https://doi.org/10.2352/ISSN.2470-1173.2020.4.MWSF-218

Motion Vector based Robust Video Hash

Huajian Liu, Sebastian Fach, Martin Steinebach
Fraunhofer Institute for Secure Information Technology (SIT), Darmstadt, Germany

Abstract

A novel robust video hashing scheme is proposed in this paper. Unlike most existing robust video hashing algorithms, the proposed video hash is generated based on the motion vectors instead of the image textures in the video stream. Therefore, neither full decoding of the video stream nor complex computation of pixel values is required. Based on analysis of motion vector properties regarding their suitability for robust hashing, an improved feature extraction mechanism is proposed and several optimization mechanisms are introduced in order to achieve better robustness and discriminability. The proposed hashing scheme is evaluated by a large and modern video data set and the experimental results demonstrate the excellent performance of the proposed hashing algorithm, which is comparable or even better than the complicated texture-based approaches.

Introduction

Robust video hashing is a technique for video identification. In contrast to cryptographic hash, robust hash is less sensitive to data modification. The robust hash value remains unchanged as long as the perceptual content is preserved. This characteristic makes it especially suitable for multimedia content identification, because most processing of multimedia content will cause modifications of the binary representation, e.g. compression, resampling, brightness/contrast adjustment, etc., which will result in very different cryptographic hash values for the perceptually same content [1].

In the literature, most existing video hashing algorithms are based on the texture of the frames. Many of them are based on image hashing methods, where a video is understood as a sequence of images [2]. However, treating videos as a sequence of images does not make use of their specific characteristics. Therefore, dedicated video hashing algorithms have been developed [3-8], which make use of the temporal feature between frames. Machine learning and deep learning techniques are also used in video hashing [9-10]. All of these algorithms still rely on the texture of the video frames. This means that a video stream must be first decoded to a sequence of images and the video hash is calculated based on the image content. Decoding a video stream into frames and working on those images is rather slow. This issue can be addressed by using motion vector to construct the video hashes. To the best of our knowledge, there is currently no video hashing scheme using motion vectors.

The goal of this work is to develop an efficient robust video hashing algorithm based on motion vectors. In order to be applicable for video identification in practical applications, the robust video hash shall be robust to common video processing, e.g. compression, brightness/contrast adjustment, scaling, spatial and temporal cropping, frame rate conversion, etc. While it stays stable after content-preserving modifications, the robust video hash shall be sensitive to perceptual content changing, i.e. the hash values shall be of good discriminability between perceptually different video content.

Motion vectors are stored separately as side data in H.264 encoded videos [11], which can be extracted without decoding the pixel data. Thus, in contrast to texture based hashes, motion vectors can be extracted without decoding the entire video content. Therefore, a hashing scheme only based on motion vectors can save a significant amount of computation during hash generation. This enables a motion vector based video hashing technique to be potentially much faster than conventional texture based methods. Efficiency is particularly important for video content, as it usually involves a huge amount of data.

This paper is organized as follows. In Section 2, the properties of motion vectors are introduced and their suitability for robust hashing is examined. The hash construction is presented in Section 3 and the hash comparison approach is given in Section 4. Experimental results are shown in Section 5. We conclude the paper in Section 6.

Properties of Motion Vectors

Motion vectors have three primary properties: position, direction and length. In this section, we analyze which properties are suitable for robust hash generation.

Positon

As any vector, a motion vector has a start and an end point. The start point is the spatial source position, while the end point is the spatial target position in the motion compensation process. The target position is always the center of the (sub-)macroblock, leading to regular, grid-like structures. This means the target position does not represent the characteristics of a video, but rather the characteristics of the H.264 standard.

Both the source and the target positions are susceptible to spatial attacks. For example, cropping, rotation, translation and mirroring will change both positions. In case of skewing and change of aspect ratio, the relation between the source and the target positions will also be altered. In addition, in re-encoding after video processing the encoder may choose different sources to predict particular (sub-)macroblocks, resulting in significant change of some motion vectors.

Therefore, the source and the target positions of motion vectors are subject to change in video processing and neither of them is suitable for robust hash generation.

Direction

Motion vectors do not always reflect the actual movement of the video content. Motion vectors are calculated by the motion estimation process in the encoder, which estimates the motion of each macroblock individually, which is not necessarily the same as the overall motion of the object represented by the block. This introduces some “chaos”, causing motion vectors not matching the overall direction of movement.

In addition, the direction of motion vectors is susceptible to some attacks, e.g. rotation, skewing and mirroring. Also re-encoding will cause severe changes of vector direction. Due to the lack of fixed point for reference, the absolute direction of motion

vectors is not suitable for robust hashing. A possible solution to cope with these circumstances is to use the difference between frames. However, our early experiments show that even with advanced filtering, the vector direction is still rather unstable after re-encoding. Therefore, the direction property is not used in our robust hashing scheme.

Length

The length of a motion vector represents the amount of movement of a (sub-)macroblock. Although in some cases the length of motion vectors could also deviate vastly from the actual object movement, our early experiments show that the length is much more stable under attacks than the direction. Only simple filtering is required to make the length property usable for robust hashing. Hence, the length of motion vectors is used to construct the robust hash in our scheme.

However, the length of a motion vector will be changed in some video operations like scaling, skewing and change of aspect ratio. Therefore, these issues have to be addressed by corresponding mechanisms and a proper feature need to be extracted based on the length of motion vectors.

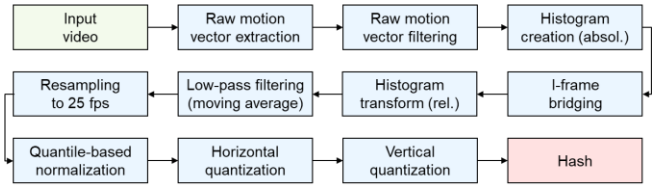


Figure 1. *Video hash generation*

Hash Construction

Based on the length of motion vectors, a histogram-based feature is extracted from the video to construct the robust hash. The hash generation consists of the following steps, as illustrated in Figure 1.

Motion Vector Extraction and Filtering

First, motion vectors are extracted frame by frame from the video stream. Usually many motion vectors have a length of zero, which are useless for hash construction and therefore are filtered out during the extraction process.

For the motion vectors with a non-zero length, their lengths are normalized by dividing the frame diagonal. The normalization helps to mitigate the threat of resolution-changing attacks like scaling and change of aspect ratio.

Feature Histogram Creation

Figure 2 shows the length histogram of the extracted non-zero motion vectors from 1000 videos in our test data set. Since the normalized lengths of most motion vectors are rather small, only the vectors in the range from 0 to 0.1 is plotted, presenting more than 99.62% of all motion vectors. The red line is the moving average, showing the trend. Note that a logarithmic scale is used for the y-axis.

For hash construction, it is more important to analyze the variation of motion vectors between videos. If motion vectors within a certain length range show more variation between videos than other vectors, they are more valuable for hash construction and should get a higher weight. On the contrary, omnipresent vector lengths should be given a lower weight or can be neglected completely.

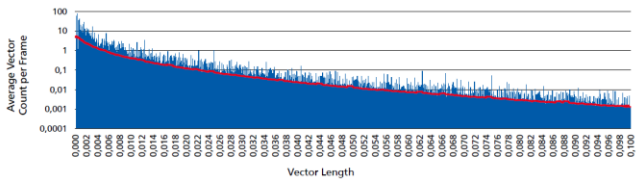


Figure 2. *Distribution of motion vector lengths*

As shown in Figure 2, the length distribution is very heterogeneous. The quantity of short motion vectors is far more than long vectors. Therefore, the variation has to be calculated using relative values to avoid biasing towards the vector lengths that occur more frequently as following

$$\mu_i = \frac{1}{N}\sum_{n=1}^{N} x_{n,i} \tag{1}$$

$$v(X_i) = \begin{cases} \frac{1}{N}\sum_{n=1}^{N} \frac{|x_{n,i}-\mu_i|}{\mu_i} & if\ \mu_i > 0 \\ 0 & if\ \mu_i = 0 \end{cases} \tag{2}$$

where $x_{n,i}$ is the average vector count per frame for the i-th bin in the length histogram of the n-th video. N is the total number of videos and μ_i is the average vector count per frame across all videos. $|x_{n,i} - \mu_i|/\mu_i$ represents the relative difference between $x_{n,i}$ and the global average μ_i. $v(X_i)$ indicates the variation of the average vector count $x_{n,i}$ within the i-th bin across all videos.

Figure 3 shows the distribution of the variation of motion vector lengths. The spikes on some bins of short length are caused by the fact that the motion vectors of these lengths only exist in one single video. The red line in Figure 3 is the moving average, which shows long motion vectors have higher variation between videos than short vectors. In other words, long motion vectors are more representative. However, as above-mentioned, the number of long vectors are far less than the short ones, which means that long vectors are valuable but they do not occur frequently. Therefore, although the shorter motion vectors have lower variation and more single spikes, they should not be ignored in the hash construction. To take both variation and quantity into account, we multiply the normalized trend lines of Figure 2 and Figure 3. The result is shown in Figure 4.

The area under the curve in Figure 4 can be divided into equally sized slices, as indicated by the blue lines as an example. In this example, the area is divided into 4 slices, which means that a feature histogram of motion vectors with 4 bins shall be created for each frame. Each bin contains the motion vectors of lengths falling in the corresponding slice, serving as a feature value. Figure 5 shows the sequence of the first bin values of all frames in a video. We refer to this sequence as a feature *signal* and each value in the signal as a *sample*.

The area under the curve in Figure 4 can be divided into more or less equally sized slices, resulting in feature histograms with different number of bins. Table 1 lists the bin ranges for histograms with different number of bins.

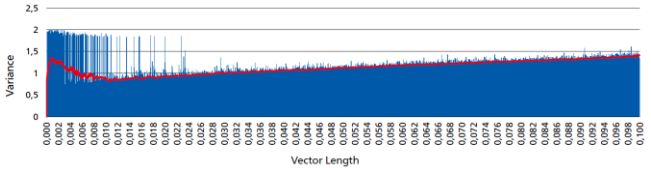


Figure 3. *Variation of motion vector lengths*

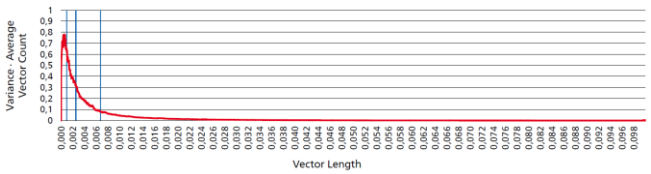


Figure 4. *Multiplication of normalized vector length and variation distribution*

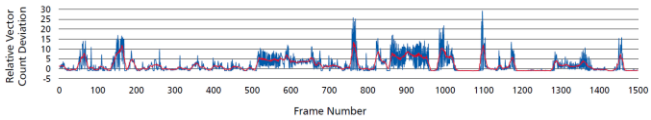


Figure 5. *Feature signal of the first bin values (test video 1080p)*

Table 1 Bin ranges of histograms with different number of bins

	3 Bins	4 Bins	5 Bins	6 Bins
Bin 1	0.00138	0.00098	0.00078	0.00065
Bin 2	0.00467	0.00254	0.00180	0.00138
Bin 3	1.0	0.00665	0.00363	0.00254
Bin 4		1.0	0.00856	0.00467
Bin 5			1.0	0.01034
Bin 6				1.0

Feature Histogram Transform

For videos with different resolutions, the amplitude of their feature signals varies greatly, because high-resolution videos tend to have much more motion vectors than low-resolution videos. To make the feature signal invariant to scaling, instead of using the absolute vector count and global average per bin, the number of vectors of each bin is divided by the total number of vectors of the current frame, leading to a relative histogram for that particular frame. Then the relative value is subtracted and subsequently divided by the relative global average. The relative global average can be derived by dividing the number of vectors in each bin by the total vector number.

I-Frame Bridging

Since there are no inter-prediction data for I-frames, the number of motion vectors for I-frames is always zero. This leads to undesired notches in the feature signal. Moreover, these notches do not occur at the same time point when comparing different videos. To remove these notches, the gap caused by I-frames is bridged by replacing the zero number of vectors by the average values of its two direct neighbor frames. If the first or the last frame has zero motion vectors, a linear interpolation based on the next two neighbors is done. If one of them is also zero, then the value of the other one is taken.

Low-pass Filtering

Even though the notches caused by I-frames are removed by bridging, adjacent sample values in feature signals still vary greatly, resulting in unstable high frequency variance. Even just re-encoding may lead to different sample values, resulting in different signal shapes not matching to each other. To mitigate this problem, the feature signal is smoothed by applying a low-pass filter to extract the long-term trend that is more stable under attacks. Moving average filters with different window sizes are used in our test. The red line in Figure 5 shows a low-pass filtered signal using a window size of 11 samples.

Resampling

Videos can have different frame rates. When the same video is re-encoded with a different frame rate, the drift over time will result in different signal shapes and length, which causes matching problem when comparing their feature signals. To address this problem, the feature signal is resampled to a fixed frame rate. In our test, 25 frames per second (fps), which is one of the commonly used frame rates, is used.

Normalization

The range of sample values vary in case of feature histograms with different number of bins. The more number of bins, the narrower is the value range. In addition, the range of sample values also vary for different vector lengths. The longer the vector, the larger is the value range. However, short vectors occur much frequently than long vectors. Therefore, using the entire global value range for normalization leads to a loss of precision.

In order to achieve a high resolution for the actual data range, a special [-1, 1] normalization is used to adjust the feature signal. The 0.2-quantile is mapped to -0.6, while the 0.8-quantile is mapped to 0.6. The sample values smaller than the 0.2-quantile and the values larger than 0.8-quantile are linearly mapped to the values in the range [-1.0, -0.6] and [0.6, 1.0], respectively. When the values are 5/3 times smaller than 0.2-quantile or larger than 0.8-quantile, they are clipped and mapped to -1.0 and 1.0, respectively. The following equation is used to determine the p-quantile.

$$x_p = \begin{cases} \frac{1}{2}\left(x_{pN} + x_{pN+1}\right) & if\ \mathrm{pN} \in \mathbb{Z} \\ x_{\lfloor pN \rfloor + 1} & if\ \mathrm{pN} \notin \mathbb{Z} \end{cases} \tag{3}$$

where N is the length of the feature signal and $\lfloor \cdot \rfloor$ indicates the floor function.

Quantization and Hash Generation

To get a more compact hash, quantization is applied on both temporal- and y-axis. On the temporal axis, this is done by dividing the samples into segments and calculating the average value for each time-segment. The quantization on temporal axis has similar effect as low-pass filtering, which can be omitted when temporal quantization is sufficient.

On the y-axis, the process is similar. The y-axis is divided into segments. The number of segments are determined by the requested resolution. For instance, for a resolution of 4 bits per sample, 16 segments shall be divided on y-axis. The sample values falling into the first segment are encoded as 0000, those in the second segment are encoded as 0001, and those in the last segment as 1111.

Hash Comparison

The normalized cross-correlation is used to compare the similarity between hashes, which is defined as

$$norm\_corr(x,y) = \frac{\sum_{i=1}^{N} x_i \cdot y_i}{\sqrt{\sum_{i=1}^{N} x_i^2 \cdot \sum_{i=1}^{N} y_i^2}} \quad (4)$$

where x and y are two feature signals and both x and y contain N samples.

Feature signals may be of different lengths, which are determined by the number of frames in the video stream. For instance, the feature signals generated from temporally cropped video clips are shorter than those of the original videos. To identify cropped videos, the feature signals are compared with time shift, resulting in a sequence of similarity values. The time shift leading to the maximal similarity indicates the best timely alignment of two videos.

Experimental Results

Test Data Set

To evaluate the proposed hash scheme, a video data set is created, which includes 2000 videos in 11 categories. The duration of each video ranges from 40 to 65 seconds. The videos in each category are randomly divided into two sets: known set and unknown set. Each set contains 1000 videos in total. The known set is used to create a reference hash database. Table 2 lists the number of videos in each category and the video lengths.

From the videos in the known set, a test set of attacked videos is created. In each category, half of the videos are randomly selected to form the test set, i.e. the attacked set contains 500 videos in total. For each video in the test set, eight kinds of attacks are applied. With different parameters in each attack, 36 attacked versions are created per video, resulting in 18000 attacked videos in the attacked set. Table 3 lists the attack categories and the applied parameter values. Note that in order to show the limit of the proposed scheme some selected attack parameters are rather extreme and the attacked videos do not have acceptable quality any longer. For instance, the perceptual quality gets severely degraded when reducing the frame rate to 50% of the original one or applying a constant rate factor of 45.

Table 2 Test video data set: known set and unknown set

Category	KNOWN		UNKNOWN	
	Count	Duration [hh:mm:ss]	Count	Duration [hh:mm:ss]
Animation Movie	31	00:31:14	31	00:30:16
Cinema	29	00:28:56	28	00:28:00
Comedy and Fun	34	00:34:06	35	00:34:33
House Building	97	01:37:10	98	01:37:55
Music Video	33	00:32:55	32	00:31:44
Nature and Travel	145	02:25:09	144	02:23:58
News	79	01:18:52	78	01:18:01
Quiz Shows	150	02:29:50	150	02:30:00
Science and Technology	129	02:08:54	130	02:10:26
Sports	123	02:03:01	124	02:04:01
Talk Shows	150	02:29:36	150	02:30:12
Total	**1000**	**16:39:47**	**1000**	**16:39:10**

Table 3 Attack categories and parameters

Attack Category	Parameter Description	Parameter Value
Random Frame Dropping	*n* frames dropped, *k* times	*n*=1,5,10 *k*=1,2,4
Frame Rage Change	multiple of the source frame rate	0.5, 0.75, 1.25,1.5
Mirroring	mirroring type	horizontal, vertical
Reduce Quality	constant rate factor (CRF)	30,35,40,45
Rotation	rotation in degrees	-15,-10,-5, 5, 10, 15
Scaling	multiple of the original dimension	0.4, 0.6, 0.8, 1.2, 1.4
Spatial Cropping	remaining part as multiple of the original dimension	0.7, 0.8, 0.9
Temporal Cropping	cropped part in seconds from the beginning	5, 10, 15

Test Parameters

In the evaluation, the parameter values listed in Table 4 are used, which are determined by our parameter optimization tests. The low-pass filter is turned off, i.e. setting the size to one sample. The test results show that additional filtering does not improve the results when a proper resolution is chosen in the horizontal quantization step. Two hashes are considered as a match if the similarity value exceeds the given threshold.

Table 4 Test Parameters

Parameter	Value
Low-pass Filter Size	1 Sa (off)
Horizontal Resolution	10 Sa
Vertical Resolution	32 Sa
Similarity Threshold	0.4

Results of Single Bin

The proposed scheme is first tested with hash values generated by a single bin. The hash values of the known set are generated by each single bin of a 4-bin feature histogram to create the hash database. Subsequently, the hash values of the unknown and the attacked sets are generated and compared with the hash database. Figure 6 and Figure 7 show the true positive rate and the false positive rate of the unknown set and the attacked set. For the known set, a true positive of 1.0 and a false positive rate of 0.0 are achieved throughout the entire threshold value range.

In Figure 6 and Figure 7, it is shown that when a similarity threshold is chosen to ensure a high true positive rate, the false positive rate keeps high. Conversely, if a low false positive rate is ensured, the true positive rate decreases significantly. The hash generated by Bin 1 achieves the best results. However, it can not achieve a satisfactory true positive rate while keeping a low false positive rate at the same time. Before the false positive rate gets too high, only a true positive rate of 0.85 can be reached. Therefore, more than one bin need to be combined to reduce the false positive rate.

Results of Combined Bins

Combining hashes from more than one bin can improve the false positive rate. However, considering all bins does not always lead to better results, but can have a negative impact on the true

positive rate and the performance. Therefore, only two or three bins, which are performing best, are combined in the following tests.

To determine the best performing bins, the results of feature histograms with different number of bins are analyzed using the same parameter settings. The results are listed in Table 5, which give the best three bins of the feature histograms containing 3 to 6 bins. Note that the first and the last bin always have the best performance, which conforms to our previous analysis that the short motion vectors have largest quantity and the longest vectors have most variation.

Based on the results of best performing bins, two bin-combination strategies are defined as follows.

Two-Bins The two best performing bins are used. For each bin, the best match is determined independently from the other bin. A pair of hashes are only considered as a match, if the results from both bins are the same.

Majority Voting The three best performing bins are used. The best match of each bin is determined independently. Only when two of three results are the same, will it be considered as a match.

Compared to the results of single bin, both the two-bins and the majority voting combination strategies achieve significant lower false positive rates at the cost of slightly lower true positive rates. The majority voting strategy gets good true positive rate, but the false positive rate is relatively higher in general. In comparison, the two-bins strategy shows better trade-off, achieving a true positive rate of 0.95 for the attacked set and a false positive rate of 0.01 for the unknown set.

Figure 8 shows the true positive rates for each individual attacks. The proposed scheme delivers very good and stable results across all attack categories. For rotation, random frame dropping and mirroring true positive rate of 98% and higher is reached in all test cases. For scaling with a factor of 0.6 and up, temporal cropping of 5 seconds, frame rate changes down to 0.75, spatial cropping to 90% and quality reduction with a CRF of 30, the aforementioned true positive rate is also achieved. Only in case of more severe attacks, the true positive rate decreases. For example, using only half of the original frame rate, downscaling the video to only 40% of the original size or reducing the quality to lower levels significantly affects the detection rates. However, most of these attacks have a perceivable impact on the content, leading to unacceptable video quality.

Note that nearly all videos that cannot be correctly detected lead to a false negative instead of a false positive. This property is crucial for practical applications of illegal video identification and screening.

Discussion

A direct comparison with other existing video hash algorithms is not easily feasible because of using different data sets and attacks. Therefore, only qualitative result comparison is provided here. A precision and recall rate of 95% each is reported in [3]. Using the same definition for precision and recall as in [2], our scheme achieves a recall rate of 94.7% and a precison rate of 99.9% on the attacked set. The ROC curve presented in [2] shows similar results. Approximately 95% of videos have been correctly detected when accepting a false positive rate of 1%. This shows that our scheme compares well to other more complex schemes in the previous work.

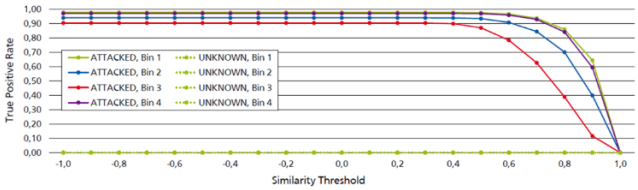


Figure 6. *True positive rate of the unknown and the attacked sets*

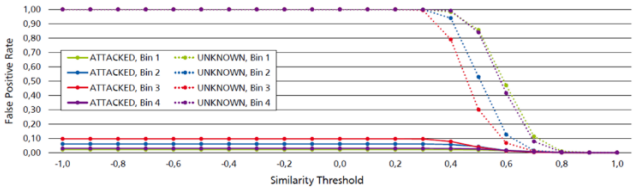


Figure 7. *False positive rate of the unknown and the attacked sets*

Table 5 Top three best-performing bins

	Primary Bin	Secondary Bin	Tertiary Bin
3 Bins	Bin 1	Bin 2	Bin 3
4 Bins	Bin 1	Bin 4	Bin 2
5 Bins	Bin 1	Bin 5	Bin 2
6 Bins	Bin 6	Bin 1	Bin 2

Conclusion

In this work, a novel motion vector based robust video hashing scheme has been proposed. In contrast to the texture-based hashing algorithms in the literature, the proposed hashing scheme does not require full decoding of the video stream, because the hash value is generated by only using the features extracted from the motion vectors. To improve the hash discriminability and the robustness, a histogram-based feature extraction mechanism is proposed and several optimization steps are introduced. The good test results demonstrate that robust video hashing based on motion vectors is feasible and performs well.

References

[1] Martin Steinebach, Huajian Liu, York Yannikos. ForBild: efficient robust image hashing, Proceedings of SPIE 8303, Media Watermarking, Security and Forensics 2012, 83030O, 2012.

[2] Li Weng and Bart Preneel. From Image Hashing to Video Hashing. Proceedings of International Conference on Multimedia Modeling. Springer. pp. 662–668, 2010.

[3] Jiande Sun et al. Video Hashing Algorithm with Weighted Matching Based on Visual Saliency. IEEE Signal Processing Letters, 19.6, pp. 328–331, 2012.

[4] Mani Malek Esmaeili, Mehrdad Fatourechi, and Rabab Kreidieh Ward. A Robust and Fast Video Copy Detection System Using Content-based Fingerprinting. IEEE Transactions on information forensics and security, 6.1, pp. 213–226, 2011.

[5] Shi Jun Xiang, Jian Quan Yang, and Ji Wu Huang. Perceptual Video Hashing Robust against Geometric Distortions. Science China Information Sciences, 55.7, pp. 1520–1527, 2012.

[6] Sunil Lee, Chang D Yoo, and Ton Kalker. Robust Video Fingerprinting based on Symmetric Pairwise Boosting. IEEE Transactions on Circuits and Systems for Video Technology 19.9, pp. 1379–1388, 2009.

[7] B. Coskun, B. Sankur, and N. Memon. Spatio–temporal transform based video hashing. IEEE Transactions on Multimedia, 8(6):1190–1208, 2006.

[8] J. Wang, J. Sun, J. Liu, X. Nie, and H. Yan. A visual saliency based video hashing algorithm. In Proceedings of IEEE International Conference on Image Processing (ICIP), pp. 645–648, 2012.

[9] Sunil Lee, Chang D Yoo, and Ton Kalker. Robust Video Fingerprinting based on Symmetric Pairwise Boosting. IEEE Transactions on Circuits and Systems for Video Technology, 19.9 pp. 1379–1388, 2009.

[10] Venice Erin Liong, Jiwen Lu, Yap-Peng Tan and Jie Zhou. Deep Video Hashing, IEEE Transactions on Multimedia, vol. 19, no. 6, June 2017.

[11] Detlev Marpe, Thomas Wiegand, and Gary J Sullivan. The H.264/MPEG4 Advanced Video Coding Standard and its Applications. IEEE communications magazine, 44.8, pp. 134–143, 2006.

Author Biography

Huajian Liu received his B.S. and M.S. degrees in electronic engineering from Dalian University of Technology, China, in 1999 and 2002, respectively, and his Ph.D. degree in computer science from Technical University of Darmstadt, Germany, in 2008. He is currently a senior research scientist at Fraunhofer Institute for Secure Information Technology (SIT). His major research interests include information security, digital watermarking, robust hashing and digital forensics.

Sebastian Axel Fach is an Automotive Security & Privacy Specialist at Continental AG, Frankfurt, Germany. Before moving to Continental in 2018, he wrote his Master Thesis in the field of Robust Video Hashing at Fraunhofer SIT. He holds a M.Sc. in Computer Science with specialization in Microelectronics and a M.Sc. in IT Security, both obtained from Technical University of Darmstadt, Germany.

Prof. Dr. Martin Steinebach is the manager of the Media Security and IT Forensics division at Fraunhofer SIT. From 2003 to 2007 he was the manager of the Media Security in IT division at Fraunhofer IPSI. He studied computer science at the Technical University of Darmstadt and finished his diploma thesis on copyright protection for digital audio in 1999. In 2003 he received his PhD at the Technical University of Darmstadt for this work on digital audio watermarking. In 2016 he became honorary professor at the TU Darmstadt.

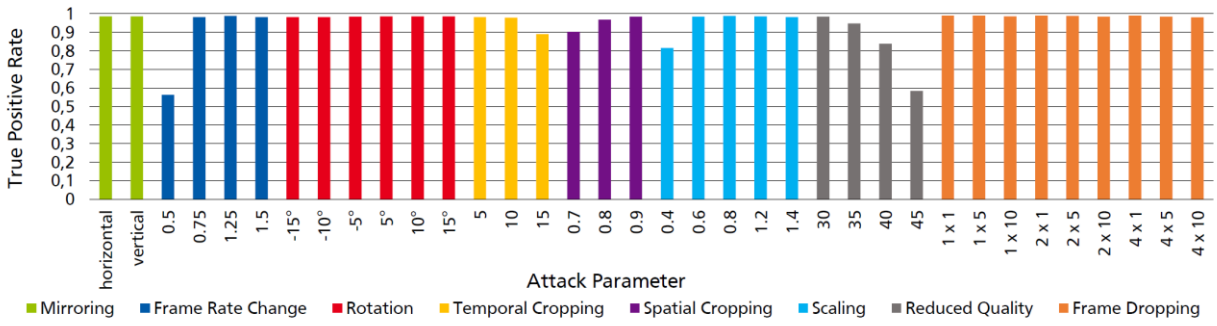


Figure 8. *True positive rate of the attacked set*

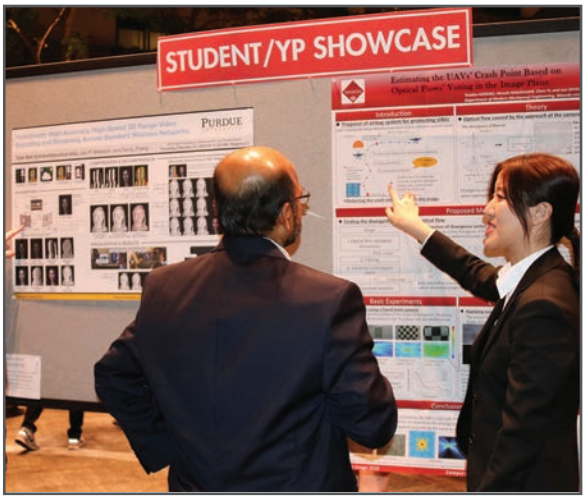
JOIN US AT THE NEXT EI!

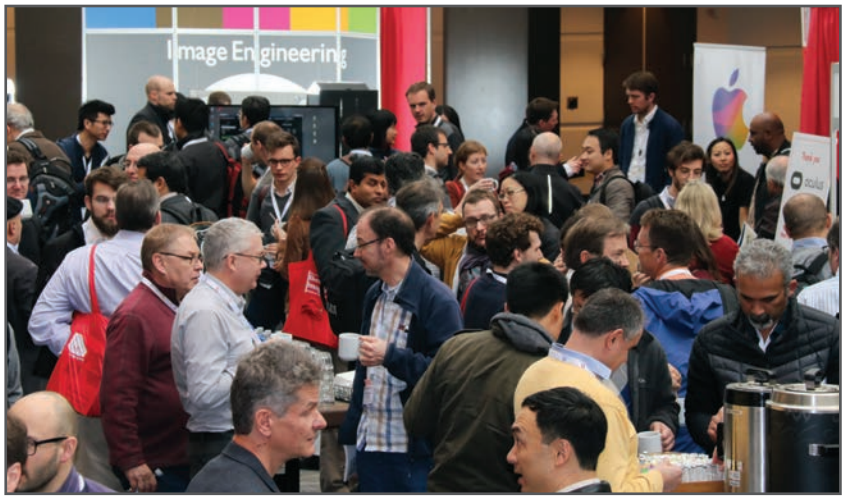
IS&T International Symposium on

Electronic Imaging

SCIENCE AND TECHNOLOGY

Imaging across applications . . . Where industry and academia meet!





• SHORT COURSES • EXHIBITS • DEMONSTRATION SESSION • PLENARY TALKS •
• INTERACTIVE PAPER SESSION • SPECIAL EVENTS • TECHNICAL SESSIONS •

www.electronicimaging.org

IS&T
imaging.org