Stochastic Non-linear Hashing for Near-Duplicate Video Retrieval using Deep Feature applicable to Large-scale Datasets

Sung-Woo Byun[1] and Seok-Pil Lee[1]
[1]Department of Computer Science, Graduate School, SangMyung University
Seoul, South Korea
[e-mail: 123234566@naver.com, esprit@smu.ac.kr]
*Corresponding author: Seok-Pil Lee

Received September 26, 2018; revised March 19, 2019; revised May 24, 2019; accepted June 29, 2019; published August 31, 2019

Abstract

With the development of video-related applications, media content has increased dramatically through applications. There is a substantial amount of near-duplicate videos (NDVs) among Internet videos, thus NDVR is important for eliminating near-duplicates from web video searches. This paper proposes a novel NDVR system that supports large-scale retrieval and contributes to the efficient and accurate retrieval performance. For this, we extracted keyframes from each video at regular intervals and then extracted both commonly used features (LBP and HSV) and new image features from each keyframe. A recent study introduced a new image feature that can provide more robust information than existing features even if there are geometric changes to and complex editing of images. We convert a vector set that consists of the extracted features to binary code through a set of hash functions so that the similarity comparison can be more efficient as similar videos are more likely to map into the same buckets. Lastly, we calculate similarity to search for NDVs; we examine the effectiveness of the NDVR system and compare this against previous NDVR systems using the public video collections CC_WEB_VIDEO. The proposed NDVR system's performance is very promising compared to previous NDVR systems.

Keywords: Near-duplicate Video, NDVR, Class Activation Maps, Hashing, supervised learning

A preliminary version of this paper was presented at APIC-IST 2018, and was selected by the conference review process

http://doi.org/10.3837/tiis.2019.08.028 **ISSN : 1976-7277**

1. Introduction

With the development of video-related applications such as video-sharing websites, video broadcasting, and advertising services, the amount of available media content has increased dramatically. Among this huge amount of media content, there is a substantial portion of duplicate and near-duplicate videos (NDVs) that have gone through video editing and redistribution [1–5]. Such NDVs largely influence video-related applications, removing them requires near-duplicate video retrieval (NDVR). The principle of NDVR is to retrieve accurate and efficient NDVs that are defined as identical or approximately identical videos that are almost exact duplicates of each other. To reflect these needs, NDVR is utilized in various fields; it is applicable in various uses such as copyright protection, video monitoring, and recommendation systems. For instance, we can use NDVR in search engines, ensuring that users can enjoy a range of videos distinct from each other rather than be faced with an endless list of the same or similar clips. Another example is using NDVR to reduce the risk that video products face of being compromised by unauthorized copying, editing, and redistribution; therefore, NDV detection is important for copyright protection.

NDVR involves searching for identical or approximately identical videos to existing videos [6] and consists of three main parts: 1) keyframe extraction, 2) feature extraction, 3) similarity computation. Keyframe extraction extracts multiple representative images from each video at regular intervals; it then extracts features to characterize each keyframe. Feature extraction generates numerical characteristics from each image using domain knowledge. Well-extracted features increase the retrieval algorithms' effectiveness. Similarity computation calculates similarities between videos using the extracted features and finally retrieve near-duplicate videos based on the similarities. NDVR system computes similarities and retrieve relevant videos by exhaustively comparing the features extracted from each keyframe between all pairwise keyframes. Here, because no single feature type exists that is sufficiently robust to capture all variations in the information, previous studies have proposed video representation methods that combine multiple types of feature. Many studies have used HSV histogram [2,3,7,8] as the global feature type such as contrast changes, or sensitivity to brightness and local binary pattern (LBP) as the local feature type [7,9,10]. Even though comparing all available keyframe pairs and using complex and high-dimension features can offer accurate retrieval results, in practice, there are time complexity problems.

This paper proposes a novel NDVR system that supports large-scale retrieval, which extends the hashing functions to non-linear functions through dimension conversion. In addition, this system contributes to the efficient and accurate retrieval performance. For this, we extract keyframes from each video at regular intervals. Then, we extract both commonly used features (LBP and HSV) and a new image feature from each keyframe. The new image feature, which recent research introduced, can provide more robust information than existing features even if there are geometric changes and complex editing of images because it involves object localization without bounding boxes. We convert a vector set that consists of the extracted features to binary code through a set of hash functions so that the similarity comparison can be efficient by making similar videos more likely to map into the same buckets. Lastly, we calculate similarity to search for NDVs. We examine and compare the NDVR system's effectiveness against previous NDVR systems using the public video collection CC_WEB_VIDEO.

The remainder of this paper goes as follows: Section 2 presents related works for NDVR systems, Sections 3 and 4 explain the basic NDVR system and the proposed method, respectively, Section 5 shows the experiment results, and Section 6 concludes this work.

2. Related Works

Most NDVR approaches carry out retrieval by extracting features from video content. One common strategy is to extract low-level features for characterizing each keyframe after selecting keyframes from videos through uniform sampling. The most common features are color information such as RGB and HSV histogram [2,3,7,8,11]; previous research often referred to these as global features. However, because the characteristics of videos can be changed by major variations such as histogram normalization and color variation, these can be used to retrieve videos that are almost identical to the query video with minor variations [7]. Compared to global features, local features are more robust to complex editing or geometric changes and they generally provide better performance when processing videos with complex scenes. Local binary pattern (LBP) is commonly utilized as the local feature [12]. Features in this category include Difference of Gaussian (DOG) [13], scale invariant feature transform (SIFT) [14], and a mixture of principal component analysis (PCA) and SIFT referred to as PCA-SIFT [15]. In recent studies using these image features, several have investigated areas such as understanding an image and finding special regions in an image [16, 17]. Meanwhile, Bolei Zhou et al. [18] used convolutional neural network (CNN) to propose deep features with localizability applicable to images. We refer to this feature as class activation maps (CAM). CAM for a particular category indicates discriminative image regions used by the convolutional neural network (CNN) to identify the category as shown in **Fig. 1**.

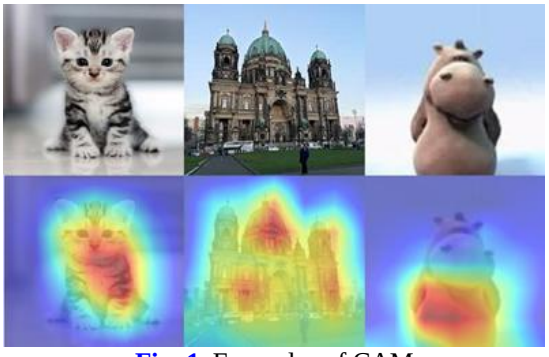

Fig. 1. Examples of CAM

Comparing complex and high-dimension features can provide accurate retrieval results but is very time-consuming. The hashing technique enables large-scale retrieval through rapid pairwise similarity comparison between videos. Classic hashing technologies include locality sensitive hashing (LSH), which converts video data into binary code through a set of random projections. Using LSH means that similar objects are more likely mapped into the same buckets. In addition, previous studies have developed various extensions of LSH. For example, J. Song et. al proposed a learning-based hashing method that jointly learns pseudo class labels and the hash code for given objects based on a discriminant embedding framework driven by linear discriminant analysis [19]. Other examples are spectral hashing (SPH) [10], self-taught

hashing (STH) [9], semi-supervised hashing (SSH) [20], and supervised hashing with kernels [21]. they construct hashcodes using different distance measures.

3. Near-duplicate video retrieval

3.1 Definition of near-duplicate video

NDVs are defined as follows in previous research: Near-duplicate web videos are identical or approximately identical videos close to exact duplicates of each other, but different in file format, encoding parameters, photometric variations (color or lighting changes), editing operations (caption, logo, and border insertion), different lengths, and certain modifications (added or removed frames). A user would clearly identify the videos as "essentially the same" [6]. NDVs are two videos that look the same or approximately the same; the two do not have to be pixel identical for us to consider them duplicates. Whether two videos are duplicates depends entirely on the type of their differences and the comparison's purpose. Originally, precise duplicate videos and near-duplicate videos were different definitions, but this paper includes exact duplicates in our definition of near-duplicate videos.

3.2 Structure of near-duplicate video retrieval

Generally, we use NDVR to search for identical or approximately identical videos and output a ranked list of videos that are relevant to a specific user-provided query. We search NDVs using a constructed retrieval system as follows:

Step 1: Keyframe extraction

The characteristics of video data include both image information and significant information such as audio data. In addition, video data has a concurrent temporal, complex, and informal structure. We generally make summarized information from such huge data by extracting keyframes from the video. The keyframe extraction method extracts keyframes from each video at regular intervals; assuming that we extract n keyframes from videos, later processing steps focus on information provided by those n extracted keyframes.

Step 2: Feature extraction

Feature extraction is a process for generating numerical characteristics based on domain knowledge about data. An important factor for feature extraction is that it requires compact and reliable features because it deals with big data elements such as video. Therefore, previous studies have proposed various approaches using different features. Global features to reflect the whole characteristic of an image are suitable for identifying copies in formatting modifications such as frame resolution changes and format conversion. Unlike global features, we can extract local features after segmenting an image into regions and computing a set of color, texture and shape features for each region. We consider such local features robust and tolerant of geometric and photometric variation. However, there are too many local points for efficient, exhaustive comparison, even between two frames. The notation of the extracted feature vector is as follows: Let $x = \{x_1, x_2, x_3 ... x_d\}$ be the feature vector for one feature type. Assuming that n keyframes are extracted from a video, each feature type includes the size $n \times d$ where d is the length of each feature. For example, we use $x_i = \{x_{i1}, x_{i2}, x_{i3} ... x_{id}\}$ to denote the first feature type for the i th keyframe.

Step 3: Hash code generation

Many previous studies have used hashing when retrieving huge data such as video. This converts an input vector into a fixed-length binary string through hash functions. Generally, a longer hash code provides better performance, but is also more time consuming. The most classic and general hashing approach is a random projection; this generates binary code (hash code) by projecting the extracting features into random lines designated as an auxiliary space, and this hash code makes similar videos more likely to map into the same buckets. This step has a set of s hash functions $\mathrm{H}_i = \{\mathrm{H}_{i1}, \mathrm{H}_{i2}, \mathrm{H}_{i3} ... \mathrm{H}_{is}\}$ each of which takes extracted features as inputs and returns a binary number. Finally, a set of hash functions generates a hash code matrix of size $n \times s$ where n is the number of keyframes.

Step 4: Similarity computation

We generate a unique hash code matrix for each video; we use the Hamming distance between generated hash codes to assess the similarity between videos. This returns a list of videos that possess the highest similarity to the query video.

4. Proposed NDVR system

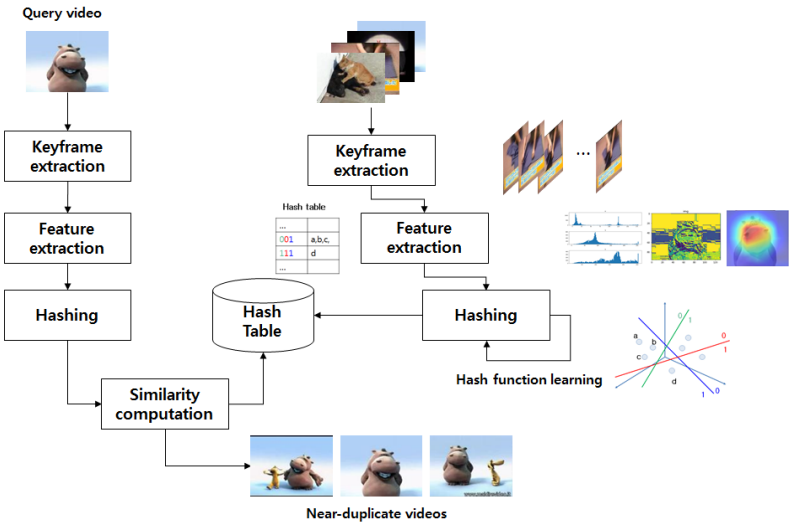

Fig. 2. Flowchart of the proposed NDVR system

Fig. 2 shows a flowchart of the proposed NDVR system. The system consists of keyframe extraction, feature extraction, hashing, and similarity computation. The keyframe extraction method extracts n keyframes from each video in regular intervals as mentioned in Section 3 and then extracts features from the n keyframes.

4.1 Keyframe extraction

As mentioned in Section 3.2, the keyframe extraction method extracts keyframes from videos at regular intervals. In this research, we set the interval such that the method extracts a keyframe every 10 seconds.

4.2 Features

1) HSV

We calculate a color histogram for each keyframe in the video that is a representative global feature and reflects the global statistics or summaries of low-level color in videos. We represent each feature as $HSV_i = \{HSV_1, HSV_2, HSV_3 ... HSV_m\}$; this includes hue, saturation, and value. The equation to normalize the histogram to an overall size is as follows:

$$NormalizedHSV_i = \{NH_1 \cdots NH_m\}, \qquad where \quad NH_i = \frac{1}{M}\sum_{j=1}^{m} HSV_j \tag{1}$$

Where M is the biggest value in histogram and HSV_{ij} is the j th value of the color histogram at keyframe i.

2) Local binary pattern

LBP indicates the texture representation of images and studies commonly utilize this as the local feature as it tends to be more robust in complex editing, photometric, and geometric changes than global features. We can extract LBP features by comparing the brightness of the eight pixels adjacent to the pixel at the center.

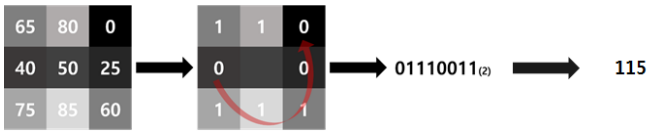

Fig. 3. How to extract an LBP feature

As shown in **Fig. 3**, if the brightness value of the surrounding pixels is greater than or equal to the central pixel, this set as 1; if smaller, this is set as 0. Then, we convert the generated binary number 01110011 to the decimal number 115. **Fig. 4** shows an example LBP feature.

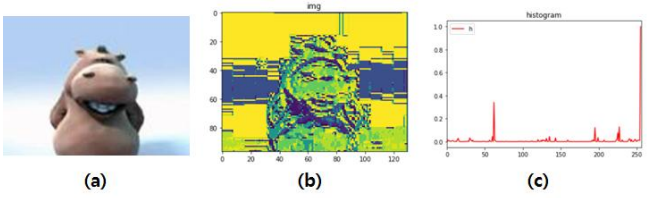

Fig. 4. An example LBP feature (a) original image (b) visualization of LBP (c) the histogram of LBP

The original LBP application uses this histogram as a texture model for the corresponding image region (e.g. the texture of grassplot, forest, land, sky, and object). Many studies used LBP in recognition or detection problems because they can express complex pattern changes even though they were originally developed to classify image textures.

3) Class activation maps

Apart from HSV and LBP, we can use many other feature extraction methods to characterize keyframes. Bolei Zhou et al. recently proposed deep features for discriminative localization. We refer to this feature as class activation maps (CAM) [18]. A class activation map for a particular category indicates the discriminative image regions used by the convolutional neural network (CNN) to identify the category as shown in **Fig. 1**. This easily

identifies the discriminative image regions in a single forward pass for a wide variety of tasks, even when we have not originally trained the network. For example, in **Fig. 5**, even if we train a network using **Fig. 5** (a), it can identify similar image regions in **Fig. 5** (b).

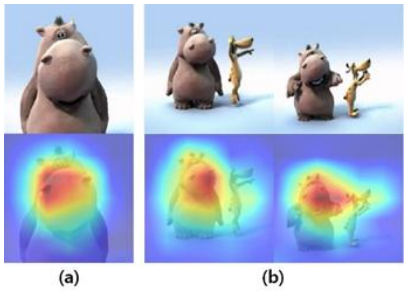

Fig. 5. Localizing class-specific image regions

This research uses a deep CNN design composed of five convolutional layers, five pooling layers, and one fully connected layer. We applied a Rectified Linear Unit (ReLu) activation function on five convolutional layers and a softmax function on one fully connected layer. The first convolutional layer has 32 5x5 filters and uses the same padding; the layer's output decreases by half through the pooling layer. Consider as an example an input image of 224x224x3 components; the resulting output of the first convolutional layer would be 112x112x32 components. The remaining convolutional layers are 5x5 and use the same padding. We identify and categorize the information of videos using the ground truth information provided by the data set. Then, we train the designed CNN using the videos as an input. **Fig. 6** (a) shows the whole architecture and (b) shows examples of CAM features extracted from the network architecture. Here, in the figures of examples, we visualize the values that correspond to the features as Heat Maps.

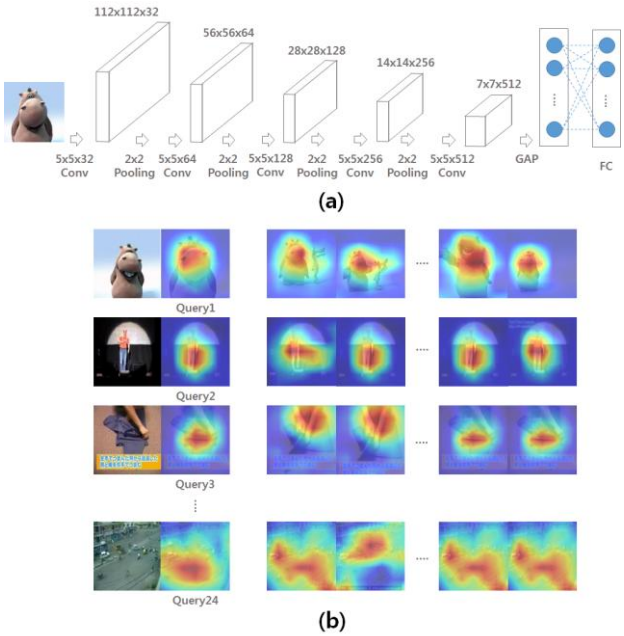

Fig. 6. (a) The network's whole architecture (b) Example CAM features extracted from network (a)

4.3 Hashing

Hashing has drawn attention in large-scale data retrieval. In recent research, Yanbin Hao et al. proposed a stochastic multiview hashing algorithm to facilitate the construction of a large-scale NDVR system [7]. They learned which hash functions applied to linear functions by maximizing the mixture of the generalized retrieval precision and recall scores. Then, they converted multiple features to binary hash code strings. In this study, we extend the hashing functions to non-linear functions through dimension conversion.

Given multiple feature vectors X from a set of n keyframes that involve all feature types, the vector $X_i = \{x_{i1}, x_{i2}, x_{i3}...x_{id}\}$ stores the features of the i th keyframe. We convert these feature vectors to binary hash codes with size s through the hash functions, which we express as $h_i = \{h_{i1}, h_{i2}, h_{i3}...h_{is}\}$ where $h_i \subset \{0,1\}$. We generate hash codes by constructing s hash functions H where $\mathrm{H}_i = \{X_i\}$. These functions are as follows:

$$\tilde{z}_{il} = \sum_{j=0}^{d-1} x_{ij} w^{(1)}{}_{ijl} + b_{il}, \qquad for \quad l = 1 \cdots m \tag{2}$$

$$z_{il} = sigmoid(\tilde{z}_{il}) \tag{3}$$

$$\tilde{h}_{ik} = \sum_{z=0}^{m} z_{iz} w^{(2)}{}_{izs} + b_{is} \qquad for \quad k = 1 \cdots s \tag{4}$$

$$h_{ik} = sigmoid(\tilde{h}_{ik}) \tag{5}$$

The above equations (2) map feature vectors to a one-dimension space by projecting with m linear functions where the size of w is $d \times m$, and the size of b is $1 \times m$. Then, we use sigmoid to make the output vector approximate to 0 or 1. The projected one-dimension vectors map into another space to extend non-linear functions, which we call dimension conversion and is equivalent to a neural network. We express this process as Equation (4), and we apply Equation (5) to make the hash code 0 or 1 by using a thresholding method. Generally, we refer to $\tilde{h}_{ik}$ as the relaxed hash code. In NDVR, one classical way to generate hash code for a video is to process the relaxed hash codes of its representative keyframes by first performing averaging and then thresholding operations [21].

$$h_{ik} = Threshold\left(\frac{1}{|ind_i|} \sum_{j \in ind_i} h_{jk}\right) \tag{6}$$

Equation (6) shows a generated hash code vector for a video in which ind_i is the set of keyframe indices for the video and $|ind_i|$ is its cardinality. Finally, we generated the $V \times s$ hash code matrix for all videos.

When making a list of output videos close to a query video, accurately measuring the similarity between the videos is very significant. Therefore, this study focuses on how to compute the optimal hash codes from the feature vectors to ensure correct similarity information between videos. When there is ground truth information available regarding the relevance between videos, constructing probabilities by rewarding actually related videos with

a score of 1 and non-related or unknown ones with a score of 0 is helpful. We refer to this as p and express this as:

$$p_{i|j} = \begin{cases} 1 & \text{if ith video is near-duplicated video with jth video} \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

Assuming that each $p_{i|j}$ represents the probability of the i th video for query j, a natural way of learning the hash functions is re-computing such probabilities in the space of hash functions and minimizing the difference between two sets of probabilities. Therefore, the probability equation q is:

$$q_{i|j} = \frac{\exp(-\left\|h_i - h_j\right\|_2^2)}{\sum_{i \neq k} \exp(-\left\|h_i - h_k\right\|_2^2)} \tag{8}$$

We increase the probability that we have extracted x_i and x_j from near-duplicated videos by learning hash functions using available ground truth information. We can assess the hashing's quality by examining how well the probabilities of p and q match. We measure the difference between two conditional probabilities p and q using the following KL-divergence method:

$$O = \sum_i \sum_j p_{ij} \cdot \log(\frac{p_{ij}}{q_{ij}}) \tag{9}$$

According to the decrease in cross-entropy value, we have increased the probability of similar objects mapping into the same group. Hash functions in Equations (2), (3), (4), and (5) consist of weight and bias parameters. Therefore, we can convert the optimization problem for hash functions into a minimization problem of composite KL-divergence values. We solve this problem by employing a gradient descent algorithm. We can compute the gradient using the following compound function derivation.

$$\frac{\partial O}{\partial w^{(2)}} = \frac{\partial O}{\partial h_i} \cdot \frac{\partial h_i}{\partial w^{(2)}} \tag{10}$$

$$\frac{\partial O}{\partial w^{(1)}} = \frac{\partial O}{\partial h_i} \cdot \frac{\partial h_i}{\partial z_i} \cdot \frac{\partial z_i}{\partial w^{(1)}} \tag{11}$$

As has been established in many machine-learning studies, placing bias on the 0^{th} weight vector means that there is no need to determine the gradient of the bias. Evidently, the targeted gradients depend on the different components of $\frac{\partial O}{\partial z_i}$, $\frac{\partial z_i}{\partial w^{(2)}}$, $\frac{\partial z_i}{\partial h}$ and $\frac{\partial z_i}{\partial w^{(1)}}$. The components can be calculated as follows.

$$\frac{\partial O}{\partial h_i} = 2\sum_j (h_i - h_j)(p_{ij} - q_{ij} + p_{ji} - q_{ji}) \tag{12}$$

$$\frac{\partial h_i}{\partial w^{(2)}} = \partial sigmoid(\tilde{h}_i) \cdot z_i \tag{13}$$

$$\frac{\partial h_i}{\partial z_i} = \partial sigmoid(\tilde{h}_i) \cdot w_i \tag{14}$$

$$\frac{\partial z_i}{\partial w^{(1)}} = \partial sigmoid(\tilde{z}_i) \cdot x_i \tag{15}$$

Substituting (12), (13), (14), and (15) into (10) and (11) permits the derivation of the complete formulations of $\frac{\partial O}{\partial w^{(2)}}$ and $\frac{\partial O}{\partial w^{(2)}}$. Generating binary code through learned hash functions means that similar videos are more likely to map into the same buckets. In addition, if we can calculate similarity using binary code, we can decrease the retrieval speed because we have only calculated the Hamming distance, which uses a bit operation. This leads to avoiding costly pairwise keyframe comparisons and can effectively improve the retrieval efficiency. In terms of video retrieval, the computational complexity of computing a hash code has a low cost of around O(d^3), where d is the length of the input vector. This phase has very simple operations such as linear combination, sigmoid, and thresholding compared to the retrieval phase, so it does not affect the retrieval time. In the retrieval phase, the bit count operations for the hamming distance calculation leads to a very fast online NDVR system. Therefore, NDVs can be found by linear search O(n)[22]. This paper demonstrates its efficiency in the results section.

5. Experiments and results

5.1 Dataset and metric

This study tested the proposed method through experiments using a publically available web video dataset. The CC_WEB_VIDEO dataset [6] consists of 12,790 video clips downloaded from video sharing websites such as YouTube, Google, and Yahoo! through keyword search, and is organized into 24 sets. The set has 398,015 keyframes in total. In previous research, two non-expert assessors were asked to watch videos from this dataset with one query at a time, and assessors labeled all videos with statuses (E: Exact duplicate, S: Similar video, V: Different version, M: Major change, L: Longer version, X: Dissimilar video, or -1: Video does not exist) according to their judgment. Therefore, this dataset provides reliable ground truth information for all video clips. In addition, the most popular video was selected as the seed video for each query for near-duplicate video retrieval.

Retrieval performance evaluations commonly use the classic metric of the mean average precision (MAP). We use the precision–recall curve and MAP.

$$\mathrm{Re}\,call = |G \cap D| / |G|, \qquad \mathrm{Pr}\,ecision = |G \cap D| / |D| \tag{10}$$

Where G is the ground truth set of redundant videos and D is the detected one.

5.1 Experimental setup

The hash code length s influences retrieval performance and efficiency, so selecting the optimal length hash code is important. We prove the retrieval performance according to the length s by testing changing the length from 500 and 1,000 with step size 50. Here, we applied LBP, HSV, and CAM for this test and we set the same experimental conditions except the hash code length.

Table 1. Applications in each class

Hash Code Length	MAP	Time(s)
S = 500	0.6798	0.33
S = 550	0.6865	0.37
…	…	…
S = 800	0.9738	0.59
S = 850	0.9765	0.65
S = 900	0.9777	0.68
S = 950	0.9861	0.75
S = 1,000	0.9898	0.78

Table 1 shows the changes in MAP performance. We can see from the results that it provides a quite similar retrieval performance in the range 800–1,000 even though the hash code lengths differ. In addition, in the case of the retrieval time, the longer the hash code, the greater the retrieval time margin. Therefore, we fixed the hash code length at s = 900 while considering the MAP and computation time.

5.2 Baseline

In this section, we describe baselines algorithms to compare our method.

1) Spectral hashing (SH) [10]

We base the spectral hashing on analyzing the k smallest single-dimension analytical Eigen functions of L_p using a rectangular approximation along every PCA direction. This uses spectral relaxation such as PCA. Then, they find the smallest element of the data whose dimensions are reduced by PCA. Finally, we convert this to binary code along with the k smallest eigenvalues.

2) Multiple feature hashing (MFH) [23]

This system proposed a sophisticated multiview method called MFH by extending SPH. MFH learns the training videos' hash codes and a group of hash functions to generate hash codes for videos outside the training set; it encodes the information provided by the HSV and LBP features as a neighbor graph and seeks a hash function to preserve the desired neighbor structure.

3) Stochastic Multiview hashing (SMVH) [7]

This method learns binary strings to characterize data samples by combining multiple feature types and auxiliary information through a stochastic matching procedure of neighborhood probabilistic models. This learns the mapping functions stochastically by maximizing a mixture of the generalized retrieval precision and recall scores. The scores are approximated by the composite Kullback–Leibler (KL) divergence computed between two probabilistic models constructed in the original feature space and a relaxed hash code space.

4) Self-taught hashing (STH) [9]

This system relies on the hashing method STH, which shares a similar hash code training procedure to SPH but achieves out-of-sample extension through a different scheme based on

linear SVM.

5) Hierarchical fusing (HF) [6]

This system combines the global and local features, by firstly using the color histogram signature to detect the NDVs with high confidence and filtering out the very novel ones, and then performing a pairwise comparison based on the local features to further determine the uncertain videos.

6) Unsupervised Stochastic Multiview hashing (USMVH) [7]

This system is an unsupervised version of SMVH

The following section describes the proposed method and the results of the overall comparison with other methods.

5.3 Results

For the experiment, we extracted 768 HSV and 256 LBP features and 1024 CAM features from each keyframe. We compited the retrieval speed using Python 3.5 running on a server with an Intel i7 4770 CPU, 16 GB RAM, and 64-bit Windows 7 operating system.

Table 2. Experiment results

Hashing method	Features	MAP	Time(s)
None	HSV and LBP	85.61%	17.27
None	HSV, LBP, and CAM	91.45%	29.50
SH [10]	HSV and LBP	86.4%	0.28
MFH [23]	HSV and LBP	92.8%	0.27
SMVH [7]	HSV and LBP	97.1%	0.28
STH [9]	HSV and LBP	93.2%	0.28
HF [6]	HSV, PCA-SIFT[24]	95.2%	8
USMVH [7]	HSV and LBP	95.5%	0.30
SMVH	HSV, Haar [25]	94.3%	0.27
SMVH	HSV, Fens [26]	96.7%	0.32
Proposed	HSV, LBP, and CAM	**98.98%**	0.75

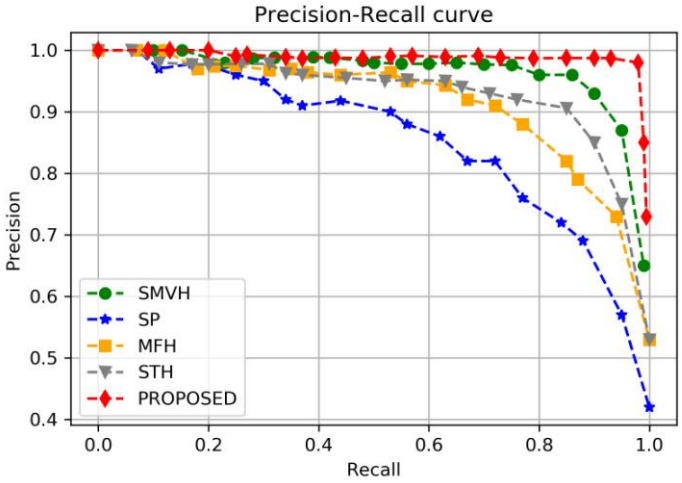

Fig. 7. Experiment results with a Precision–Recall curve

Table 2 summarizes the MAP performance of all methods and the retrieval speed for CC_WEB_VIDEO. In addition, **Fig. 7** shows the Precision-Recall curve for the experiment results. According to this result, LBP, HSV, and CAM features provide better performance than cases that use LBP and HSV. In addition, the proposed method (98.98%) is better than the other hashing method with LBP and HSV. **Table 2** also compares the retrieval speed. The proposed method's set of feature vectors is relatively larger than other methods that use LBP and HSV, so the proposed method needs more hash functions. Therefore, we can see that the proposed method has a longer retrieval time. However, the retrieval speed is dramarically decreased compared to not using hashing methods.

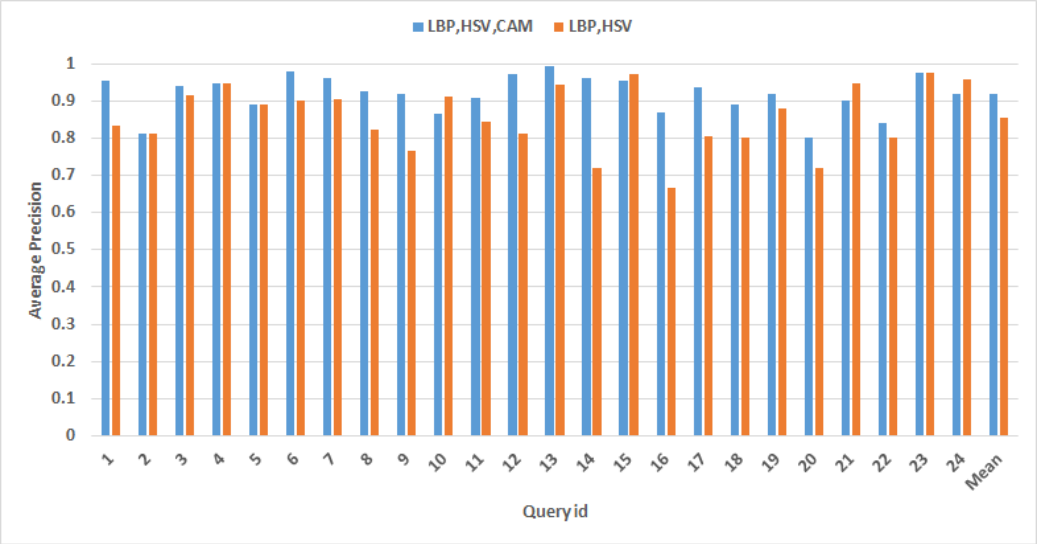

Fig. 8. AP performance comparison

Finally, we tested the average precision (AP) performance of a different feature set over each of the 24 queries. For most queries, a feature set that includes LBP, HSV, and CAM shows better performance that one that only includes LBP and HSV. Although there are a few individual cases such as Q10, Q16, and Q24 for which the LBP and HSV feature set shows better performance than LBP, HSV, and CAM, this does not change the overall conclusion when taking all queries into account.

5. Conclusion

This paper proposes a novel NDVR system that supports large-scale retrieval. For this, we extracted keyframes from each video at regular intervals. Then, we extracted both commonly used features (LBP, HSV) and the new image feature from each keyframe. We accurately retrieved the NDVs by considering the new image feature provided auxiliary information such as the object localization of keyframes. The extracted features make up a vector set that we convert to simple binary strings through a set of mapping functions such that the similarity comparison can be efficient. Lastly, we calculated the similarity to search for NDVs. We examined the NDVR system's effectiveness and compared it against previous NDVR systems using the public video collection CC_WEB_VIDEO. The proposed method dealt with important accuracy issues in recent NDVR studies and contributed to performance improvement.

Acknowledgment

This research was supported by a 2019 Research Grant from Sangmyung University.

References

[1] Liu, J., Huang, Z., Cai, H., Shen, H. T., Ngo, C. W., and Wang, W., "Near-duplicate video retrieval: Current research and future trends," *ACM Comput. Surv.*, vol. 45, no. 4, Art. no. 44., pp. 218–227, 2013. Article (CrossRef Link)

[2] J. Song, Y. Yang, Z. Huang, H. T. Shen, and R. Hong, "Multiple feature hashing for real-time large scale near-duplicate video retrieval," in *Proc. of 19th ACM Int. Conf. Multimedia*, pp. 423–432, 2011 Article (CrossRef Link)

[3] M. Cherubini, R. De Oliveira, and N. Oliver, "Understanding near-duplicate videos: A user-centric approach," in *Proc. of 17th ACM Int. Conf. Multimedia*, pp. 35–44, 2009. Article (CrossRef Link)

[4] H. T. Shen, X. Zhou, Z. Huang, J. Shao, and X. Zhou, "UQLIPS: A real-time near-duplicate video clip detection system," in *Proc. 33rd Int. Conf. Very Large Data Bases*, pp. 1374–1377, 2007.

[5] H.-K. Tan, C.-W. Ngo, R. Hong, and T.-S. Chua, "Scalable detection of partial near-duplicate videos by visual-temporal consistency," in *Proc. of 17th ACM Int. Conf. Multimedia*, pp. 145–154, 2009. Article (CrossRef Link)

[6] X. Wu, A. G. Hauptmann, and C.-W. Ngo, "Practical elimination of near-duplicates from web video search," in *Proc. of 15th ACM Int. Conf. Multimedia*, pp. 218-227, 2007. Article (CrossRef Link)

[7] Yanbin Hao, Tingting Mu, Richang Hong, Meng Wang, Ning An, John Y. Goulermas, "Stochastic Multiview Hashing for Large-Scale Near-Duplicate Video Retrieval," *IEEE Transactions on Multimedia*, Vol. 19, No. 1, pp. 1-14, 2016. Article (CrossRef Link)

[8] L. Shang, L. Yang, F. Wang, K.-P. Chan, and X.-S. Hua, "Real-time large scale near-duplicate web video retrieval," in *Proc. of 18th ACM Int. Conf. Multimedia*, pp. 531–540, 2010. Article (CrossRef Link)

[9] D. Zhang, J. Wang, D. Cai, and J. Lu, "Self-taught hashing for fast similarity search," in *Proc. of 33rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, pp. 18–25, 2010. Article (CrossRef Link)

[10] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Proc. of Adv. Neural Inf. Process. Syst. Conf.*, pp. 1753–1760, 2009.

[11] J. Yuan, L.-Y. Duan, Q. Tian, S. Ranganath, and C. Xu, "Fast and robust short video clip search for copy detection," in *Proc. of Adv. Multimedia Inf. Process. Conf.*, pp. 479–488, 2004. Article (CrossRef Link)

[12] G. Zhao and M. Pietikainen, "Dynamic texture recognition using local binary patterns with an application to facial expressions," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 6, pp. 915–928, Jun. 2007. Article (CrossRef Link)

[13] D.-G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004. Article (CrossRef Link)

[14] D.-G. Lowe, "Object recognition from local scale-invariant features," in *Proc. of Int. Conf. Comput. Vis.*, pp. 1150–1157, 1999. Article (CrossRef Link)

[15] Y. Ke and R. Sukthankar, "PCA-SIFT: A more distinctive representation for local image descriptors," in *Proc. of IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recog.*, pp. 506–513, Jun.-Jul. 2004. Article (CrossRef Link)

[16] Chenggang Yan, Liang Li, Chunjie Zhang, Bingtao Liu, Yongdong Zhang, Qionghai Dai, "Cross-modality Bridging and Knowledge Transferring for Image Understanding," *IEEE Transactions on Multimedia. (Early Access)*, pp. 1-1, 2019 Article (CrossRef Link)

[17] Chenggang Yan, Liang Li, Chunjie Zhang, Bingtao Liu, Yongdong Zhang, Qionghai Dai, "A Fast Uyghur Text Detector for Complex Background Images," *IEEE Transactions on Multimedia*,Vol. 20, Issue. 12, pp. 3389-3398, 2018. Article (CrossRef Link)

[18] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, Antonio Torralba, "Learning Deep Features for Discriminative Localization," in *Proc. of Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*, PP. 2921-2929, 2016. Article (CrossRef Link)
[19] J. Song, L. Gao, Y. Yan, D. Zhang, and N. Sebe, "Supervised hashing with pseudo labels for scalable multimedia retrieval," in *Proc. of 23rd ACM Int. Conf. Multimedia*, pp. 827–830, 2015. Article (CrossRef Link)
[20] R. Salakhutdinov and G. E. Hinton, "Learning a nonlinear embedding by preserving class neighbourhood structure," in *Proc. of 11th Int. Conf. Artif. Intell. Statist*., pp. 412–419, 2007.
[21] W. Liu, J.Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang, "Supervised hashing with kernels," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog*., pp. 2074–2081, 2012. Article (CrossRef Link)
[22] A. Gionis et al., "Similarity search in high dimensions via hashing," in *Proc. of 25th Int. Conf. Very Large Data Bases*, pp. 518–529, 1999.
[23] J. Song, Y. Yang, Z. Huang, H. T. Shen, and J. Luo, "Effective multiple feature hashing for large-scale near-duplicate video retrieval," *IEEE Trans. Multimedia*, vol. 15, no. 8, pp. 1997–2008, Dec. 2013. Article (CrossRef Link)
[24] David G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004. Article (CrossRef Link)
[25] P. Viola, M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. of IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, 2001. Article (CrossRef Link)
[26] Mustafa Ozuysal, Michael Calonder, Vincent Lepetit, Pascal Fua, "Fast Keypoint Recognition Using Random Ferns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 3, pp. 448–461, Jan. 2009. Article (CrossRef Link)

Sung-Woo Byun received his BS degree from the department of digital media technology from SangMyung University, Seoul, Korea. He is currently a PhD student in the department of computer science, SangMyung University. His main research interests include signal processing, artificial intelligence, and personalized media processing.

Seok-Pil Lee received his BS and MS degrees in electrical engineering from Yonsei University, Seoul, Korea, in 1990 and 1992, respectively. In 1997, he earned a PhD degree in electrical engineering, also at Yonsei University. From 1997 to 2002, he worked as a senior research staff member at Daewoo Electronics, Seoul, Korea. From 2002 to 2012, he worked as a head of a digital media research center of the Korea Electronics Technology Institute. He worked also as a research staff member at Georgia Tech., Atlanta, USA from 2010 to 2011. He is currently a Professor at the department of electronic engineering, SangMyung University. His research interests include artificial intelligence, audio digital processing, and multimedia searching