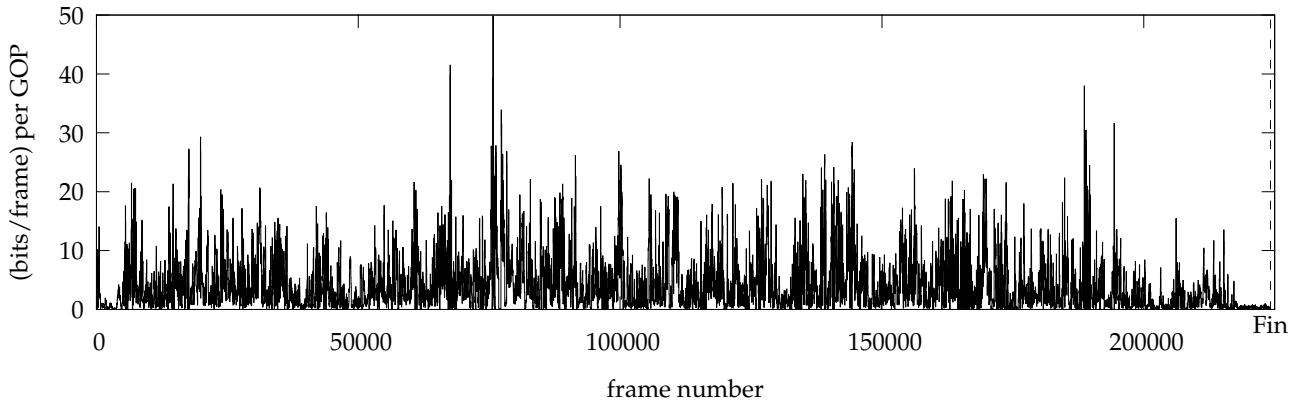
We presented a Blu-ray watermark embedding and detection framework which offers robustness to transcoding and scaling. In addition, we showed how different videos and bit stream characteristics influence the embedding capacity and run time. Furthermore, we showed that our approach is highly parallelizable subject to hard disk limitations, revealing that the hard disk's access time is as crucial for achieving maximum speedup as the hard disk's transfer rate.

From a practical point of view, we discussed that splitting the bit stream enables parallelized embedding in the first place. Furthermore, the design choice to only mark non-reference frames helps avoiding temporal drift, thereby making the quality control loop in the embedder less complex. In conclusion, we showed that the robustness and run time of our framework suffice to meet industry-level requirements.

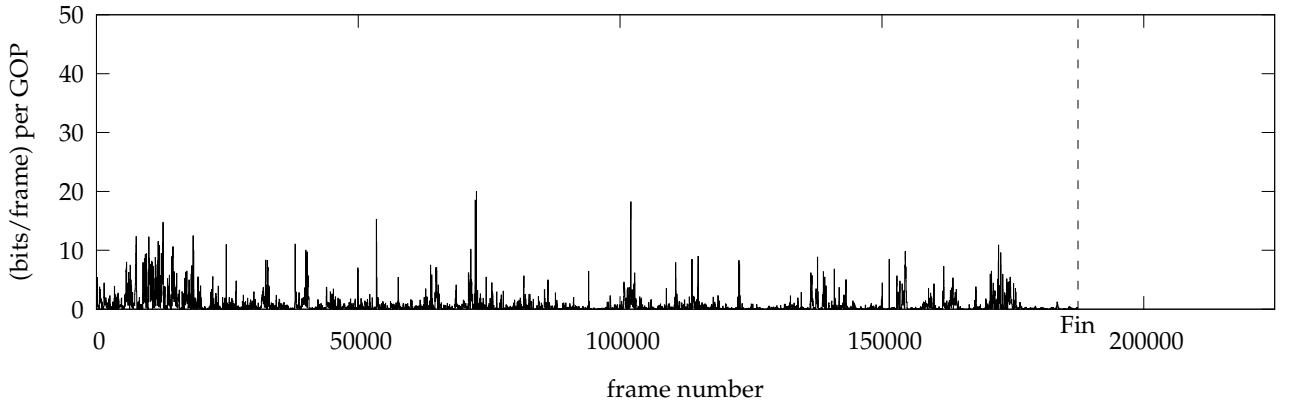
From a theoretical point of view, we gained knowledge about the requirements for an industry-level wa-

Table 3 Embedding capacity with frame and GOP statistics for a set of exemplary Blu-ray main movies

Movie name	Capacity (N)	Capacity (G)	# Frames	% b frames	#GOPs	#Open GOPs
1492	1,253,766	680,313	224,208	56.44	10,169	0
American Beauty	361,339	251,432	174,874	34.66	1,016	1
Cazzia	599,521	237,560	130,056	57.22	7,890	0
Enemy at the Gates	146,445	46,974	187,447	45.82	5,723	297
Gandhi	650,533	285,677	274,486	45.30	7,965	197
Independence Day	255,108	138,870	208,272	32.89	2,839	1
Maya	285,539	131,738	132,673	34.94	1,633	0
Resident Evil: Extinction	1,078,844	363,975	135,246	52.35	8480	0
Thor	402,095	229,693	164,920	32.36	1814	1



(a) 1492 (680,313 bits in 224,208 frames)



(b) Enemy at the Gates (46,974 bits in 130,056 frames)

Fig. 10 Location of the embedding capacity in the given videos. Note that the y and x axes are to the same scale, the dashed vertical line denotes the end of the corresponding video.

termark application. Namely, properties which are often thought of as irrelevant in science, like length changes, are important in practice since they entail complicated changes in the rest of the Blu-ray image. Other considerations which are usually treated with higher priority in science, e.g., blind watermarking, are of less or no concern. In conclusion, the design of watermarking methods should further improve the preservation of source properties, i.e., more than just format compliance, to boost applicability.

References

1. Chen, T., Liu, S., Yao, H., and Gao, W. (2006). Spatial Video Watermarking Based on Stability of DC Coefficients. In Yeung, D., Liu, Z.-Q., Wang, X.-Z., and Yan, H., editors, *Advances in Machine Learning and Cybernetics*, volume 3930 of *Lecture Notes in*

Computer Science, pages 1033–1042. Springer Berlin Heidelberg.

2. Cock, J. D., Notebaert, S., Lambert, P., and de Walle, R. V. (2010). Requantization transcoding for H.264/AVC video coding. *Signal Processing: Image Communication*, 25(4):235–254.
3. Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3, J. and ISO/IEC JTC1/SC29/WG11 (2012). High efficiency video coding (HEVC) text specification draft 8. `http://phenix.it-sudparis.eu/jct/doc_end_user/current_document.php?id=6465`.
4. Cox, I. J., Miller, M. L., Bloom, J. A., Fridrich, J., and Kalker, T. (2007). *Digital Watermarking and Steganography*. Morgan Kaufmann.
5. Hartung, F. and Kutter, M. (1999). Multimedia watermarking techniques. In *Proceedings of the IEEE, Special Issue on Protection of Multimedia Content*, volume 87, pages 1079–1107.
6. ISO/IEC 14496-2 (2004). Information technology – coding of audio-visual objects, Part 2: Visual.
7. ITU-T H.264 (2007). Advanced video coding for generic audiovisual services. `http://www.itu.int/rec/T-REC-H.264-200711-I/en`.
8. Lin, E. T. and Delp, E. J. (2004). Temporal synchronization in video watermarking. *IEEE Transactions on Signal Processing*, 52(10):3007–3022.
9. Marpe, D., Schwarz, H., and Wiegand, T. (2003). Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):620–636.
10. Sinha, R. K., Machado, F. S., and Sellman, C. (2010). Don't Think Twice, It's All Right: Music Piracy and Pricing in a DRM-Free Environment. *Journal of Marketing*, 74(2):40–54.
11. Stütz, T., Autrusseau, F., and Uhl, A. (2013). Inter-frame H.264/CAVLC structure-preserving substitution watermarking. Technical Report 2013–02, Deptartment of Computer Sciences, University of Salzburg, Salzburg, Austria. Available at `http://www.cosy.sbg.ac.at/research/tr.html`.
12. Stütz, T., Autrusseau, F., and Uhl, A. (2014). Non-blind structure-preserving substitution watermarking of H.264/CAVLC inter-frames. *IEEE Transactions on Multimedia*. to appear.
13. Zou, D. and Bloom, J. (2010). H.264 stream replacement watermarking with CABAC encoding. In *Proceedings of the IEEE International Conference on Multimedia and Expo, ICME '10*, pages 117–121, Singapore.