Fast Video Deduplication and Localization With Temporal Consistence Re-Ranking

Chris Henry, *Student Member, IEEE*, Li Song, *Senior Member, IEEE*, and Zhu Li, *Senior Member, IEEE*

***Abstract*— The use of social media networks and mobile devices has experienced tremendous growth in recent years. This has led to a surge in the number of videos recorded and uploaded to social media platforms like TikTok and YouTube. However, this increase has also resulted in the rise of illegal duplicate videos, which are essentially the same as the original videos but with minor editing effects and variations in coding. In addition, the large number of duplicate videos is a major storage and communication efficiency issue. The task of finding duplicate videos from a large repository is referred to as video deduplication. Video deduplication is a crucial task for applications like saving storage space and detecting copyright infringement. This work proposes a fast and robust location-aware video deduplication system capable of retrieving duplicate videos from a large repository extremely quickly. In addition, the proposed system has the ability to find the precise location of the query video in the retrieved videos. To identify and localize short video clips against large video repositories, we utilize robust image-level features from keypoint aggregation and deep learning along with an efficient KNN search of query frames with a multiple k-d tree setup, giving us a set of candidate video clips. Then, a fast temporal consistence pruning algorithm re-ranks the clip-level candidates and identifies the matching clip along with its temporal location in a sequence in an efficient way. The system was tested on 1 million frame/145 hour and 4.5 million frame/636 hour repositories generated via the large-scale FIVR-200K and VCSL datasets, respectively. The proposed system achieves a recall of 98.8% and 94.1% for the FIVR-200K and VCSL datasets, respectively. A query frame is searched as fast as 83.96ms and 462.59ms from a 1 million frame/145 hour and a 4.5 million frame/636 hour repository, respectively. These experimental results demonstrate that our system is highly accurate and that the time consumption is extremely low for retrieving video along with its timestamp information from large-scale repositories.**

***Index Terms*— Video deduplication, video retrieval, near-duplicate video retrieval, video copy detection, fisher vector.**

Manuscript received 19 January 2024; revised 9 May 2024 and 30 May 2024; accepted 20 June 2024. Date of publication 28 June 2024; date of current version 27 November 2024. This work was supported in part by NSF under Award 2148382. This article was recommended by Associate Editor L. Nie. *(Corresponding author: Zhu Li.)*

Chris Henry and Zhu Li are with the Department of Computer Science and Electrical Engineering, University of Missouri–Kansas City, Kansas City, MO 64110 USA (e-mail: chffn@umsystem.edu; zhu.li@ieee.org).

Li Song is with the Institute of Image Communication and Network Engineering, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: song_li@sjtu.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSVT.2024.3420422.

Digital Object Identifier 10.1109/TCSVT.2024.3420422

I. Introduction

In RECENT years, the easy accessibility to mobile devices and increasing usage of social media platforms like YouTube,[1] Facebook,[2] and TikTok[3] have led to explosive growth in the amount of video content shared online. For instance, as of May 2019, the rate of video content uploaded to YouTube surpassed 500 hours per minute.[4] As a result, an increase in the amount of illegal pirated video content shared has been witnessed. These illegal pirate videos essentially contain content identical to the original videos. However, these videos are edited by adding slight variations to evade detection by copy detection systems. These variations are usually added by transformations such as modifying the aspect ratio, changing color, changing frame rate, padding, overlaying text, flipping, etc. Such videos are known as duplicates or near-duplicates. In addition, storing this huge amount of video content is a challenging issue that further worsens the problem.

The detection of such duplicate videos is referred to as video deduplication. Video deduplication systems aim to identify and remove duplicate videos from a large collection of videos. A task similar (but distinct) to video deduplication is near-duplicate video retrieval (NDVR) [1], [2], [3], [4], which aims to retrieve near-duplicate videos from a large-scale video repository or database. Near-duplicates are similar to original videos but not exactly the same. Both NDVR and video deduplication systems are useful for applications like copyright infringement detection [5], [6], [7], [8], video search/recommendation [9], [10], [11], etc. Video deduplication is more useful for applications that generally require duplicate video removal due to limited storage resources. Such tasks include video content management, video surveillance [12], [13], etc. NDVR, on the other hand, is more useful for content-based video search and recommendation systems.

Most video deduplication/NDVR systems work by extracting features from frames/videos and then computing a similarity score to obtain duplicates/near-duplicates. These systems can be broadly divided into frame-level [7], [14], [15], [16], [17], [18] and video-level [19], [20], [21], [22], [23], [24], [25] methods. Frame-level methods extract features from frames in the video, whereas video-level methods rep-

[1] https://www.youtube.com
[2] https://www.facebook.com
[3] https://www.tiktok.com
[4] https://en.wikipedia.org/wiki/YouTube

resent the entire video with a global representation. While recent methods have shown promising results in retrieving duplicate/near-duplicate videos, most methods are not location-aware and have limited scalability. Generally, video deduplication systems are computationally expensive; consequently, such systems require a considerably large amount of time to retrieve videos from a large-scale video repository. Therefore, a fast and robust location-aware video deduplication system is crucial for applications requiring real-time processing.

In this work, we propose a frame-level video deduplication system that is fast, robust, and location-aware. The proposed system is based on fisher vector, VGG feature, and thumbnail feature extraction which are used to represent a video frame. This representation is then used for frame retrieval from large-scale 1 million frame (145 hour) and 4.5 million frame repositories using our multiple k-d tree setup. A temporal consistence pruning strategy is proposed that utilizes sequence ID and global timestamp information to retrieve duplicate videos along with their precise location accurately. The proposed system was tested on the large-scale FIVR-200K dataset [26] and VCSL dataset [27]. Experimental results validate that the retrieval results are highly accurate and the system can process a query frame within a few milliseconds. The main contributions of this paper are mentioned below:

1) We propose a fast and robust location-aware video deduplication system. Given a query video, the system is able to retrieve the duplicate video from a large 636 hours video repository along with its precise location.
2) The system is fast, requiring as low as 83.96ms and 462.59ms to search a query frame from a 145 hour and 636 hour video repository, respectively.

The remainder of the paper is organized as follows. Section II presents the related works, whereas the proposed method is described in Section III. The experimental setup and experimental results are discussed in Section IV and Section V, respectively. Conclusions are presented in the last section of this paper.

II. Related Works

A. Frame-Level Retrieval Methods

Frame-based methods generally represent individual video frames and use nearest neighbor search to retrieve relevant frames. The retrieved video frames are used to compute a video similarity score via post-filtering.

A video copy detection system that matched individual frames and verified their spatiotemporal consistency was proposed in [7]. Hessian-Affine region detector [28] was used to extract local patches from frames and Scale-Invariant Feature Transform (SIFT) [29] or CS-LBP [30] descriptors were used as the descriptors. Video frames were represented using SIFT and bag-of-words representation by the study in [14]. Incorrect matches were filtered out via a weak geometric consistency. Temporal-concentration SIFT (TCSIFT) that encoded temporal information by tracking SIFT was proposed in [15]. The work by [16] used binary temporal alignment to efficiently find a match. Fast CenSurE keypoint detector and BRIEF descriptor were used for feature detection and description, respectively. The study in [17] utilized a scalable K-means clustering technique for learning a visual vocabulary based on the color correlograms of training images. The study leveraged inverted file indexing for efficient video retrieval. In [18], compact spatiotemporal features based on feature selection and a w-shingling scheme are used to represent videos and a modified inverted file index was constructed for real-time video retrieval.

B. Video-Level Retrieval Methods

Video-level retrieval methods usually represent an entire video as a global signature and a similarity metric is used to compute similarity between videos in the embedding space.

A video clip representation model referred to as a bounded coordinate system was proposed in [19]. The model was based on Principal Component Analysis (PCA). The study in [20] proposed multiple feature hashing that used multiple image features to learn a group of hash functions that map video keyframes into the Hamming space. A series of binary codes were used to generate signatures for the video dataset. In [21], a deep video hashing method was proposed that used CNN to represent a video via meaningful binary codes. A self-supervised video hashing framework [22] that used an encoder–decoder architecture for generating binary codes to represent videos. A global video representation was generated in [23] by using intermediate CNN features via a layer-based feature aggregation scheme. In [24], intermediate CNN features were used to generate global video signatures together with a deep metric learning framework. The metric learning framework used Triplet loss [31] to minimize the distance between relevant videos and maximize the distance between irrelevant videos. A video representation framework dubbed as Temporal Context Aggregation for Video Retrieval (TCA) was proposed in [25]. TCA incorporated temporal information among frame-level features via temporal context aggregation that used a self-attention mechanism. The framework was trained via a supervised contrastive learning method.

III. Video Clip Identification and Localization

This section describes the proposed video deduplication system. The block diagram of the proposed system is presented in Fig. 1. The proposed method can be divided into the following steps:

A. Feature Generation

1) Fisher Vector: This subsection introduces fisher vector aggregation for image classification. Fisher vector aggregation is used in computer vision and image processing to encode local features from an image. The MPEG-CDVS Standard [32] used the scalable compressed Fisher Vector (SCFV) representation for the task of visual search and achieved high matching accuracy with minimal memory requirements. The studies in [33] and [34] also used fisher vector for the task of video deduplication. Inspired by [32], [33], and [34], we also incorporate fisher vector in our work.

Let $X = \{x_t, t = 1, \ldots, T\}$ represent the set of T local descriptors (such as SIFT [29])) extracted from an image. The main idea of fisher vector aggregation is to model the distribution of local features X via a probability density function u_λ with parameters λ. Gaussian mixture model (GMM)

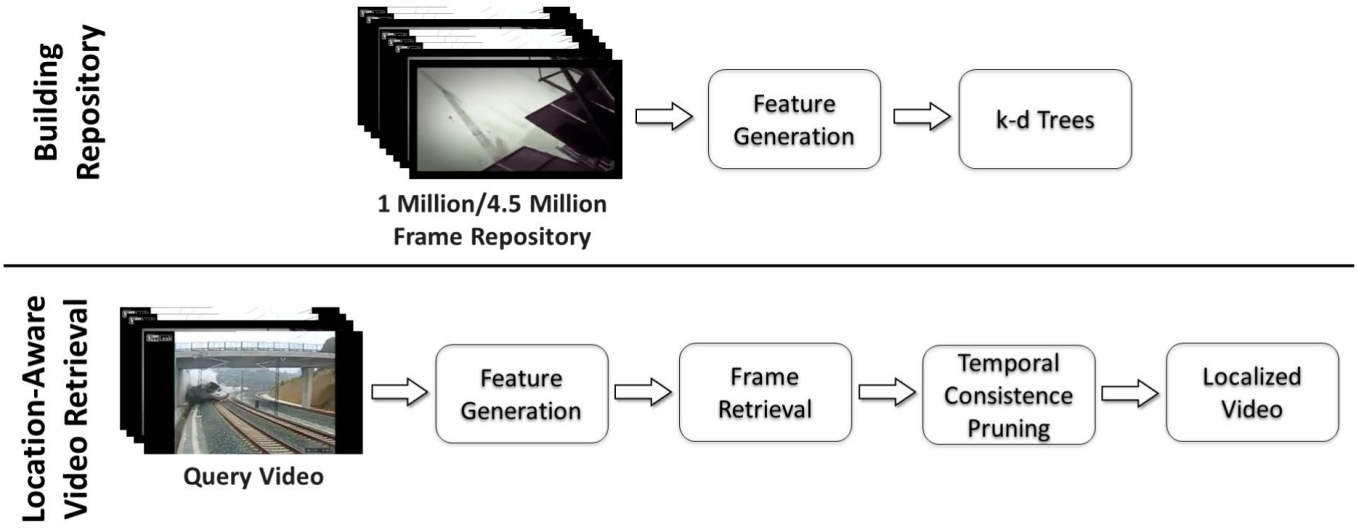


Fig. 1. Overall workflow of the proposed approach.

is generally used to model the probability density function u_λ. This is followed by computing the gradient of log-likelihood function with respect to the model parameters. The gradient of the log-likelihood can be mathematically formulated as:

$$G_\lambda^X = \nabla_\lambda \log u_\lambda(X) \tag{1}$$

The gradient of the log-likelihood function of the model with respect to its parameters λ indicates how each parameter influences the generation process of X. The parameters $\lambda = \{w_k, \mu_k, \sigma_k, k = 1, \ldots, K\}$ where w_k, μ_k, and σ_k represent the mixture weights, means, and diagonal covariance matrices of GMM containing K number of components. The fisher vector, a $2Kd$-dimensional super vector, is obtained by concatenating the gradients $G_{\mu,k}^X$ and $G_{\sigma,k}^X$ with respect to μ_k and σ_k of the k-th GMM component. These gradients can be computed as follows:

$$G_{\mu,k}^X = \frac{1}{T\sqrt{w_i}} \sum_{t=1}^{T} \gamma_t(k) \left(\frac{x_t - \mu_k}{\sigma_k} \right) \tag{2}$$

$$G_{\sigma,k}^X = \frac{1}{T\sqrt{2w_i}} \sum_{t=1}^{T} \gamma_t(k) \left[\frac{(x_t - \mu_k)^2}{{\sigma_k}^2} - 1 \right] \tag{3}$$

where $\gamma_t(k)$ represents the weight of local feature x_t for the k-th Gaussian component and is defined as:

$$\gamma_t(k) = \frac{\pi_k \mu_k(x_t)}{\sum_{j=1}^{K} w_j \mu_j(x_t)} \tag{4}$$

This work uses SIFT keypoints as local image features due to the scale and rotation invariance properties of SIFT features. We fit GMM using SIFT keypoint descriptors [29] which were extracted from frames in the CDVS dataset [32]. Each SIFT keypoint descriptor is a 128-d vector. We project the 128-d vector to a 16-d vector using Principal Component Analysis (PCA). The 16-d vector is then used for training the GMM with 64 number of components. The projected SIFT keypoint descriptors were randomly sampled from the entire dataset. The outcome of fitting the GMM is a visual vocabulary of dominant image features and their distributions. The 16-d SIFT keypoint descriptors and the trained GMM model are used to generate the fisher vectors. The dimension of the generated fisher vector is 1024-d. The fisher vector is reduced to a d_{fisher} dimension vector via PCA and will be denoted by f_{fisher} in the following text.

2) VGG Feature: The VGG feature is generated by inputting the frame to a pretrained VGG16 [35] (trained on ImageNet [36]). VGG16 is a convolutional neural network architecture that was introduced in 2014 by the Visual Geometry Group (VGG) at the University of Oxford. It consists of 13 convolutional layers and 3 fully connected (fc) layers and achieved outstanding results on image recognition tasks. The last layer (output layer) is removed from the VGG16 and a 4096-d feature is extracted from the 'fc2' (fully connected 2) layer of VGG16. This 4096-d feature vector is projected to a lower dimension of d_{VGG} via PCA. The VGG feature is denoted as f_{VGG}.

3) Thumbnail Feature: The thumbnail feature is computed by converting the video frame to grayscale, followed by resizing the video frame to a resolution of 12×12 pixels. The thumbnail is transformed into a 144-d vector with values ranging between $0 - 1$. The global mean is also subtracted from the thumbnail feature. The thumbnail feature is reduced to a d_{thumb} dimension vector via PCA. The thumbnail feature is denoted as f_{thumb}.

B. Efficient Frame Retrieval via Multiple k-d Tree Setup

The proposed features are used for efficient frame retrieval using k-d trees. A k-d tree is a data structure that is used for organizing points in a k-dimensional space. A k-d tree enables efficient nearest neighbor searches by partitioning the k-dimensional space into smaller regions based on the points' coordinates.

We use multiple k-d tree setup for the frame retrieval system. A single k-d tree could also be built by merging the three features into one. However, this would degrade the performance of the k-d tree. This is because a k-d tree's performance degrades with high-dimensional data due to the

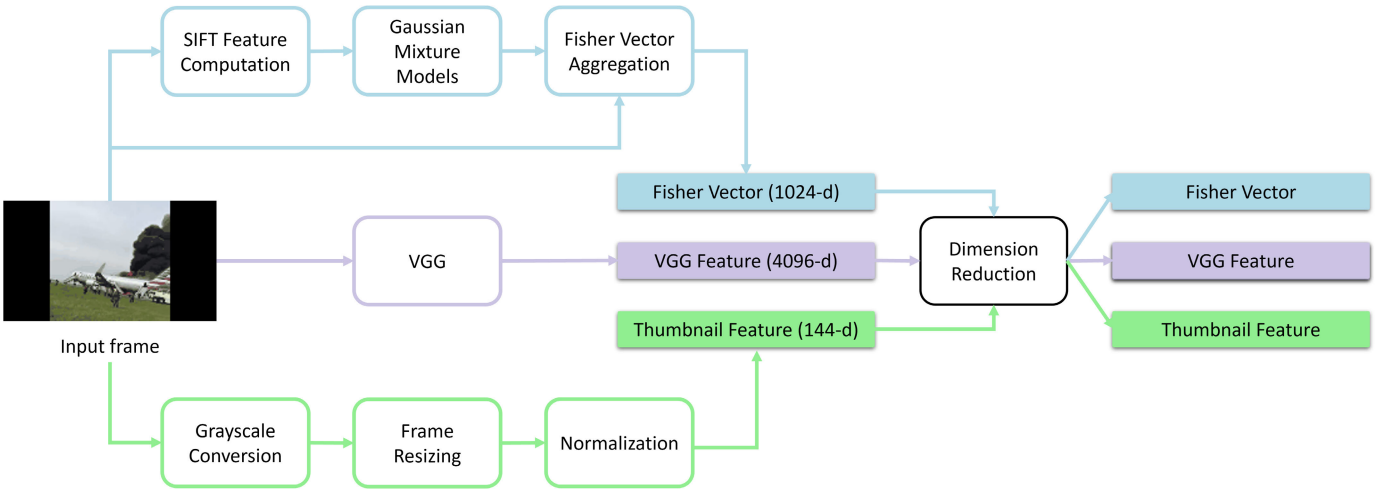


Fig. 2. Overview of the feature generation step.

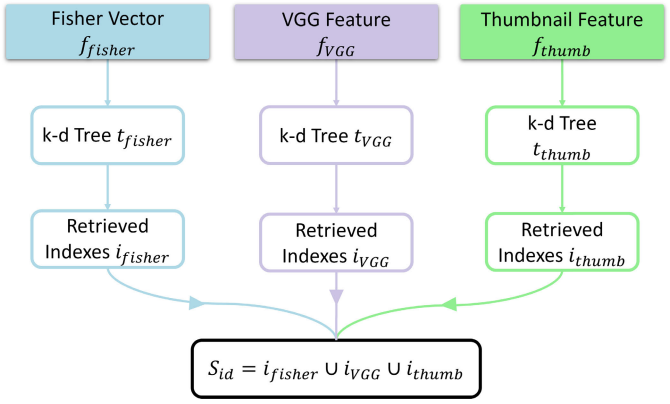


Fig. 3. Block diagram for Efficient Frame Retrieval via Multiple k-d Tree Setup.

curse of dimensionality. In addition, each feature has its own latent space, so merging these features into one might not work out well. Therefore, a separate k-d tree is built for each of the three features generated via the process mentioned in Section III-A. This keeps the feature dimension small for the k-d tree and also maximizes the chances of correct frame retrieval in at least one of the three k-d trees. In case one k-d tree gives incorrect retrieval results, the remaining trees can cover for it. The overall frame retrieval setup is shown in Fig. 3.

During query time, the k-nearest neighbor (KNN) search is used to search for the samples in the repository that are nearest to the query. The time consumption for KNN search increases as the number of neighbors K increases. The KNN search results return k number of frames. k depends upon the K value selected during KNN search. The k-d trees are used to index 1-million and 4.5-million frame repositories which are generated by uniformly sampling video frames at regular intervals from the FIVR-200K dataset [26] and VCSL dataset [27], respectively. Each frame is associated with its video ID V_{id} and the global timestamp ts_{global}.

Given the three features (f_{fisher}, f_{VGG}, and f_{thumb}), we generate three k-d trees t_{fisher}, t_{VGG}, and t_{thumb} using f_{fisher}, f_{VGG}, and f_{thumb}, respectively. Each k-d tree returns the k neighbors nearest to the query frame. The neighbors are returned as indexes in the frame repository. The returned indexes from each k-d tree are merged to form a set $S_{IDs} = \{I_{fisher} \cup I_{VGG} \cup I_{thumb}\}$ containing the frame IDs from each k-d tree. I_{fisher}, I_{VGG}, and I_{thumb} are the indexes of retrieved frame IDs from t_{fisher}, t_{VGG}, and t_{thumb}, respectively. S_{id} may contain many false positives; however, these false positives will be pruned via the proposed temporal consistence pruning strategy, as explained in the following subsection.

C. Re-Ranking and Localization With Temporal Consistence Pruning

In this step, we utilize the temporal coherence constraint to remove the false positives obtained during the frame retrieval step. This step uses the global timestamp ts_{global} and video ID V_{id} information to retrieve duplicate videos from the large-scale repository. The overall temporal consistence pruning algorithm is shown in Algorithm 1.

A query video $Q = \{q(\tau_1), q(\tau_2), \ldots, q(\tau_m)\}$ is represented by query frames $q(\tau)$ extracted from query video sampled at interval τ. For each query frame $q(\tau)$, the fisher vector f_{fisher}, VGG feature f_{VGG}, and thumbnail feature f_{thumb} are generated via the procedure explained in Section III-A. These features are inputted to the three k-d trees (t_{fisher}, t_{VGG}, and t_{thumb}) to retrieve matching frames for each of the query frames. The multiple k-d tree setup returns indexes of retrieved frames $S_{id}^{q(\tau_m)}$ for each query frame $q(\tau_m)$. m number of $S_{id}^{q(\tau_m)}$ are returned by the frame retrieval setup. Each index retrieved in $S_{id}^{q(\tau_m)}$ is associated with its video ID V_{id} and global timestamp ts_{global} in the repository. So given query video $Q = \{q(\tau_1), q(\tau_2), \ldots, q(\tau_m)\}$, the frame retrieval setup returns $S_{id}^{q} = \{S_{id}^{q(\tau_1)}, S_{id}^{q(\tau_2)}, \ldots, S_{id}^{q(\tau_m)}\}$ where $S_{id}^{q(\tau_m)}$ contains the indexes of the retrieved frames in the repository for query frame $q(\tau_m)$.

We utilize the video ID V_{id} and global timestamp ts_{global} to prune the retrieved frames' indexes S_{id}^{q}. The temporal consistence pruning strategy can be divided into 3 steps - video ID pruning, timestamp pruning, and localized video retrieval. In video ID pruning, we compare the video IDs of the retrieved frames in each set in S_{id}^{q}. Only the same video IDs are kept while the remaining are pruned. The video IDs in $S_{id}^{q(\tau_{m+1})}$ are compared with the preceding set $S_{id}^{q(\tau_m)}$ and video IDs are

Algorithm 1 Temporal Consistence Pruning

Input: $S_{id}^{q} = [S_{id}^{q(\tau_1)}, S_{id}^{q(\tau_2)}, \ldots, S_{id}^{q(\tau_m)}]$ where $S_{id}^{q(\tau_m)}$ contains the indexes of the retrieved frames for query frame $q(\tau_m)$.

Output: Videos IDs and the associated timestamps.

1: Compare video ID of each retrieved frame in $S_{id}^{q(\tau_{m+1})}$ with unique video IDs in previous set $S_{id}^{q(\tau_m)}$ until the last set in S_{id}^{q}.

2: Update $S_{id}^{q(\tau_{m+1})}$ by removing frames whose video IDs do not match those in $S_{id}^{q(\tau_m)}$ until the last set in S_{id}^{q}.

3: Compare timestamp difference of each retrieved frame in $S_{id}^{q(\tau_{m+1})}$ with timestamp of each frame in previous set $S_{id}^{q(\tau_m)}$ until the last set in S_{id}^{q}. The timestamp difference should be equal to τ.

4: Update $S_{id}^{q(\tau_{m+1})}$ by retaining frames that satisfy the timestamp difference constraint in step 3 until the last set in S_{id}^{q}.

5: Retrieve videos and their timestamps by extracting retrieved which match both video ID and timestamp constraint in all sets in S_{id}^{q}.

only kept in $S_{id}^{q(\tau_{m+1})}$ if the video ID in $S_{id}^{q(\tau_m)}$ is also found in $S_{id}^{q(\tau_m+1)}$. This removes the false positive frames that were retrieved. However, even after video ID pruning, many false positives remain in S_{id}^{q}. These false positives are taken care of by the timestamp pruning.

In timestamp pruning, we leverage the global timestamp ts_{global} information available in the repository. The frame sampling interval τ used to sample the query frames from the query video is known. This information about τ can be leveraged to further prune the remaining frames retrieved. Since the query video is sampled at regular interval τ, the difference between a frame in $S_{id}^{q(\tau_{m+1})}$ and a frame in preceding $S_{id}^{q(\tau_m)}$ should be equivalent to τ. Based on this timestamp constraint, we further reduce the number of false positives and are left with the relevant frames.

Finally, we are left with frames that satisfy the video ID and timestamp constraint. The final videos are retrieved by extracting frames that have the same video ID and timestamps increasing by an interval of τ. Since we know the timestamp associated with each retrieved frame, our algorithm is also able to accurately localize the video. This is better illustrated in Fig. 4. Please see the light green and light blue colored boxes which represent the final retrieved videos along with their time stamp. Note that the same video was returned with two different timestamps. However, it must be noted that both the timestamps differ only by 0.5 seconds which is acceptable since it is possible for the frames to be static at such short intervals.

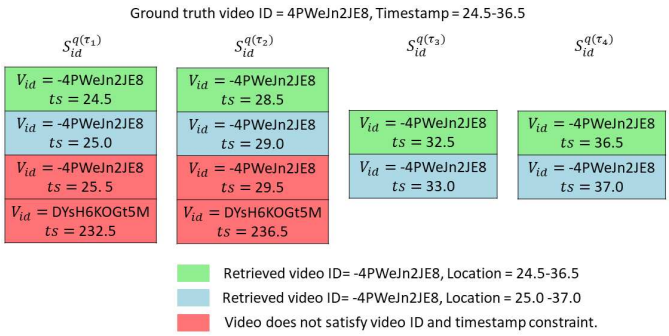


Fig. 4. Intuition behind the temporal consistence pruning strategy. The blue and green colored sequences are retrieved while the red colored sequences are rejected. Note: Figure best viewed in color.

IV. Experimental Setup

A. Dataset Details

This paper used the following datasets:

1) FIVR-200K Dataset: The FIVR-200K dataset [26] is a large-scale dataset for the problem of Fine-grained Incident Video Retrieval (FVIR). It consists of 225, 960 YouTube videos which are associated with 100 selected video queries and 4, 687 Wikipedia events. Videos are categorized into Duplicate Scene Videos (DSVs), Complementary Scene Videos (CSVs), and Incident Scene Videos (ISVs). In our work, we are particularly interested in DSVs which refer to videos sharing at least one scene regardless of the transformations applied. DSVs are close to exact duplicates of each other; however, they can be different in terms of transformations such as photometric variations, editing, length, etc. A total of 7, 558 videos are labeled as DSVs; however, only 4, 960 videos were available for download at the time of writing this manuscript.

We use these 4, 960 videos to build a 1-million frame repository (1, 016, 035 frames exactly) which is equivalent to about 145 hours of video. The frames were extracted at a fixed interval of 0.5 seconds which translates to 2 frames being extracted per second of video. This large-scale repository is used as the database of videos in which we will search for the query video. We generate two test sets of varying difficulty levels for testing the proposed system. These test sets will be referred to as FIVR-200K-Normal and FIVR-200K-Hard throughout the remaining paper. For generating the query videos for testing, we chose 1, 000 videos randomly from the FIVR-200K dataset [26] in a way that ensures at least 10 videos were selected from each of the 100 video queries and also ensures that the selected videos have a duration of at least 41 seconds. We randomly extract a 40 second clip from each video and sample frames at an interval of 10 seconds, generating a total of 5 frames per 40 seconds query video. For each query video, we have the video ID and timestamp information as the ground truth. We refer to videos that have a match in the 1-million frame repository as positive videos.

For the FIVR-200K-Normal test set, the frames were augmented with transformations that included changing hue, saturation, gamma, adding blur, and JPEG compression. A tougher set of augmentations containing rotation, horizontal flipping, changing hue, saturation, gamma, adding blur, and JPEG compression were used for generating the FIVR-200K-Hard test set. Samples from both FIVR-200K-Normal and FIVR-200K-Hard can be visualized in Fig. 5. Notice the frames and red boxes in the top right corner of Fig. 5 which show that the augmented frame is challenging to retrieve due to flipping and rotation as compared to the original frame. As can be seen in the top left corner of Fig. 5, the yellow boxes show

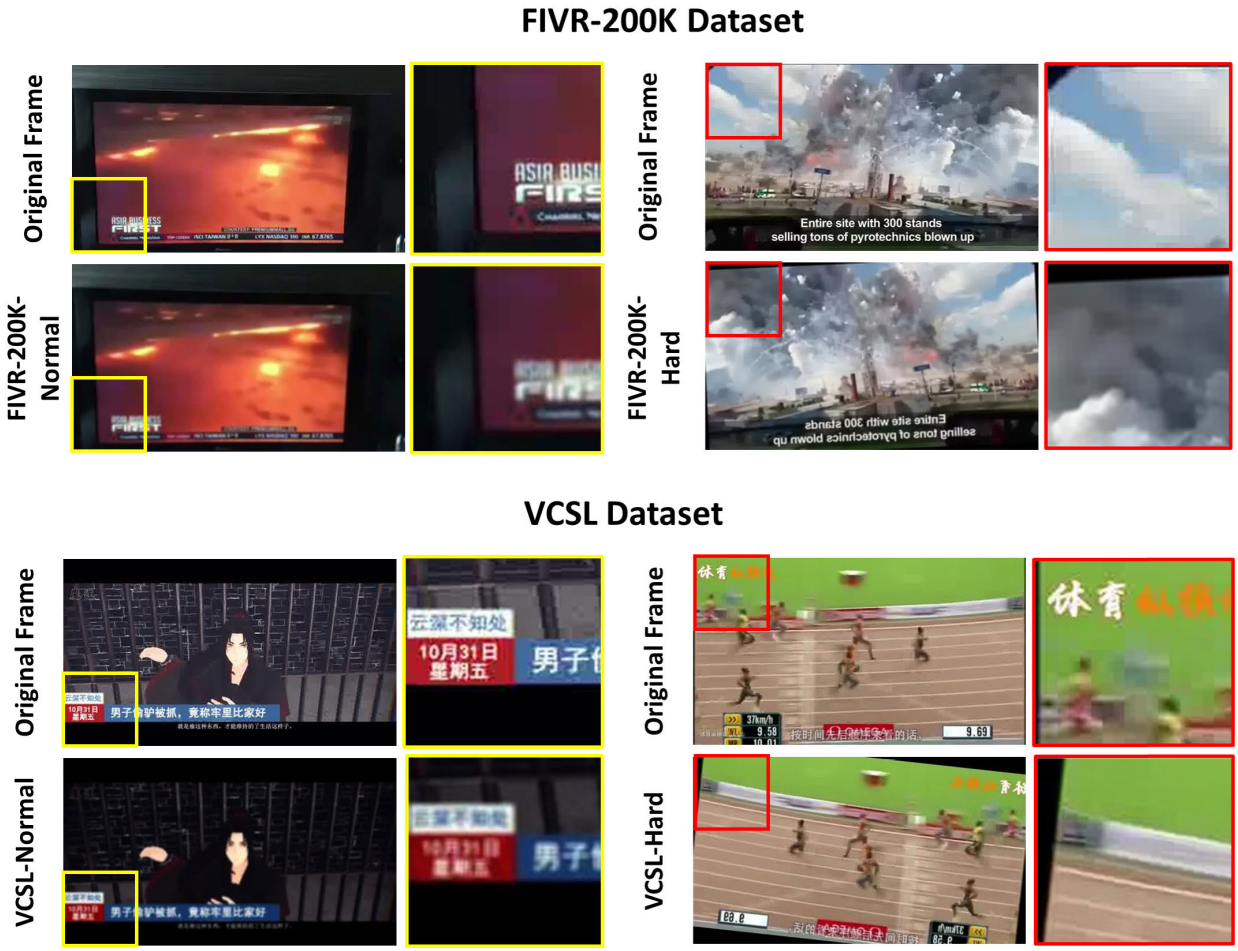


Fig. 5. Dataset samples for FIVR-200K-Normal (top left), FIVR-200K-Hard (top right), VCSL-Normal (bottom left), and VCSL-Hard (bottom right). Notice that the hard version of the dataset contained vertical flipping and rotation, essentially changing the entire frame. Note: Figure best viewed in color.

that the augmented frame used for testing is extremely blurry as compared to the original frame.

2) VCSL Dataset: VCSL (Video Copy Segment Localization) [27] is a large-scale segment-level annotated video copy dataset introduced in 2022. It contains 160, 000 realistic video copy pairs which include more than 280, 000 localized copied segment pairs. The dataset covers a wide range of video duration in addition to diverse video categories. The precise start and end timestamps are provided for all the copied segments inside each video pair. The dataset provides a total of 9, 207 video links from YouTube and BiliBili.[5] At the time of writing this manuscript, only 6, 649 videos were available for downloading.

These 6, 649 videos were sampled every 0.5 seconds, generating 2 frames every second for each video. A large-scale 636 hours of video repository containing 4.5-million frames (4, 524, 789-frames exactly) was generated for the VCSL dataset [27]. Similar to the FIVR-200K dataset [26], we create VCSL-Normal and VCSL-Hard test sets. We randomly chose 1, 000 videos from the VCSL dataset [27] and augmented them with various transformations to create the test data. We applied the same sets of transformations that were used to create FIVR-Normal and FIVR-Hard. We extract 40 second chunks from each test video and sample at an interval of 10 seconds to represent a query video with 5 frames. Each query video is accompanied by its video ID and timestamp information which is used as ground truth to estimate the performance of our proposed system. Samples from VCSL-Normal and VCSL-Hard are visualized in Fig. 5. Notice the difficulty level in the hard test set for the VCSL dataset (see Fig. 5's lower right corner).

3) Negative Videos: To evaluate the performance of videos that have no match in the repository, we generate negative videos. We randomly downloaded videos from the 'DW Documentary[6]' YouTube channel. We extract 1, 000 clips with a duration of 40 seconds from the downloaded videos and represent each negative video as 5 frames sampled at an interval of 10 seconds.

B. Implementation Details

The system was implemented on a desktop computer with Intel Core i7-12700 CPU and 64 gigabytes of RAM. Python programming language was used to implement the video deduplication system. MATLAB was used to extract SIFT keypoints, perform GMM training, and compute fisher vectors. PyTorch framework was used to extract VGG features from

[5]https://www.bilibili.com/

[6]https://www.youtube.com/c/DWDocumentary

TABLE I
RECALL (%) AT DIFFERENT K VALUES FOR POSITIVE QUERIES

Dataset	Feature Dimension (FV, Thumbnail, VGG)	Normal				Hard			
		Recall (K=64)	Recall (K=128)	Recall (K=256)	Recall (K=512)	Recall (K=64)	Recall (K=128)	Recall (K=256)	Recall (K=512)
FIVR-200K Dataset	32, 32, 32	96.5	97.3	97.39	**98.0**	84.7	87.2	90.6	**92.9**
	64, 32, 32	96.7	97.39	97.89	**98.2**	86	89.6	92.4	**94.89**
	128, 64, 64	97.8	98.3	98.4	**98.8**	89.8	92.7	95.3	**96.5**
VCSL Dataset	32, 32, 32	82.8	85.9	88.1	**90.3**	66	71.7	75.5	**80**
	64, 32, 32	84.5	87.7	89.8	**91**	69.89	75	78.5	**83**
	128, 64, 64	89.6	91.8	93.4	**94.1**	77	81.5	83.7	**87.7**

VGG16 on a single NVIDIA RTX A5000. We used leaf size (the maximum number of points allowed in a leaf node) of 32 when building k-d trees.

C. Optimization Guidelines

This subsection provides guidelines for optimizing the proposed video deduplication system. An important parameter is the number of PCA components/feature dimension. A common way to determine the number of components is to analyze the cumulative explained variance ratio as a function of the number of components. Choose the number of components by visually inspecting the plot and selecting the point where the curve starts to reach a plateau. Generally, more components/higher feature dimension can better represent the data, leading to better results but higher time consumption. Lower feature dimensions decrease the time consumption but achieve lower recall. A recall-time consumption trade-off should be determined based on the application.

Regarding thumbnail size, an optimal thumbnail size should perform frame abstraction and produce a low-dimensional embedding to reduce computation cost. In our case, 12×12 was empirically chosen. Similarly, for fisher vector, lower dimensions are preferred to effectively deal with computation and memory constraints. For k-d tree generation, lower dimensional data is better since k-d trees become inefficient for high-dimensional data. As for the leaf size (maximum number of points allowed in a leaf node), a smaller leaf size produces a deeper k-d tree but faster query times, while a larger leaf size produces a shallower k-d tree but slower query times. Another important parameter is the value K used during k-d tree search. A higher K value achieves better results with greater time consumption, whereas lower K results in faster search times but lower recall rates.

V. EXPERIMENTAL RESULT

The proposed system was evaluated on FIVR-200K [26] and VCSL [27] datasets - both of which are two large-scale video copy detection datasets.

A. Duplicate Video Retrieval

In this section, we evaluate the performance of our proposed video deduplication system in terms of recall for different values of K and different feature dimensions (d_{fisher}, d_{VGG}, and d_{thumb}). Table I presents the recall for normal and hard test sets generated from FIVR-200K [26] and VCSL [27] datasets. Given a positive query, the results in Table I evaluate our system's ability to retrieve the same video from the repository correctly.

As evident in Table I, the recall increases with the increase in K and feature dimensionality. The proposed system achieves a maximum recall of 98.8% at $K = 512$, $d_{fisher} = 128$, and $d_{VGG}/d_{thumb} = 64$ and a minimum recall of 96.5% for $K = 64$, $d_{fisher}/d_{VGG}/d_{thumb} = 32$ for FIVR-200K-Normal. In case of FIVR-200K-Hard, best recall of 96.5% is obtained at $K = 512$, $d_{fisher} = 128$, and $d_{VGG}/d_{thumb} = 64$. Similar to FIVR-200K dataset [26], the best recall is obtained at $K = 512$, $d_{fisher} = 128$, and $d_{VGG}/d_{thumb} = 64$ and worst recall at $K = 64$, $d_{fisher}/d_{VGG}/d_{thumb} = 32$ for the VCSL dataset [27]. The recall varies with K because the greater the number of retrieved frames from the k-d trees, the greater the chances of correct frames being retrieved, ultimately leading to retrieving the correct video. Regarding feature dimension, higher feature dimensions tend to generate more distinctive representations, leading to better recall at the expense of increased time consumption (discussed in the following subsections).

We only report recall because of how the 1-million and 4.5-million frame repositories are built. The repositories contain multiple same/similar videos from the FIVR-200K dataset [26] and VCSL dataset [27]. Sometimes, our system retrieves multiple videos from the same query class with timestamps different than the ground truth timestamp. This can be better visualized in Fig. 6 which shows this exact phenomenon. In Fig. 6, the video ID 'tOhDhU6Lh00' is actually from the same class as the ground truth video and is visually almost the same.

B. Duplicate Video Retrieval With Localization

This subsection evaluates the system for duplicate video retrieval with localization. In addition to the video ID, our system also retrieves the timestamps. A localized video is considered a correct match if it matches the ground truth video id and ground truth timestamps (with a tolerance of $\pm$0s, $\pm$1s, and $\pm$5s). A tolerance of $\pm$1 means that the

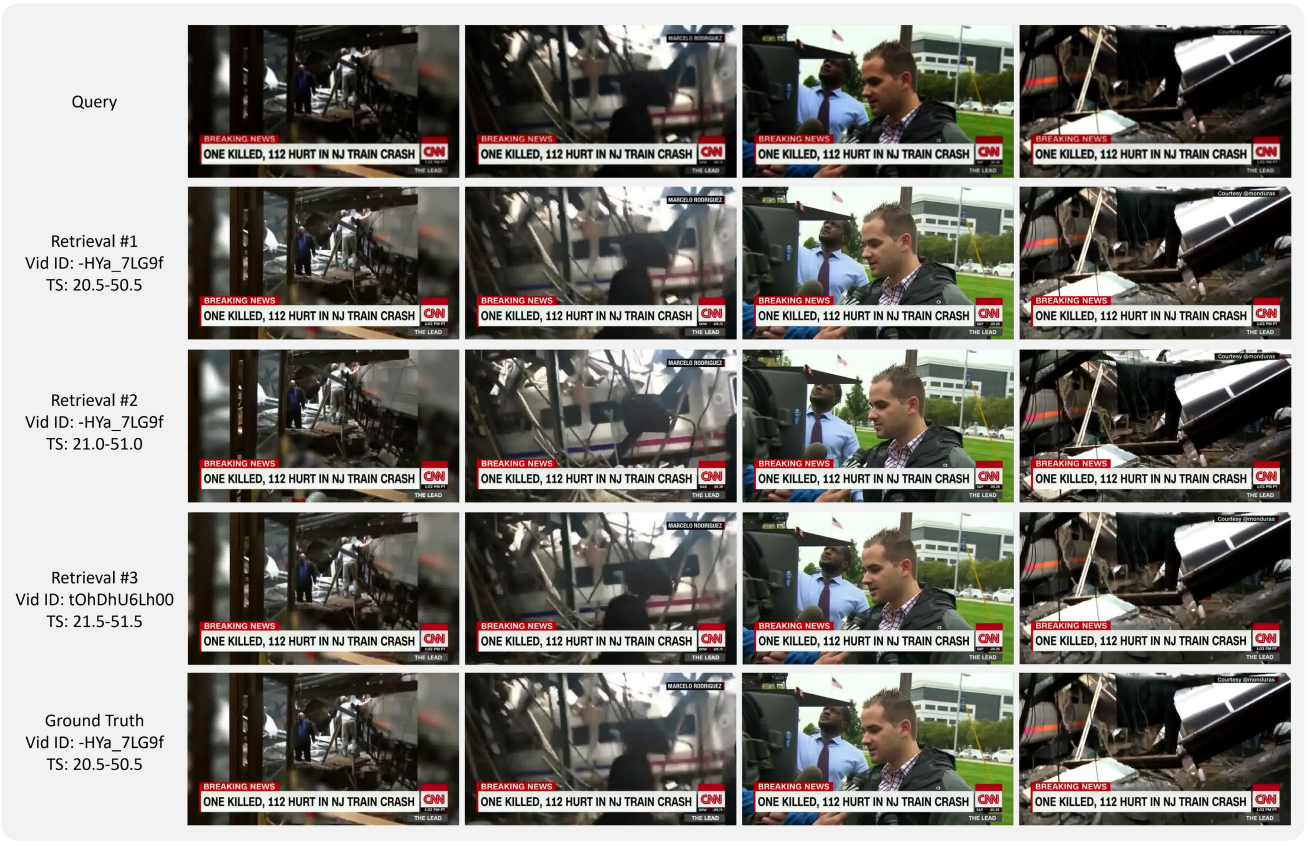


Fig. 6. Video retrieval results by our system. Note: Figure best viewed in color.

TABLE II
TABLE SHOWING TIMESTAMP ACCURACY (%) AT VARYING TOLERANCE VALUES

Dataset	Accuracy	Normal				Hard			
		K=64	K=128	K=256	K=512	K=64	K=128	K=256	K=512
FIVR-200K	Timestamp Accuracy ±0	91.6	94.1	95.5	**96.6**	74.2	80.4	84.5	**88.9**
	Timestamp Accuracy ±1	92.4	94.6	96.1	**97.3**	75.7	82	86	**89.8**
	Timestamp Accuracy ±5	92.6	94.8	96.1	**97.5**	76.2	82.3	86.2	**89.8**
VCSL Dataset	Timestamp Accuracy ±0	77.5	82.4	85.5	**88.3**	55.7	62	68.4	**73.9**
	Timestamp Accuracy ±1	78.1	82.9	86.2	**88.7**	57	63.6	69.5	**75**
	Timestamp Accuracy ±5	78.2	83.1	86.4	**88.7**	57.3	64.3	69.6	**75.2**

predicted timestamp is within ±1s seconds of the ground truth timestamps. This tolerance was added for fair evaluation since videos in both datasets tend to have static scenes spanning a couple of seconds. In such situations, our system retrieves the same video ID and varying timestamps since the frames are exactly the same. This can be better visualized in Fig. 6.

Table II presents the timestamp accuracy for feature dimensions $d_{fisher} = 128$, $d_{VGG} = 64$, and $d_{thumb} = 64$ at varying values of K for both FIVR-200K [26] and VCSL [27] datasets. As observed in Table II, the accuracy increases with the increasing value of K. This is true for normal and hard test sets from both FIVR-200K [26] and VCSL [27] datasets. The maximum timestamp accuracy ±5 of 97.5% and 89.8% was achieved at the highest value of $K = 512$ for FIVR-200K-Normal and FIVR-200K-Hard, respectively. The same holds true for the VCSL dataset [27] which achieves a maximum timestamp accuracy ±5 of 88.7% and 75.2% for the normal and hard test sets, respectively. Higher values of K retrieve

TABLE III
RECALL (%) AT DIFFERENT K VALUES FOR NEGATIVE QUERIES

Dataset	Search Repository	Feature Dimension (FV, Thumbnail, VGG)	Recall (K=64)	Recall (K=128)	Recall (K=256)	Recall (K=512)
DW Documentary (Negative Queries)	FIVR-200K 1-M frame/145 hr	32, 32, 32	100.0	100.0	99.9	99.8
		64, 32, 32	100.0	100.0	99.9	99.8
		128, 64, 64	100.0	99.9	99.9	99.7
	VCSL 4.5-M frame/636 hr	32, 32, 32	100.0	100.0	99.9	99.8
		64, 32, 32	100.0	100.0	100.0	99.9
		128, 64, 64	100.0	100.0	100.0	99.9

more frames, increasing the likelihood of retrieving the correct video ID with the correct timestamps. The results in Table II validates that our system can accurately localize the video from the 636 hour VCSL repository with a timestamp accuracy 88.7% (normal) and 75% (hard) with a small tolerance of ± 1 at $K = 512$. For the 145 hour FIVR-200K repository, timestamp accuracy of 97.3% (normal) and 89.8% (hard) was achieved for a tolerance of ± 1 at $K = 512$. Considering the huge size of repositories, being able to localize the video within ± 1 seconds reflects the robustness of the proposed video deduplication system.

Another important metric for evaluating our video deduplication system is its ability to reject false positives. Given a negative query, our system must not retrieve any videos. Table III presents the results for negative queries generated via videos downloaded from 'DW Documentary[7]'. The negative queries were searched against the 1 million frame/145 hour and 4.5 million frame/636 hour repositories generated via FIVR-200K [26] and VCSL [27] datasets, respectively. For negative videos, contrary to positive queries, the fewer frames retrieved by k-d trees, the greater the chances that no frames are left after the temporal consistency pruning; hence, no video is retrieved. This is exactly what can be observed by the results for negative queries, as shown in Table III. We can see that for the 1 million frame and 4.5 frame search repository, a recall of 100% is achieved at $K = 64$ while the lowest recall is obtained for the highest K value. This trend is also true for the various feature dimensions. As evident in Table III, the proposed system performs extremely well at rejecting false positives given negative queries as input.

C. Effect of Feature Dimension Reduction

In this subsection, we study the effect of dimension reduction on fisher vector f_{fisher}, thumbnail feature f_{thumb}, and VGG feature f_{VGG}. Figure 7 presents accuracy vs. feature dimension plots for each of the three features. We reduce the dimension of f_{fisher} and f_{VGG} to 32, 64, 128, 256, and 512 and calculate the timestamp accuracy ± 5 for different values of K. In the case of f_{thumb}, the feature dimension is reduced to 16, 32, and 64 before calculating the timestamp accuracy ± 5 at different K values. As evident in Fig. 7, higher dimensions and higher K values increase the timestamp accuracy ± 5. Notice that the maximum timestamp accuracy ± 5 of 61.3% and 95.2% was achieved at $K = 512$ and feature dimension 512 for f_{fisher} and f_{VGG}, respectively. For f_{thumb}, maximum timestamp accuracy ± 5 of 62.7% was obtained at $K = 512$ and feature dimension 512. Higher dimensions can better represent the data, leading to better performance. Dimensions higher than the ones shown in Fig. 7 were not tested since k-d trees become inefficient with high-dimensional data, leading to worst performance.

[7]https://www.youtube.com/c/DWDocumentary

TABLE IV
AVERAGE TIME IN MS/FRAME REQUIRED FOR RAW FEATURE GENERATION AND PCA PROJECTION FOR FIVR-200K-NORMAL AND VCSL-NORMAL DATASETS

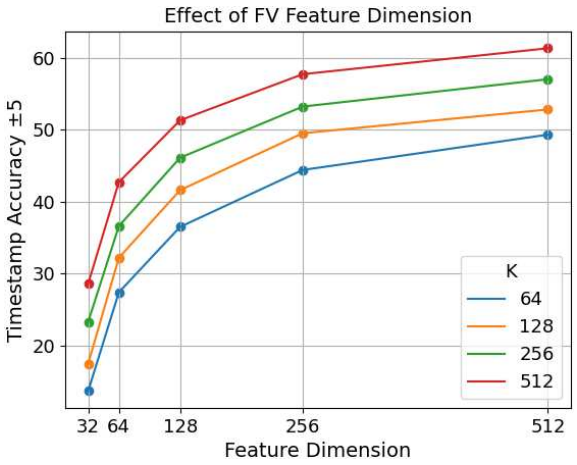
Time Consumption (ms/frame)	Feature	FIVR-200K Normal	VCSL Normal
Feature Generation	Fisher Vector[a] (1024-d)	177	118
	Thumbnail[a] (144-d)	1.6	1.3
	VGG[b] (4096-d)	8.1	7
PCA Projection	Fisher Vector[a] (128-d)	0.33	0.64
	Thumbnail[a] (64-d)	0.19	0.18
	VGG[a] (64-d)	0.84	2.58

[a] Results computed using CPU (i7-12700).
[b] Results computed using a single NVIDIA RTX A5000 GPU.

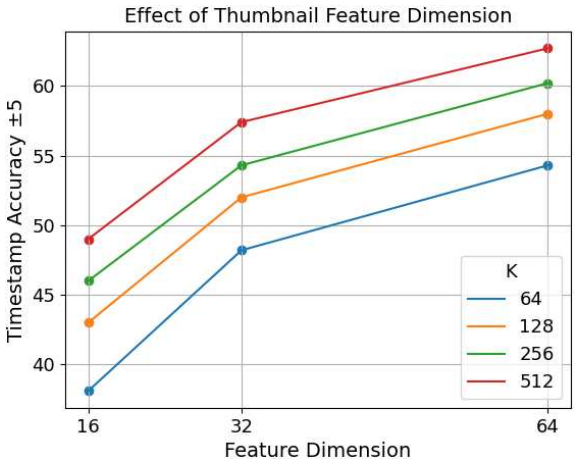
D. Time Consumption

This subsection presents the time consumption of the proposed video deduplication system for different steps involved in building the repository and during test query processing. It must be noted that all the results presented regarding time consumption are approximated and vary based on multiple factors, such as the hardware used. The time to save features to disk was excluded from these results since these times highly vary with the type of storage used (HDD or SSD). In addition, the time consumption results for PCA projection represent the time it takes to project a feature to a lower dimension given the input feature and trained PCA model.

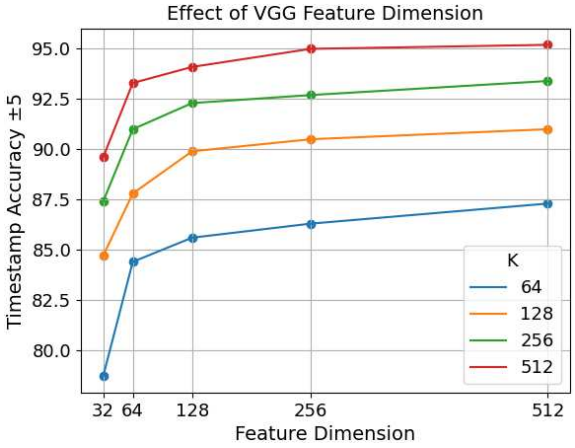
Table IV presents the time consumption for raw feature generation which includes raw feature generation followed



(a) Timestamp accuracy ± 5 (%) vs. FV feature dimension.



(b) Timestamp accuracy ± 5 (%) vs. Thumbnail feature dimension.



(c) Timestamp accuracy ± 5 (%) vs. VGG feature dimension.

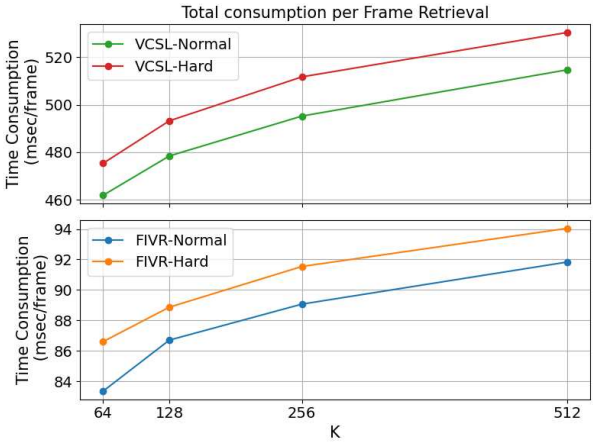
Fig. 7. Plots showing the effect of (a) FV, (b) thumbnail, and (c) VGG feature dimension on timestamp accuracy ± 5 (%) for FIVR-200k-Normal test set. Colored lines in each plot represent different values of K. Note: Figure best viewed in color.

by dimension reduction. As can be observed in Table IV, thumbnail and VGG feature generation is extremely fast, taking about 1.3 ms/frame and 7 ms/frame for VCSL-Normal, respectively. The fisher vector generation takes about 177 ms/frame and 118 ms/frame for FIVR-200K-Normal and VCSL-Normal, respectively. Fisher vector generation is a bit slower because it was implemented in MATLAB which is slower than Python (used to implement thumbnail and VGG feature generation). Regarding projecting features to a lower dimension via PCA, it takes a minimal amount of time to process. It consumes about 0.33/0.19/0.84 ms/frame (fisher vector/thumbnail/VGG) and 0.64/0.18/2.58 ms/frame (fisher vector/thumbnail/VGG) for FIVR-200K-Normal and VCSL-Normal, respectively.

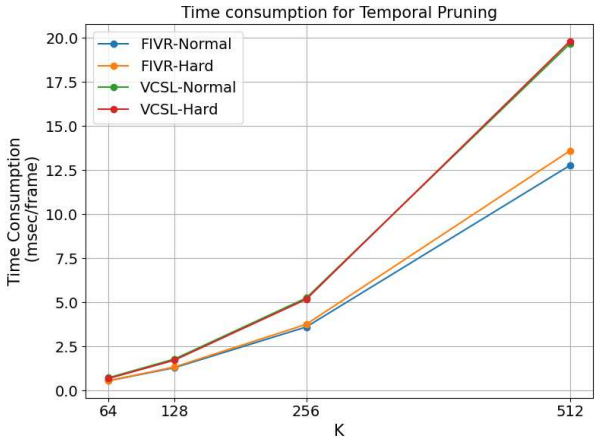
Table V presents the time consumption (in seconds) for building the repository, i.e., tree generation for the FIVR-200K [26] and VCSL [27] datasets. Given the projected features, the tree for fisher vector, thumbnail, and VGG feature can be generated in about 2.72s, 1.55s, and 1.64s, respectively, for the 1-Million frame/145 hour repository (FIVR-200K dataset [26]). The 4.5-Million frame/636 hour repository consumes about 14.71s, 8.13s, and 10.50s to build fisher vector, thumbnail, and VGG feature repository, respectively. It can be observed in Table V that a higher feature dimension and larger repository size increase the time to build a repository. The repository building time for normal and hard test sets is extremely fast due to the relatively small dataset size.

Fig. 8 presents the time consumption plots for frame retrieval (Fig. 8a) and temporal pruning (Fig. 8b). The total time/frame is shown in Fig. 8c. The results are presented for the FIVR-200K-Normal, FIVR-200K-Hard, VCSL-Normal, and VCSL-Hard test sets. Feature dimension of $d_{fisher} = 128$, $d_{thumb} = 64$, and $d_{VGG} = 64$ were used for computing these results. Frame retrieval time refers to the time it consumes to retrieve the frames from the frame retrieval step (Section III-B), temporal pruning is the time it takes to prune the frames retrieved during frame retrieval, and total time/frame is the total time it takes to process a video frame. All the results in Fig. 8 represent time consumption per frame. Time consumption per frame can better evaluate the system since a query video can be represented with a variable number of frames. All CPU cores were utilized during the k-d tree search.

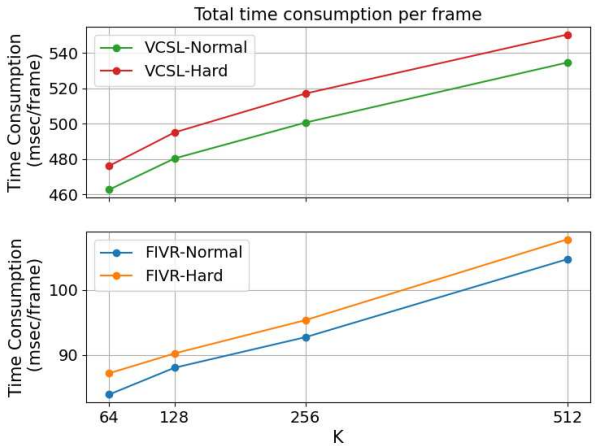
As evident in all the subfigures in Fig. 8, the time consumption for each task is directly proportional to the K value. In addition, the time consumption increases with the increase



(a) Time consumption per frame in msec for frame retrieval.



(b) Time consumption per frame in msec for temporal pruning.



(c) Total time consumption per frame in msec.

Fig. 8. Plots showing time consumption per frame in msec for (a) frame retrieval and (b) temporal pruning. The total time consumption per frame is shown in (c). Each line in every plot represents a different dataset. Feature dimension of $d_{fisher} = 128$, $d_{thumb} = 64$, and $d_{VGG} = 64$ were used for these results. Note: Figure best viewed in color.

TABLE V
TIME IN SECONDS REQUIRED FOR GENERATING KD-TREE (LEAF SIZE OF 32) FOR EACH OF THE THREE FEATURES

Time Consumption (s)	Feature	FIVR-200K Dataset [26]			VCSL Dataset [27]		
		1-M Frame/145 hr Repository	Normal	Hard	4.5-M Frame/636 hr Repository	Normal	Hard
Tree Generation	Fisher Vector (128-d)	2.72	0.01	0.758	14.71	0.7	0.0137
	Thumbnail (64-d)	1.55	0.022	0.019	8.13	0.005	0.011
	VGG (64-d)	1.64	0.005	0.004	10.50	0.004	0.0077

in the size of the search repository. This can be validated by the results for VCSL-Normal and VCSL-Hard datasets which were computed for the 4.5-million frame/636 hour repository. The results for the FIVR-200K-Normal and FIVR-200K-Hard test sets show significantly lower time consumption since these were computed for the 1 million frame/145 hour search repository. Our temporal consistence pruning is extremely fast, as observed in Fig. 8b with prune times as low as 0.56ms for FIVR-200K-Normal and FIVR-200K-Hard at $K = 64$. Most of the time is consumed searching the query frame against the search repository. It takes as low as 83.36 ms and 461.82 ms for FIVR-200K-Normal and VCSL-Normal test sets, respectively, at $K = 64$. Overall, it consumes 84ms and 463ms to process a query frame at $K = 64$ for the FIVR-200K-Normal and VCSL-Normal test sets, respectively. These results validate that our system is fast in processing the query

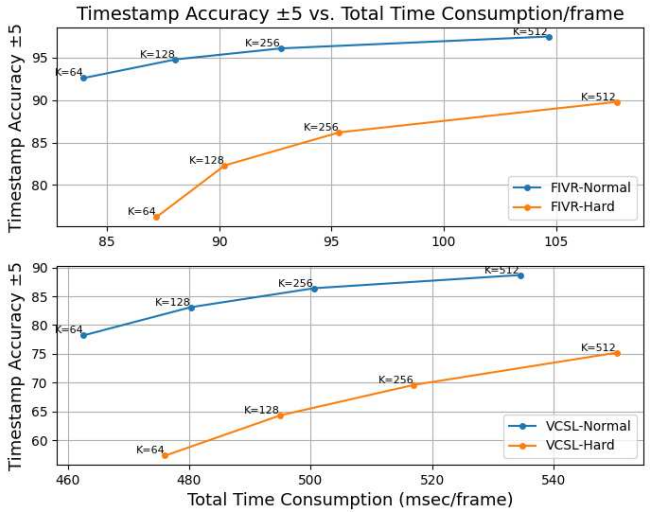


Fig. 9. Timestamp accuracy ±5 (%) vs. total time consumption/frame (in msec) plot for different K values. Each line represents a different dataset. Note: Figure best viewed in color.

TABLE VI
RESULTS FOR ABLATION STUDY (REMOVING EACH OF THE THREE FEATURES ONCE AT A TIME)

Fisher Vector	VGG Feature	Thumbnail Feature	FIVR-200K-Normal (K=64)	
			Recall (%)	Timestamp Accuracy ±5 (%)
×	✓	✓	96.5	90.3
✓	×	✓	87.1	73
✓	✓	×	96.1	89.1
✓	✓	✓	97.8	92.6

frames. A graph showing the timestamp accuracy ±5 versus time consumption at different K values can be seen in Fig. 9.

E. Ablation Study

We conduct an ablation study that signifies the importance of each of the three features used in our work. We experiment by removing one feature at a time and evaluating the system using only two features. In Table VI, we present the results for three different feature combinations - VGG feature/thumbnail feature, fisher vector/thumbnail feature, and fisher vector/VGG feature. For each combination, we present the recall and timestamp accuracy ±5 seconds for FIVR-200K-Normal and VCSL-Normal test sets. All these experiments are performed at $K = 64$ and feature dimensions of $d_{fisher} = 128$, $d_{VGG} = 64$, and $d_{thumb} = 64$. The importance of each of the three features is validated by the results presented in Table VI. The recall for FIVR-200K-Normal reduces to 96.5%, 87.1%, and 96.1% from 97.8% after removing fisher vector f_{fisher}, VGG feature f_{VGG}, and thumbnail feature f_{thumb}, respectively. A similar trend can be seen for timestamp accuracy ±5. These results show that the VGG feature is the most important feature as compared to the fisher vector and thumbnail feature. That being said, the fisher vector and thumbnail features also contribute to improving the recall. These results confirm that each feature is important and contributes towards improving performance.

VI. CONCLUSION

This study proposed a robust video deduplication system for large video repositories. The system is fast, consuming only a few milliseconds to search a query frame from a large-scale video repository of about 1 million frame/145 hour and 4.5 million frame/636 hours. The system can retrieve the duplicate video from the repository along with its precise location. The system is based on fisher vector, VGG feature, and thumbnail feature that are used to represent a video frame in latent space. A multiple k-d tree setup was designed for efficient frame retrieval along with a temporal consistence pruning strategy that can prune the retrieved video frames to retrieve the video ID and the timestamp. The system is extremely useful for tasks like saving storage space and copyright infringement detection. The system was evaluated on the large-scale FIVR-200K and VCSL datasets and the experimental results validate our claims.

Regarding future work, a promising direction is to incorporate neural processing unit-based acceleration for query feature generation on mobile devices and fully exploit the multi-core and multi-GPU resources on the cloud side for video retrieval.

REFERENCES

[1] D. Xu, T. J. Cham, S. Yan, L. Duan, and S.-F. Chang, "Near duplicate identification with spatially aligned pyramid matching," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 20, no. 8, pp. 1068–1079, Aug. 2010.

[2] K. Liao et al., "IR feature embedded BOF indexing method for near-duplicate video retrieval," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 12, pp. 3743–3753, Dec. 2019.

[3] H.-s. Min, J. Y. Choi, W. De Neve, and Y. M. Ro, "Near-duplicate video clip detection using model-free semantic concept detection and adaptive semantic distance measurement," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 8, pp. 1174–1187, Aug. 2012.

[4] C.-Y. Chiu and H.-M. Wang, "Time-series linear search for video copies based on compact signature manipulation and containment relation modeling," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 20, no. 11, pp. 1603–1613, Nov. 2010.

[5] M. M. Esmaeili, M. Fatourechi, and R. K. Ward, "A robust and fast video copy detection system using content-based fingerprinting," *IEEE Trans. Inf. Forensics Security*, vol. 6, no. 1, pp. 213–226, Mar. 2011.

[6] C. Kim and B. Vasudev, "Spatiotemporal sequence matching for efficient video copy detection," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 1, pp. 127–132, Jan. 2005.

[7] M. Douze, H. Jegou, and C. Schmid, "An image-based approach to video copy detection with spatio-temporal post-filtering," *IEEE Trans. Multimedia*, vol. 12, no. 4, pp. 257–266, Jun. 2010.

[8] S. He et al., "TransVCL: Attention-enhanced video copy localization network with flexible supervision," in *Proc. AAAI Conf. Artif. Intell.*, 2023, vol. 37, no. 1, pp. 799–807.

[9] S. Sowmyayani and P. A. J. Rani, "Content based video retrieval system using two stream convolutional neural network," *Multimedia Tools Appl.*, vol. 82, no. 16, pp. 24465–24483, Jul. 2023.

[10] Z. Li and M. Zhu, "A light-weight relevance feedback solution for large scale content-based video retrieval," *Int. J. Comput. Sci. Issues*, vol. 10, no. 1, p. 382, 2013.

[11] L. Shen, R. Hong, H. Zhang, X. Tian, and M. Wang, "Video retrieval with similarity-preserving deep temporal hashing," *ACM Trans. Multimedia Comput., Commun., Appl.*, vol. 15, no. 4, pp. 1–16, Nov. 2019.

[12] X. Wang, "Intelligent multi-camera video surveillance: A review," *Pattern Recognit. Lett.*, vol. 34, no. 1, pp. 3–19, Jan. 2013.

[13] C. Rougier, J. Meunier, A. St-Arnaud, and J. Rousseau, "Robust video surveillance for fall detection based on human shape deformation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 21, no. 5, pp. 611–622, May 2011.

[14] Y.-G. Jiang, Y. Jiang, and J. Wang, "VCDB: A large-scale database for partial copy detection in videos," in *Proc. Eur. Conf. Comput. Vis.*, Zurich, Switzerland. Cham, Switzerland: Springer, 2014, pp. 357–371.
[15] Y. Zhu, X. Huang, Q. Huang, and Q. Tian, "Large-scale video copy retrieval with temporal-concentration SIFT," *Neurocomputing*, vol. 187, pp. 83–91, Apr. 2016.
[16] Y. Zhang and X. Zhang, "Effective real-scenario video copy detection," in *Proc. 23rd Int. Conf. Pattern Recognit. (ICPR)*, Dec. 2016, pp. 3951–3956.
[17] Y. Cai et al., "Million-scale near-duplicate video retrieval system," in *Proc. 19th ACM Int. Conf. Multimedia*, Nov. 2011, pp. 837–838.
[18] L. Shang, L. Yang, F. Wang, K.-P. Chan, and X.-S. Hua, "Real-time large scale near-duplicate web video retrieval," in *Proc. 18th ACM Int. Conf. Multimedia*, Oct. 2010, pp. 531–540.
[19] Z. Huang, H. T. Shen, J. Shao, X. Zhou, and B. Cui, "Bounded coordinate system indexing for real-time video clip search," *ACM Trans. Inf. Syst.*, vol. 27, no. 3, pp. 1–33, May 2009.
[20] 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.
[21] V. E. Liong, J. Lu, Y.-P. Tan, and J. Zhou, "Deep video hashing," *IEEE Trans. Multimedia*, vol. 19, no. 6, pp. 1209–1219, Jun. 2017.
[22] J. Song, H. Zhang, X. Li, L. Gao, M. Wang, and R. Hong, "Self-supervised video hashing with hierarchical binary auto-encoder," *IEEE Trans. Image Process.*, vol. 27, no. 7, pp. 3210–3221, Jul. 2018.
[23] G. Kordopatis-Zilos, S. Papadopoulos, I. Patras, and Y. Kompatsiaris, "Near-duplicate video retrieval by aggregating intermediate CNN layers," in *Proc. MultiMedia Modeling, 23rd Int. Conf. (MMM)*, Reykjavik, Iceland. Cham, Switzerland: Springer, Jan. 2017, pp. 251–263.
[24] G. Kordopatis-Zilos, S. Papadopoulos, I. Patras, and Y. Kompatsiaris, "Near-duplicate video retrieval with deep metric learning," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops (ICCVW)*, Venice, Italy, Oct. 2017, pp. 347–356.
[25] J. Shao, X. Wen, B. Zhao, and X. Xue, "Temporal context aggregation for video retrieval with contrastive learning," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2021, pp. 3268–3278.
[26] G. Kordopatis-Zilos, S. Papadopoulos, I. Patras, and I. Kompatsiaris, "FIVR: Fine-grained incident video retrieval," *IEEE Trans. Multimedia*, vol. 21, no. 10, pp. 2638–2652, Oct. 2019.
[27] S. He et al., "A large-scale comprehensive dataset and copy-overlap aware evaluation protocol for segment-level video copy detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 21054–21063.
[28] K. Mikolajczyk and K. Mikolajczyk, "Scale & affine invariant interest point detectors," *Int. J. Comput. Vis.*, vol. 60, no. 1, pp. 63–86, Oct. 2004.
[29] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, Nov. 2004.
[30] M. Heikkilä, M. Pietikäinen, and C. Schmid, "Description of interest regions with local binary patterns," *Pattern Recognit.*, vol. 42, no. 3, pp. 425–436, Mar. 2009.
[31] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 815–823.
[32] L.-Y. Duan et al., "Overview of the MPEG-CDVS standard," *IEEE Trans. Image Process.*, vol. 25, no. 1, pp. 179–194, Jan. 2016.
[33] C. Henry, R. Liao, R. Lin, Z. Zhang, H. Sun, and Z. Li, "Lightweight Fisher vector transfer learning for video deduplication," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Jun. 2023, pp. 1–5.
[34] C. Henry, R. Liao, R. Lin, Z. Zhang, H. Sun, and Z. Li, "Fast and robust video deduplication," in *Proc. 2nd Mile-High Video Conf.*, May 2023, p. 160.
[35] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.
[36] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Miami, FL, USA, Jun. 2009, pp. 248–255.

Chris Henry (Student Member, IEEE) received the B.E. degree in electronic engineering from Hamdard University, Pakistan, in 2015, and the M.E. degree in IT convergence engineering from Gachon University, South Korea, in 2019. He is currently pursuing the Ph.D. degree in electrical and computer engineering with the University of Missouri–Kansas City, Kansas City, MO, USA. He is also affiliated with the Multimedia Computing and Communication Laboratory, University of Missouri–Kansas City. His research interests include image processing, computer vision, and deep learning.

Li Song (Senior Member, IEEE) received the B.E. and M.S. degrees in engineering and the Ph.D. degree in electrical engineering from Shanghai Jiao Tong University, Shanghai, China, in 1997, 2000, and 2005, respectively. From 2011 to 2012, he was a Visiting Professor with Santa Clara University, Santa Clara, CA, USA. He is currently a Full Professor with the Department of Electronic Engineering, Shanghai Jiao Tong University. He has more than 200 publications, more than 50 granted patents, and 18 standard technical proposals in the field of video coding and image processing. His research interests include video processing and multimedia systems. He was a recipient of the National Science and Technology Progress Award in 2015, the Okawa Foundation Research Grant in 2012, the Second Place Award from IEEE ICME-Twitch Grand Challenge in 2017, the Best 10% Paper Award from IEEE VCIP in 2016, and the Best Paper Award from the International Conference on Wireless Communications and Signal Processing in 2010. He was the area or session chair for various international conferences and workshops. He has been an Associate Editor of *Multidimensional Systems and Signal Processing* since 2012. He was a Guest Editor of IEEE TRANSACTIONS ON BROADCASTING, a special issue on the quality of experience of advanced broadcast services, in June 2018.

Zhu Li (Senior Member, IEEE) received the Ph.D. degree in electrical and computer engineering from Northwestern University in 2004. He was an AFRL Summer Faculty Member of the UAV Research Center, U.S. Air Force Academy (USAFA), from 2016 to 2018 and from 2020 to 2023. He was a Senior Staff Researcher with the Samsung Research America's Multimedia Standards Research Laboratory, Richardson, TX, USA, from 2012 to 2015, a Senior Staff Researcher with the FutureWei Technology's Media Laboratory, Bridgewater, NJ, USA, from 2010 to 2012, an Assistant Professor with the Department of Computing, The Hong Kong Polytechnic University, from 2008 to 2010, and a Principal Staff Research Engineer with the Multimedia Research Laboratory (MRL), Motorola Labs, from 2000 to 2008. He is currently a Professor with the Department of Computer Science and Electrical Engineering, University of Missouri–Kansas City, Kansas City, where he is also the Director of the NSF I/UCRC Center for Big Learning (CBL). His research interests include point cloud and light field compression, graph signal processing and deep learning in the next-gen visual compression, and image processing and understanding. He has more than 50 issued or pending patents, and more than 190 publications in book chapters, journals, and conferences in these areas. He received the Best Paper Award from the IEEE International Conference on Multimedia and Expo (ICME), Toronto, in 2006, and the IEEE International Conference on Image Processing (ICIP), San Antonio, in 2007. He is the Associate Editor-in-Chief of IEEE TRANSACTIONS ON CIRCUITS AND SYSTEM FOR VIDEO TECHNOLOGY from 2016 to 2019, and an Associate Editor of IEEE TRANSACTIONS ON IMAGE PROCESSING since 2020, IEEE TRANSACTIONS ON MULTIMEDIA from 2015 to 2018, and IEEE TRANSACTIONS ON CIRCUITS AND SYSTEM FOR VIDEO TECHNOLOGY from 2016 to 2019.