Saoudi and Jai-Andaloussi *J Big Data* (2021) 8:87
https://doi.org/10.1186/s40537-021-00479-x

Journal of Big Data

RESEARCH Open Access

A distributed Content-Based Video Retrieval system for large datasets

El Mehdi Saoudi[1*] and Said Jai-Andaloussi[1,2]

*Correspondence:
elmehdi.saoudi@gmail.com
[1] Department of Mathematics and Informatics, Faculty of Sciences, Hassan II University, Casablanca, Morocco
Full list of author information is available at the end of the article

Abstract

With the rapid growth in the amount of video data, efficient video indexing and retrieval methods have become one of the most critical challenges in multimedia management. For this purpose, Content-Based Video Retrieval (CBVR) is nowadays an active area of research. In this article, a CBVR system providing similar videos from a large multimedia dataset based on query video has been proposed. This approach uses vector motion-based signatures to describe the visual content and uses machine learning techniques to extract key frames for rapid browsing and efficient video indexing. The proposed method has been implemented on both single machine and real-time distributed cluster to evaluate the real-time performance aspect, especially when the number and size of videos are large. Experiments were performed using various benchmark action and activity recognition datasets and the results reveal the effectiveness of the proposed method in both accuracy and processing time compared to previous studies.

Keywords: Content-Based Video Retrieval (CBVR), Key frame selection, Distributed real-time processing

Introduction

The rapid increase in video data invokes the necessity for efficient indexing and retrieval systems. In the literature, two categories of indexing and retrieval systems have been reported. The first category focuses on annotation-based approaches, which employ textual information, attributes, or keyword annotations to represent the video content. The second uses visual features, such as motion, texture, shape, or colors, which describe the low-level visual content. However, with the massive amount of video data being generated every day, using textual descriptions is no longer practical. Because they take an enormous amount of time. In addition, they do not take into consideration the complexity, heterogeneity, and rich dynamic content present in the video. Therefore, it seems crucial to have automatic video content analysis systems capable of representing, modeling, indexing, retrieving, browsing, or searching information stored in large multimedia databases. These techniques are grouped into a single concept of Content-Based Video Retrieval systems (*CBVR)* [1, 2].

Various CBVR approaches based on low-level features are developed. For example [3] uses histogram-based color descriptors [4, 5] use motion and color information [6, 7]

combine color and texture features [8] is based on shape-based retrieval of video objects, and a relevant survey of indexing and retrieval techniques based on the low-level features can be found in [9]. Motion descriptors are important features of CBVR, classified into frame-based descriptors including: motion activity, warping parameters, camera motion, and object-based descriptors including: parametric motion and motion trajectory [10–12]. In combination with other low-level features such as color, motion-based indexing has proved a significant improvement in the performance of CBVR systems [13]. The color features are the most commonly used visual feature information in video retrieval since they are simple to extract in comparison to shape and texture features. The color feature is most robust to the complex background and independent of image orientation and size. Several color feature extraction methods have been proposed. The color histogram [14, 15] is the most widely used technique for representing color features. Maheen et al. [16] uses the color correlogram method. Li et al. [17] uses color coherence vectors. Alamdar et al. [18] uses a color distribution entropy (CDE) method.

Video analysis requires a set of features to describe visual information. Such features are generally extracted from the pixel values of the video frames. The pixel domain typically requires a large processing time because of the considerable amount of involved data. Instead, video analysis in the compressed domain requires minimal computation, which limits time and storage requirements. The majority of the daily generated video data are stored in compressed form for various purposes like streaming or distribution. With the rapid increase in the amount of video data, content analysis in compressed video starts generally with partitioning the video into groups of related frames named Group of pictures (GOP), then one or several key frames or representative frames can be extracted for every GOP. The visual contents of such representative frames are then used to represent the video for indexing and retrieval. There are several key frame extraction techniques available. The most basic approach is to extract the first and last frame of each shot as key frames. Even if these methods are simple and efficient, they often lose a lot of visual information to represent the shot. The approach in [19] presents a key frame selection method using motion analysis, [20] use a key frame extraction technique with unsupervised clustering [21] use a key frame extraction method based on automatic threshold and edge matching rate. It also exists visual histogram-based color key frame extraction techniques that can be applied to color features [22].

In this paper, we propose a CBVR system for large datasets, able to perform real-time retrieval. The proposed system processes compressed videos of different frame sizes and uses motion information and residual image data to create video signatures. This approach uses key frames to represent video content. Key frames were selected using a machine learning algorithm. To compare two sequences, a video clip representation model [23] that captures the dominating content and content changing trends of each clip is used as a technique for measuring similarity. We compared our method with the state-of-the-art methods on benchmark datasets for general performance evaluation. This approach was implemented in both, serial processing on a single machine and real-time distributed processing on a cluster of machines to evaluate real-time performance. The remainder of this paper is organized as follows. "Related works" section addresses related work in the area of CBVR using low-level features and the key frame selection. "Proposed method" section introduces our developed CBVR system. "System

architecture and implementation" section describes the system architecture and implementation. The experimental results and analysis are given in "The experimental results and analysis". Finally, "Conclusion" section concludes this paper.

Related works

In this section, we discuss some existing research approaches in CBVR. At the end of the last century, many institutes have started projects related to intelligent access to digital video. The Informedia Digital Video Library project of the Carnegie Mellon University [24] and the VideoQ project of Columbia University, which is the first on-line video search engine supporting automatic object-based indexing and spatio-temporal queries [25], are some pioneering works in this field. Recently, research work focused on video processing, in particular, shot boundary detection, video summarization, video segmentation, video object detection, and content-based video retrieval. Various CBVR approaches have been presented in the literature review in recent times. We proposed two works in this area. In [26], we developed a video retrieval system based on motion histogram and residual data of the video content using batch processing based on Apache Hadoop. In addition, in [27], a distributed real-time processing has been proposed. However, these two works used the entire video to create and compare the video signatures, which reduced significantly the precision and the processing time.

In the following, we present many relevant approaches for video retrieval, particularly those related to the proposed approach, which are: low-level visual feature extraction for video indexing and retrieval, and key frame selection for video summarization.

Low-level visual feature extraction

In the content-based video retrieval (CBVR) domain, various techniques of low-level feature extraction are reported. Hashemi et al. [28] developed automatic video analysis methods to study behaviors and child responses related to attention and facial expression. A facial landmark detector and tracker were deployed to track 49 facial landmark points. Thereafter, they estimate the child's head movement by tracking the distances and pixel-wise displacements of central facial landmarks. Agahian et al. [29] proposed a human action recognition framework with pose-based spatiotemporal features. They define a pose descriptor consisting of three elements. The first element contains the normalized coordinates of the raw skeleton joint information. The second element contains the temporal displacement information relative to a predefined temporal offset and the third element keeps the displacement information pertinent to the previous timestamp in the temporal resolution. Fan et al. [30] proposed an approach for End-to-End learning of motion representation for Video Understanding based on optical flow. Barmpoutis et al. [31] proposed a fire detection method from images using texture analysis. It consists of two main steps. In the first step, each image is fed as input into a faster R-CNN network for the detection and localization of candidate fire regions. In the second step, the extracted candidate fire regions are divided into rectangular patches (blocks), which are modeled using linear dynamical systems and projected to a Grassmann manifold. Venkateswarlal et al. [32] presented an ensemble of texture and shape descriptors using support vector machine classification for face recognition. In the proposed methodology, color, texture, and orientation feature descriptors are considered, such as HOG,

ICS, LBP, SHIFT, and color dominant structure descriptors. El ouadrhiri et al. [33] proposed a CBVR system based on the bounded coordinates of motion histogram. The system characterizes videos with signatures by using spatio-temporal features (e.g., motion direction, intensity, and residual information features) and uses the Bounded Coordinate System (BCS) as a technique for measuring similarity. Zhou et al. [34] proposed a shot boundary detection method for CBVR systems based on multi-level features. The approach uses image color features, and local descriptors and combines a kind of motion area extraction algorithm to achieve shot boundary detection. Firstly, they select candidate transition segments via color histogram and speed-up the robust features. Then, they perform cut transition detection through uneven slice matching, pixel difference, and color histogram. Finally, they perform gradual transition detection by motion area extraction, scale-invariant feature transform, and even slice matching. Sasithradevi et al. [35] proposed a shape and color-based model for shot boundary detection. The model collects the spatial color-shape attributes within a region and its spatial layout by partitioning the frame into regions at different resolutions. The HOG descriptor is then used to compute the edge-oriented histogram for each cell at every pyramidal level. Spolaôr et al. [36] proposed a systematic review of the relevant literature on CBVR techniques. Especially, the techniques of segmentation, reduction of dimensions, and extraction of low-level visual features. They found that strategies for cut-based segmentation, color-based indexing, k-means based dimensionality reduction, and data clustering have been the most frequent choices in recent papers.

Key frame selection

In recent years, many studies have been realized in the key frame extraction domain. Aote et al. [37] presented an automatic video annotation framework based on two-level key frame extraction mechanism. The proposed idea consists of fine-tuning the key frame extraction process, which extracts the key frames in two levels. At the first level, the first frame in the shot is considered as a key frame. However, to remove the redundancy, it enters into the second level and finds the optimal set of key frames by using the fuzzy c-means clustering technique. Color and texture features are used for feature extraction. Wu et al. [38] proposed a key frame-based clustering algorithm by integrating important properties of video such as color channels to gather similar frames into clusters. Finally, all clusters' centers will be collected as static video summarization. Das et al. [39] proposed a video key frame extraction algorithm that leads to video watermarking. The key frame extraction algorithm is based on use boundary luminosity analysis that is proven efficient regarding fast camera movement and moving objects in the video frames. The low-frequency DCT coefficients of key frames are chosen for watermarking. Luo et al. [40] proposed a key frame extraction method of surveillance video based on moving object detection and image similarity. The work consists of three modules. The first module is used to divide the video into several segments using moving object detection. The Second module reduces the redundant frames from the key video sequence using PSNR (Peak Signal-to-Noise Ratio). The last module extracts the wanted key frames using image similarity. Kimar et al. [41] presented an Eratosthenes sieve based key frame extraction technique for event summarization in videos. Spatial frequency, spectral residual, and color are the visual features extracted to distinguish such

key frames. Zhang et al. [42] proposed a motion-state-adaptive video summarization method based on spatio-temporal analysis. The proposed method utilizes spatiotemporal slices to analyze object motion trajectories and selects motion state changes as a metric to summarize videos. Then an attention curve based on the STS-CS model is formed to extract the key frames. Loukas et al. [43] proposed a key frame extraction method from individual shots of the operational video. The laparoscopic video was first segmented into video shots using an objectness model, which was trained to capture significant changes in the endoscope field of view. Each frame of a shot was then decomposed into three saliency maps to model the preference of human vision to regions with higher differentiation with respect to color, motion, and texture. Key frame was extracted from each state based on the highest state conditional probability of the corresponding saliency vector. Chen et al. [44] proposed a key frame selection method based on object detection and image quality. In this work, the deep learning-based target detection method is used to classify the key frames in the video by establishing a convolutional neural network model, which makes the target detection based on deep learning possible in the application of key frame extraction. Raikwar et al. [45] proposed framework that consists of two steps: First, the size of the input video shot is reduced by eliminating those frames of the shot which are not distinguishable by a human eye. Then the motion energy between the remaining frames of the input video shot is calculated and those frames are extracted as key frames in which the optical flow becomes maximum. Asim et al. [46] proposed a video summarization method to detect shot boundaries based on a combination of color features extracted from patches of a video frame instead of a whole frame. The approach uses color histogram, histogram of oriented gradients (HOG), saturation, and contrast to represent each frame. The key frames extracted from each shot, that are too similar are eliminated. The remaining key frames are then combined to create the final summary.

Proposed method

The proposed CBVR system uses motion features and residual image data to build the video signature. Furthermore, a key frame extraction method is applied to provide a compact video representation that contains salient and important video content information. In this system, the processing is performed in the compressed domain which offers relevant information pertaining to the visual content in the form of transform coefficients, quantization steps, coded block patterns, and especially motion vectors (VMs) with minimal computing resources. The subsections below describe the details of the proposed approach.

Key frame selection

In this approach, first, the video sequence is divided into GOP. In the GOP structure, three different frame types are used I-Frame (Intra-compressed), P-Frame (forward Predicted), and B-Frame (Bi-directional predicted). I-frame is an intra-coded picture without inter-frame prediction. P-frame is a predictive coded picture with an inter-frame prediction from the previous picture. B-frame is a bidirectional predictive-coded picture with inter-frame prediction both from a previous picture and a future picture [47], Fig. 1 shows an example of a GOP structure. In a GOP, there are high redundancies between

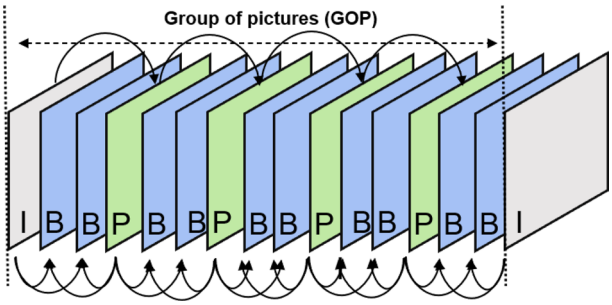

Fig. 1 The group of pictures (GOP) structure

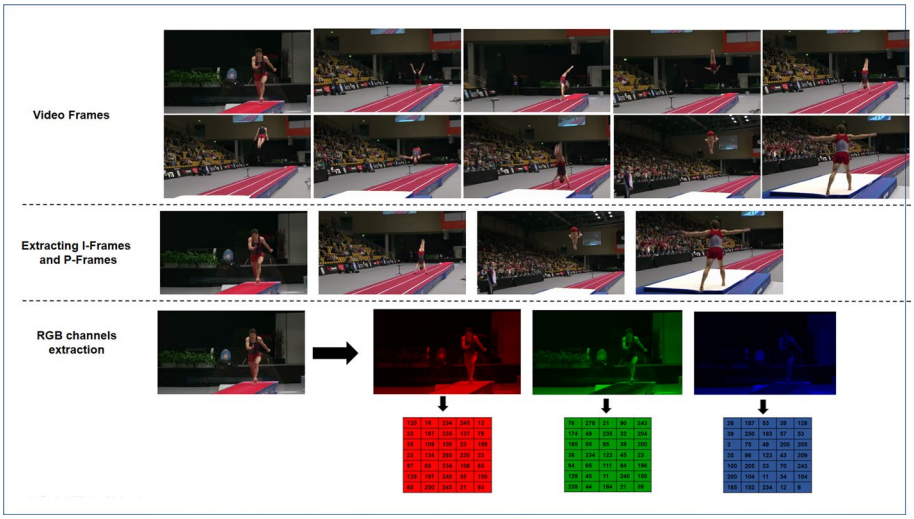

Fig. 2 The three RGB color channels extraction

frames. In our proposed approach, only I-Frames and P-frames of each GOP are selected as candidate key frames, because they contain all salient content required by this system [48]. The proposed key frame extraction method includes three main steps, which will be detailed below.

Color feature extraction

The proposed method uses a color-based frame feature extraction technique. First of all, we extract I-frames and P-frames for each GOP of an input video using FFMPEG. The frame on which the processing is performed is generally assumed to be an array of pixel values, each pixel has three values to define it. This means that the frame is defined not by a matrix of values but by three matrices. In this case, each pixel has values describing the levels of the three primary colors, red, green, and blue that are found in the pixel. These three values combine to define the overall color of the pixel. This method of describing pixels is called RGB classification. Thus, our method consists in extracting separately the three color channels red, green, and blue for the extracted I and P

frames. The extracted values will be stored in different matrices, as shown in Fig. 2. The extracted RGB matrices are then used as input for the next step.

The second step consists of measuring the similarity distance between the extracted frames while maintaining their sequencing order. In practice, we have implemented an image matching algorithm based on the Zero-mean Normalized Cross-Correlation (ZNCC) similarity measurement technique [49]. The idea is to determine the correlation coefficient between these frames.

Distance measure

Several relevant methodologies for measuring distances between frames are proposed in publications [50]. The ZNCC is one of the widely used methods for similarity measurement due to its efficiency in template matching. In this work, the ZNCC function is used with Eq. 1 to calculate the similarity distance between consecutive frames. ZNCC computes the correlation between the target image and the template. The distance range is the interval [−1,1] (1 for "perfect match", 0 for "no correlation", and -1 for "negative correlation"), in our context which is based on color features to measure the distance, we only consider the positive scores as similar. However, for every extracted matrix, the ZNCC coefficient between red, green, and blue channels of successive frames is determined and the similarity result is saved.

$$\mathrm{ZNCC} = \frac{\sum_{i=1}^{\mathrm{MN}} (\mathrm{x_i} - \bar{\mathrm{x}})\left(\mathrm{y_i} - \bar{\mathrm{y}}\right)}{\sqrt{\sum_{\mathrm{i=1}}^{\mathrm{MN}} (\mathrm{x_i} - \bar{\mathrm{x}})^2 \sum_{\mathrm{i=1}}^{\mathrm{MN}} \left(\mathrm{y_i} - \bar{\mathrm{y}}\right)^2}} \tag{1}$$

where: $\bar{x}$, and $\bar{y}$ are the mean intensity values of two frames x and y of size MN pixels. The mean of the resultant matrix is calculated to determine the distance.

Once the distances between frames are calculated, we exploit them to determine the key frames. For this, the k-means clustering algorithm is used. The objective is to group similar frames from each GOP together, using the calculated ZNCC distances. After the k-means algorithm converges to an optimal solution, the closest frame to the centroïd of each cluster will be designated as a key frame.

Frames clustering

Clustering is a method of grouping a set of objects into clusters so that the objects in the same cluster have high similarity, however, are very dissimilar to objects in other clusters [51]. Various clustering techniques have been presented [52]. The k-means clustering algorithm is one of the most fundamental unsupervised learning algorithms that solve clustering problems [53].

The proposed approach uses the k-means clustering algorithm based on the ZNCC distance between successive frames previously calculated, to group frames with high similarity in the same cluster. The Initialization of k (the number of clusters) is a crucial task for any clustering problem. In the scientific literature, various approaches have been proposed to determine the number of clusters for the k-means clustering calculation [54]. The current work will combine k-means with the elbow method [55] for finding the optimal value of k. This technique calculates the Sum Square Error *SSE* of each point to its nearest centroid with different values of k using Eq. 2 where n is the number of

observations, x_i is the value of the i_{th} observation, and 0 is the mean of all observations. The objective is to determine a small value of k that still has a low *SSE*, and the elbow usually represents where we start to have a lower return by increasing k. In this work, we run k-means clustering on the datasets for a range of values of k from 2 to 8 for every GOP.

$$\mathrm{SSE} = \sum_{\mathrm{i}=1}^{\mathrm{n}} (\mathrm{x_i} - \bar{\mathrm{x}})^2 \tag{2}$$

The following algorithm describes this procedure in detail:

Algorithm 1: The adopted k-means clustering algorithm.

Input:

- Set of N frames represented by their ZNCC distance values, denoted by x;
- The desired number of clusters, denoted by k;

Output:

A set of key frames $\mathrm{I_{KF}} = \{\mathrm{I_{KF_1}}, \mathrm{I_{KF_2}}, ..., \mathrm{I_{KF_k}}\}$;

begin

1- Cluster centers ($\mathrm{C_k}$) initialization ;

repeat

2- Assign all frames to the closest cluster centroïd

$$\mathrm{x_i} \in \mathrm{C_k} \;\; if \;\; \forall_{\mathrm{j}} \, |\mathrm{x_i} - \mu_{\mathrm{k}}| = \min |\mathrm{x_i} - \mu_{\mathrm{j}}| \tag{3}$$

where, μ_k : cluster k centroïd;

3- Recompute the centroïds of newly formed clusters;

$$\mu_k = \frac{1}{N} \sum_{x_i \in C_k} x_i \tag{4}$$

until *Centroids of newly formed clusters do not change*;

4- Select the closest frame to the centroïd of each cluster $\mathrm{I_{KF}}$ using the Euclidean distance.

End

At the end of this step, the closest frame to the centroïd of each cluster will be designated as a key frame. Knowing that the I-frame is the frame with which each GOP begins and it contains most of the vital information, we decided to add it to the list of designated key frames. The adopted key frame extraction method is summarized in Fig. 3.

Video signature

The efficient analysis of video data requires an effective and compact description based on their content. For that, a video signature is computed on the basis of the motion information and residual image data extracted from key frames. Therefore, the signature will be represented by two categories from the stream:

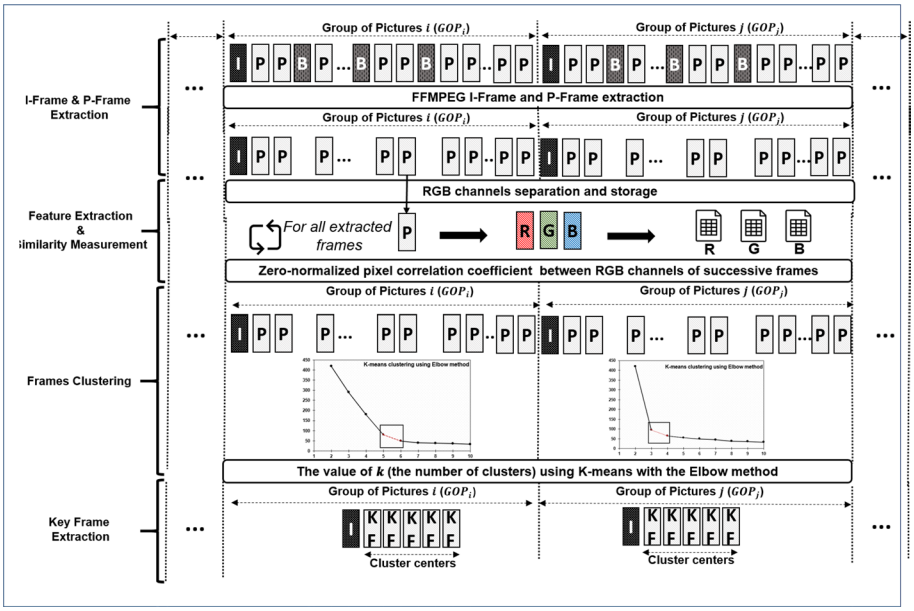

Fig. 3 The key frame extraction method

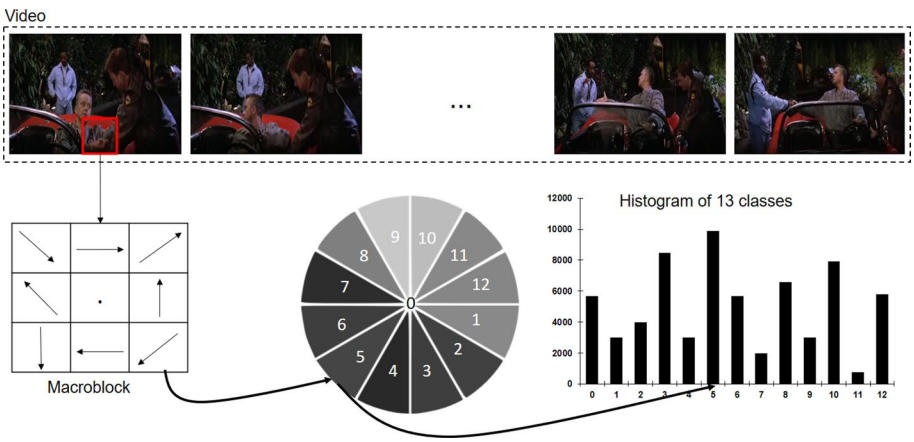

Fig. 4 Motion vector histogram

1) The motion vector histogram;
2) The residual error which represents the difference between predicted and original frames.

Motion histogram

Firstly, motion vectors are extracted from the video with a personalized *FFMPEG* library. Then the motion histogram is created based on the motion vectors of key frames. Each motion histogram represents one frame. Due to a lot of directions of motion (360^o) which can display in a frame, 12 possibilities and a separate bin $M = 0$ for zero-length motion vectors are considered (Fig. 4).

The motion histogram is calculated with Eq. 6. Therefore, the first part of the video signature will be represented by:

1. *Direction:* The highest amount of vector $\mu(\mathrm{x,y})$ on the histogram calculated by Eq. 5, where μ represents the motion vector change of a macro-block between $\mathrm{Frame_i}$ and $\mathrm{Frame_{i+1}}$, $\mathrm{x} = (\mathrm{x_{Frame_{i+1}}} - \mathrm{x_{Frame_i}})$, $\mathrm{y} = (\mathrm{y_{Frame_{i+1}}} - \mathrm{y_{Frame_i}})$;
2. *Class:* The ID of the dominant direction;
3. *Intensity:* The median of total motion vectors of the dominant class is calculated with Eq. 7.

$$\Omega(\mu) = \begin{cases} arcoss\ \frac{x}{|\mu|}\ ,\ y\ \geq 0 \\ 2\pi - arcoss\ \frac{x}{|\mu|}\ ,\ y < 0 \end{cases} \tag{5}$$

$$\mathrm{Histogram}(\mu) = \begin{cases} 0\ ,\ \mu = (0,0) \\ 1 + \left(\left[\Omega(\mu) \frac{M}{2\pi} + \frac{1}{2} \right] mod M \right)\ ,\ Otherwise \end{cases} \tag{6}$$

where, μ represents the motion vector change of a macro-block, and M is the number of classes in the histogram (13 classes, from 0 to 12).

$$\mathrm{Intensity}_\mu = \frac{1}{\mathrm{D}} \sum_{\mathrm{i=1}}^{\mathrm{D}} |\mu|\ \ ;\ (\mathrm{D{:}Direction}) \tag{7}$$

Motion compensation of residual information

To define the second part of the signature, which represents the residual information, we used the Generalized Gaussian Distribution method (GGD) [56], determined by Eq. 8 as a statistical model for motion compensation of residual information. For each macro-block of key frames, the residual information is calculated as the difference between the predicted content of the macro-block and the actual content of the macro-block.

$$\mathrm{P}(\mathrm{x}, \alpha, \beta) = \frac{\beta}{2\alpha\Gamma\left(\frac{1}{\beta}\right)} \mathrm{e}^{-\left(\frac{|\alpha|}{\alpha}\right)\beta} \tag{8}$$

The gamma function is $\Gamma(\mathrm{x}) = \int_0^\alpha \mathrm{e}^{-\mathrm{t}} \mathrm{t}^{\mathrm{x-t}} \mathrm{dt}\ ;\ \mathrm{x} > 0$, where:

- α: A scale factor corresponds to the standard deviation of the Gaussian distribution;
- β: A shape parameter.

These two parameters are calculated using a maximum likelihood estimator of the $GGD(\hat{\alpha}, \hat{\beta})$. Varanasi et al [56] proved that the exclusive solution of $(\hat{\alpha}, \hat{\beta})$ is taken by the following equations:

$$\begin{cases} \hat{\alpha} = \left(\frac{\hat{\beta}}{L} \sum_{i=1}^{L} |x_i| \right)^{\frac{1}{\hat{\beta}}} \\ 1 + \frac{\Psi\left(\frac{1}{\hat{\beta}}\right)}{\hat{\beta}} - \frac{\sum_{i=1}^{L} x_i^{\hat{\beta}} \log|x_i|}{\sum_{i=1}^{L} |x_i|^{\hat{\beta}}} + \frac{\log\left(\hat{\beta} \frac{\sum_{i=1}^{L} |x_i|^{\hat{\beta}}}{L} \right)}{\hat{\beta}} = 0 \end{cases} \tag{9}$$

Supposing that each x_i(coefficient for one key frame) is independent and L is the total of key frame's blocks, and the digamma function is $\Psi(\mathrm{t}) = \frac{\Gamma^{(\mathrm{t})}}{\Gamma(\mathrm{t})}$. At the end of this section, all data composing the signature are extracted and each video is now represented with the following signature:

$$\mathrm{Signature}_{\mathrm{Video_i}} = \begin{Bmatrix} D_{KF1},\ C_{KF1}, I_{KF1}, \alpha_{KF1}, \beta_{KF1} \\ D_{KF2},\ C_{KF2}, I_{KF2}, \alpha_{KF2}, \beta_{KF2} \\ D_{KF3},\ C_{KF3}, I_{KF3}, \alpha_{KF3}, \beta_{KF3} \\ D_{KFn},\ C_{KFn}, I_{KFn}, \alpha_{KFn}, \beta_{KFn} \end{Bmatrix} \tag{10}$$

where n is the number of the last key frame of a video.

Signature optimization

At this point, each video is presented with a signature of 5 parameters. To make the processing much simple and fast, especially with large datasets, we used the Principal Component Analysis (PCA) method to reduce the dimension of the signature while maintaining up to 90% of its performance [57]. The central idea of principal component analysis is to reduce the dimensionality of a data set consisting of a large number of interrelated variables while retaining as much as possible of the variation present in the data set [58, 59]. As a result, a two-dimensional representation was adopted as shown in Fig. 5.

Similarity measurement

For similarity measurement between videos, we adopted a video representation model called Bounded Coordinate System (BCS), which is the first single representative capturing the dominating content and content changing trends of a video. It summarizes a video by a coordinate system, where each of its coordinate axes is identified by the PCA and bounded by the range of data projections along the axis. To measure the similarity between two BCSs, Huang et al. [23] consider the operations including translation, rotation, and scaling for coordinate system matching. To capture the data information more

Fig. 5 Two-dimensional representation of the signature

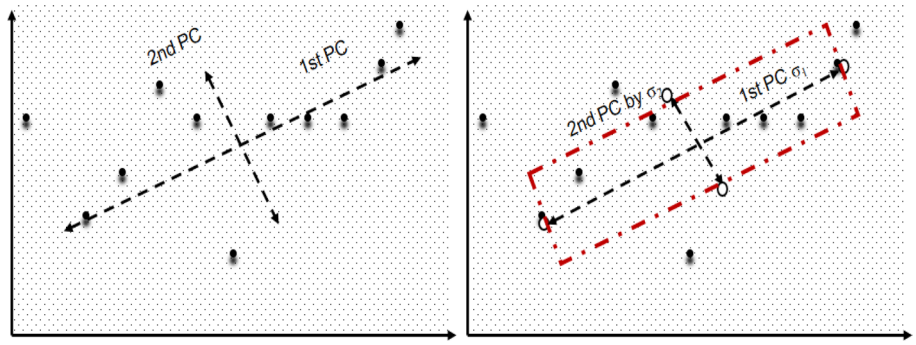

Fig. 6 Bounded principal components

exactly and avoid the negative effect of noise, BCS redefine the length of the principal component by the standard deviation (*i.e.*, σ) of data projections (Fig. 6).

For a principal component, denoted as Φ_i which identifies a line (or direction), its Bounded Principal Component (BPC), denoted as $\ddot{\Phi}_i$, identifies a segment of the line bounded by σ_i, where σ_i is the standard deviation of projections on σ_i away from the mean for all data points. The length of Bounded Principal Component $\left\|\ddot{\Phi}_i\right\|$ is $2\sigma_i$.

The similarity measure of BCS integrates two distances: the distance between two origins by translation, and the distance between each pair of bounded axes by rotation and scaling [23]. Let a video clip $X = \{x_1, x_2, x_3, \ldots, x_n\}$, where x_i is a d-dimensional feature vector, its Bounded Coordinate System $BCS(X) = (O, \ddot{\Phi}_1, \ddot{\Phi}_2, ..., \ddot{\Phi}_d)$ is determined by the mean for all x_i denoted as O (the origin of the coordinate system) and d orientations and ranges (the bounded axes of coordinate system ($\ddot{\Phi}_i$)). To compute the distance between two videos X and Y, where $BCS(X) = (O, \ddot{\Phi}_{X_1}, \ddot{\Phi}_{X_2}, ..., \ddot{\Phi}_{X_d})$ and $BCS(Y) = (O, \ddot{\Phi}_{Y_1}, \ddot{\Phi}_{Y_2}, ..., \ddot{\Phi}_{Y_d})$, the Eq. 11 is used:

$$D(BCS(X), BCS(Y)) = \left\|O^X - O^Y\right\| + \frac{\left(\sum_{i=1}^{d^Y} \left\|\ddot{\Phi}_i^X - \ddot{\Phi}_i^Y\right\| + \sum_{i=d^Y+1}^{d^X} \left\|\ddot{\Phi}_i^X\right\|\right)}{2} \quad (11)$$

The distance function defined above takes into account two factors:

1) The distance between the two origins (translation) calculated by $\left\|O^X - O^Y\right\|$.
2) The distance between each pair of bounded axes that represent the rotation and the change of scale, calculated by $\frac{\sum_{i=1}^{d^Y} \left\|\ddot{\Phi}_i^X - \ddot{\Phi}_i^Y\right\| + \sum_{i=d^Y+1}^{d^X} \left\|\ddot{\Phi}_i^X\right\|}{2}$.

As mentioned in Fig. 7, the first distance demonstrates the global difference between two sets of points, while the second distance demonstrates the average difference of all corresponding bounded principal components which reflect the content changing trends and ranges [23].

System architecture and implementation

In CBVR systems, there are typically two main phases: feature extraction and retrieval. Both are CPU consuming and time-consuming due to the highly complex and repetitive operations. In most cases, computer vision applications use a small

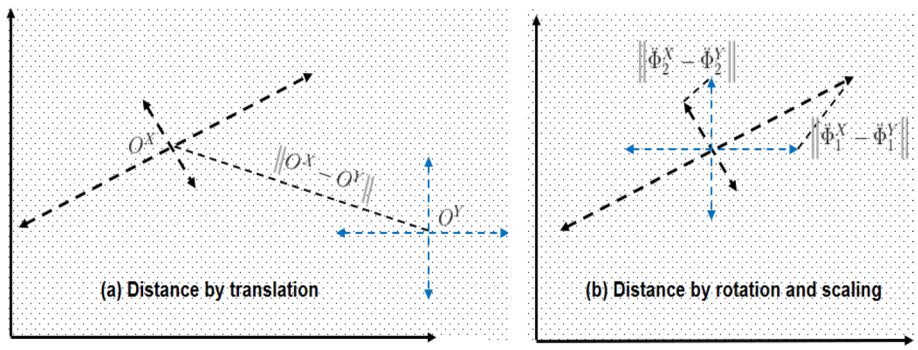

Fig. 7 BCS similarity measure

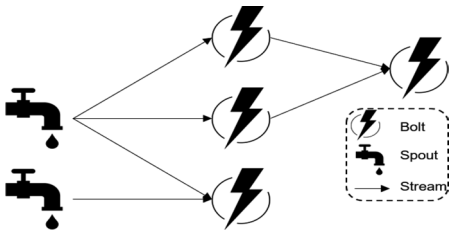

Fig. 8 The storm topology

amount of videos due to the difficulty in acquiring computational resources and storage for large datasets. As the progress of unstructured data increases, analytical systems should assimilate and interpret images and videos as well as interpret structured data such as text. Furthermore, real-time analysis on large-scale datasets is becoming increasingly important in many fields including medicine, video surveillance, advertisements, etc. To deal with these issues, the proposed CBVR approach is implemented using a distributed real-time video computation framework based on Apache Storm cluster. Furthermore, for reasons of precision evaluation, the approach was implemented also using serial processing on a single machine.

Storm distributed real-time computation system: a brief description

Apache Storm is a free and open-source distributed real-time computation system. It has many use cases: real time analytics, online machine learning, continuous computation, distributed RPC, ETL, and more. In a Storm cluster, there are two kinds of nodes: the master node and the worker nodes. The master node runs a daemon called *Nimbus*. Nimbus is responsible for distributing code around the cluster, assigning tasks to machines, and monitoring for failures. Each worker node runs a daemon called the *Supervisor*. The supervisor listens for the work assigned to its machine and starts and stops worker processes as necessary based on what Nimbus has assigned to it. Coordination between the master node and the processing nodes is done through a ZooKeeper cluster [60]. To do real-time computation on Storm, you create what are called *topologies*. A topology is composed of three main components: stream, spout,

and bolt as shown in Fig. 8. The core abstraction in Storm is the *stream*. A stream is an unbounded sequence of tuples. A spout is a source of streams. A bolt consumes any number of input streams, does some processing, and possibly emits new streams [61].

Architecture and implementation

The proposed Storm architecture consists of two topologies. We developed the first topology (*T1*) with six main steps and the second (*T2*) with two steps. Each element of the topologies will be detailed. The structure of the proposed Storm topologies is presented in Fig. 9.

- T1 1st Step: The Hadoop Distributed File System (HDFS) Spouts read all videos loaded by user file by file from HDFS using multiple instances for parallel processing. The HDFS Spouts allow feeding data into Storm from a monitored HDFS directory [61];
- T1 2nd Step: Bolts 1 receive videos from the HDFS Spout, select key frames, and extract the motion vectors, then they submit the results to the next bolt of the first topology for signature calculation;
- T1 3rd Step: Bolt 2 receive tuples from Bolts 1 to calculate video signature parameters which are: *Direction*, *Class* and *Intensity* by using VM-Sig Bolts and *Alpha*, *Beta* by using Res-Sig Bolts. Both parts of the signature are then moved to the next bolts;
- T1 4th Step: Bolts 3 consume streams from Bolts 2, aggregate the two signatures parts, and generate a stream that is later processed by Bolt 4;
- T1 5th Step: Bolts 4 receive tuples from Bolts 3, reduce the signature dimension using the PCA method, then submit the new signature the next bolts of the topology;
- T1 6th Step: Bolts 5 consume streams from Bolts 4, calculate the Bounded Coordinate System distances (translation and rotation), and store the results into an HDFS folder which will be used as a data source for the second topology.
- T2 1st Step: At the start of the second topology, the HDFS Spouts read the offline BCS values file (this file was prepared in the offline phase) which contains the trans-

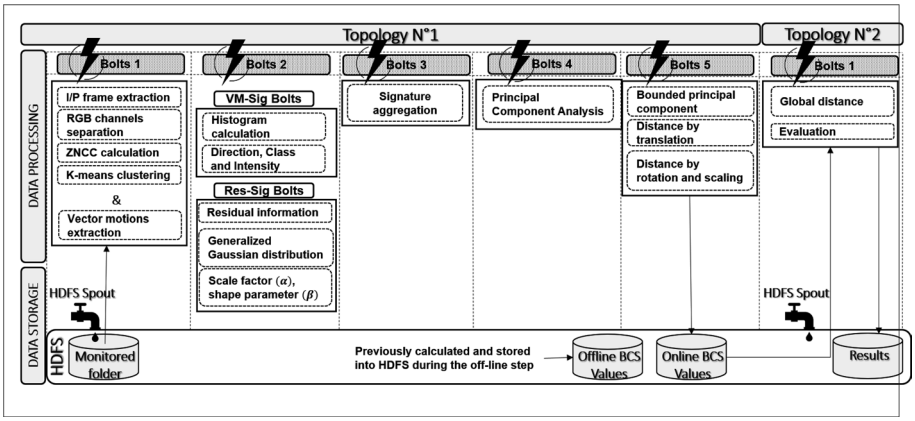

Fig. 9 The proposed CBVR approach using storm real-time distributed system

lation and rotation distances of all videos of the dataset, plus the BCS values file of the current online video. Then it submits them to the Bolt;

- T2 2nd Step: Bolts 1 receive the streams, calculate the similarity between the online video query and the videos stored on the database, using BCS distances. Then they determine the five best videos similar to the videos in question, and stores the results on HDFS.

The experimental results and analysis

Overview of exeperimental evaluations

To examine the performance of our method, we conducted many evaluation tasks. Our pilot studies were performed to solve the following issues:

1) How effective is our CBVR method? Can it rival existing solutions?
2) Can the proposed key frame extraction method improve the retrieval performance?
3) Can the real-time distributed computation improve the retrieval performance in large datasets by speed up the processing time?

Theoretically, to compare various approaches of video retrieval, every one of them must be tested on identical data sets and measured through identical metrics. Therefore, the proposed approach was evaluated through different benchmark datasets comprising HOLLYWOOD2 Human Actions and Scenes Dataset [62, 63], HMDB51 Human Motion Database [64], UCF50 Action Recognition Data Set [65, 66] and Olympic Sports Dataset [67]. The performance of our approach is compared with state-of-the-art CBVR methods.

Performance evaluation of the proposed method

Parameter study

In this section, the selection of the number of clusters k, is introduced, since it defines the number of key frames, and it is significant for the efficiency of our method. In this

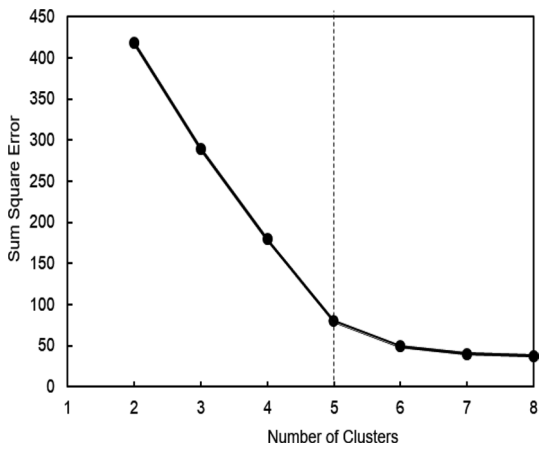

Fig. 10 Determination of the best cluster using the Elbow method

work, the number of clusters k is selected based on the elbow method (see page 7). Thus, we ran k-means clustering on the data sets for a range of values of k from 2 to 8, and for each value of k, we compute the sum of squared errors (SSE) and we plot a line chart as shown in Fig. 10. The algorithm is stopped once the centroids of newly formed clusters are not changing. Even after multiple iterations, if we are getting the same centroïde for all clusters, we can say that the algorithm is not learning any new pattern and it is a sign to stop it.

Based on Fig. 10, we notice that there was no significant change for the SSE value from $k = 2$ to $k = 4$, but the change is quite drastic at the value of $k = 5$. Then we start to have a lower return of SSE by increasing k (6, 7, and 8). The objective is to determine the small value of k that still has a low *SSE*. The elbow usually represents where we start to have a lower return by increasing k. The dashed solid line in Fig. 10 indicates the number of clusters suggested on the basis of the available data. Accordingly, $k = 5$ is the best number of clusters to use. This is reinforced by the number of iterations needed to form the final clusters (4 iterations) as shown in Table 1. In the same table, we can observe that the number of iterations in $k = 5$ is less than the number needed if we are using another cluster.

Results and discussion

The proposed approach is implemented and assessed on a single node and on a real-time distributed cluster of nodes. All nodes are with an Intel(R) Core (TM) i3-6100 CPU @3.70Ghz with Ubuntu 16.04 server, 8 GB RAM, and 320 GB of disk capacity. Results of each dataset and comparisons with state-of-the-art approaches are discussed in separate sections.

HMDB51 Human Motion Database HMDB51 dataset is collected from various sources, mostly from movies, and a small proportion from public databases such as the Prelinger archive, YouTube, and Google videos. The dataset contains 6849 clips divided into 51 action categories, each containing a minimum of 101 clips. The action categories can be grouped into five types related to general facial actions, general facial action with object manipulation, general body movements, body movements with object interaction, and body movements with human interaction. We are respecting the initial protocol using three train-test splits [68]. For each action category, we selected sets of 70 training and 30 testing clips. The average accuracy over the three splits is considered as

Table 1 Comparison of the number of clusters with SSE and the number of iterations needed

Number of clusters	SSE Values	SSE Difference	Number of iterations
2	419	0	5
3	290	129	7
4	182	108	5
5	**80**	**102**	**4**
6	52	28	6
7	40	12	5
8	38	2	5

Table 2 Performance comparison of our results to the state of art

HMDB51		**UCF50**	
Method	**Accuracy (%)**	**Method**	**Accuracy (%)**
Phan et al. [69]	31.5	Yan et al. [76]	41.45
Zheng et al. [70]	64.9	Banerjee et al. [77]	78
Wang et al. [71]	68.3	Zuo et al. [78]	90.3
Our method	72.37	Our method	79.95
HOLLYWOOD2		**Olympic-Sports**	
Method	**MAP (%)**	**Method**	**MAP (%)**
Zhang et al. [72]	55.10	Sumer et al. [74]	67.9
El Ouadrhiri et al. [33]	64.10	Sanakoyeu et al. [75]	78
Yi et al. [73]	76.70	Yi et al. [79]	94.10
Our method	67.08	Our method	84.91

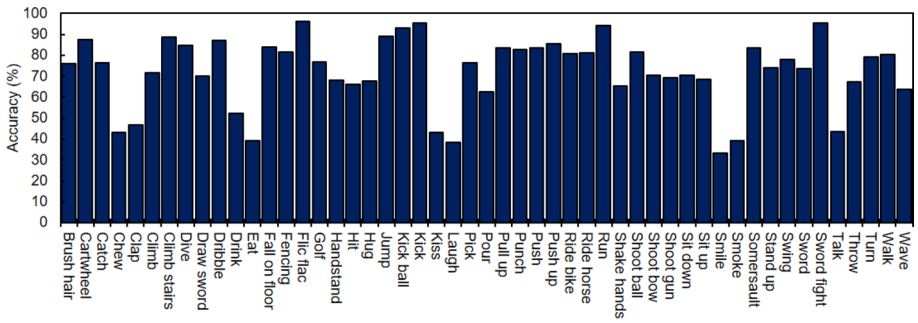

Fig. 11 The detailed accuracy by class for the HMDB51 dataset

performance measure. It should be underlined that in all tests we process the original videos, not the stabilized ones.

Using the HMDB51 dataset, the proposed approach is compared with similar approaches reported in the literature, including Phan et al. [69], Zheng et al. [70], and Wang et al. [71]. The obtained results on the test set of HMDB51 dataset are presented in Table 2. The detailed accuracy by class is displayed in Fig. 11. On this dataset, Phan et al. reported 31.5% accuracy while Zheng et al., and Wang et al. obtained 64.9%, and 68.3%, respectively. Our method increases the accuracy score by 4% on this dataset. In this challenging dataset, our approach produced good results. From Fig 11, it can be observed that not many categories accuracies are superior to 90%, but the majority of them are in the range of 70% to 90%. In all classes, the accuracy exceeds 40% except *Eating*, *Laugh*, and *Smile*. Because of the overlapping of visual content and motion information with the videos of other classes. Furthermore, motion vectors in these three classes are less present. Thus, the analysis of such videos is a more challenging task.

Hollywood2 Dataset The Hollywood dataset contains 12 classes of human actions, distributed over 1707 video clips with more than 14.1 h of video in total, split into two parts, training dataset (823 videos) and testing dataset (884 videos). This dataset is

very comprehensive and considered as a benchmark in action recognition literature. Moreover, this dataset is one of the most challenging for action recognition evaluation, due to the fact that many overlapping actions are performed within each category. For example, an action *Sit down* is performed next to an eating table, which is in turn also represented in the action *Eating*. The accuracy of the proposed approach is evaluated by the Mean Average Precision (MAP) metric. Average precision is calculated using Eq. 12.

$$\mathrm{MAP} = \frac{\sum_{k=1}^{n} (\mathrm{P(k)} \times \mathrm{rel(k)})}{\mathrm{N_r}} \tag{12}$$

where k is the rank in the sequence of retrieved videos, n is the number of retrieved videos, P(k) is the precision at cut-off k in the list, $rel(k)$ is an indicator function equaling 1, if the video at rank k is a relevant video, zero otherwise.

Using this dataset, our approach is compared to different state-of-the-art methods, notably El Ouadrhiri et al. [33], Zhang et al. [72] and Yi et al. [73]. The MAP achieved on this dataset is shown in Table 2, the detailed precision by class is presented in Fig. 12. From the obtained results, it is observed that the proposed approach is more accurate compared to El Ouadrhiri et al. [33] and Zhang et al. [72]. While the results reported in Yi et al. [73] remain better. Results shown in Fig. 12 illustrate that the *Run, DriveCar, GetOutCar*, and the *FightPerson* classes perform better compared to the rest. This demonstrates that our proposed method achieves higher accuracy values on video scenes that contain more motion vectors. It would be expected that the proposed method may have some drawbacks when motion vectors in the processed videos are less present.

Olympic Sports Dataset The Olympic Sports Dataset contains videos of athletes practicing different sports. The used release contains 16 sports, represented by a total of 783 video clips. We use 649 sequences for training and 134 sequences for testing as recommended. For comparison purposes, we report the MAP. Table 2 compares our approach with other techniques. The detailed accuracy by class is shown in Fig. 13.

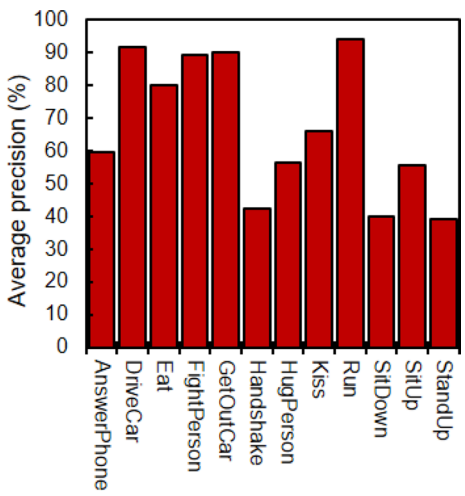

Fig. 12 The detailed accuracy by class for the Hollywood2 dataset

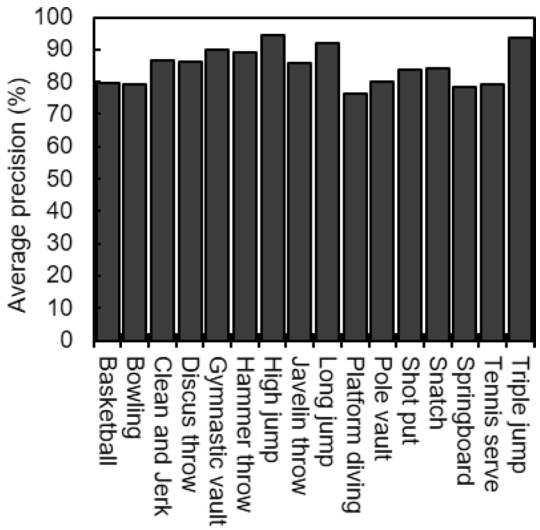

Fig. 13 The detailed accuracy by class for the Olympic Sports dataset

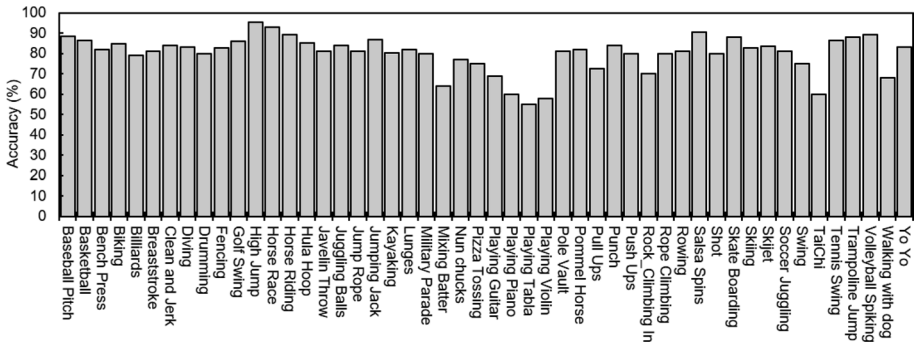

Fig. 14 The detailed accuracy by class for the UCF50 dataset

According to the table, Sumer et al. [74] reported 67.9% by using self-supervised learning. Sanakoyeu et al. [75] proposed a deep unsupervised learning of visual similarity method and they achieved 78%. Yi et al. track the motion key-point trajectories with an optical flow rectification algorithm and report 94.1%. Our approach outperforms Sumer et al. [74] and Sanakoyeu et al. [75] by over 17 and 6 percentage points, respectively, but she did not achieve higher accuracy than all methods.

UCF50 Action Recognition Dataset UCF50 is an action recognition dataset with 50 action categories, consisting of real-world videos taken from YouTube. It is one of the most used and challenging datasets in the action recognition literature due to large variations in camera motion, object appearance and pose, object scale, viewpoint, cluttered background, illumination conditions, etc.

Using the UCF50 dataset, the proposed method is compared with three similar approaches, including Yan et al. [76], Banerjee et al. [77] and Zuo et al. [78] . For all dataset classes, the videos are split into 25 groups. There is a minimum of four action clips in each group. The video clips in the same group may share some common features, such as the same person, similar background, similar viewpoint, etc.

In accordance with the recommendations [65], in this dataset, we use the *leave one group out cross-validation* as an experimental setup and we report the average accuracy over all classes. The achieved accuracy is presented in Table 2, the detailed accuracy by class is shown in Fig. 14.

According to Table 2, the proposed approach has achieved higher accuracy for this dataset when compared to Yan et al. [76] and Banerjee et al. [77] which achieves 41.54% and 78% accuracy, respectively, but it does not exceed the results presented in Zuo et al. [78]. Figure 14 shows that most of the classes achieved more than 80% accuracy results, which explains the efficiency of our approach in the real-world activity recognition process.

Performance evaluation of the real-time distributed processing

Despite the good precision, the proposed CBVR approach is relatively time-consuming, particularly when the amount of video data to process becomes large. Some applications with real-time requirements demand consistent video storage and analysis capabilities. For that reason, the architecture was implemented on a real-time distributed computing platform with Storm system. Experimental results are presented and explained above.

The experiments were initiated on a Storm single node cluster, then we gradually increase the size of the cluster to reach 10 nodes with an architecture that brings together Zookeeper, Nimbus, and Storm UI on the master machine. The other machines act as workers (supervisors). In this real-time distributed processing system the performances depend both on the setting of the scalability parameter (number of nodes), and the parallelism parameter (number of threads). Improving this performance by studying the correlation between the number of threads and nodes is presented in terms of factor average processing time (m). Consequently, the factor of average processing time (m), is the time required to complete all the steps of our proposed Storm topologies (see page 13). It is calculated by Eq. 13, and the global factor of processing time is calculated by Eq. 14:

$$\mathrm{m} = \frac{1 + \frac{\mathrm{T_s} + \varepsilon}{\mathrm{T_p}}}{\mathrm{N} \times \mathrm{T}} \tag{13}$$

where:

- $T_p = \sum T_{P_i}$: Processing time of step i;
- $T_S = \sum T_{S_i}$: Time of transaction and storage the data of step i;
- $\varepsilon =$ Startup's time;
- N: Number of nodes;
- T: Number of Threads;
- E: Set of nodes.

$$\mathrm{F} = \max(\mathrm{m_i})_{\mathrm{i} \in \mathrm{E}};\ \mathrm{m_i} = \left[\frac{1 + \frac{\mathrm{T_s} + \varepsilon}{\mathrm{T_p}}}{\mathrm{T}} \right]_{\mathrm{N_i}} \tag{14}$$

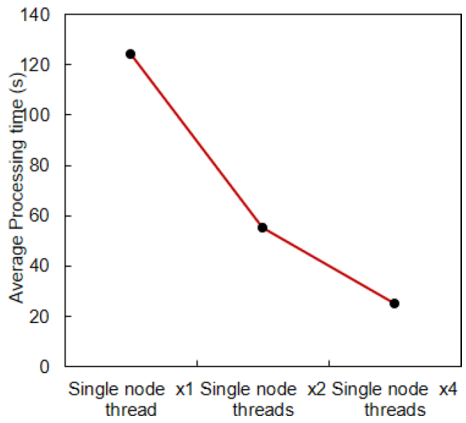

Fig. 15 The parallelism mechanism

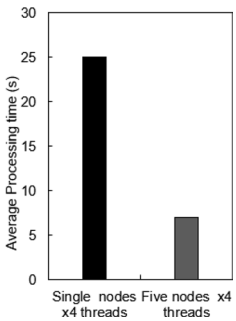

Fig. 16 The horizontal scaling

In the first experiment, we calculate the average factor of processing time to execute all processes of the two topologies on a single Storm machine (1 thread). Then we increase the number of executors (threads) on the same machine. To perform the experiment, 100 videos with various lengths from the Hollywood 2 dataset testing part are used. HDFS Spout is used to simulate real-time video streams by reading video files simultaneously from a determined HDFS folder [80]. Figure 15 presents the obtained results. From Fig. 15, we observe that with four threads, the average factor of processing time(m) is 80% better than that obtained with a single thread Storm machine. This demonstrates that increasing the number of threads allows us to take advantage of the power of parallel processing with multiple threads, where each instance can work independently and can contribute to the processing of data.

In order to get closer real-time video processing, the next experience process topologies on a Storm cluster of five machines with four parallel threads in each. The results are shown in Fig. 16. It is clear that horizontal scaling improves processing time by sharing the processing of tasks. Even though the progress is remarkable, the processing time does not satisfy the requirements of real-time applications.

In the last experiment, a cluster of 10 machines has been set up. Fig. 17 presents a comparison of the performances with the two previous experiments. From the results, it can be concluded that the processing time has been improved with 10 nodes cluster

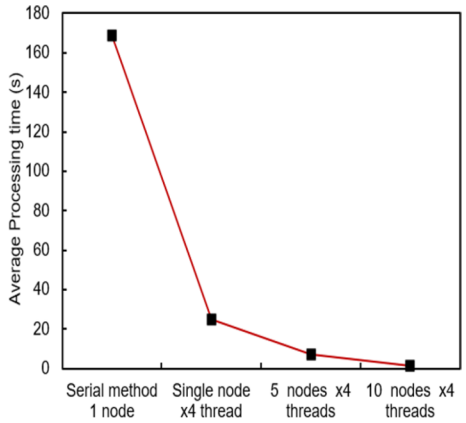

Fig. 17 Comparison of processing time

Table 3 Performance comparison of our results to the state of art

Approach	Average processing time	
	Hollywood2 Dataset	UCF Dataset
Udding et al. [81]	–	68 s
Xu et al. [82]	–	33.02 s
Wang et al. [83] Trajectory based feature method(2015)	12.29 s	–
Saoudi et al. [27]	2.7 s	–
Wang et al. [83] Fisher vectors method (2015)	1.77 s	–
Proposed approach (5 Nodes ×4 Threads)	7 s	12.8 s
Proposed approach (10 Nodes ×4 Threads)	1.6 s	5.3 s

and started to meets the requirements of real-time applications. To achieve near real-time video processing with the proposed architecture, a suitable number of supervisors is required. As illustrated in the Figure, using a large number of supervisors improved clearly the average processing time to reach the 1.6 s by sharing the processing of tasks.

To evaluate the performance of the proposed approach, we compared our results with those of three other distributed processing approaches including Udding et al. [81], Xu et al. [82], Wang et al. [83], and Saoudi et al. [27]. These approaches were performed on a distributed architecture using clusters of 4, 3, 9, and 10 machines, respectively. The results are reported in Table 3. From the table, it can be seen that our work outperforms other state-of-the-art works, which indicates that the increase in the number of executors by changing the default Storm parallelism factor from 1 to 4 was beneficial to improve the processing time. Contrary to other studies that mainly focus on horizontal scalability (adding more physical machines to the cluster), this work also took advantage of vertical scalability (increasing the number of executors (threads) in node). The results also confirm that a reduced number of nodes in a distributed processing cluster increases the execution time. Accordingly, a suitable number of nodes are required to achieve real-time processing. However, it should also be noted that the technique of using key frames to represent the video content instead of processing the entire video has contributed to the performance improvement.

Conclusion

A novel Content-Based Video Retrieval system for large datasets was presented in this paper. In the proposed approach, motion information (motion vectors and motion-compensated residual data) is extracted from compressed video streams to build a video signature. This method uses a key frame extraction algorithm to provide a compact video representation that contains salient and important video content information. This approach is advantageous in terms of computation time, compared to competing methods that process the entire video. Once the videos are characterized by a signature, they are compared using a similarity measurement model, which allows quick comparisons. The proposed CBVR approach was implemented using both a single machine and a real-time distributed cluster to evaluate the real-time performance aspect. Experimental results reveal that our method significantly improves retrieval accuracy over state-of-the-art methods. It proves also that the selected key frames are meaningful and capture the video content comprehensively. Results of the real-time distributed implementation prove that the proposed topologies have considerably improved the processing time to approximate real-time performance.

However, more works need to be done. In the future, other visual features and their fusion will be evaluated. Furthermore, we plan to add more machines to our cluster for further tests. Moreover, a group of parameters in Storm can affect the performance greatly, so we decide to do more research on this topic. Our future direction of research will involve the use of Deep Learning (DL) techniques such as Convolutional Neural Networks (CNN) to improve precision and processing time performance. Furthermore, a user interface will be designed to visualize the data query results.

Abbreviations

CBVR: Content-Based Video Retrieval; GOP: Group of pictures; ZNCC: Zero-mean Normalized Cross-Correlation; GGD: Generalized Gaussian Distribution method; PCA: Principal component analysis; BCS: Bounded Coordinate System; HDFS: Hadoop Distributed File System; MAP: Mean Average Precision.

Acknowledgements

Not applicable.

Authors' contributions

All authors read and approved the final manuscript.

Funding

Not applicable. This research received no specific grant from any funding agency.

Availability of data and materials

The datasets used and analyzed during the current study are available at the following links: HMDB51: http://serre-lab.clps.brown.edu/wp-content/uploads/2013/10/hmdb51_org.rar; HOLLYWOOD2: ftp://ftp.irisa.fr/local/vistas/actions/Hollywood2-actions.tar.gz; UCF50: https://www.crcv.ucf.edu/data/UCF50.rar; Olympic Sports: http://vision.stanford.edu/Datasets/OlympicSports/

Declarations

Ethics approval and consent to participate

The author confirms the sole responsibility for this manuscript. The author read and approved the final manuscript

Competing interests

The authors declare that they have no competing interests.

Author details

[1]Department of Mathematics and Informatics, Faculty of Sciences, Hassan II University, Casablanca, Morocco. [2]Faculty of Sciences, Hassan II University, Casablanca, Morocco.

Received: 18 February 2021 Accepted: 2 June 2021
Published online: 09 June 2021

References

1. Patel B. Content based video retrieval systems. International Journal of UbiComp. 2012;3.
2. Priya R, Shanmugam DTN, Baskaran DR. A content based video retrieval analysis system with extensive features by using Kullback-Leibler. Int J Comput Intell Syst. 2014;7(2):242–63.
3. Mistry YD. Textural and color descriptor fusion for efficient content-based image retrieval algorithm. Iran J Comput Sci. 2020;3(3):169–83.
4. Cha Y-J, Chen JG, Büyüköztürk O. Output-only computer vision based damage detection using phase-based optical flow and unscented Kalman filters. Eng Struct. 2017;132:300–13.
5. Minaei S, Kiani S, Ayyari M, Ghasemi-Varnamkhasti M. A portable computer-vision-based expert system for saffron color quality characterization. J Appl Res Med Aromat Plants. 2017;7:124–30.
6. Ghyar BS, Birajdar GK. Computer vision based approach to detect rice leaf diseases using texture and color descriptors. In: 2017 International Conference on Inventive Computing and Informatics (ICICI), 2017. p. 1074–1078. IEEE.
7. Jana S, Basak S, Parekh R. Automatic fruit recognition from natural images using color and texture features. In: 2017 Devices for Integrated Circuit (DevIC), 2017. p. 620–624. IEEE.
8. Zhou Y, Habermann M, Xu W, Habibie I, Theobalt C, Xu F. Monocular real-time hand shape and motion capture using multi-modal data. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020. p. 5346–5355.
9. Zhou W, Li H, Tian Q. Recent advance in content-based image retrieval: A literature survey. arXiv preprint arXiv:1706.06064 2017.
10. Liu X, He G-F, Peng S-J, Cheung Y-M, Tang YY. Efficient human motion retrieval via temporal adjacent bag of words and discriminative neighborhood preserving dictionary learning. IEEE Trans Hum Mach Syst. 2017;47(6):763–76.
11. Khan MH, Li F, Farid MS, Grzegorzek M. Gait recognition using motion trajectory analysis. In: International Conference on Computer Recognition Systems, Springer; 2017. p. 73–82.
12. Cho J, Lee M, Chang HJ, Oh S. Robust action recognition using local motion and group sparsity. Pattern Recognit. 2014;47(5):1813–25.
13. Sedmidubsky J, Elias P, Zezula P. Effective and efficient similarity searching in motion capture data. Multimedia Tools Appl. 2018;77(10):12073–94.
14. Ali H, Lali M, Nawaz MZ, Sharif M, Saleem B. Symptom based automated detection of citrus diseases using color histogram and textural descriptors. Comput Electr Agric. 2017;138:92–104.
15. Liu P, Guo J-M, Chamnongthai K, Prasetyo H. Fusion of color histogram and lbp-based features for texture image retrieval and classification. Inf Sci. 2017;390:95–111.
16. Maheen JB, Aneesh R. Machine learning algorithm for fire detection using color correlogram. In: 2019 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT), vol. 1, 2019. p. 1411–1418. IEEE.
17. Li Y, Liu M. Aerial image classification using color coherence vectors and rotation & uniform invariant lbp descriptors. In: 2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), 2018. p. 653–656. IEEE.
18. Alamdar F, Keyvanpour M.: A new color feature extraction method based on dynamic color distribution entropy of neighborhoods. arXiv preprint arXiv:1201.3337 2012.
19. Ejaz N, Baik SW, Majeed H, Chang H, Mehmood I. Multi-scale contrast and relative motion-based key frame extraction. EURASIP J Image Video Process. 2018;2018(1):1–11.
20. Lu G, Zhou Y, Li X, Yan P. Unsupervised, efficient and scalable key-frame selection for automatic summarization of surveillance videos. Multimedia Tools Appl. 2017;76(5):6309–31.
21. Dhagdi MST, Deshmukh P. Keyframe based video summarization using automatic threshold & edge matching rate. Int J Sci Res Publ. 2012;2(7):1–12.
22. Mounika BR, Prakash O, Khare A. Fusion of zero-normalized pixel correlation coefficient and higher-order color moments for keyframe extraction. In: Recent Trends in Communication, Computing, and Electronics, Springer, ???; 2019. p. 357–364.
23. Huang Z, Shen HT, Shao J, Zhou X, Cui B. Bounded coordinate system indexing for real-time video clip search. ACM Trans Inf Syst (TOIS). 2009;27(3):1–33.
24. Wactlar HD, Kanade T, Smith MA, Stevens SM. Intelligent access to digital video: Informedia project. Computer. 1996;29(5):46–52.
25. Chang S-F, Chen W, Meng HJ, Sundaram H, Zhong D. A fully automated content-based video search engine supporting spatiotemporal queries. IEEE Trans Circuit Syst Video Technol. 1998;8(5):602–15.
26. Saoudi EM, El Ouadrhiri AA, El Warrak O, Andaloussi SJ, Sekkaki A. Improving content based video retrieval performance by using hadoop-mapreduce model. In: 2018 23rd Conference of Open Innovations Association (FRUCT), 2018. p. 1–6. IEEE.
27. Saoudi EM, El Ouadrhiri AA, Andaloussi SJ, El Warrak O, Sekkaki A. Content based video retrieval by using distributed real-time system based on storm. Int J Embed Real-Time Commun Syst (IJERTCS). 2019;10(4):60–80.
28. Hashemi J, Dawson G, Carpenter KL, Campbell K, Qiu Q, Espinosa S, Marsan S, Baker JP, Egger HL, Sapiro G. Computer vision analysis for quantification of autism risk behaviors. IEEE Transactions on Affective Computing. 2018.
29. Agahian S, Negin F, Köse C. An efficient human action recognition framework with pose-based spatiotemporal features. Eng Sci Technol Int J. 2020;23(1):196–203.
30. Fan L, Huang W, Gan C, Ermon S, Gong B, Huang J. End-to-end learning of motion representation for video understanding. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018. p. 6016–6025.

31. Barmpoutis P, Dimitropoulos K, Kaza K, Grammalidis N. Fire detection from images using faster r-cnn and multidimensional texture analysis. In: ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2019. p. 8301–8305. IEEE.
32. VenkateswarLal P, Nitta GR, Prasad A. Ensemble of texture and shape descriptors using support vector machine classification for face recognition. Journal of Ambient Intelligence and Humanized Computing. 2019;1–8.
33. El Ouadrhiri AA, Saoudi EM, Andaloussi SJ, Ouchetto O, Sekkaki A. Content based video retrieval based on bounded coordinate of motion histogram. In: 2017 4th International Conference on Control, Decision and Information Technologies (CoDIT), 2017. p. 0573–0578. IEEE.
34. Zhou S, Wu X, Qi Y, Luo S, Xie X. Video shot boundary detection based on multi-level features collaboration. Signal Image Video Process. 2021;15(3):627–35.
35. Sasithradevi A, Roomi SMM. A new pyramidal opponent color-shape model based video shot boundary detection. J Visual Commun Image Represent. 2020;67:102754.
36. Spolaôr N, Lee HD, Takaki WSR, Ensina LA, Coy CSR, Wu FC. A systematic review on content-based video retrieval. Eng Appl Artif Intell. 2020;90:103557.
37. Aote SS, Potnurwar A. An automatic video annotation framework based on two level keyframe extraction mechanism. Multimedia Tools Appl. 2019;78(11):14465–84.
38. Wu J, Zhong S-H, Jiang J, Yang Y. A novel clustering method for static video summarization. Multimedia Tools and Applications. 2017;76(7):9625–41.
39. Das S, Banerjee M, Chaudhuri A. An improved video key-frame extraction algorithm leads to video watermarking. Int J Inf Technol. 2018;10(1):21–34.
40. Luo Y, Zhou H, Tan Q, Chen X, Yun M. Key frame extraction of surveillance video based on moving object detection and image similarity. Pattern Recognit Image Anal. 2018;28(2):225–31.
41. Kumar K, Shrimankar DD, Singh N. Eratosthenes sieve based key-frame extraction technique for event summarization in videos. Multimedia Tools Appl. 2018;77(6):7383–404.
42. Zhang Y, Tao R, Wang Y. Motion-state-adaptive video summarization via spatiotemporal analysis. IEEE Transactions on Circuits and Systems for Video Technology. 2016;27(6):1340–52.
43. Loukas C, Varytimidis C, Rapantzikos K, Kanakis MA. Keyframe extraction from laparoscopic videos based on visual saliency detection. Comput Methods Programs Biomed. 2018;165:13–23.
44. Chen M, Han X, Zhang H, Lin G, Kamruzzaman M. Quality-guided key frames selection from video stream based on object detection. J Visual Commun Image Represent. 2019;65:102678.
45. Raikwar SC, Bhatnagar C, Jalal AS. A framework for key frame extraction from surveillance video. In: 2014 International Conference on Computer and Communication Technology (ICCCT), 2014. p. 297–300. IEEE.
46. Asim M, Almaadeed N, Al-Máadeed S, Bouridane A, Beghdadi A. A key frame based video summarization using color features. In: 2018 Colour and Visual Computing Symposium (CVCS), 2018. p. 1–6. IEEE.
47. Lee J, Dickinson BW. Rate-distortion optimized frame type selection for mpeg encoding. IEEE Trans Circuit Syst Video Technol. 1997;7(3):501–10.
48. Babu RV, Tom M, Wadekar P. A survey on compressed domain video analysis techniques. Multimedia Tools Appl. 2016;75(2):1043–78.
49. Di Stefano L, Mattoccia S, Tombari F. Zncc-based template matching using bounded partial correlation. Pattern Recogni Lett. 2005;26(14):2129–34.
50. Chambon S, Crouzil A. Similarity measures for image matching despite occlusions in stereo vision. Pattern Recognit. 2011;44(9):2063–75.
51. Li MJ, Ng MK, Cheung Y-M, Huang JZ. Agglomerative fuzzy k-means clustering algorithm with selection of number of clusters. IEEE Trans Knowl Data Eng. 2008;20(11):1519–34.
52. Karypis MSG, Kumar V, Steinbach M. A comparison of document clustering techniques. In: TextMining Workshop at KDD2000 (May 2000) 2000.
53. Hartigan JA, Wong MA. Algorithm as 136: a k-means clustering algorithm. J Royal Stat Soc Series C. 1979;28(1):100–8.
54. Khan SS, Ahmad A. Cluster center initialization algorithm for k-means clustering. Pattern Recognit Lett. 2004;25(11):1293–302.
55. Kodinariya TM, Makwana PR. Review on determining number of cluster in k-means clustering. Int J. 2013;1(6):90–5.
56. Varanasi MK, Aazhang B. Parametric generalized gaussian density estimation. J Acoust Soc Am. 1989;86(4):1404–15.
57. Lever J, Krzywinski M, Altman N. Points of significance: Principal component analysis. Nature Publishing Group; 2017.
58. Arowolo MO, Adebiyi MO, Adebiyi AA. An efficient pca ensemble learning approach for prediction of rna-seq malaria vector gene expression data classification. Int J Eng Res Technol. 2020;13(1):163–9.
59. Arowolo MO, Adebiyi MO, Aremu C, Adebiyi AA. A survey of dimension reduction and classification methods for rna-seq data on malaria vector. J Big Data. 2021;8(1):1–17.
60. Foundation TAS. Apache ZooKeeper. https://zookeeper.apache.org/. 2020; Online Accessed 31 July 2020.
61. Foundation TAS. Apache Storm. https://storm.apache.org/. 2019; Online Accessed 31 July 2020.
62. Marszałek M, Laptev I, Schmid C. HOLLYWOOD2 Human Actions and Scenes Dataset. https://www.di.ens.fr/~laptev/actions/. 2008; Online Accessed 31 July 2020.
63. Marszałek M, Laptev I, Schmid C. Actions in context. In: IEEE Conference on Computer Vision & Pattern Recognition 2009.
64. Kuehne H, Jhuang H, Garrote E, Poggio T, Serre T. HMDB: a large human motion database. https://serre-lab.clps.brown.edu/resource/hmdb-a-large-human-motion-database/. 2011; Online Accessed 31 July 2020
65. Reddy KK, Shah M. Recognizing 50 human action categories of web videos. Mach Vision Appl. 2013;24(5):971–81.
66. Reddy KK, Shah M. UCF50 - Action Recognition Data Set. https://www.crcv.ucf.edu/data/UCF50.php. 2012; Online Accessed 31 July 2020.
67. Juan Carlos Niebles C-WC, Fei-Fei L. Olympic Sports Dataset. http://vision.stanford.edu/Datasets/OlympicSports/. 2010; Online Accessed 31 July 2020.
68. Kuehne H, Jhuang H, Garrote E, Poggio T, Serre T. Hmdb: a large video database for human motion recognition. In: 2011 International Conference on Computer Vision, 2011. p. 2556–2563. IEEE.

69. Phan H-H, Vu N-S, Nguyen V-L, Quoy M. Action recognition based on motion of oriented magnitude patterns and feature selection. IET Comput Vision. 2018;12(5):735–43.
70. Zheng J, Cao X, Zhang B, Zhen X, Su X. Deep ensemble machine for video classification. IEEE Trans Neural Netw Learn Syst. 2018;30(2):553–65.
71. Wang L, Ge L, Li R, Fang Y. Three-stream cnns for action recognition. Pattern Recogn Lett. 2017;92:33–40.
72. Zhang K, Zhang L. Extracting hierarchical spatial and temporal features for human action recognition. Multimedia Tools Appl. 2018;77(13):16053–68.
73. Yi Y, Cheng Y, Xu C. Mining human movement evolution for complex action recognition. Expert Syst Appl. 2017;78:259–72.
74. Sumer O, Dencker T, Ommer B. Self-supervised learning of pose embeddings from spatiotemporal relations in videos. In: Proceedings of the IEEE International Conference on Computer Vision, 2017. p. 4298–4307.
75. Sanakoyeu A, Bautista MA, Ommer B. Deep unsupervised learning of visual similarities. Pattern Recognit. 2018;78:331–43.
76. Yan X, Hu S, Ye Y. Multi-task clustering of human actions by sharing information. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017. p. 6401–6409.
77. Banerjee B, Murino V. Efficient pooling of image based cnn features for action recognition in videos. In: 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2017. p. 2637–2641. IEEE.
78. Zuo Z, Yang L, Liu Y, Chao F, Song R, Qu Y. Histogram of fuzzy local spatio-temporal descriptors for video action recognition. IEEE Trans Ind Inf. 2019;16(6):4059–67.
79. Yi Y, Wang H. Motion keypoint trajectory and covariance descriptor for human action recognition. Visual Comput. 2018;34(3):391–403.
80. Foundation TAS. HDFS Spout. https://storm.apache.org/releases/2.1.0/storm-hdfs.html. 2020; Online Accessed 12 Feb 2020.
81. Uddin MA, Joolee JB, Alam A, Lee Y-K. Human action recognition using adaptive local motion descriptor in spark. IEEE Access. 2017;5:21157–67.
82. Xu W, Uddin MA, Dolgorsuren B, Akhond MR, Khan KU, Hossain MI, Lee Y-K. Similarity estimation for large-scale human action video data on spark. Appl Sci. 2018;8(5):778.
83. Wang H, Zheng X, Xiao B. Large-scale human action recognition with spark. In: 2015 IEEE 17th International Workshop on Multimedia Signal Processing (MMSP), 2015. p. 1–6. IEEE.

Publisher's Note