Multimed Tools Appl (2019) 78:5233–5254
DOI 10.1007/s11042-017-4969-2

Multimedia file forensics system exploiting file similarity search

Min-Ja Kim[1] · **Chuck Yoo**[1] · **Young-Woong Ko**[2]

Received: 8 January 2017 / Revised: 12 April 2017 / Accepted: 20 June 2017 /
Published online: 5 July 2017

Abstract With the fast increase of multimedia contents, efficient forensics investigation methods for multimedia files have been required. In multimedia files, the similarity means that the identical media (audio and video) data are existing among multimedia files. This paper proposes an efficient multimedia file forensics system based on file similarity search of video contents. The proposed system needs two key techniques. First is a media-aware information detection technique. The first critical step for the similarity search is to find the meaningful keyframes or key sequences in the shots through a multimedia file, in order to recognize altered files from the same source file. Second is a video fingerprint-based technique (VFB) for file similarity search. The byte for byte comparison is an inefficient similarity searching method for large files such as multimedia. The VFB technique is an efficient method to extract video features from the large multimedia files. It also provides an independent media-aware identification method for detecting alterations to the source video file (e.g., frame rates, resolutions, and formats, etc.). In this paper, we focus on two key challenges: to generate robust video fingerprints by finding meaningful boundaries of a multimedia file, and to measure video similarity by using fingerprint-based matching. Our evaluation shows that the proposed system is possible to apply to realistic multimedia file forensics tools.

Keywords Multimedia file forensics · File similarity search · Video fingerprint · media-aware information detection · fingerprint-based matching

✉ Young-Woong Ko
yuko@hallym.ac.kr

Min-Ja Kim
mjkim@gmail.com; hxy@korea.ac.kr

[1] Department of Computer Science and Engineering, Korea University, Seoul, South Korea

[2] Department of Computer Engineering, Hallym University, Chuncheon, Gangwon 200-702, Republic of Korea

1 Introduction

Over the past few years, thanks to the development of the network environment, multimedia contents have been consistently increasing on the Internet as well as in the mobile environment [9]. The proliferation of User Generated Content (UGC) shows explosive demand for multimedia contents. The development of various mobile devices that support multimedia has accelerated the usage demands for multimedia contents. Due to the spread of mobile devices, it has been essential to generate media contents supporting diverse formats from original sources, in order to make playback possible on various devices [12]. To this end, the same video source could be created in diverse formats or different resolutions by many content providers. Generally, content providers distribute videos to content distributers. The obtained content from content providers has to be re-encoded according to various device options in order to support different resolutions, codecs, or formats. The same video can be produced in many different resolutions or formats by many different content providers. This means that there are alternative versions of the same content available on different devices and hosting sites [24]. Table 1 indicates that different types of video streams, audio streams, and containers supported differently according to mobile phone models. Owing to this diversity being the cause of the heterogeneity for multimedia contents, newly re-encoded versions can have lots of differences on the binary level from the original versions. This results in serious difficulties in finding malicious multimedia contents. The modified media contents which have the different formats or frame rates, or qualities can be seen as the same or similar to the naked eye. When it compares the modified versions with the source version at the binary level, the general expectations are that the result of similarity between two files should be close to almost 99%. However contrary to these expectations, it produces unreliable results even though there are very small changes.

We argue that adapting existing file similarity search schemes (e.g., the block hashing scheme) into multimedia files could yield worse results. There are two reasons behind this. The first is that the block hashing schemes among file similarity search approaches are not appropriate to apply to multimedia files. Even when two videos are composed of a set of identical samples, media samples (audio and video) could be placed in a different order. For example, video and audio samples can be interleaved in the different order from those in the source. The difference of interleaving leads to change of metadata regarding file structure. Consequently, the block similarities from two versions through the binary comparison could be

Table 1 Different types of video and audio streams, and containers supported by mobile phone models

Manufacturer	Model	Container	Video stream	Audio stream
Apple	IPhone 4	MOV	H.264	MP4A
	IPhone 6, 6S, 6SPlUS	MOV, MP4, AVI	H.264, MP4V	AAC-LC
Google	Nexus 7	3GP	H.264	MP4A
LG	KU990	3GP, AVI	DivX, MP4V, H263	MP3, AMR-NB
Nokia	6710, E61i, E65	3GP, MP4	H.263, MP4V	MP4A, AMR-NB
Motorola	MileStone	3GP	MP4V	MP4A
Samsung	Galaxy S7 edge	MP4, 3GP, 3G2, WMV, ASF, AVI, FLV, etc.	M4 V, H.263, H.264, FLV1, FLV4, etc.	MP3, M4A, 3GA, AAC, OGG, WMA, AMR, FLAC, etc.

significantly different. The second reason is that parallel encoding can affect the encoding output. In case of using the same encoder with the same preset, re-encoding sometimes generates different outputs on the binary level. This is affected by the number of CPU cores employed by the encoder [4]. Even for seemingly identical video contents, irregularity in the encoding process could produce completely different binaries. In other words, the exact same content could be also encoded into a completely different binary. For example, the same music video provided by different mobile or web sites could be created into different binaries depending on the encoders and devices. When applying existing file forensics methods to video contents, in these above two cases, the rates of similarity and similarity across multimedia files may be more less.

In this paper, we propose an effective multimedia forensics system based on file similarity search. We present two key mechanisms. The first is media-aware information detection. To search similarities among video files, it is important to obtain a semantic understanding of the video contents. This allows independence from common video processing steps, including lossy compression, resizing, frame rate or formats change, etc. To this end, a video file is segmented into shots. The shots are divided into different groups based their visual similarity. The segmented shot can be served as a video chunk. A keyframe will be selected as the representative frame of the shot. Therefore, each keyframe – that is, the first frame extracted from each shot – can be a media-aware boundary anchor. The second mechanism is the video fingerprint-based techniques (VFB). Multimedia files may undergo diverse distortions such as resizing, and format and frame rate change. Therefore, the binary data of media files cannot directly be compared to each other. The perceptual video hashing mechanism provides an efficient way to determine whether two media files are "perceptual identical", which is processed without changing its semantics. One example of "perceptual identical" approach is to determine whether one document is a copy of another one. A hash, which is a message digest that is generated by a one-way cryptographic calculation, is used to identify an object if the hash of the original copy exists. In other words, an extracted hash value from a keyframe is regarded as a video fingerprint. A video fingerprint would be an identifier of each keyframe. We use a video fingerprint-based matching technique in order to get results of similarity search. Video fingerprint-based matching compares whether the same video fingerprint from the original content exists in the new version.

The main goal of our research is to obtain a similarity result from multimedia files. It provides the independence and robustness against transformations of video processing through the comparison of the media-aware information itself. The focus of this paper is on the media-aware information detection as extracting keyframes and generating video fingerprints. In conclusion, our proposed system should find the similarity by using video fingerprints. The evaluation results show the performance of similarity analysis through the video fingerprint-based matching.

The remainder of the paper is organized as follows. In Section 2, we briefly review file similarity search schemes and video fingerprinting technique. Section 3 describes our system design and architecture. In Section 4, we explain media-aware information detection mechanism including the shot boundary detection and key frame extraction. In Section 5, we describe video fingerprint-based techniques associated with the way of generating video fingerprints and matching them. We evaluate our multimedia similarity search and fingerprint-based matching scheme, in Section 6, and conclude in Section 7.

2 Related work

2.1 File similarity search schemes

The primary method in file forensics is based on the analysis of file similarity and redundancy. File similarity and redundancy search schemes have been used in many areas, such as file synchronization and network deduplication (e.g., Active Sync [20], HotSync [14], CPISync [27], LLRFS [33], Tpsync [32], Rsync [29], Venti [25], and LBFS [21]). Block hashing schemes are the most important key idea used for file similarity and redundancy search. Block hashing schemes divide the file into numerous small blocks, and the hash keys for identifying each of the blocks are extracted by the hash function. The process of splitting a file into smaller blocks is called chunking. Chunk-based deduplication is a very effective technique for removing data redundancy [10, 15, 25]. Data redundancy means that identical chunks exist in another file. File similarity indicates how often data redundancy occurs in another file. The first process divides input data streams into fixed or variable-length chunks. A cryptographic hash function is a particular mathematical algorithm that serves as the security mechanism that produces a hash value or message digest for a specific data object. Fig.1 shows the difference of methodological approach by comparing the fixed-length and variable-length chunking methods. A cryptographic hash function is a particular mathematical algorithm that serves as the security mechanism that produces a hash value or message digest for a specific data object.

As shown in Fig. 2, the cryptographic hash value of each chunk can be the chunk ID, as an identifier of each chunk. A chunk ID is used to determine whether that chunk was backed up before. Chunks with the same chunk ID are assumed identical. New chunks are stored, and references are updated for duplicate chunks. Chunk-based deduplication is very effective for backup workloads. Backups tend to be changed, mainly through small changes, additions, and deletions [35].

There are two well-known chunking methods to split a file into multiple blocks. Fixed-length chunking [25] lets a file be divided into a number of fixed-sized blocks, and then applies the hash function to extract the hash key from each block. Fixed-length chunking is very simple, light-weight scheme and is easier to implement than variable-length chunking [15]. However, the drawback of fixed-length chunking approach is the data-shifting problem. For instance, if some of the data are inserted or deleted at the boundary of block, the result of hash value can be completely altered differently. Thus, in terms of accuracy through the similarity analysis, it usually results in bad performance. Content-based chunking or variable-length chunking is where each block is partitioned by using special predefined value called the “anchor”. This scheme can prevent the data-shifting problem of the fixed-length chunking approach as with the fast duplication approach. Therefore, variable-length chunking is more

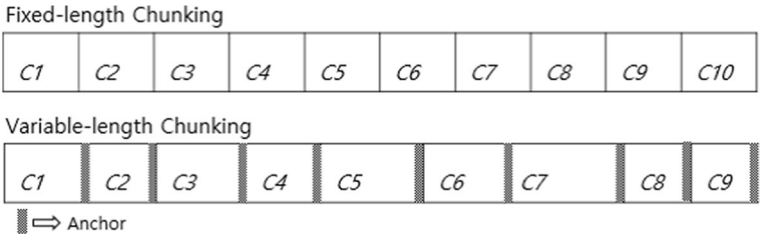

Fig. 1 The concept of FLC and VLC

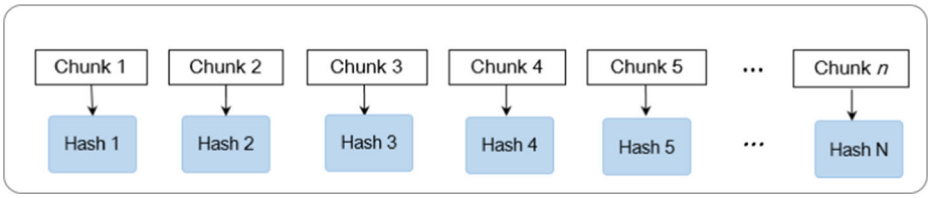

Fig. 2 Block hashing schemes for file similarity search

flexible than fixed-length chunking but requires more computation time to achieve good file similarity.

2.2 Video fingerprinting

The video fingerprinting technique is an efficient solution for video content identification and management of multimedia files in UGC sites or P2P networks, and for video copy detection such as copyright violation. The "identical" or "similar" in video means that there is "perceptual consistency" when people watch the video. However, the "identical" is not the same concept as the "similar". The "similar" in videos refers to inclusion of any kind of distortions and transformations to multimedia contents. Transformations in multimedia include the changes, such as to format, codec, resolution, frame rate, bitrate. The video is the series of images, and can be distorted by cropping, scaling, frame dropping, text inserting. Video copy detection has been actively studied [6, 31]. The video copy is a modified video derived from another one, by means of diverse spatial-temporal transformations [6]. Basically, video copy detection means determining similarity with other copies. It is quite important to search for similarity between the original and copied version in multimedia files. "Robust video hashing" is another term explaining video fingerprinting. Robust video hashing is a well-known approach for being fast and robust [7]. It was basically derived from cryptographic hashing like MD5, SHA-1, LSH(Locality Sensitive Hashing)[11] and SHA-256. The goal of it is to make robust hashes to transformations and distortions that do not change human's perception when people view the video content with the human eye. For this reason, it is often called perceptual hashing. Perceptual hashing schemes have been studied by applying them to various media types (e.g., audio, video, image, text). Coskun [7] proposed the robust video hash function based on low-frequency components of the 3D DCT (Discrete Cosine Transform) using the normalized video sequence. Steinebach [28] introduced cropping-resistant robust image hashing by combining image segmentation and block mean image hashing. Roover [8] proposed the image hashing algorithm that is robust to geometrical operation by using the Radon transform. In [1, 13], existing audio hashing algorithms are explained.

A video fingerprint is a string with the signature extracted from the video content. Video similarity can be obtained by comparing the signature in two versions of a video. Video fingerprinting algorithms can be classified into four main features of videos: spatial, temporal, color, and transform domain. Spatial features include luminance patterns, edge histograms, and gradient patterns. Temporal features are computed between frames with the same temporal direction, and include motion vector patterns and frame difference measures. Color feature utilizes the color space, such as RGB or YUV. Transform-domain is to compute coefficients of video using a DCT or Wavelet transform. DCT converts the pixel values to frequency values. It is the most robust scheme to detect similarity, but has high computation complexity. In [7], the DCT-based algorithm is proposed, and uses the lower frequencies in order to capture the

"perceptual feature" of the image. With these four features, video fingerprinting schemes can reach the compromise with performance properties [17], including uniqueness, robustness, and compactness.

3 Overview of the system design

This paper proposes an efficient multimedia file forensics system. In general, multimedia characteristic are large size, complexity, and heterogeneity. For this reason, the design of multimedia file forensics system is required for high-level understanding about image and video processing, or computer vision. In this section, we describe the design goals and overall architecture for the system with considering multimedia characteristics.

Fig. 3 illustrates the schematic diagram of the system. As shown in Fig. 3, the system processes the following three steps to get results of media forensics investigation:

Media-aware information detection.

Video fingerprint-based techniques.

Similarity measurement.

The detailed mechanisms of these three steps is described in the following sections 4 and 5.

3.1 Design goals and approaches

We define three design goals and corresponding approaches as the most important properties in our system as follow.

Accuracy This system should find the same or similar contents that have undergone various processing, transformation, and manipulations, while not distinguishing perceptually identical content as false copies. The ultimate goal is to achieve high accuracy. However, this is challenging because of the multimedia heterogeneity. (e.g., cropping, scaling, blurring, changing frame rates and format, editing partial) For achieving this goal, our approach is to extract the video fingerprints, which is robust to as many transformations as possible.

Computational efficiency The system should have low computational overheads to obtain results of similarity measurement, and ensure fast response time. The ultimate goal is to extract the video fingerprints by minimizing computation complexity, and to utilize fast matching. For achieving this goal, our approach is to utilize as simple a video fingerprinting algorithm as possible without sacrificing accuracy, and make the metadata compactly by concatenating compact hash values. This enables a fast calculation and comparison.

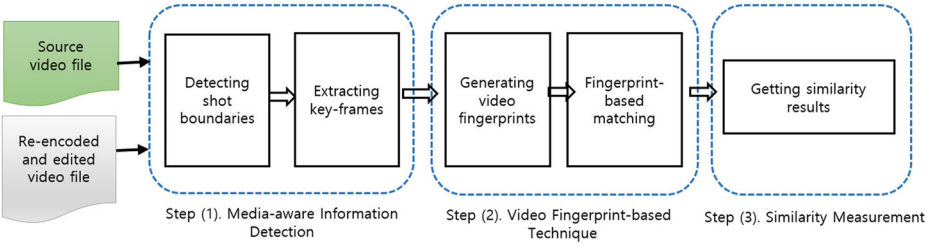

Fig. 3 The schematic diagram of the system

Matching and search efficiency The system should optimize the metadata size to improve the efficiency in matching and search. The ultimate goal is to reduce the amount of comparison. For achieving this goal, we will attempt to extract key frames to the vicinity of 1% of the total number of video frames.

3.2 Overall architecture

The system has three main processing steps: (1) media-aware information detection, (2) video fingerprint-based technique, and (3) similarity measurement.

Step (1) focuses on shot boundary detection and extracting key-frames from the detected shots. Media-aware information detection utilizes shot boundary detection [5] in order to extract discriminating features of the video as "perceptual identical." Video is a set of consecutive frames, and can be segmented into shots. The first frame of each shot can be used as the boundary anchor of each segmented video chunk. Extracted key frames play an important role as a representative of each shot. They can be basic unit frames to detect the similarity between comparable videos.

Step (2) generates the video fingerprints from extracted key frames, and make the hash list. The hash list is created by concatenating a generated fingerprint sequentially. For the system, the hash list would be the final result for this step. We would use the hash list in order to proceed the forensics investigation of multimedia files. It provides a simple and lightweight way for comparison and calculation due to the compactness of hash.

Finally, step (3) obtains a result of similarity by comparing the hash lists of the new version and the original one. Our system can get a result by comparing the two sets of hash lists generated from each file. The result indicates how many matches there are, if any, between the two hash lists.

Fig 4 shows a more detailed technical methodology for the three steps.

4 Media-aware information detection

In this section, we explain how to detect media-aware information through content-based analysis of video. Many studies on analyzing video based on the content have been conducted

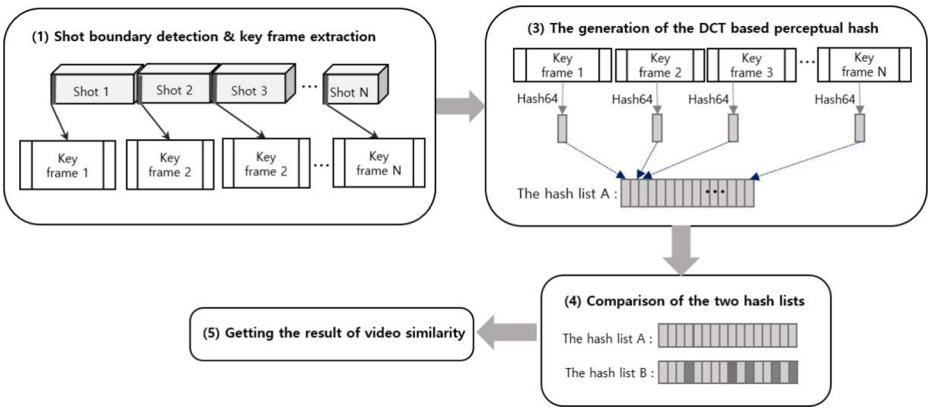

Fig. 4 The detailed architecture of the proposed system

for several years. The techniques of the relevant studies can be broadly classified into two parts: shot boundary detection, and key frame extraction. The goal of these two techniques is to detect semantic units in multimedia files, and to select key frames from them to construct compact representations as representative information. Fig.5 illustrates an overview of shot boundary detection and key frame extraction. As shown in the figure, video can be segmented into shots and scenes as large meaningful units. Video shots are the basic meaningful unit of video analysis.

4.1 Shot boundary detection

Shot boundary detection is the first step that accomplishes video analysis, and it helps to construct the system more easily without referencing all video frames. This technique is to detect the boundaries between shots in order to make video data into more compact forms. It finds the beginnings and ends of the shots, by splitting a video into semantic segments based on visual similarities and temporal relations. Shots are defined as sequences of images captured without interruption by a camera [17]. A video scene consists of several semantically related shots. Therefore, a scene is a longer temporal segment than a shot. Shot boundary detection algorithms are operated by extracting one or more features from a video frame.

There are four main approaches in these features for shot boundary detection.

- Color/luminance histogram: It computes the color of every pixel in each frame and makes a histogram using these values. The histograms of adjacent frames are compared successively to each other. It is easy to compute and widely used [18].
- Luminance values: It compares the luminance values, but is susceptible to illumination changes. To overcome disadvantage like this, it needs to use one more averages of the values in appropriate color space.
- Edges: It is to look for edges in each frame. It compares edge information from successive frames. The detection of significant change by comparison means that there is a shot boundary.
- Motion: It exploits movement information of objects in a video. When motion changes abruptly, shot change is taken to have occurred.

Shot detection algorithms can be designed by combining these approaches. In addition, powerful features for object recognition, such as SIFT and SURF, are utilized for shout

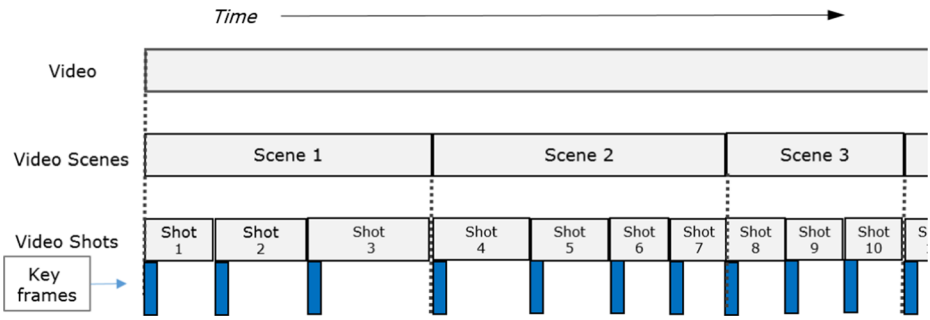

Fig. 5 The shot detection and keyframe extraction

boundary detection [3, 16]. It is the most important process for content-based video analysis; it directly affects the accuracy and efficiency of media-aware information detection.

In this paper, we use Open Source Computer Vision Library (OpenCV) with C++ for the implementation of the shot boundary detection. OpenCV is an open source computer vision and machine learning software library [22]. Our approach is to detect shot boundary by comparing histogram similarity based on Hue Saturation Value (HSV) color space of video frame. HSV separates luminance or the image intensity, from the color information. It is possible to remove the value of intensity and then handle with large variations of illumination by using HSV. For measuring the similarity between a pair of frames, $frame_i$ and $frame_{i+1}$, We must choose one of the several kinds of ways to perform histogram matching: Correlation, Chi-square, Histogram intersection, and Bhattacharyya. In this paper, Chi-square distance is employed for this work; the formula is as follows:

$$d_c(H_i, H_{i+1}) = \sum_{j=1}^{N} \frac{(H_i(j) - H_{i+1}(j))^2}{H_i(j) + H_{i+1}(j)} \tag{1}$$

$$\begin{aligned} &where\ H_i^{'} = \frac{1}{N}\sum_{j=1}^{N} H_i(j) \\ &N\ is\ the\ number\ of\ bin\ in\ histogram. \end{aligned} \tag{2}$$

Fig. 6 shows the actual example of our shot boundary detection approach. When comparing re-encoded version with transformations like format, resolution, codec, and partial edit, with original source, the result shows the same shot boundaries between the two versions.

4.2 Key frame extraction

Key frame extraction finds the most representative frames in a video. The ultimate goal is to extract semantically meaningful information from a video. The keyframe extraction method could be thought of simply as a method for extracting I-frames from video stream. This is because I-frames are reference frames coded without reference to any frames (P or B frame)

Fig. 6 Actual example of shot boundary detection

except themselves. However, there are several constraints in selecting I-frames as key frames. The original video could be re-encoded or edited by cutting out or concatenating partial frames or through transformations. In this case, the locations of I-frames might be chosen differently with the original version. Even if the original and re-encoded video have content overlap, the two versions could have completely different binary. As a result, I-frames in the output could be different from those in the original video, and not be selected as representative frames. Accuracy, as one of our design goals should be robust to various processing activities and transformations. In this sense, selecting I-frames as key frames is not appropriate.

Video segmentation is accomplished by finding the significant differences based on the similarity of a video sequence, by analyzing the low level video features. By selecting a representative frame from each of the video segments, it can eliminate redundant data in the video. Therefore, the task of extracting key frames is the first step, which can reduce the large amount of video data without sacrificing comparison factors. Many algorithms for extracting key frames are proposed on the basis of shot detection, visual feature (e.g., color, edge, and texture), and motion analysis. These methods are overlapped with shot boundary detection method in many aspects, as was already mentioned in 4.1. To extract key frames, this paper uses a shot boundary detection scheme. A key frame can be selected mainly on the first frame, intermediate frame, and last frame from each shot. It also can be chosen by combining the first and last frames [2]. We select the first frame of each shot as the key frame, as shown in Fig. 5.

5 Video fingerprint-based techniques

The proposed system utilizes video fingerprint-based techniques. The main functions of these techniques are to generate video fingerprints, and measure the performance on the basis of video fingerprint-based matching. The system uses video fingerprints in order to compare video files. The use of video fingerprints provides low storage complexity and low matching time. In this section, we explain that how to generate video fingerprints into compact video representation, and measure the similarity by an efficient matching method.

5.1 Video fingerprint generation

Video fingerprint technique uses cryptographic hash functions, such as MD5, SHA-1, and SHA-256. A cryptographic hash function is a mathematical function, which takes the input data and returns a fixed-size short binary string. The value returned by a hash function is called the hash value, message digest, or digital fingerprint. Cryptographic hashing techniques are generally used to identify specific objects (or chunks) in binary data, like executable files. However, cryptographic hashing is very sensitive, with a small change. There are significant difficulties in applying cryptographic hashing to the multimedia contents, which have different representation versions, such as format conversion and compression. Therefore, robust video hashing schemes for video fingerprinting are proposed to identify or authenticate the multimedia contents with various versions. In this paper, we apply perceptual image hashing schemes for the generation of video fingerprints. There are four different types of perceptual hash algorithm as follows:

Block Mean Value Based Hash (BMB) In [34], Yang proposed the normalized block mean value based image perceptual hashing algorithm, and presented four methods for it.

Basically, it normalizes the original image into a preset size, and divides it into blocks. Next, it calculates the mean value Mi from a corresponding block and obtains the median value Md as: $Md = median\ (Mi) \quad (i = 1, 2, \ldots, N)$.

Finally, it normalizes the mean value and obtains the hash values h as follows:

$$h(i) = \begin{cases} 0, & Mi < Md \\ 1, & Mi \geq Md \end{cases}$$

The main advantage of this algorithm is that it works well with fast speed.

DCT Based Hash (DCT) The properties of the DCT can be utilized for perceptual hash algorithms. Low-frequency DCT coefficients are stable values under most manipulations of images. The majority of signal information tends to be concentrated on a few low-frequency components of the DCT. Low-frequency components are mostly distributed close to the top-left of the coefficient array, and they comprise perceptually important information. DCT based hashing algorithms generally utilize this property. The main advantage of this algorithm is that it is stable to manipulations, such as small turns, blur, and image compression. This algorithm ensures fast comparison speed owing to the small size of the hashes.

Marr-Hildreth Operator Based Hash(MH) The Marr-Hildreth Operator Based Hash algorithm calculates the perceptual hash based on detecting edges. It basically uses the Laplacian of the Gaussian function. It is simple but it has two main disadvantages, "false edges" and the localization error that can occur at curved edges.

Radial Variance Based Hash(RADIAL) The Radon transform is one of the perceptual image hash functions. The Radon transform is the integral transform, which consists of the integral of a function over a straight line. It is robust against various image processing and geometrical transformations. The Radial Variance Based Hash algorithm is based on the Radon transform, but that was designed to improve the Radon transform. The basic idea is to create the dispersed beam vector based on the Radon transform [26].

Several studies show the benchmark results for the four perceptual hash algorithms [30, 36]. According to the results, in terms of speed, Block Mean Based Hash algorithm is faster than all the other algorithms. To measure the discriminative capabilities of perceptual hash functions, the inter score distribution can be used and the results show that the Marr-Hildreth operator based image hash has the best performance for discriminative abilities. The DCT based image hash seems to give the second best performance in this part. The intra score distribution was used to test the robustness against various common image operations (e.g., flipping, resizing, compression, and rotation). Table 2 summarizes the results of the intra score distribution against four perceptual hash algorithms. As shown in Table 2, these hash functions have

Table 2 The results of the intra score distribution against four perceptual hash algorithms

Image operation	The best performance	The worst performance
Flipping	None	All (especially MH)
Rotation	BMB, RADIAL	MH
Resizing	BMB, DCT	RADIAL
Compression	RADIAL, BMB	MH

different properties. The intra score distribution could be improved a little more by combining these hash functions.

In the results, the block mean value based image hash function has the best performance in terms of speed. The DCT based image hash function is the slowest. The Marr-Hildreth operator and The DCT based image hash function have superior performance in terms of discriminative abilities. However, the Marr-Hildreth operator based hash shows that it has the limiting factors owing to the results of the intra score distribution. The DCT based hash algorithm is relatively superior to every performance except for the speed.

We design the algorithm of the video fingerprint based on the perceptual image hash algorithms mentioned above. Our choice focuses on the "accuracy" that is one of our design goals for the video fingerprint generation. Thus, we choose the DCT based algorithm in our system. This is because low-frequency DCT coefficient, which is based on the characteristic of the DCT, can be a "perceptual entity" of the image.

We decided to proceed with the DCT based algorithm based on the scheme proposed in [7, 23]. The steps of the algorithm used to generate video fingerprints are as follow:

(1) Convert from RGB to YCbCr for eliminating of unnecessary high frequencies.
(2) Use the median filter [19] for reduction of noise on Y metrics.
(3) Normalize the image into a preset size (i.e., Resize the image to 32 × 32)
(4) Apply the DCT on the Y matrices to get DCT coefficients.
(5) Construct the hash.

– Crop the left-top (8 × 8) for the low-frequency coefficients of Y DCT metrics.
– The DCT coefficients are denoted as $C(i)$, i = 1,2,...,64.
– The median, m, is defined as:

$$m = (C(32) + C(33))\Big/2 \tag{3}$$

– Quantization is performed as follows [7]:

$$hs_{(i)} = \left\{ \begin{array}{ll} 1, & C(i) \geq m \\ 0, & C(i) < m \end{array} \right\} \tag{4}$$

where $hs_{(i)}$ is the i^{th} bit of the perceptual hash.

In conclusion, video fingerprints (hs) for each keyframes are generated by the above algorithm.

5.2 Fingerprint-based matching

The basic idea of the fingerprint-based matching is to find similarity and redundancy through comparison of fingerprints from the original source and new version. The video

stream is divided into many chunks through the shot boundary detection, and the key frame is selected as the representative of each chunk. Therefore, the video fingerprints are generated, one per keyframe. The fingerprint is the extracted hash value from each keyframe by hash function. Each hash value becomes a chunk ID and serves as an identifier. Fingerprint-based matching could eliminate the overhead of a byte-to-byte comparison. For fingerprint-based matching, we generate a compact 64-bit summary (Hash64), to calculate each chunk's hash value. From these two sorting lists, we compare two set of hash values for each file. For this system, the similarity could be the final result for the fingerprint-based matching scheme. We use the hash lists for checking the similarity and redundancy in order to proceed with the forensics investigation against multimedia files.

Algorithm 1 Video similarity Quantifying Algorithm

Input: VF_{input}, VF_{target}, $count_{input}$, $count_{target}$ for measuring video similarity
Output: videoSimilarity
begin
allCount ← ($count_{input}$ <= $count_{target}$) ? $count_{input}$: $count_{target}$
matchCount ← 0
for i ← 0 to $count_{input}$ **do**
for j ← 0 to $count_{target}$ **do**
dist ← CalculateHammingDistance (VF_{input}[i], VF_{target}[j])
matchCount ← CalculateMatchCount (dist)
end
end
videoSimilarity ← matchCount / allCount
return videoSimilarity
end

Algorithm 1 describes a pseudocode for measuring video similarity. To calculate the video similarity, two types of hash lists are needed. The first one is generated from the target file (VFinput), and the other one (VFtarget) is already stored on the storage system. According to Algorithm 1, hash values are sequentially compared to each other. Initially, the "allCount" is set to a small number among the count of video fingerprints of the target and the input file, and the "matchCount" is set to 0.

The "matchCount" indicates how many identical video fingerprints are existing between the two files. To calculate matchCount, we compare the Hamming distance between the two hash values for the target and input file. If the two hashes are sufficiently close in terms of Hamming distance, they are regarded as likely to match. Hence, the number of "matchCount" is increased by 1. The "videoSimilarity" is calculated by dividing the "matchCount" by the "allCount". Ultimately, "videoSimilarity'" is used as the result of the measurement of video similarity.

5.3 Similarity measurement

Video similarity ratio(VSR) is calculated as shown in eq. (5). It indicates that how much the same contents exist between the two files. N is the total number of the extracted key frames. D

is the total number of hash matching by comparing the hash values of the original content and is calculated as (6).

$$\mathrm{D} = \sum_{i=1}^{N} \sum_{j=1}^{K} matched\left(hammingDist\left(hs_{modified}(i), \quad hs_{original}(j)\right)\right) \tag{6}$$

$$\begin{aligned} &Video \quad Similarity \quad Ratio\,(VSR) \\ &= \frac{D}{N} \\ &N = the \quad number \quad of \quad total \quad hashes \\ &D = the \quad number \quad of \quad matched \quad hashes \end{aligned} \tag{5}$$

where $hs_{modified}$ is the hash list for modified version, $hs_{original}$ is the hash list for original version.

As a result, video similarity ratio(VSR) serves as the final result of this system. Fig. 7 shows the concept of hash list composed of matched hashes and mismatched hashes.

6 Evaluation

In this section, we evaluate the efficiency of media-aware information detection and the fingerprint-based matching scheme. The goal of our evaluation is to verify that the generated video fingerprints serve as representatives of a video properly – that is, as sufficiently perceptual entities –, and the result of similarity measurement is to demonstrate the robustness against various transformations and distortions for a video.

6.1 Methodology and setup

We present an experimental methodology and the configuration to evaluate the system: implementation, data set, and transformation factors.

Implementaion We implement the shot boundary detection and key frame extraction by using OpenCV library with C++. For the generation of video fingerprints, we implement the generation algorithms by extending, applying the pHash library [23]. Fingerprint-based matching algorithms are implemented as described in “Algorithm 1”.

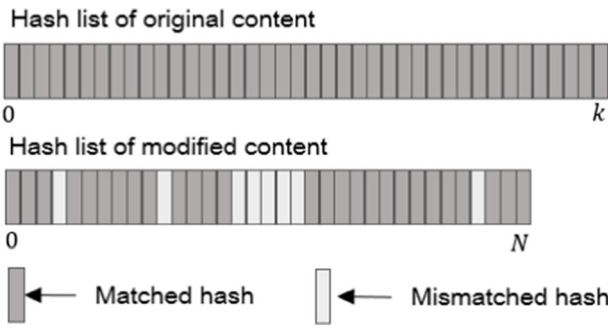

Fig. 7 Example of the generated hash lists

Dataset We use a dataset indicated in Table 3 for experiments in order to analyze the efficiency of the proposed system. The five videos are used to evaluate the shout boundary detection and key frame extraction. In order to evaluate the capability of the key frame extraction, we choose the videos of various genres, having the different length and dynamic events.

Transformation factors We apply three transforms: changing the format(e.g, from mov to mp4), codec (H.264, MP4V), and resolution. Test videos are modified for comparison with the original content, they would be re-encoded by the information shown in Table 4.

6.2 Performance of Media-aware boundary detection

Performance results of shot boundary detection and key frame extraction are described in this section. We evaluate the performance of them in terms of how much match between the key frames extracted from the two files. Fig. 10 illustrates a portion of the captured images for the keyframes in order to make it easier to analyze the results of shot boundary detection and key frame extraction. There are three transformations on the original content for the experiment: format, codec, and resolution. The results show that the same keyframes are generated almost identically from both the original content and the re-encoded content through three variants. However, the explicit analysis for whether how much the same key frames are generated from both files is required. The recall and precision rate are used to measure the effectiveness of key frame extraction, and can be defined as:

$$Precision\,(P_R) = \frac{T_P}{T_P + F_P} \tag{7}$$

$$Recall\,(R_R) = \frac{T_P}{T_P + F_N} \tag{8}$$

Where T_P is the number of correct extraction for key frames, F_N(the false negative) is the number of missed detection, and F_P(the false positive) is the number of wrong detection.

Table 3 Dataset for experiments

Genre	Length(mm:ss)	Data size(MB)	Codec	Format	Resolution(WxH)
Movie tailer[1]	2:57	27.6	MP4V	MP4	854 × 480
Animation[2]	9:56	397	H.264	MOV	1280 × 720
Documentary[3]	41:46	241	AVC1	MP4	480 × 320
Sports[4]	10:09	73.4	H.264	MP4	480 × 320
Music video[5]	4:00	28.1	H.264	MP4	854 × 480

[1] The Legend of Tarzan Official Trailer, https://www.youtube.com/watch?v=v5r6FrEgg5M

[2] Big Buck Bunny, https://peach.blender.org/

[3] The Universe, https://www.youtube.com/watch?v=kLYSFkJ3jIg

[4] Football game, Gareth Bale Ultimate Skills, https://www.youtube.com/watch?v=CeRo4-W3eYU

[5] TWICE “CHEER UP” Music video. https://www.youtube.com/watch?v=c7rCyll5AeY

Table 4 Transformation factors for the re-encoding

Genre	Length(mm:ss)	Data size(MB)	Codec	Format	Resolution(WxH)
Movie tailer	2:57	749	H.264	AVI	480 × 320
Animation	9:56	95.6	MP4V	MP4	720 × 480
Documentary	41:46	632	H.264	AVI	720 × 480
Sports	10:09	154	H.264	AVI	720 × 480
Music video	4:00	37.8	MP4V	MOV	480 × 340

Table 5 shows the experimental results for the shot boundary detection and key frame extraction. The results of shot boundary detection and key frame extraction against test videos are compared with those of the original contents. The performance results are obtained based on eq. (7) and (8).

Fig. 8 shows the percentage of total number of entire video frames and generated key frames. Fig. 9 shows the accuracy of key frame extracion. The video of football game(i.e, sports genre) shows slightly worse performance than other test videos; the recall and precision rate indicates less than 90%. Football game that was taken from far distance is composed of successive video frames of a green background, which players' small movements are only present in it. This result show that our shot boundary detection method can have a relatively low performance against a video consisting of similar frames with no change. However, we get results of an average recall of 95.2% and precision of 92.2%. The average key frame extraction ratio has 1.03% of the entire video frames. Experimental results show that shot boundary detection method is robust against various transformations for codecs, formats, and resolutions Fig. 10.

6.3 Performance of fingerprint-based matching

We evaluate the video fingerprint matching results between the two files (the original content and re-encoded content) in order to evaluate the accuracy of generated video fingerprints. Video fingerprints are generally not perfectly identical as the other versions of the same content. For this reason, video fingerprint-based matching scheme cannot exploit a simple method as searching metadata from the database or comparing the metadata strings within a file. It needs a similarity search. Generally, the distance metrics like Manhattan (L1) and Euclidean distance (L2) can be used to measure the similarity between two video fingerprints. Hash64 generates a video fingerprint consisting of binary signatures. For the binary signatures, Hamming distance can be used, and it provides a good measure of similarity. In this paper, we use Hamming distance for video fingerprint matching.

Table 5 Experimental results for shot boundary detection and key frame extraction

Genre	Total # of frames	Total # of key frames	Key frame extraction ratio (%)	Correct Key frame detection (T_P)	False -(F_N)	False +(F_P)	Recall (R_R)	Precision (P_R)
Movie tailer	5323	86	1.62	84	3	5	0.96	0.94
Animation	14,315	140	0.97	127	2	14	0.98	0.90
Documentary	72,674	194	0.27	190	4	8	0.98	0.96
Sports(Football)	17,938	123	0.69	107	16	16	0.87	0.87
Music video	5771	91	1.58	87	2	5	0.97	0.94

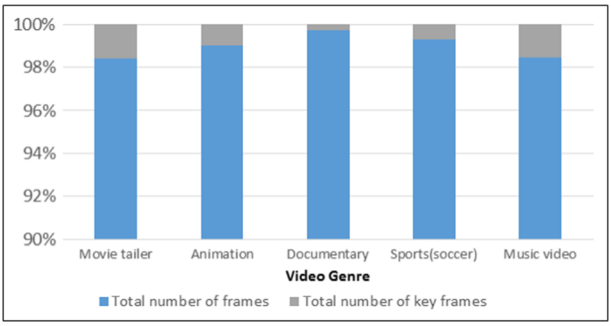

Fig. 8 The percentage of the generated frames and key frames

Table 6 shows the results of video similarity ratio. The documentary content shows the best performance for video similarity result. Since this documentary is interleaved by interview scenes and the universe-related scenes, it has the definite shot boundaries among the video frames. For sports video, it shows the lowest performance with respect to video similarity ratio. As shown in Table 6, the video similarity ratio is 84.5% and the accuracy of video fingerprinting matching is 97.1%. Accuracy of video fingerprinting matching, $R_{accuracy}$ can be measured as follow:

$$R_{accuracy} = \frac{N_{fingerprintMatching}}{N_{keyframeMatching}} \tag{9}$$

where $N_{keyframeMatching}$ is the total number of key frames to match each other exactly between the original and the modified content, $N_{fingerprintMatching}$ is the total number of video fingerprints to match each other from the two key frames. Whereas the accuracy of video fingerprinting is high, the reason for the low video similarity ratio is because of a small value of $N_{keyframeMatching}$, as shown in eq. (9). Therefore, the performance degradation for sports video is because the shot boundary detection errors.

In summary, the average accuracy of video fingerprinting is 97.86%. This result indicates that the proposed hash function is robust against signal processing, including any transformations. On the other hand, the average video similarity ratio is 91.8%. This result is lower than the accuracy of video fingerprinting. It indicates that accurate shot boundary detection process should be performed preferentially well and this gives significant effect on the overall system performance.

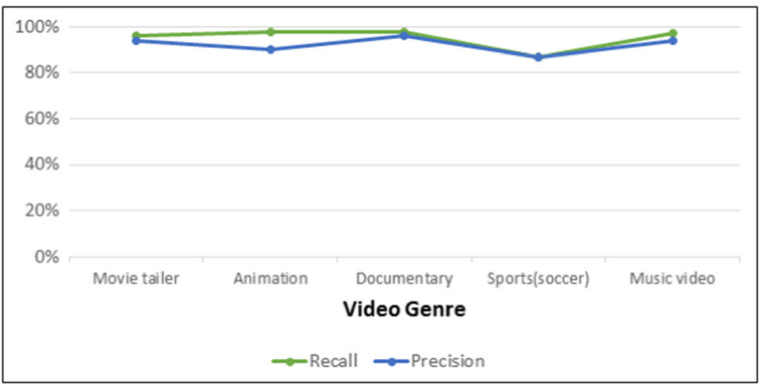

Fig. 9 Accuracy of key frame extraction

Fig. 10 Image capturing for the result of the shot boundary detection and key frame extraction

Table 6 The results of video fingerprint-based matching

Genre	Video similarity ratio (VSR)	Accuracy of video fingerprinting	Total # of key frame matches	Total # of matched video fingerprints
Movie tailer	95.3	97.6	84	82
Animation	90.0	99.2	127	126
Documentary	96.9	98.9	190	188
Sports	84.5	97.1	107	104
Music video	92.3	96.5	87	84

7 Conclusion

In this paper, we propose an efficient multimedia file forensics system based on file similarity search. The proposed system is based on media-aware information detection and a video fingerprint-based matching scheme. The system divides a media file into many shots through the shot boundary detection scheme. Then, it extracts key frames from each shots, and calculates the hash values from each of the key frames by using hash function. A video fingerprint is generated from each key frame using the DCT based perceptual hash function. Each video fingerprint serves as the identifier of each of the key frames, and is used as the standard unit for a shot-to-shot comparison. The system uses the fingerprint-based matching scheme for file similarity search. A fingerprint-based matching scheme has two benefits. The first is that because fingerprints are small objects, they incur low storage overheads and are efficiently matched. The second is that it has low computational overheads for detecting a similarity. The results of experiment were obtained by comparing the original content and the modified content undergone the format, codec, resolution changes. Our experiment results show that the average accuracy of video fingerprinting is 97.86% and the average video similarity ratio is 91.8% as a result of the matching between the original content and the modified content. Therefore, the proposed system can be used to determine whether the illegal and unauthorized distribution against original content by using the similarity search between multimedia files.

Acknowledgements This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Science, ICT and future Planning (2014R1A2A1A11054160). And this research was supported by The Leading Human Resource Training Program of Regional Neo industry through the National Research Foundation of Korea(NRF) funded by the Ministry of Science, ICT and future Planning (2016H1D5A1910630)

References

1. Allamanche, E., Herre, J., Hellmuth, O., Fröba, B., Kastner, T., & Cremer, M. (2001). Content-based Identification of Audio Material Using MPEG-7 Low Level Description. In ISMIR
2. Anand A, Balachandran A, Akella A, Sekar V, Seshan S (2016) Enhancing video accessibility and availability using information-bound references. IEEE/ACM Trans Networking 24(2):1223–1236
3. Baber, J., Afzulpurkar, N., Dailey, M. N., & Bakhtyar, M. (2011). Shot boundary detection from videos using entropy and local descriptor. In 2011 17th International Conference on Digital Signal Processing (DSP) (pp. 1–6). IEEE
4. Bae, S., Nam, G., & Park, K. (2014) Effective content-based video caching with cache-friendly encoding and media-aware chunking. InProceedings of the 5th ACM Multimedia Systems Conference (pp. 203–212). ACM
5. Chasanis VT, Likas AC, Galatsanos NP (2009) Scene detection in videos using shot clustering and sequence alignment. IEEE Trans Multimed 11(1):89–100
6. S. Chen, J. Wang, Y. Ouyang, B. Wang, Q. Tian, H. Lu. (2010) Multi-level trajectory modeling for video copy detection. Proc IEEE Int Conf Acoustics Speech Sign Process (ICASSP'10), 2378–2381
7. Coskun, B., & Sankur, B. (2004) Robust video hash extraction. In Signal Processing and Communications Applications Conference, 2004. Proceedings of the IEEE 12th (pp. 292–295). IEEE
8. De Roover C, De Vleeschouwer C, Lefèbvre F, Macq B (2005) Robust video hashing based on radial projections of key frames. IEEE Trans Signal Process 53(10):4020–4037
9. Eshghi K, Lillibridge M, Wilcock L, Belrose G, Hawkes R (2007) Jumbo Store: Providing Efficient Incremental Upload and Versioning for a Utility Rendering Service. FAST 7:123–138

10. Forman, G., Eshghi, K. Chiocchetti, S. (2005) Finding similar files in large document repositories. In Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining (pp. 394–400). ACM
11. Gionis A, Indyk P, Motwani R (1999) Similarity search in high dimensions via hashing. VLDB 99(6):518–529
12. Gloe T, Fischer A, Kirchner M (2014) Forensic analysis of video file formats. Digit Investig 11:S68–S76
13. Haitsma J, Kalker T, Oostveen J (2001) Robust audio hashing for content identification. Int Workshop Content-Based Multimed Index 4:117–124
14. HotSync, P. (2007) Palm Developer Online Documentation
15. Ko Y-W, Jung H-M, Lee W-Y, Kim M-J, Yoo C (2013) Stride Static Chunking Algorithm for Deduplication System. IEICE Trans Inf Syst 96(7):1544–1547
16. Li J, Ding Y, Shi Y, Li W (2010) A Divide-And-Rule Scheme For Shot Boundary Detection Based on SIFT. JDCTA 4(3):202–214
17. Lu, J. (2009). Video fingerprinting for copy identification: from research to industry applications. In IS&T/SPIE Electronic Imaging (pp. 725402–725402). International Society for Optics and Photonics
18. Mas J, Fernandez G (2003) Video shot boundary detection based on color histogram. Notebook Papers TRECVID2003. NIST, Gaithersburg, Maryland
19. Median filter, http://en.wikipedia.org/wiki/Median_filter
20. Meunier, P., Nystrom, S., Kamara, S., Yost, S., Alexander, K., Noland, D., Crane, J. (2002) ActiveSync, TCP/IP and 802.11 b Wireless Vulnerabilities of WinCE-based PDAs, pp. 145–150. IEEE
21. Muthitacharoen A, Chen B, Mazieres D (2001) A low-bandwidth network file system. ACM SIGOPS OperA Syst Rev 35:174–187
22. OpenCV. Open source computer vision, http://www.opencv.org
23. pHash. The Open source perceptual hash library, http://phash.org
24. Pucha, H., Andersen, D. G., & Kaminsky, M. (2007). Exploiting Similarity for Multi-Source Downloads Using File Handprints. In NSDI
25. Quinlan S, Dorward S (2002) Venti: A New Approach to Archival Storage. FAST 2:89–101
26. Standaert, F. X., Lefebvre, E., Rouvroy, G., Macq, B., Quisquater, J. J., & Legat, J. D. (2005). Practical evaluation of a radial soft hash algorithm. International Conference on Information Technology: Coding and Computing (ITCC'05)-Volume II (Vol. 2, pp. 89–94). IEEE
27. Starobinski D, Trachtenberg A, Agarwal S (2003) Efficient PDA synchronization. IEEE Trans Mob Comput 2:40–51
28. Steinebach, M., Liu, H., & Yannikos, Y. (2014). Efficient Cropping-Resistant Robust Image Hashing. In Availability, Reliability and Security (ARES), 2014 Ninth International Conference on (pp. 579-585). IEEE
29. Tridgell, A. (1999) Efficient algorithms for sorting and synchronization. PhD thesis, The Australian National University
30. Vukelic, B., & Baca, M. (2014). Comparison of RADIAL variance based and Maar-Hildreth operator perceptual image hash functions on biometric templates. In Central European Conference on Information and Intelligent Systems (p. 286). Faculty of Organization and Informatics Varazdin
31. Wu, C., Zhu, J., & Zhang, J. (2012). A content-based video copy detection method with randomly projected binary features. 2012 I.E. Computer Society Conference on Computer Vision and Pattern Recognition Workshops (pp. 21–26). IEEE
32. Xu D, Sheng Y, Ju D, Wu J, Wang D (2011) High Effective Two-round Remote File Fast Synchronization Algorithm. Jisuanji Kexue yu Tansuo 5:38–49
33. Yan, H., Irmak, U., Suel, T. (2008) Algorithms for low-latency remote file synchronization, pp. 156–160. IEEE
34. Yang, B., Gu, F., & Niu, X. (2006). Block mean value based image perceptual hashing. In 2006 International Conference on Intelligent Information Hiding and Multimedia (pp. 167–172). IEEE
35. You, L., & Karamanolis, C. T. (2004). Evaluation of Efficient Archival Storage Techniques. In MSST (pp. 227–232)
36. Zauner, C. (2010). Implementation and benchmarking of perceptual image hash functions. na

Min Ja Kim she received the B.S. degree in computer engineering from Dongduk Women's University in 2000 and M.S. degree in Computer Science from Korea University in 2002. She is currently pursuing her Ph.D degree in College of Informatics, Korea University, Seoul, Korea. Her research interests include Operating System, File system and multimedia streaming.

Chuck Yoo received the B.S. degree in electronics engineering from Seoul National University, Seoul, Korea and the M.S. and Ph.D. in computer science in University of Michigan. From 1990 to 1995, he worked as researcher in Sun Microsystems Lab. He is now a Professor in College of Information and Communications, Korea University, Seoul, Korea. His research interests include Operating System, Virtualization and multimedia streaming.

Young Woong Ko received both a M.S. and Ph.D. in computer science from Korea University, Seoul, Korea, in 1999 and 2003, respectively. He is now a professor in Department of Computer engineering, Hallym University in Korea. His research interests include operating systems, embedded systems and multimedia systems.