Fast Compressed Domain Copy Detection with Motion Vector Imaging

Yuanyuan Yang[1], Yixiong Zou[2], Yemin Shi[2], Qingsheng Yuan[4], Yaowei Wang[3], Yonghong Tian[2*]
[1] Beijing University of Posts and Telecommunications
[2] Peking University
[3] Beijing Institute of Technology
[4] School of Cyber Security, University of Chinese Academy of Sciences

Abstract

With an increasing number of videos uploaded to the Internet, how to fast detect copy videos in compressed domain has been paid greater attention to. Many researchers have tried using information in motion vector to be the feature. However, in these methods motion vectors are used as histogram, which lacks structural information in detail. To address this problem, in this paper we propose a new way of using Motion Vector Imaging. We first extract motion vector from a compressed video, and then project them onto a canvas to generate a MVI which contains detail motion information. Based on these MVIs, a siamese deep neural network is utilized to train on pairs from dataset and one side of the network is applied to extract features. Finally, a cascade system using MVI model and I frames is used to do fast copy detection. Results on public dataset CC_WEB_VIDEO show that MVI can achieve high recall rate and precision rate at a high speed.

1 Introduction

With an increasing number of people connected to the Internet, there are more and more videos uploaded to the Internet everyday. Besides those original videos, there also exists a great many near duplicate videos copied from original ones, which may do great harm to the copyright of each publisher. These near duplicate videos apply many transformations to the original videos such as transform of bit rate, frame rate, resolution and so on. Generally speaking, most videos on the Internet are stored in compression, which takes a relatively great time to decode to get each image to pixel domain. However, as the number of uploaded videos increases rapidly, it is more and more infeasible to do copy detection in pixel domain. To address this problem, compressed domain copy detection begins to draw people's attention.

*Corresponding author: Yonghong Tian (yhtian@pku.edu.cn)

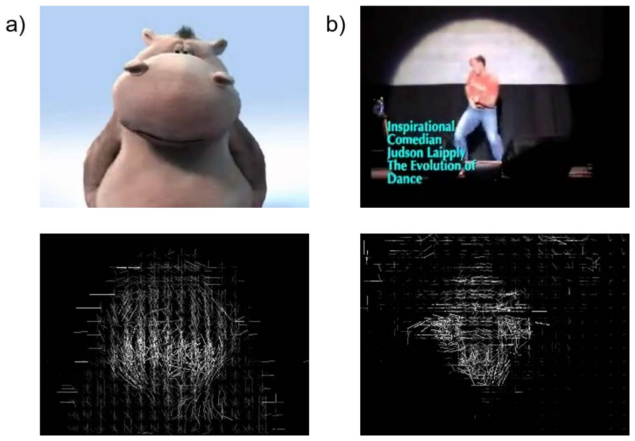

Figure 1: Motion Vector Imaging(MVI) can represent movements in both videos. (a) A cartoon hippo is singing, and we can see its mouth in the MVI, which represents the movement of its mouth.(b) A man is dancing with a light shaded on him, and we can see the lighted part is just where he is dancing.

Unlike pixel domain Content-based copy detection (CBCD), compressed domain CBCD doesn't have features like SIFT or SURF. It can only use features extracted from coding streams, which makes it hard for researchers to design features. [7] proposed to use easy available information in the stream such as bit per frame (BPF) of each GOP, histogram of macro-block size of each frame and Gaussian weighted average motion intensity of each frame. As these information can almost be retrieved even without decoding, the processing speed is really high while the performance is not well satisfied.

Another way of compressed domain CBCD is to use information in Discrete Cosine Transform (DCT) coefficient as [8] did. After Discrete Cosine Transform, each frame can be divided into DC and AC coefficients. AC coefficients can represent texture information varying from low

0-7695-6354-6/18/$31.00 ©2018 IEEE
DOI 10.1109/MIPR.2018.00086

frequency to high frequency as DCT block goes from left-top to right-down. The author divides each elements into simple, middle and complex texture to represent low, middle and high frequency respectively. Then they count the number of each kind of element and use this as the feature. [14] uses energies of the first four sub-bands of each coefficient matrix to be the feature by summing up the absolute values of each coefficient. And [18] uses edge information contained in the left-top 4 elements of DCT coefficient matrix to do the job.

Apart from DCT coefficient describing spatial information in I frame, researchers also pay attention to motion vector describing temporal motion information between frames. Motion vector is a motion estimation of macro-blocks between frames and exists in inter frames such as P frame and B frame. In [17], the authors propose to use large value (intensity) motion vectors and the less phase angle changed component to generate watermark for each inter frame. As [17] uses fixed threshold to filter motion vector with large intensity, [9] uses Average Motion Vector magnitude (AMV) to be the adaptable threshold, and divide each frame into 9 regions to average over each region. Then it filters each regions again to describe motion activity in each region as well as direction to form the Motion Activity (MA) words and finally use K-means clustering method to get the feature. Besides paying attention to features, some researchers also make efforts to reduce noise. In [13] researchers argue that the displacement between 2 consecutive frames can be trivial when the object moves slowly or even keep stationary, leaving the motion magnitude really small and may contain some error. Thus [13] proposes to extract motion vectors between every t^{th} frame and $(t+n)^{th}$ frame rather than $(t+1)^{th}$ frame so as to make motion vector more robust. However, this method requires searching method for each video which is still time demanding.

However, methods described above all use handcrafted global information which lacks structural information in detail. To address this problem, we use method similar to method proposed in [10]. As with action recognition, CBCD using motion vector also utilizes information contained in the moving objects. [10] uses optical flow to form a dense trajectory and project it to a canvas to get texture image, then uses Deep Neural Network (DNN) to extract feature so as to get Deep Trajectory Descriptor (DTD). Similarly, as shown in Fig. 1, different actions can be represented by projected motion vectors, thus motion vector here can also play a role like optical flow to describe motion of each object.

Following this idea, we propose our Motion Vector Imaging (MVI) method and use deep learning method to learn the feature to do compressed domain CBCD. We first extract motion vectors of all inter frames and then project motion vectors of a fixed period to a canvas to get a MVI which contains both global information and local information. After that, a siamese deep neural network is utilized to train on pairs from dataset and one side of the network is applied to extract features. Eventually we use a cascade system in which the first layer is MVI model and the second layer is I frame model to do the compressed domain CBCD.

The rest of this paper is organized as follows: In section 2 we briefly summarize the related works of CBCD. In section 3 we propose MVI and in section 4 we introduce the cascade system to do the fast copy detection. Implementation details as well as experiments are described in section 5. Finally we conclude this paper in section 6.

2 Related Works

There have been several works trying to conduct CBCD in the compressed domain. Babu [2] proposed to extract feature in MPEG compressed domain. The features extracted from motion vectors were fed to Hidden Markov Model (HMM) for classification. [3] proposed a method using motion flow history (MFH) and motion history image (MHI) in MPEG compressed domain to recognize human action. Various handcrafted features were extracted from the static MFH and MHI images, and histogram of the horizontal and vertical components of the MVs were utilized to form the Projected-1D feature. Also, a 2D Polar feature was developed using histogram of magnitude and orientation of MVs. The extracted features were used to train different types of classifiers including KNN, Neural network, SVM and the Bayes for action recognition. Later Khalid Tahboub [12] proposed a motion vector based method to use robust hashing function with projection on random matrices to generate the hashing bits. A sequence of these bits serves as the signature for the video.

Besides using motion vector only, many works are also based on combination of motion vector and other compressed domain information. Manu Tom.R [15] proposed a novel algorithm for activity recognition using information from QPs, MB partition types, and MVs in the H.264/AVC compressed video. The gradient information of QP over the space is utilized to form QP Gradient Image (QGI) which provides vital clue regarding motion occurrence and the spread of the action. This information, along with the magnitude and orientation of the motion vector, is utilized together to represent a video. And [1] proposed a technique fuses the macroblock types information and the motion field information generated by using the motion vectors in the MPEG stream to capture the intrinsic content of the video.

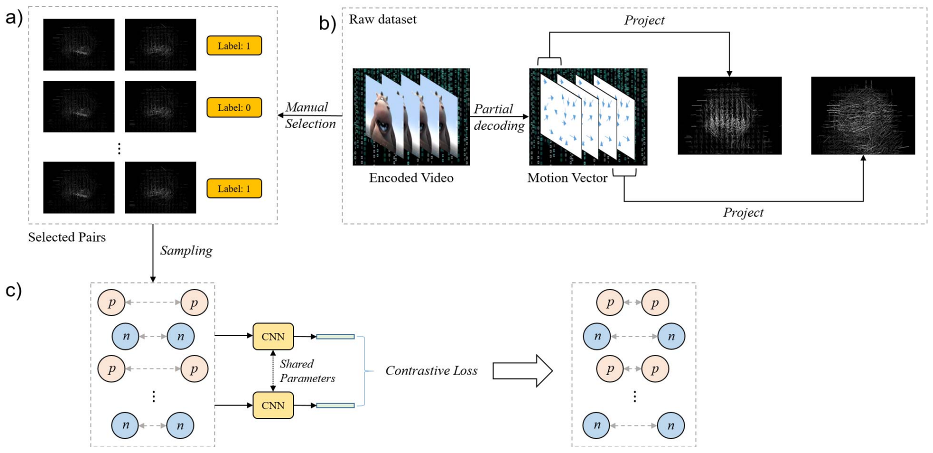

Figure 2: Framework of MVI. We first convert encoded videos in datasets to MVI, which needs partial decoding, then select pairs for training and testing, and finally use a Siamese network to learn a distance embedding by sampling from selected pairs. p denotes positive(copied) pairs and n denotes negative(not copied) pairs.

3 MVI

Figure 2(b) illustrates the pipeline of our MVI extraction. We first extract the motion vector of our input video by partial decoding, and then project motion vector in a period to a canvas to get the motion vector imaging(MVI). Then a video can be described by several consecutive imaging and we can use CNN to train or extract features on it.

Motion vector is from compressed domain of encoding system such as H.264, which describes the motion estimation between successive frames and can be seen as a kind of trajectory. Frame can be divided in to I frame, P frame and B frame. I frame is the beginning of a GOP which doesn't contain motion vector. But as I frame is quite rare in a GOP, this loss of motion vector can be ignored. P frame and B frame contain motion vector and B frame contains bi-directional motion vector. We take all the motion vectors of these frame as the motion descriptor and project them to an empty canvas. As motion vector indicates the movement between macro-blocks which is difficult to draw, we choose to draw the movement of each center of macro-blocks instead.

Let (x_t^k, y_t^k) denotes the center point p_t^k of relevant macro-block at frame t and k denotes the index of this macro-block. Given a compressed video of H.264, when we partially decode it we can get movement

$$\Delta d_t^k = (\Delta x_t^k, \Delta y_t^k) = (x_{t+1}^k - x_t^k, y_{t+1}^k - y_t^k) \quad (1)$$

directly.

A video can be divided into several successive parts with a fixed time T_0. Suppose for each period l we get a set of movements

$$M_l = \{\Delta d_0^0, \Delta d_0^1, \cdots, \Delta d_t^k\} \quad (2)$$

Then for each period l we can project M_l to an imaging I_l, that is to say, draw a line from p_t^k to p_{t+1}^k for each Δd_t^k as follows:

$$I_l = \begin{cases} \sqrt{(\Delta x_t^k)^2 + (\Delta y_t^k)^2} & if(x,y) inline, \\ 0 & otherwise \end{cases} \quad (3)$$

With this equation we can convert a compressed video into a set of successive imaging which we call Motion Vector Imaging.

Because of fixed T_0, there are always some residual time at the tail of a video. For short videos residual period also contain important information compared with the whole video while for long video it may import some error. As a consequence, we set a threshold P for fixed period count to determine whether to keep it.

Unlike [10], we don't take an adaptive way to choose the count of frames for two reasons. On the one hand, calculating overwrite ratio needs to count overwrite pixels on a pixel level, which is quite time demanding. On the other hand, fixed T_0 can make the model robust for change of frame-per-second(FPS) and GOP size.

Unlike action classification, MVI here doesn't need to de-noise also for two reason. First, de-noising method may be time consuming that may lower the speed on which the compressed domain copy detection stressed. Second, the noise contained by the query video also exists in the reference video. As long as these videos' noise are the same, we can still detect them.

4 MVI Cascade System

Framework of our MVI cascade system is shown in Fig. 3, in which we first use partially decoded MVI to filter suspected videos and then use fully decoded method to find the copy videos. In this section we will first introduce the deep model we use to train and extract features for MVI, then we describe the cascade system we use to achieve the fast copy detection.

4.1 MVI Model

Copy detection is essentially an embedding learning task. Given pairs of images labeled as copy or not, we need to increase the distance of not copied ones and decrease that of copied ones. And as copy detection needs a threshold to balance recall rate and precision rate, contrastive loss with a margin is suitable here. Thus we decide to use a Siamese network to train it.

Figure 2(c) illustrates the training period of out model. As you can see, we first randomly sample pairs from selected pairs, then we input a pair of images to a pairwise CNN whose parameters are shared, after that CNN extracts a feature vector for each image. Finally these features are sent to contrastive loss function which can be written as

$$loss = \frac{1}{2N}\sum_{n=0}^{N} yd^2 + (1-y)max(margin-d,0)^2 \quad (4)$$

where y indicates the label, 1 for copy and 0 for not copy, and margin is typically set to 1. d denotes L2 normalized distance between 2 images. With increased inter-class distance and intra-class distance, we can distinguish these two types of pairs easily.

4.2 Cascade System

As shown in Fig. 3, at the beginning we have a large amount of raw videos to detect, as the partially decoding method is much faster but less accurate than the fully decoding method, we put our partially decoding model (MVI) in the first layer to filter out many easy videos. Then those filtered suspected videos are sent to the fully decoding model to get the final result. Because videos have been reduced on a large scale, it's feasible to implement a slower but more accurate model in the second layer. Thus in the first layer, we should pay more attention to high recall rate while in the second layer we should stress more on high precision rate.

As for the fully decoding method, apart from precision, we also want it to have a relatively high speed. So we only pick the key frame (I frame) to decode. I frame is at the beginning of each GOP and there is only 1 I frame in a GOP. The count of other kinds of frame is always tens of times of that of I frame, leaving out most of frames while decoding. Moreover, decoding P frame or B frame need to refer to decoded past frames while decoding I frame doesn't. As a result, decoding only I frame will be much faster than decoding the whole video.

Both MVI model and I frame model convert a compressed video into several successive images, thus we need matching strategies to convert distances between images to distance between videos. As for matching strategies, we simply use a fixed size M to be the matching length and find the smallest normalized distance of M consecutive images between two videos. This can be represented in each layer in Fig. 3, where two lines of dots represent MVIs or I frames of two videos and orange ones mean those can match each other. Video distance can be written as

$$distance = \frac{1}{M}\sum_{i=0}^{M} d_i \quad (5)$$

, where d_i denotes L2 normalized distance of a pair within consecutive M pairs.

With distance for each video, we can specify a threshold T to judge whether it is a copy. For MVI layer, T should be set as a larger number than that of I frame layer, as larger T means less videos will be missed, which also means higher recall rate.

5 Experiment

In this chapter we will first introduce our implementation details and then report the experimental results and our exploration experiment with MVI.

5.1 Implementation Details

We use Caffe [6] to implement our deep model. In terms of base network CNN, we tested GoogleNet [11] and Inception network with batch normalization [5] and finally use the latter one.

In terms of training data, as there are no provided pairs for us, we need to generate them ourselves. We marked some videos that are totally the same in context but different in file formats, encoding parameters and so on, and then use these videos to generate training and testing pairs.

Before training our Siamese network on our own pairs, we pre-trained base network on ImageNet [4]. Then we use pre-trained model to extract feature for each MVI and send them to matching strategy with matching length 1 to get a rough result. With single frame match we can get a threshold resulting in a high precision rate and low recall rate. This is the threshold under which distance between images are quite small with no judging error. Thus we can use this threshold to run the judging again to find those pairs whose distance are smaller than it. If there are more than 5 images in two pairs are the similar with the same MVI id, we have confidence to say that the relevant video pairs are all the same from the beginning to the end in context. Moreover, to avoid mistakes we finally draw those pairs and check them

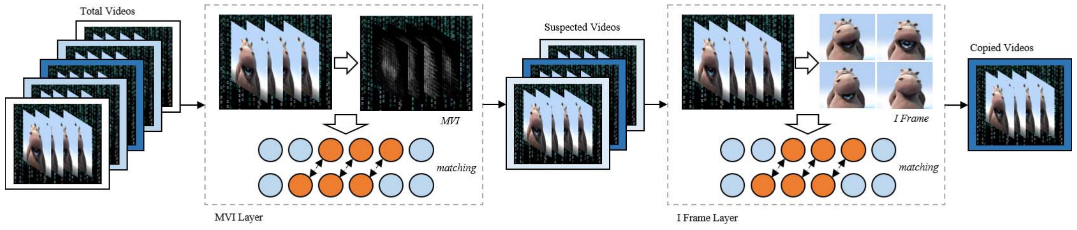

Figure 3: Cascade system pipeline. We first input total videos into the MVI layer which do copy detection with high recall rate at a high speed. In this step suspected videos will be found, and will be input into I frame layer which do copy detection with high precision rate at a sightly lower speed. And finally the copied videos will be found.

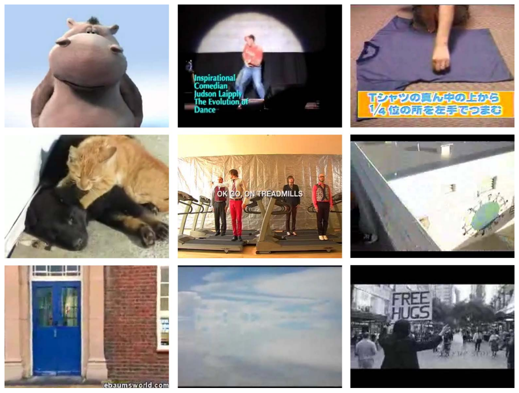

Figure 4: Samples of CC_WEB_VIDEO. Great variations in context exist in this dataset, which makes it suitable for performance testing.

manually to get pairs labeled 1 (similar pairs). To get pairs labeled 0 (different pairs), we use seed videos' MVI which are different from themselves. Relevant videos used here are training set, which are excluded from the rest of videos that form the testing set. With these pairs in training set, we can fine tune our model.

In out experiment we set a fixed MVI extraction time T_0 to 4.17s and fixed residual count P to 4. And the matching length M is set as

$$M = min(c_1, c_2) \tag{6}$$

where c_1 and c_2 denote the number of MVIs of 2 videos respectively.

For I frame model, as pre-trained model is good enough to get a satisfied result, we didn't fine tune on target dataset. And because GOP size for each video may be different, the matching length M is set to 3 instead of the length of each video.

5.2 Dataset

In our experiment we use CC_WEB_VIDEO [16] to test our cascade system.

Table 1: Performance on CC_WEB_VIDEOS. High recall rate can be achieved by MVI model at a high speed, and both high recall rate and precision rate can be achieved by our cascade system at a relatively high speed.

Method	Recall(%)	Precision(%)	Speed(s/video)
MVI	95.8	50.1	0.572
Cascade	95.3	91.08	1.34

CC_WEB_VIDEO is a near duplicate web videos dataset published by Xiao Wu, el. It selects 24 queries designed to retrieve the most viewed and top favorite videos from YouTube in November, 2006. Each Video in the same directory are approximately identical, but different in file formats or some other parameters. In all it contains 24 directories and 12790 videos with 2/7 of which are copy videos. Samples of these videos can be shown in Fig. 4.

5.3 Result and Exploration Experiments

The result and speed on CC_WEB_VIDEO are listed in Table 1. As you can see, on CC_WEB_VIDEOS we can achieve near 95% recall rate with higher than 50% precision. As stated before, MVI layer needs to stress on high recall rate, so as not to leave out any suspected video, thus this performance is high enough. On the other hand, the speed of MVI is high, as it only takes 0.572s for each video for detection. Such high speed is achieved by partially decoding, compared with fully decoding. Performance of our cascade system is also listed in Table 1. With high recall rate achieved by MVI, I frame layer needs to stress on high precision rate. Based on MVI, Cascade system achieves 95% recall rate with 91.08% precision rate at the speed of 1.34s/video, which means in I frame layer, most of false positive samples are filtered. Even though I frame layer may be a little bit slower than MVI, it can indeed promote to precision rate.

As stated before, we use MVI generated by fixed time. However, generation by GOP is also another option. Thus

Table 2: Comparison of MVI generated by GOP and fixed time. Keep the recall rates the same, MVI_t can achieve about 15% higher precision rate, compared with MVI_g.

Method	Recall(%)	Precision(%)
MVI_t	88.5	90.1
MVI_g	88.4	74.5

we compare performance of these two options in Table 2. MVI_t means generation by fixed time and MVI_g means generation by GOP. As you can see, we alter threshold so as to keep recall rate of them almost the same, so that we can compare their performance by precision easily. With recall rate at 88%, MVI_t achieves 90% precision while MVI_g only achieves 75%. This difference is a strong evidence of why we choose MVI_t. Generation by fixed time can tolerate the variation made by different FPS, GOP size, and so on. However, on the other hand, with variation made by MVI_g, it can still achieve recall rate near 90% and precision rate near 75%, which in turn proves the effectiveness of MVI.

6 Conclusion

In this paper we propose a novel way, MVI, to model motion in compressed domain and use a cascade system containing MVI and I frame model to do compressed domain copy detection. We first extract motion vector from compressed domain and then project them to a canvas for a fixed period. After that we use a Siamese network to train them and use one side of the network to extract feature for copy detection. Performance of tested our models on CC_WEB_VIDEO shows its effectiveness.

Acknowledgement. This work is partially supported by grants from the National Basic Research Program of China under grant 2015CB351806, the National Natural Science Foundation of China under contract No. U1611461, No. 61390515, No. 61425025, No. 61650202, and No. 61633002.

References

[1] A. S. Abbass, A. A. A. Youssif, and A. Z. Ghalwash. Hybrid-based compressed domain video fingerprinting technique. *Computer & Information Science*, 5(5), 2012.

[2] R. V. Babu, B. Anantharaman, K. R. Ramakrishnan, and S. H. Srinivasan. Compressed domain action classification using hmm. *Pattern Recognition Letters*, 23(10):1203–1213, 2001.

[3] R. V. Babu and K. R. Ramakrishnan. Recognition of human actions using motion history information extracted from the compressed video . *Image & Vision Computing*, 22(8):597–607, 2004.

[4] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and F. F. Li. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255, 2009.

[5] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *Computer Science*, 2015.

[6] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[7] C. Kas and H. Nicolas. Compressed domain copy detection of scalable svc videos. In *International Workshop on Content-Based Multimedia Indexing*, pages 89–94, 2009.

[8] Z. Li and J. Chen. Efficient compressed domain video copy detection. In *Management and Service Science (MASS), 2010 International Conference on*, pages 1–4, 2010.

[9] R. Roopalakshmi and G. R. M. Reddy. *Content-Based Video Copy Detection Scheme Using Motion Activity and Acoustic Features*. Springer International Publishing, 2014.

[10] Y. Shi, Y. Tian, Y. Wang, and T. Huang. Sequential deep trajectory descriptor for action recognition with three-stream cnn. 2016.

[11] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[12] K. Tahboub, N. J. Gadgil, M. L. Comer, and E. J. Delp. An hevc compressed domain content-based video signature for copy detection and video retrieval. In *IS&T/SPIE Electronic Imaging*, pages 557–580, 2014.

[13] K. Tademir and A. E. etin. Content-based video copy detection based on motion vectors estimated using a lower frame rate. *Signal, Image and Video Processing*, 8(6):1049–1057, 2014.

[14] Y. Tian, M. Jiang, L. Mou, X. Fang, and T. Huang. A multimodal video copy detection approach with sequential pyramid matching. In *IEEE International Conference on Image Processing, ICIP 2011, Brussels, Belgium, September*, pages 3629–3632, 2011.

[15] M. Tom, R. V. Babu, and R. G. Praveen. Compressed domain human action recognition in h.264/avc video streams. *Multimedia Tools and Applications*, 74(21):9323–9338, 2015.

[16] X. Wu, A. G. Hauptmann, and C. W. Ngo. Practical elimination of near-duplicates from web video search. In *ACM International Conference on Multimedia*, pages 218–227, 2007.

[17] J. Zhang, J. G. Li, and L. Zhang. Video watermark technique in motion vector. In *Computer Graphics and Image Processing, 2001 Proceedings of XIV Brazilian Symposium on*, pages 179–182, 2001.

[18] Z. Zhang and J. Zou. A relative phases algorithm of key edges for detecting compressed video copies. *Journal of Xian Jiaotong University*, 44(10):8–11, 2010.