Perceptual Video Hashing based on 3D SPIHT Coding of 3D DWT Coefficients

Ravi Kumar Vadapalli and P. K. Bora

Department of Electronics and Communication Engineering, Indian Institute of Technology Guwahati, Guwahati 781039, India, Email: {vadapalli, prabin}@iitg.ernet.in

Abstract—Perceptual video hashing captures the perceptual content of a video sequence into a short binary string called the hash. The interest for the perceptual video hashing arises from its applications, namely video authentication and indexing video databases. The hash should uniquely characterize the video, independent of the perceptually non-significant distortions of its content. We propose a robust video hashing algorithm based on the the 3D Set Partitioning In Hierarchical Trees (SPIHT) coding of the Three Dimensional Discrete Wavelet Transform (3D DWT) coefficients of the video. The motivation for selecting the 3D SPIHT coding of the 3D DWT coefficients is that the 3D SPIHT algorithm encodes video progressively. So, the DWT partitioning information obtained in the first few passes of the scalable encoder gives us the most important features of the video. With each sorting pass, the hashing algorithm forms binary maps of DWT partitions (significant and non-significant 3D DWT coefficients and 3D DWT trees) obtained in that pass and concatenates these maps to obtain the hash. Simulation results show the superior performance of the proposed algorithm.

I. INTRODUCTION

Along with the access to a wide range of image and video data, the user today is equipped with powerful tools to manipulate digital imagery and videos. To restore credibility to digital videos, there is a need for techniques which would tolerate the trivial modifications but detect malicious content changes. Two techniques are being reported in the literature to authenticate videos, namely watermarking [1] and digital signature [2] based methods. In watermarking some data (called the watermark) is embedded into the video. The watermark can be extracted later by the owner of the video to check the integrity and authenticity of the video. In digital signature based methods, a short binary string (called the hash) extracted from the video accompanies the video. To check the authenticity and integrity of the video, the hash is extracted from it and cross-checked with the accompanying hash. To prevent attackers from modifying the hash itself, it is encrypted by a secret key. Unless the hash captures the perceptual content of videos the hash extracted from the received video would be entirely different from that extracted at the source end. This is because most of the videos undergo operations like compression, frame rate changes, filtering, spatial geometrical transformations (like shifting, scaling and shearing, bending etc., ) before reaching the intended user. So, cryptographic hash functions cannot be used here, as the hash generated by them would be sensitive to even a single bit change in video. To tolerate all these incidental modifications but detect malicious manipulations the hashing function should capture visual content of the video rather than the video data. Such a technique is known as Perceptual Video Hashing, Soft Hashing or Robust Video Hashing.

In addition to video authentication perceptual video hashing finds extensive applications in indexing videos in a database. With increasing popularity of digital photography, more and more videos are created and stored every day. One cannot say if a video already exists in the database or not without exhaustively searching through all the entries. Further, two visually identical videos may have completely different digital representations (for example, the original and compressed ones) which would further complicate the search. This problem demands for a perceptual hash function which would precisely capture the visual content of videos and can be used for fast and more efficient search.

This paper describes a new method for perceptual video hashing based on robust feature extraction from the 3D SPIHT coding of 3D DWT coefficients. The rest of the paper is organized as follows. In section II, Perceptual video hash function and its desirable properties are mathematically defined. Section III gives a short review of existing video hashing algorithms. Section IV describes the proposed method. Section V gives the experimental setup describing the attacks employed. Section VI presents results and a discussion on them. Conclusions are drawn in section VII.

II. PERCEPTUAL VIDEO HASH FUNCTION AND ITS DESIRED PROPERTIES

The desirable properties of a perceptual image hash function are mathematically defined by Monga and Evans [3]. We extend them to define the desirable properties of video hash functions. Let $ν$ and $κ$ denote a set of videos with finite cardinality and space of secret keys respectively. Let $V \in ν$, $k \in κ$ be the input video, key to the perceptual hash function $H(\ldots)$ respectively and $h = H(V, k)$ be the hash extracted by the hash function. Let $V_{ident} \in ν$ and $V_{diff} \in ν$ denote perceptually similar and distinct videos respectively. Let $θ_1$, $θ_2$ satisfy $0 < θ_1 < θ_2 < 1$. The properties of the perceptual hash function are:

1) Perceptual Robustness:

$$P(H(V, K) = H(V_{ident}, K)) \geq 1 - θ_1$$

2) Fault Resilience:

$$P(H(V, K) \neq H(V_{diff}, K)) \geq 1 - θ_2$$

3) Security of hash:

$$P(H(V, K) = h) \approx (1/2^q), \forall h \in \{0, 1\}^q$$

The property 1 states that for a given key $K$, all perceptually identical videos should give a same hash with high probability. This is required to achieve robustness.
against non-significant distortions. The property 2 states that for a given key $K$, all perceptually distinct videos should give different hashes with high probability. This property is required to achieve fragility against malicious manipulations. The property 2 states that it should be infeasible for an attacker to find a video with the same hash as that of a given video. The property 3 adds security to the hash function; it states that as the key is varied the output hash value must be approximately uniformly distributed among all possible $q$ bit outputs. Along with these properties, it is desirable to have a smaller hash and a faster hashing algorithm as it would reduce the memory and computational requirements.

III. RELATED WORKS

In one of the first attempts, R. Du and J. Fridrich [4] developed a hashing technique for the MPEG-2 video. They proposed calculating the hash for each frame independently from their quantized DCT coefficients and then embedding the hash along with the frame index into nearest 'B' frames. This technique is fragile to frame rate alterations and requires a robust watermarking algorithm to embed the hash. Hamon et al [5] developed a simple technique for hashing minimally changing videos like the news reading scene. Since the frames are similar to each other, the temporal average of a group of frames is taken and Gaussian filtered to remove detailed information and noise. The resulting image represents the entire GOF whose histogram is then used in generation of the hash. But this technique is limited to minimally changing videos. Lefevvre et al [6] extended their image hashing technique based on the Radon transform to video hashing. Their idea is to select some key frames in the video and to apply their image hashing technique to the key frames hoping that the key frames would tolerate acceptable modifications. The drawback of this (and any key frame based) approach is that an attacker can modify the video in such a way that key frames are not affected but the other frames are affected. Sun et al [7] proposed a scheme which would tolerate quality, temporal and spatial scalability. To achieve robustness against changes in the quantization step size, the hash is generated by keeping a large step size and to have robustness against the frame rate changes, the hash generated from a frame is inserted into its neighboring frames so that, even when a frame is missing its hash will be there. To maintain the robustness against spatial resolution changes, the hash is calculated from spatially down sampled frames. Since the hash size is large, the authors proposed embedding it into the video instead of transmitting it separately. Since watermarking is employed to carry the hash, the watermarking should be robust against scalability to allow hash extraction. Coskun et al [8] developed a 3-D transform domain video hashing based on the 3D DCT of video. They collected lowest AC coefficients of the 3D DCT and median thresholded them to generate the hash. The idea is that the low frequency components remain almost same for minor modifications and at same time remain sensitive to malicious modifications as they contain predominant energy. Since the applied transform is spatio-temporal it will satisfy the above property in both the spatial and temporal domains and thus able to authenticate the video. To add security to above algorithm, the authors proposed using a random basis set (RBT) instead of the DCT as the transform. Coskun's approach [8] has got clear distinction from other approaches by treating video as a single entity instead of treating it as a group of entities (frames). In fact, as explained in the next section, this treatment is necessary to have a scalable hash.

IV. PROPOSED HASHING METHOD

In scalable video coding, [9] a single video code stream can be decoded at different distortion rates and spatio-temporal resolutions. To support scalable video coding as one of the non-significant distortions, we aim at developing a hashing algorithm which would extract a scalable hash. We would like the hash extracted from a decoded video at different rates or resolutions to indicate that it is authentic and not to confuse from being tampered. At the same time, it should indicate any malicious modifications.

The 3D DWT offers a multi-resolution representation of video which is desirable in scalable coding of video. It promises to be the chosen transformation for a scalable video codec. To achieve high coding gains along with scalable coding, specialized progressive coding algorithms like the Embedded image coding using Zero trees of Wavelet coefficients (EZW) [10] and SPIHT [11], [12] (which exploit the relationship among 3D DWT coefficients at different scales) have to employed. So the natural choice for a scalable codec would be to use these progressive coding algorithms to code the 3D DWT coefficients. Since we aim at generating a scalable hash, the proposed algorithm mimics the steps of a scalable codec and uses the partitioning information as robust features to calculate the hash. It follows the 3D SPIHT coding of 3D DWT coefficients and uses the partitioning information of the video from the 3D SPIHT algorithm as the hash. The 3D SPIHT coding is selected over the EZW because of its better partitioning capabilities which yield robust features. The block diagram of hash generation is shown in Figure 1.

A. Hash Generation

1) Video Normalization: Two video sequences of the same perceptual content may be represented in different color spaces and different spatial and temporal resolutions. To have a comparison of hashes obtained from them, they need to be converted to a standard color space and to standard dimensions before calculating their hashes. We use the luminance components of the video as its standard representation. Before calculating the hash we obtain the luminance component of given video and then convert it into a standard dimension video signal by spatial and temporal smoothing followed by sub-sampling.

2) 3D DWT: The mean gray level of the normalized video is subtracted from it before applying the 3D DWT so that the hash extracted will be robust to brightness changes (which do not alter the perceptual quality of the video) are nullified. An $l$-level 3D DWT is applied on the mean subtracted normalized video. The lifting approach [13] has been followed to implement the 3D DWT because of the advantages it offers over the filter bank implementation.
3) Robust Feature Extraction from 3D SPIHT Coding:
It has been observed that if a DWT coefficient at a given scale has a magnitude less than a certain threshold then in most cases all of its children (DWT coefficients obtained from same spatio temporal location in the video but at smaller scale) will have magnitudes less than the same threshold [11]. The 3D SPIHT algorithm is a clever way of partitioning the 3D DWT coefficients into such sets and coding these sets, so that number of code bits required per coefficient is less. In the first sorting pass, the threshold is kept high and with each pass the threshold is halved. At each pass, the sets which satisfy above relation will be placed in the List of Insignificant Sets (LIS) and the sets which violate the above relation will be placed in the List of Insignificant Pixels (LIP), and these lists are then coded. In actual implementation of the SPIHT, the LIS is again divided into two types namely the LIS of type-A and LIS of type-B. We use the notations LISA and LISB to denote them throughout this paper. Yang et al [14] observed that the LISA, LISB and LIP of coarsest scale sub-bands (4 in case of image) obtained in a 2D SPIHT coding algorithm of images in the first three sorting passes can be used as features for image hashing. But they have not made any explicit attempt to maintain the robustness of these features under attacks which add non-significant distortions. We have observed that the LISA, LISB and LIP of the approximation sub-band (but not all the coarsest scale sub-bands) obtained in the first few sorting passes of the 3D SPIHT algorithm are robust features. The proposed algorithm uses binary maps to represent LISA, LISB and LIP of the approximation sub-band obtained in the first few sorting passes. Suppose in the $p^{th}$ sorting pass LISA has the pixel at location $(i,j,k)$ (of the approximation sub-band), then the binary map correspondence to $p^{th}$ pass of LISA would have 1 at $(i,j,k)$. Similarly 0 is used to represent the absence of a pixel in a list. The binary maps are then concatenated to get the hash. The 3D DWT coefficients of two perceptually similar videos may have different absolute maximum values, which may lead to different thresholds under the first sorting pass of 3D SPIHT. To have a correct comparison of the features obtained from the 3D SPIHT, maps compared should be obtained from the same thresholds. To allow such a comparison, the threshold $th$ under the first sorting pass is appended to the hash and the binary maps corresponding to two more maps are also included in the hash (the reason for this will be more clear after going through 'hash alignment' in the hash verification section).

4) Hash Encryption: Unless the hash is dependent on a secret key, the perceptual hashing framework is vulnerable to attacks as the attacker can attack the video and modify the hash accordingly [2]. To add the secret key functionality to the algorithm we encrypt the original hash obtained from the previous stage using a secure cryptographic hash algorithm to obtain the final hash. The secret key will be sent to the receiver through a secure channel which the attacker can not tap.

B. Hash verification

The block diagram of the hash verification is as shown in Figure 2. The hash is extracted from a test video following the same procedure described above and the original hash is obtained from the encrypted hash by decrypting it.

![Fig. 2. Block Diagram of the Hash Verification Algorithm](image)

The hash extracted from the test video is compared with the original hash and if they match the video is said to be authentic, Otherwise it is considered as tampered. The hash comparison is of two stages, namely hash alignment and distance calculation.

1) Hash Alignment: Let $th_1$ and $th_2$ denote the thresholds under first sorting pass of the 3D SPIHT coding of the two videos under comparison. They are obtained from the hashes ($hash_1$, $hash_2$) of the respective videos. The hash alignment is as follows:

$$n_1 = \log_2(th_1)$$
$$n_2 = \log_2(th_2)$$

$$\text{if } ((n_1 - n_2 = 1) \text{ or } (n_1 - n_2 = 2))$$

binary maps corresponding to first $(n_1 - n_2)$ sorting passes of $hash_1$ are discarded to align it with $hash_2$

$$\text{elseif } (n_2 - n_1 = 1) \text{ or } (n_2 - n_1 = 2)$$

binary maps corresponding to first $(n_2 - n_1)$ sorting passes of $hash_2$ are discarded to align it with $hash_1$

else

no need of the hash alignment.

In some of the above cases hash alignment discards maps obtained in some of the first passes. So, to have enough maps even after discarding some of them, binary maps corresponding to two more passes are also included in the hash in the hash generation stage.

2) Distance Calculation: We use the Hamming distance between the two hashes $hash_1$ and $hash_2$ after alignment as the distance between them. If the distance is less than a pre-defined threshold, the corresponding videos are perceptually similar and different otherwise.

V. EXPERIMENTAL SETUP

14 standard uncompressed test video sequences namely Akiyo, Antibes, Bike, Cheer, Coast, Container, Football, Foreman, Garden, Mob, Mosaic, News, Stefan and Tempete are considered. Each video is of spatial dimensions $288 \times 352$ with 128 frames and frame rate 15 fps. Their luminance components are obtained and reduced to spatio-temporal dimensions of $64 \times 64 \times 16$ in the normalization stage. The mean gray level of the normalized video is subtracted and then subjected to the 3-level 3D DWT decomposition. We used the Haar wavelet on the temporal domain and the CDF 9/7 wavelet on the spatial domain. The Haar wavelet is chosen for the temporal domain because it can be applied even when the number of frames is less without any boundary problems. The CDF 9/7 is chosen for the spatial domain because of its better energy compaction characteristics and the fact that the scope of the boundary problem in the spatial domain is less [13]. Applying the first level 3D DWT will split the video into eight sub-bands. The approximation sub-band will be split into eight more sub-bands in the next level
of the 3D DWT and so on. The 3-level 3D DWT splits the normalized video of size 64 X 64 X 16 into 21 detail bands and 1 approximation band each of size 8 X 8 X 2. In the experiments, we have considered the lists LIP, LISA and LISB obtained in the first five sorting passes as the features. So the length of hash will be

\[
\text{length of the hash} = \text{number of maps} \times \text{number of thresholds} \times \text{dimensions of approximation sub-band} = 3 \times (5 + 2) \times (8 \times 8 \times 2) = 2688 \text{ bits}
\]

A. Attack Description

Each test video is subjected to different kinds of attacks that add perceptually non-significant distortions. The attacks are designed to test the robustness of the hash under perceptually non-significant distortions and some of them are similar to the attacks presented in the Coskun’s work. The attack strengths used here lead to low visual quality videos in most cases. We apply them to test the algorithm under “extreme stress” cases expecting the hash to maintain its robustness even under these extreme modifications.

1) Scalable decoding: Since the proposed hashing algorithm offers robustness to spatial and rate scalability, there is no need to design attacks to test the hash performance against them. To test the hash performance under temporal scalable decoding, we have generated two videos with frame rates doubled and halved respectively from each test video by temporal filtering and interpolation/decimation.

2) AWGN: Zero mean additive white Gaussian noises with variances 5, 10 and 20 are added to get 3 noisy videos. Even though the perceptual quality of the video becomes poor under addition of the AWGN with variance 10 and above, we have included the AWGN of variance 20 to stress test the algorithm. We expect the hash to exhibit robustness even for the AWGN with variance 20 because it is extracted from the approximation sub-band.

3) Blurring: Guassian filters with variances 5, 10, 30, 50 and 100 respectively are applied on each frame to get the blurred videos. Here again large variances are considered to stress test the algorithm.

4) Brightness Increase/Decrease: Brightness of the video is increased/decreased by adding/subtracting \( r \) percent of the mean frame to/from each frame in the video. Values of \( r \) are chosen as 10, 30, and 50.

5) Contrast Increase/Decrease: The contrast of the video is increased/decreased by stretching the gray levels of video. We have considered three cases of contrast decrease by mapping 0 - 255 gray levels to 10 – 245, 50 – 200 and 88 – 168. Similarly we designed three contrast increase attacks by mapping 10 – 245, 50 – 200 and 88 – 168 gray levels to 0 – 255.

6) Frame drop: The frame dropping attack is aimed at simulating a lossy channel. When the packets carrying frame headers are lost, the frames are completely lost. These lost frames are then linearly interpolated from their neighboring frames. We have randomly dropped 10, 30, 50 and 70 percent of total frames in four frame drop attacks.

7) Frame rotation/circular shifting: Frame rotation and circular shifting are also important attacks that add non-significant distortions where the perceptual quality of video does not change much. We have considered two rotation attacks and one shifting attack. Each frame of the video is rotated by 1 and 3 degrees counterclockwise to get rotated video. Each frame is circularly shifted both in rows and columns by 1 percent to simulate shifting attack.

B. Parameters for Result Analysis

Two parameters called inter-hash statistics and intra-hash statistics are used in the literature of perceptual hashing to analyze the performance of a hashing algorithm.

1) Intra-hash statistics: Intra-hash statistics present the robustness of a hashing algorithm. They are extracted from the distances between hashes of perceptually similar videos. Videos perceptually similar to a given video are obtained by subjecting it to the attacks described in the section V-A. Since we expect the hashes from similar videos to be close in distance metric (Hamming distance in our algorithm), the average and maximum intra-hash distances should be small. We use average and maximum intra-hash distances as the intra-hash statistics to check the robustness of our algorithm.

2) Inter-hash statistics: Inter-hash statistics are extracted from the distances between hashes of perceptually distinct videos to measure the fault resilience of a hashing algorithm. Perceptually distinct videos from a given video are obtained by subjecting the remaining test video sequences to the same attacks described above. The average and minimum inter-hash distances should be large because we expect the hashes from distinct videos to be far apart in the distance. We use average and minimum inter-hash distances as the inter-hash statistics to check fragility of the algorithm.

VI. RESULTS AND DISCUSSION

Figure 3 shows the inter and intra-hash statistics of the proposed method. In the figure, the average inter-hash distance, minimum inter-hash distance, maximum intra-hash distance and average intra-hash distance are shown in the respective order for each attack. For example, the first group of four bars gives the average inter-hash, minimum inter-hash, maximum intra and average intra-distances of the AWGN (with variances 5, 10 and 20) added videos.

![Fig. 3. Inter-hash and Intra-hash Statistics of the Proposed Method against AWGN, Blurr (BL), Brightness Decrease (BD), Brightness Increase (BI), Contrast Decrease (CD), Contrast Increase (CI), Frame Dropping (FD), Frame Rate Changes (FRC) and Frame Shifting (FS) attacks.](image)

For a good hashing algorithm,

1) these statistics should decrease in magnitudes;
2) the inter and intra-hash statistics should be above and below a threshold (the threshold is drawn as a dotted line in the Figure) respectively and
the minimum inter-hash distance and maximum intra-hash distance should have a healthy margin between them, so that perceptually similar videos can be separated from distinct videos with high confidence. From Figure 3 we observe that the proposed method satisfies the three requirements for all attacks except the contrast change attack. For comparison, we show the performance of existing 3D DCT based method [8] in Figure 4 for the same attacks. We use the same statistics to compare the performances. We observe that the 3D DCT based method fails to meet the above requirements for four different kinds of attacks namely the AWGN, temporal scalability, frame rotation and circular shifting of frames.

**Fig. 4.** Inter-hash and intra-hash Statistics of the 3D DCT based Method against AWGN, Blur (BL), Brightness Decrease (BD), Brightness Increase (BI), Contrast Decrease (CD), Contrast Increase (CI), Frame Dropping (FD), Frame Rate Changes (FRC) and Frame Shifting (FS) attacks.

Contrast stretching in the spatial domain manifests itself as the magnitude stretching of the DWT coefficients which would alter the partitioning sets (LISA LISB and LIP) obtained with each sorting pass of the 3D SPIHT algorithm. Since we extract the partitioning sets as the features, the hash is not robust to large contrast changes. The 3D DCT based method thresholds low to mid frequency AC coefficients which will not be affected drastically under these attacks. The partitioning sets of the 3D DWT undergo less alteration compared to AC DCT coefficients under geometrical transformations (frame rotation and circular shifting) thereby showing the better performance of the proposed method.

**VII. CONCLUSION**

A perceptual video hashing algorithm is proposed. The algorithm mimics the encoding steps of a 3D SPIHT coding of 3D DWT coefficients to generate the hash. The partitioning sets obtained in the 3D SPIHT algorithm; namely LISA, LISB and LIP of the approximation sub-band of first five sorting passes have been used as robust video features. The features extracted by the proposed algorithm remained robust to distortions introduced by scalable decoding because the algorithm tracing the steps of the 3D SPIHT codec. The performance of the algorithm is evaluated by testing it with different attacked videos. While the proposed algorithm is not robust against contrast attacks, its performance against all other set of attacks is good.

The algorithm needs to be extensively tested for attacks introducing significant distortions (malicious manipulations like visible logo insertion). Currently we are developing a processing stage of the 3D DWT coefficients before coding them with the 3D SPIHT algorithm. The motivation is to make the hash robust to contrast attacks and increase the robustness of hash to rotation and shifting attacks. One can also use the motion-compensated 3D DWT and observe if there is any improvement in the performance against temporal attacks.

**REFERENCES**


