Fast and Blind Detection of Rate-Distortion-Preserving Video Watermarks

Anonymous Author(s)

(a) Watermarked video crop ($QP_w = 27$).

(b) Attacked video crop ($QP_a = 37$).

Figure 1: The proposed method is between 1400 and 5700 faster in detecting watermarks from (a) watermarked videos. The robustness of fast detection is reduced in comparison with slow detection, although it can still detect the watermark after applying (b) content-preserving recompression attacks, even when these reduce the video quality.

ABSTRACT

Forensic watermarking enables the tracing of digital pirates that leak copyright-protected multimedia. To prevent a negative impact on the video quality or bit rate, rate-distortion-preserving watermarking exists, which represents a watermark as compression artifacts. However, this method has two main disadvantages; the detection has a high complexity and it is non-blind. Although a method based on perceptual hashing exists that speeds up the detection of a fallback watermarking system, it decreases its robustness. Therefore, this paper proposes a novel fast detection method that has less impact on the robustness than related work. Our method optimized NS-DCT-DST hashes for rate-distortion-preserving watermarking, which are more robust to content-preserving attacks. Moreover, a blind version is proposed which does not require the original video for hash extraction. As such, the detection is up to 5700 times faster, at the cost of a modest decrease in robustness. In fact, the proposed method shows good robustness to content-preserving recompression attacks when using hashes that are as small as 432 bytes. This is much smaller than comparable performance of related work. In conclusion, this paper enables fast adversary tracing using watermarks that do not impact the video's compression efficiency.

WSDF '22, August 23–26, 2022, Vienna, Austria

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

CCS CONCEPTS

• **Security and privacy** → **Digital rights management**; • **Computing methodologies** → *Image processing*; • **Information systems** → Multimedia streaming.

KEYWORDS

Video Watermarking, Traitor Tracing, Perceptual Hashing, Watermark Detection, Blindness

ACM Reference Format:

Anonymous Author(s). 2022. Fast and Blind Detection of Rate-Distortion-Preserving Video Watermarks. In *Proceedings of Workshop on Digital Forensics (WSDF '22).* ACM, New York, NY, USA, 7 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Forensic watermarking is a popular solution for tracking digital pirates after they leak copyright-protected videos [1, 13]. Two main challenges are imperceptibility and robustness. That is, the watermark should not decrease the video quality nor increase its bit rate. Additionally, the watermark should resist attacks that attempt to delete the watermark. Typically, there is a trade-off between imperceptibility and robustness: a watermark that is more perceptible is harder to delete, whereas an invisible one may be easily deleted by simple signal-processing attacks such as recompression.

Although most robust watermarking methods claim to be imperceptible, they objectively decrease the video's quality. An exception to this is the rate-distortion-preserving watermarking method by Mareen *et al.*, which embeds a robust watermark without affecting the video's quality and bit rate [11]. Although this method has a high level of robustness, the imperceptibility comes at the cost

of a high embedding and detection complexity. Additionally, the method is non-blind: the original watermarked videos are required during the detection process. Requiring all watermarked videos for detection is a potential security thread and hence not desirable.

This paper solves the two main disadvantages of the rate-distortion-preserving method by proposing a faster watermark detection method that utilizes perceptual hashing. Note that perceptual hashes or content-based fingerprints were typically designed for other applications though, such as content identification or video copy detection. For example, it can be used to automatically discover if a video found on the internet is present in a certain database, i.e., to automatically find copyright-protected videos. To do this in an efficient way, a small perceptual hash is extracted from each video segment. The perceptual hashes are small in file size such that the database can be queried quickly and efficiently. Moreover, the perceptual hash is robust against content-preserving attacks. This means that the perceptual hash should not change much when exposed to common signal-processing manipulations such as recompression. In contrast, perceptual hashes of two videos with different content should be significantly different from each other. These two features (i.e., fast querying and high robustness) are desired in the watermarking use case as well. Hence, this paper proposes to utilize perceptual hashes to perform fast and robust watermark detection. It has been shown that applying perceptual hashes on watermarks has merit as well [12]. However, the robustness of this fallback system decreased due to using the hashes, and the system is still non-blind.

Our paper investigates fast detection of rate-distortion-preserving watermarking using a perceptual hashing method optimized for our use case. Additionally, the impact on the robustness is minimized, and a blind variant is proposed as well.

In summary, the main contributions of this paper are:

- This paper proposes a perceptual hashing method optimized for rate-distortion-preserving watermarks. The perceptual hashing method is based on similar ideas of the technique by Khelifi and Bouridane, which was made for content identification and authentication purposes [9]. In contrast, the proposed hash extraction method is adapted and optimized for the use case of watermarking. Additionally, a blind version of the perceptual hashing method is proposed.
- This paper proposes a fast watermark detection method for rate-distortion-preserving watermarks that utilizes the proposed perceptual hashes.
- This work demonstrates that the proposed solution is robust against attacks. For example, a small hash of 432 Bytes (B) is resistant against recompression attacks that do not significantly decrease the video's quality, whereas a higher level of robustness is achieved when using larger hashes of approximately 5 kiloByte (kB).
- Additionally, blind detection showcases an acceptable level of robustness for larger hashes of 40 kB.
- When the hashes of the watermarked videos are extracted before watermark detection, the detection is accelerated up to a factor of 5700.

The rest of this paper is organized as follows. First, Section 2 discusses the rate-distortion-preserving watermarking method in more detail, as well as some perceptual hashing methods. Then, Section 3 presents the proposed watermark detection method, which is evaluated in Section 4. Finally, Section 5 concludes this paper.

2 RELATED WORK

2.1 Rate-Distortion-Preserving Watermarking

This section briefly describes rate-distortion-preserving watermarking. A more detailed description can be found in [11].

The watermark is embedded during H.265/HEVC video compression. More specifically, the video encoder is modified such that it variates the quantization parameters (QPs) of the Coding Tree Units (CTUs). Due to this QP variation, different compression artifacts are introduced. Note that these compression artifacts are *not* drift-error artifacts that occur during e.g. packet loss, but rather ordinary artifacts that are always introduced during lossy video compression. Besides from this subtle video encoder modification, the compression process operates as usual. Hence, the rate-distortion performance is preserved.

Every watermarked video contains different compression artifacts because they are each compressed with a different seed for QP variation (i.e., the watermark input). In other words, the collection of compression artifacts is a representation of the watermark, and is used for watermark detection. The detection is done by comparing an uncompressed attacked video to all uncompressed watermarked videos. Although the detection is very robust, it is slow and requires access to all watermarked videos.

In short, the method embeds a watermark consisting of compression artifacts without affecting the rate-distortion performance. Although the system is robust, the detection method has the main disadvantages that it is slow and that all watermarked videos are required during watermark detection (i.e., it is non-blind). These disadvantages are tackled in Section 3.

2.2 Perceptual Hashing

As explained in Section 1, state-of-the-art perceptual hashing methods are typically designed for content identification, i.e., not for watermarking purposes [3–5, 7, 10, 14–16]. This section briefly describes some related perceptual hashing methods for videos. For a more detailed analysis, we refer to recent surveys [7, 10].

The discrete cosine transform (DCT) is a popular transformation in video compression, and also in perceptual hashing. For example, Coskun *et al.* proposed to perform perceptual hashing using the three-dimensional (3D) DCT on the 3D video signal [6]. Esmaeili *et al.* used the two-dimensional (2D) DCT for hashing, and included the temporal domain by using temporally informative representative images (TIRIs) [8]. A TIRI summarizes a short 3D video signal in a 2D image. As such, the TIRI-DCT hash summarizes both temporal and spatial information, and is robust to attacks. More recently, Khelifi *et al.* proposed a perceptual hashing algorithm utilizing both the DCT and discrete sine transform (DST) [9]. In their algorithm, they proposed the normalizing shift (NS): the number of samples that is required for shifting a digital signal so that it matches a certain pattern according to the DCT/DST coefficients. This normalizing shift is a sort of calibration that makes the method

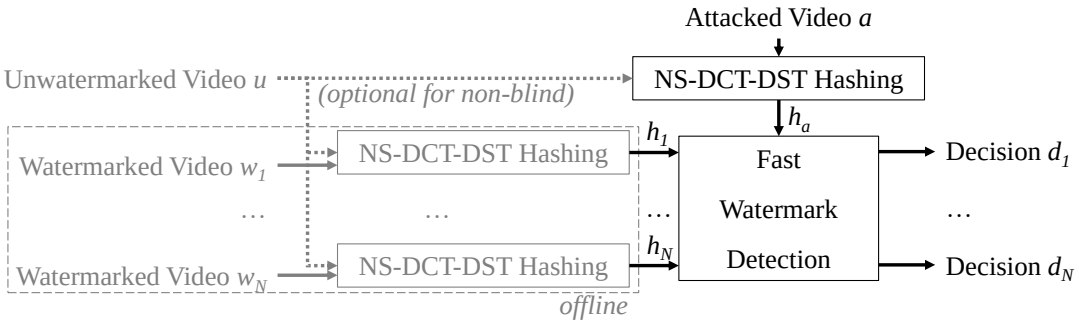

Figure 2: Diagram of the proposed fast watermark detection.

robust to signal processing operations. In this way, their NS-DCT-DST hashing algorithm is robust to attacks such as filtering, noise addition, and compression.

In short, perceptual hashing methods were designed for copy detection and exhibits great properties such as robustness to signal processing attacks, and fast look-up. These characteristics are also desired in a watermark detection system. Therefore, Mareen *et al.* proposed to apply the TIRI-DCT hashing algorithm of Esmaeili *et al.* on their fallback watermarking method in order to speed up the detection. However, the robustness of their system decreased due to using the hashes. Our paper further investigates the application of perceptual hashing for watermarking purposes in Section 3, with less impact on the robustness. That is, we aim to speed up the slow watermark detection method of the rate-distortion-preserving watermarking system of Section 2.1. Additionally, we investigate how to make the watermark detection blind.

3 PROPOSED METHOD

This section proposes a fast and blind watermark detection system, targeted at rate-distortion-preserving watermarked videos. The system utilizes perceptual hashes, as these summarize a video signal in a concise and robust way.

A diagram of the proposed system is shown in Fig. 2. First, a perceptual hash of each watermarked video w_i, $i \in \{1, 2, \ldots, N\}$ (with N the number of watermarked videos) is extracted, as well as from the attacked video a. This is done using the adapted NS-DCT-DST hashing algorithm described in Section 3.1. This can optionally be done in a blind way, i.e., without using the unwatermarked video. The hash extractions of the watermarked videos can happen offline, before watermark detection. Then, these small hashes can be stored instead of the large watermarked videos.

After extracting the small hashes, they are fed to the fast watermark detection system, which is presented in Section 3.2. Finally, the complexity reduction of the fast detection system is discussed in Section 3.3.

3.1 NS-DCT-DST Hashing of Watermarks

This section presents the perceptual hash extraction method. Because of space limitations, we reduced the number of equations and explanations of the hashing method, but instead kindly refer to the hashing method by Khelifi and Bouridane for more detailed motivations [9]. Note that the original hashing method was designed for content identification and authentication, whereas this paper optimized the method for the entirely different use case of watermarking.

Fig. 3 summarizes the proposed perceptual hashing method. The input video v has a resolution $W \times H$ and F frames at R frames per second (fps). First, v is preprocessed by extracting the luminance component and downsampling (with anti-alias filtering) it to R_d fps, resulting in the preprocessed video p_v that contains $K = F \cdot \frac{R_d}{R}$ frames. This video v is either the attacked video a or a watermarked video w_i, as seen in Fig. 2. Optionally for the non-blind version, p_u, the preprocessed version of the unwatermarked video u, can be subtracted from p_v. We call the resulting preprocessed and optionally subtracted signal p. This subtraction increases the robustness by removing the underlying video-content signal, i.e., it enables us to hash only the watermark signal. However, requiring the uncompressed video u makes the method non-blind, which may be undesired in certain applications. Therefore, this step is optional and we evaluate both blind and non-blind versions in Section 4.2.

After the preprocessing, we calculate the differential luminance block means (DLBMs) in the horizontal, vertical, and temporal directions. That is, p is divided in $J \times I$ blocks of size $\left\lfloor \frac{W}{J} \right\rfloor \times \left\lfloor \frac{H}{I} \right\rfloor$ pixels, and the mean luminance value is calculated for each block. This results in the 3D array M of block means. Then, the difference between each two blocks is calculated in the horizontal, vertical and temporal direction, resulting in the 3D arrays D_H, D_V, and D_T, respectively. This is done in a circular way, as presented in (1) for D_H, and analogously for D_V and D_T. Note that the used notations index the array vertically first (i), followed by a horizontal (j) and temporal (k) indexation. We omitted the equations for D_V and D_T because of the limited space and their high redundancy. As such, the 3D arrays D_H, D_V, and D_T have a size of $I \times J \times K$.

$$D_H(i,j,k) = \begin{cases} M(i,j+1,k) - M(i,j,k) & \text{if } j < J-1 \\ M(i,0,k) - M(i,j,k) & \text{if } j = J-1 \end{cases} \quad (1)$$

After calculating the DLBMs, the 3D arrays D_H, D_V, and D_T are transformed into 2D arrays C_H, C_V, and C_T of calibrated signals containing normalizing shifts (NS). The NS is the number of samples that is required for shifting the signal so that it matches a certain pattern according to certain DCT/DST coefficients. For example, we traverse the 3D array D_H horizontally, resulting in a 1D signal $x(n)$ with $n = 0, 1, \ldots, L-1$. On this signal, a normalization

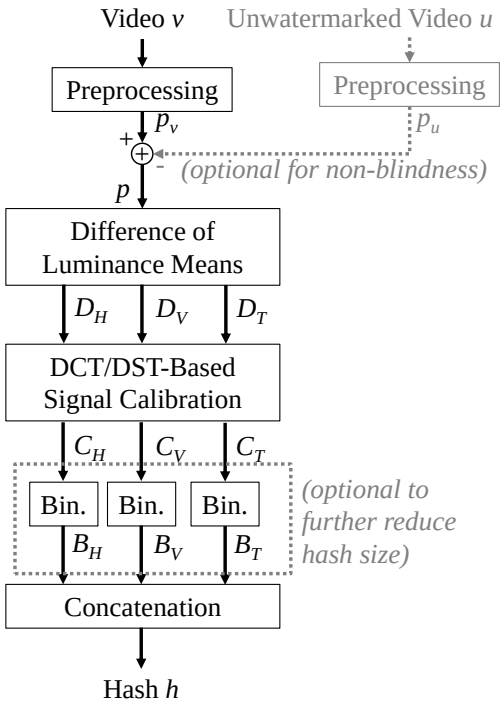

Figure 3: Diagram of the proposed NS-DCT-DST hashing.

technique is applied that provides the same output signal even if the input signal has undergone a circular translation. Without going into details, this is found for the horizontally traversed D_H using (2), and analogously for the vertically and temporally traversed arrays. In this equation, α is a predefined reference angle for normalization, set between 0 and 2π for normalization. Additionally, $DST_{\{D_{H,i,k}\}}(1)$ is the first DST coefficient, and $DCT_{\{D_{H,i,k}\}}(2)$ is the second DCT coefficient of the horizontally-traversed sequence $D_{H,i,k}$. As a result, the arrays C_H, C_V, and C_T have a size of $I \times K$, $J \times K$, and $I \times J$, respectively.

$$C_H(i,k) = \left\lfloor \frac{L \cdot \arctan\left(\frac{\mathrm{DST}_{\{D_{H,i,k}\}}(1)}{\mathrm{DCT}_{\{D_{H,i,k}\}}(2)}\right) + J\alpha}{2\pi} \right\rfloor \mod J \quad (2)$$

The signal calibration using normalizing shifts has two advantages. First, it reduces the size of a 3D DLBM array to a 2D array, by extracting a single NS feature from each DLBM signal. Second, it increases the robustness because it is based on the shift-invariance property. That is, the resulting hash is invariant to shifts in the input signal. For a more detailed explanation of the advantages of using the NS, we kindly refer to the work by Khelifi and Bouridane [9].

Next, the 2D arrays C_H, C_V, and C_T are optionally binarized into B_H, B_V, and B_T. That is, each 8-bit integer value of the arrays is converted to a single bit: 1 if the value is larger than the mean of the 2D array, and 0 if it is smaller or equal than the mean. The binarization idea is similar as those proposed in other hashing algorithms, such as in TIRI-DCT [8]. As such, the hash size optionally is further reduced with a factor of 8. We leave this step optional, as it may reduce the robustness of the system, as evaluated in Section 4.2.

Finally, the three 2D arrays are flattened and concatenated to form the output hash h with size $IK + JK + IJ$. This hash is significantly smaller than the input video v, and can be used for fast watermark detection, described in Section 3.2.

3.2 Fast Watermark Detection using Perceptual Hashes

The proposed detection differs from the slow detection of the rate-distortion-preserving system [11] in the hash comparison. The short hash comparison is much faster than pixel-by-pixel comparisons of uncompressed videos.

As shown in Fig. 2, the hash from an attacked video a is given as input of watermark detection, along with the hashes of all watermarked videos w_i. First, the last step of the hashing algorithm of Section 3.1 is reversed: all hashes are de-concatenated into the (binarized) 2D arrays B_H, B_V, and B_T. Then, the distance between the hash of the attacked video is calculated against all hashes from the watermarked video, using the identification measure proposed by Khelifi and Bouridane [9]. This is done by considering the distance of the three (binarized) arrays individually, and finally adding them to one another. The distance d_H between two horizontal 2D arrays B_H^1 and B_H^2 is defined in (3), and it is performed analogously for the vertical and temporal arrays, resulting in d_V and d_T. Because the normalizing shifts are determined in a forward direction only, the equation additionally considers the case where a change in the video splits the normalizing shift to the beginning of the sequence as it exceeds the boundary. Therefore, the equation additionally considers the subtractions between $B_H^1(i,k)$ and $B_H^2(i,k)$ with J subtracted or added to it. Then, the distance d is equal to $d = d_H + d_V + d_T$.

$$d_H(B_H^1, B_H^2) = \sum_{i=0}^{I} \sum_{k=0}^{K} \min\{|B_H^1(i,k) - B_H^2(i,k)|, \\ |B_H^1(i,k) - B_H^2(i,k) - J|, \\ |B_H^1(i,k) - B_H^2(i,k) + J|\} \quad (3)$$

After calculating the distance d_i, $i \in \{1, 2, \ldots, N\}$ between the attacked video and each of the watermarked videos, the rest of the process is equal to that of the slow detection system of Section 2.1. These steps are shortly discussed here, and we kindly refer the reader to the work by Mareen *et al.* for a more detailed explanation [11]. For each distance value d_i, outlier detection is performed by calculating the z-score z_i, as defined in (4). The z-score z_i indicates the number of standard deviations σ_A that a distance value d_i differs from the mean μ_A of the distribution of distance values corresponding to watermarks that are absent in the video. That is, we assume that all watermarks are absent, except for the one corresponding to the smallest distance value.

$$z_i = \frac{d_i - \mu_A}{\sigma_A} \quad (4)$$

Then, each z-score is compared to a threshold T in order to get a decision d_i of the corresponding watermark's presence or absence. The threshold T is calculated for a certain false-positive

(FP) probability P_{fp} using the Gaussian method, as defined in (5). In the equation, erfc^{-1} is the inverse of the complimentary error function. For example, for $P_{\text{fp}} = 10^{-6}$, the corresponding threshold is $T \approx -4.8$.

$$T = -\sqrt{2}\,\text{erfc}^{-1}(2P_{\text{fp}}) \tag{5}$$

3.3 Theoretical Complexity

This section discusses the theoretical complexity of hash extraction and fast watermark detection, in comparison to slow detection. The slow detection method compares every pixel of an attacked video a of size $W \times H \times F$ to all N watermarked videos. It has a complexity of $\Theta(N \cdot F \cdot W \cdot H \cdot B)$, in which B is the number of bits per pixel.

The computational complexity of the hash extraction is analyzed in three steps. First, the complexity of preprocessing is $\Theta(F \cdot W \cdot H)$. Then, the DLBMs calculation is performed in $\Theta(W \cdot H \cdot K + 3 \cdot I \cdot J \cdot K)$. Lastly, the signal calibration has a complexity of $\Theta(I \cdot J \cdot K)$. Although the hash extraction has to be performed for every watermarked video, this can be done prior to watermark detection.

The fast detection method compares hashes of size $(IK + JK + IJ) \cdot B_H$, where B_H is the number of bits per element in the optionally binarized arrays of calibrated signals. Without binarization, these are 8-bit integers, whereas each element is a single bit when binarization is performed. As such, the complexity of comparing N hashes is $\Theta(N \cdot (IK + JK + IJ) \cdot B_H))$. Since I, J, K, and possibly B_H are significantly lower than W, H, F, and B, respectively, the proposed method is much faster than the slow variant. Although the proposed method still scales linearly in the number of watermarked videos N, it is several orders of magnitude faster than the slow detection method, such that it is practical to apply in realistic scenarios.

As an example, consider a video segment with resolution $W \times H = 1920 \times 1080$ pixels, $F = 500$ frames at 50 fps, and $B = 8$ bits per luminance pixel value. Assume the following parameters are used for perceptual hashing: $I = J = 128, R = 10, K = 100, B_H = 1$. This results in hashes of size $L = (IK + JK + IJ) \cdot B_H = 41,984$ bits $=$ 5248 Bytes. As a result, for these parameters, we get a theoretical speed-up of $\frac{N \cdot F \cdot W \cdot H \cdot B}{N \cdot L} \approx 197,560$. Note that the actual speed-up may differ in practice. This is measured in Section 4.3.

4 EXPERIMENTAL RESULTS

4.1 Experimental Setup

To evaluate the performance of the proposed fast detection method, we used five test sequences with resolution 1920×1080: *BQTerrace, Cactus, Kimono1, ParkJoy, and ParkScene* [2]. These are 10 seconds long and contain between 240 and 600 frames (i.e., they have a framerate between 24 and 60 fps). The watermarking was done using the rate-distortion-preserving algorithm [11], implemented using the HEVC reference Model (HM) version 16.15. Each test sequence was compressed using 4 default QPs (22, 27, 32, and 37), further denoted as QP_w. These default QPs were varied during watermarking with 20 different seeds or watermarked IDs. Thus, every test sequence was converted into $4 \times 20 = 80$ watermarked videos.

The perceptual hash extraction and distance algorithms were implemented in MATLAB. Several parameters were swept to explore hashes with various sizes: the number of blocks in the horizontal (I) and vertical (J) direction was set to 16, 32, 64, and 128, the resample rate R_d was set to 5 and 10, the binarization process was either applied or not applied, and both the non-blind and blind variant were executed. Note that only a few selected configurations are reported in this paper, though, because of space limitations.

The robustness experiments were performed using recompression attacks with 6 QPs (22, 27, 32, 37, 42, and 47), further denoted as QP_a. Note that a recompression with $\text{QP}_a = 47$ results in a very low-quality video. Watermark detection was performed for each attacked video, using the slow watermark detection [11], the proposed fast watermark detection using NS-DCT-DST hashes, as well as the proposed fast detection method using TIRI-DCT hashes from the state of the art [12]. The FP probability was set to $P_{\text{fp}} = 10^{-6}$, corresponding to a threshold $T \approx -4.7534$. Grouping all detection results into a single value is done by using the false-negative rate (FNR), which is calculated as in (6).

$$\text{FNR} = \frac{\#\text{FN Detections}}{\text{Total Number of Detections}} \tag{6}$$

4.2 Robustness

Table 1 shows selected robustness results of the experiments. More specifically, the table shows the results for the slow configuration of the rate-distortion-preserving detection method [11], as well as for five configurations using the proposed fast detection method that utilizes the proposed NS-DCT-DST hashes, and for the state-of-the-art TIRI-DCT hashing algorithm. The parameters I and J, the optional binarization and blindness were varied, while the parameter R_d was fixed to 10 fps because it led to the best trade-off between hash size and robustness. Additionally, results from a single configuration using the TIRI-DCT method used in the state of the art is presented [8, 12].

The first slow-detection configuration in Table 1 shows the results from the state-of-the-art rate-distortion-preserving detection method [11]. As one can see, most reported FNR values are 0%; the FNR is non-zero only for high QP_a values in combination with low QP_w values. In other words, the slow detection method has a very high level of robustness. However, it is slow and requires all uncompressed watermarked files. For the test sequences, this is between 746 and 1869 MB, which is over 10,000 times higher than hash sizes of the proposed fast detection system.

Fast Config. 1 in Table 1 shows the robustness results for the proposed fast detection with $I = J = 16$ and without binarization, resulting in perceptual hashes of approximately 3 kB. Using this configuration, the FNR is zero as long as the recompression attack does not significantly decrease the video quality, i.e., $\text{QP}_a \leq \text{QP}_w$.

Fast Config. 2 in Table 1 is the same as Fast Config. 1, yet *with* binarization. Although the hash sizes are 8 times smaller, the robustness performance is comparable to Config. 1. That is, the FNR is zero as long as $\text{QP}_a \leq \text{QP}_w$.

Fast Config. 3 in Table 1 is the same as Fast Config. 2, yet with $I = J = 128$ instead of $I = J = 16$, and thus has a hash size of approx. 5 kB. As a result of the larger hash size, the robustness increases significantly: this configuration can resist stronger recompression attacks. For example, a watermarked video with $\text{QP}_w = 27$ (see Fig. 1a) that is attacked with $\text{QP}_a = 37$ (see Fig. 1b) now has

Table 1: Robustness results for recompression attacks.

		False-Negative Rate (%)					
QP_w	QP_a =	22	27	32	37	42	47
Slow detection [11]: 746 - 1869 MB							
22		0	0	0	0	2	87
27		0	0	0	0	0	30
32		0	0	0	0	0	0
37		0	0	0	0	0	0
Fast Config. 1: $I = J = 16$, no bin.: 3456 B							
22		0	6	81	99	100	100
27		0	0	1	75	93	100
32		0	0	0	7	89	98
37		0	0	0	0	17	92
Fast Config. 2: $I = J = 16$, bin.: 432 B							
22		0	19	87	100	100	100
27		0	0	7	78	100	100
32		0	0	0	18	96	100
37		0	0	0	0	40	97
Fast Config. 3: $I = J = 128$, bin.: 5248 B							
22		0	0	0	67	98	100
27		0	0	0	0	74	100
32		0	0	0	0	1	88
37		0	0	0	0	0	6
Fast Config. 4: $I = J = 128$, no bin., blind: 41,984 B							
22		0	7	60	99	100	100
27		0	0	0	40	96	100
32		0	0	0	0	32	98
37		0	0	0	0	0	30
TIRI-DCT [8, 12] Config. 1: $b_t = 16$, $d = 0$: 570 B							
22		49	96	98	100	100	100
27		0	19	78	100	99	100
32		0	0	0	50	96	99
37		0	0	0	2	28	93
TIRI-DCT [8, 12] Config. 2: $b_t = 8$, $d = 2$: 5016 B							
22		0	34	90	98	100	100
27		0	0	0	56	97	100
32		0	0	0	0	45	96
37		0	0	0	0	0	24

a 0% FNR. Thus, this demonstrates that binarization is a powerful feature to decrease the hash size and/or increase the robustness.

Fast Config. 4 in Table 1 is the *blind* version of Fast Config. 3, *without* binarization. No binarization was performed because blind hashing with binarization performed significantly worse. This configuration results in hash sizes of approx. 42 kB. The robustness is similar to the that of Fast Config. 3, of which the hash size was approx. 100 times smaller. However, these results demonstrate the potential of blind hashing: we demonstrate that it is not essential to subtract the original video before extracting hashes.

Finally, the last two configurations in Table 1 show the results for watermark detection using TIRI-DCT hashes of 570 B and 20,026 B, respectively [8, 12]. TIRI-DCT Config. 1 is approximately as large as the size of Fast Config. 2, yet it performs worse. Additionally, TIRI-DCT Config. 2 is approximately as large as the NS-DCT-DST hash of Fast Config. 3, yet its robustness performance is significantly worse. For example, consider a watermarked video with $QP_w = 27$ (as shown in Fig. 1a)that is attacked with $QP_a = 37$ (as shown in Fig. 1b). For Fast Config. 3, this results in a 0% FNR, whereas it has a 56% FNR for TIRI-DCT Config. 2. This demonstrates that the proposed method outperforms the state of the art.

In conclusion, we demonstrated that the proposed fast detection method using perceptual hashes has a modest level of robustness when using very small hashes, and improves in performance when using larger hashes. Additionally, we showed that binarization is a good strategy to reduce the hash size / increase the robustness. However, binarization had to be disabled during blind hashing to get a basic level of robustness. Future work should investigate how to improve the performance of the blind hashing algorithm, for example by inspecting which features are most robust and only selecting those in the hash extraction process.

4.3 Time measurements

This subsection reports measurements related to the complexity of the proposed hashing and fast detection methods. For the time measurements, the *ParkScene* sequence containing 240 frames was used (of which crops are shown in Fig. 1). The experiments were performed using the high-level programming language MATLAB, and hence the time results could be significantly improved using a low-level language such as C++. In order to not take the efficiency of the programming language into account, we report the measurements of the perceptual hashing and fast detection method as a factor of the slow detection method.

As a baseline, it takes the slow detection method approx. 24.3 seconds to compare two uncompressed YUV files. To extract the perceptual hash, it takes approximately 1.37 times longer than comparing two uncompressed YUV files. This value is averaged over all test configurations, which all are approximately equally fast. It should be stressed that extracting the perceptual hashes can be done offline, prior to watermark detection. As such, only the small perceptual hashes need to be saved and be given as input to the fast watermark detection system of Fig. 2. As such, the speed of the hash extraction method is of less importance.

Comparing two perceptual hashes is between approx. 1400 and 5700 times faster than comparing two uncompressed YUV files, for the swept parameters and test configurations given in Section 4.1. These observed speed-ups are much smaller than the theoretical speed-ups calculated in Section 3.3. This is partly due to constant factors that are ignored in the asymptotic complexity notation, and due to MATLAB being very inefficient. Nonetheless, this speed-up highlights the main advantage of the proposed detection method: it is several orders of magnitude faster than the slow state-of-the-art watermark detection method.

5 CONCLUSION

Rate-distortion-preserving watermark detection is slow and requires all watermarked videos. As high complexity and non-blindness are undesirable features in practical multimedia security applications, this paper proposed a fast detection method that uses perceptual hashes. Additionally, a blind hash extraction variant is proposed. The proposed hashing is based on the normalizing shifts of the DCT and DST.

The experiments confirm that the proposed detection is robust. That is, when using very short hashes of approximately 0.4 kB, the fast detection method is robust against recompression attacks that do not significantly reduce the quality and outperforms the state of the art. A higher level of robustness can be obtained by increasing the hash size to, e.g., approx. 5 kB. Moreover, the blind version is robust to recompression attacks that do not significantly reduce the quality when hashes of approx. 40 KB are used. Reducing the size of the blind hashes as well as increasing its robustness are future work opportunities.

In conclusion, this paper tackled the main disadvantages of the rate-distortion-preserving watermarking system. The proposed detection is much faster and does not require access to the large watermarked videos, but rather only to short precomputed perceptual hashes. As such, one can perform watermarking without impacting the video's compression efficiency, and without worrying about a high complexity of watermark detection.

REFERENCES

[1] M. Asikuzzaman and M. R. Pickering. 2018. An Overview of Digital Video Watermarking. *IEEE Trans. Circuits Syst. Video Technol.* 28, 9 (2018), 2131–2153. https://doi.org/10.1109/TCSVT.2017.2712162

[2] F. Bossen. 2013. *Common test conditions and software reference configurations.* Technical Report JCTVC-L1100. ITU-T JCT-VC.

[3] Uwe Breidenbach, Martin Steinebach, and Huajian Liu. 2020. Privacy-Enhanced Robust Image Hashing with Bloom Filters. In *Proceedings of the 15th International Conference on Availability, Reliability and Security* (Virtual Event, Ireland) *(ARES '20)*. Association for Computing Machinery, New York, NY, USA, Article 56, 10 pages. https://doi.org/10.1145/3407023.3409212

[4] C. Chou, H. Chen, Y. Chen, C. Ho, and S. Lee. 2013. Near-duplicate video retrieval and localization using pattern set based dynamic programming. In *IEEE Int. Conf. Multimedia Expo (ICME)*. 1–6. https://doi.org/10.1109/ICME.2013.6607616

[5] R. Cook. 2011. An efficient, robust video fingerprinting system. In *IEEE Int. Conf. Multimedia Expo (ICME)*. 1–6. https://doi.org/10.1109/ICME.2011.6012135

[6] B. Coskun, B. Sankur, and N. Memon. 2006. Spatio–Temporal Transform Based Video Hashing. *IEEE Trans. Multimedia* 8, 6 (2006), 1190–1208. https://doi.org/10.1109/TMM.2006.884614

[7] K. Du, A. T.S. Ho, and R. Cong. 2020. Perceptual hashing for image authentication: A survey. *Signal Process.: Image Commun.* 81 (2020), 115713. https://doi.org/10.1016/j.image.2019.115713

[8] M. M. Esmaeili, M. Fatourechi, and R. K. Ward. 2011. A Robust and Fast Video Copy Detection System Using Content-Based Fingerprinting. *IEEE Trans. Inf. Forensics Security* 6, 1 (2011), 213–226. https://doi.org/10.1109/TIFS.2010.2097593

[9] F. Khelifi and A. Bouridane. 2019. Perceptual Video Hashing for Content Identification and Authentication. *IEEE Trans. Circuits Syst. Video Technol.* 29, 1 (2019), 50–67. https://doi.org/10.1109/TCSVT.2017.2776159

[10] J. Law-To, L. Chen, A. Joly, I. Laptev, O. Buisson, V. Gouet-Brunet, N. Boujemaa, and F. Stentiford. 2007. Video copy detection: a comparative study. In *Proc. Int. Conf. Image Video Retrieval*. 371–378.

[11] H. Mareen, M. Courteaux, J. De Praeter, M. Asikuzzaman, G. Van Wallendael, and P. Lambert. 2020. Rate-Distortion-Preserving Forensic Watermarking Using Quantization Parameter Variation. *IEEE Access* 8 (2020), 63700–63709.

[12] H. Mareen, N. Van Kets, P. Lambert, and G. Van Wallendael. 2021. Fast Fallback Watermark Detection Using Perceptual Hashes. *Electronics* 10, 10 (2021). https://doi.org/10.3390/electronics10101155

[13] David Megías, Minoru Kuribayashi, Andrea Rosales, and Wojciech Mazurczyk. 2021. DISSIMILAR: Towards Fake News Detection Using Information Hiding, Signal Processing and Machine Learning. In *The 16th International Conference on Availability, Reliability and Security* (Vienna, Austria) *(ARES 2021)*. Association for Computing Machinery, New York, NY, USA, Article 66, 9 pages. https://doi.org/10.1145/3465481.3470088

[14] Martin Steinebach, Sebastian Lutz, and Huajian Liu. 2019. Privacy and Robust Hashes. In *Proceedings of the 14th International Conference on Availability, Reliability and Security* (Canterbury, CA, United Kingdom) *(ARES '19)*. Association for Computing Machinery, New York, NY, USA, Article 70, 8 pages. https://doi.org/10.1145/3339252.3340105

[15] N. Vretos, N. Nikolaidis, and I. Pitas. 2009. A perceptual hashing algorithm using latent dirichlet allocation. In *IEEE Int. Conf. Multimedia Expo (ICME)*. 362–365. https://doi.org/10.1109/ICME.2009.5202509

[16] J. Zhou and O. C. Au. 2011. Security evaluation of a perceptual image hashing scheme based on virtual watermark detection. In *IEEE Int. Conf. Multimedia Expo (ICME)*. 1–6. https://doi.org/10.1109/ICME.2011.6011882