

INDEXED DATASET FROM YOUTUBE FOR A CONTENT-BASED VIDEO SEARCH ENGINE

Ahmad Sedky Adly*

Computer Science Dept.,
Faculty of Information
Technology, Misr University
for Science & Technology,
Giza, Egypt
sedky@must.edu

Islam Hegazy

Computer Science Dept.,
Faculty of Computer &
Information Sciences, Ain
Shams University,
Cairo, Egypt
islheg@cis.asu.edu.eg

Taha Elarif

Computer Science Dept.,
Faculty of Computer &
Information Sciences, Ain
Shams University,
Cairo, Egypt
taha_elarif@cis.asu.edu.eg

M. S. Abdelwahab

Computer Science Dept.,
Faculty of Information
Technology, Misr University
for Science & Technology,
Giza, Egypt
mswahab@must.edu.eg

Received 2021- 3-21; Revised 2021-6-3; Accepted 2021-6-16

Abstract: Numerous researches on content-based video indexing and retrieval besides video search engines are tied to a large-scaled video dataset. Unfortunately, reduction in open-sourced datasets resulted in complications for novel approaches exploration. Although, video datasets that index video files located on public video streaming services have other purposes, such as annotation, learning, classification, and other computer vision areas, with little interest in indexing public video links for purpose of searching and retrieval. This paper introduces a novel large-scaled dataset based on YouTube video links to evaluate the proposed content-based video search engine, gathered 1088 videos, that represent more than 65 hours of video, 11,000 video shots, and 66,000 unmarked and marked keyframes, 80 different object names used for marking. Moreover, a state-of-the-art features vector, and combinational-based matching, beneficial to the accuracy, speed, and precision of the video retrieval process. Any video record in the dataset is represented by three features: temporal combination vector, object combination vector with shot annotations, and 6 keyframes, sideways with other metadata. Video classification for the dataset was also imposed to expand the efficiency of retrieval of video-based queries. A two-phased approach has been used based on object and event classification, storing video records in aggregations related to feature vectors extracted. While object aggregation stores video records with the maximal occurrence of extracted object/concept from all shots, event aggregation classify based on groups according to the number of shots per video. This study indexed 58 out of 80 different object/concept categories, each has 9 shot number groups.

* Corresponding author: Ahmad Sedky Adly

Computer Science Dept.,

Faculty of Information Technology, Misr University for Science & Technology, Giza, Egypt

E-mail address: sedky@must.edu

Keywords: *Content-based video search engine, content-based video indexing and retrieval, CBVSE, CBVIR, feature extraction, search engine, video search, video retrieval, video indexing, YouTube indexing.*

1. Introduction

Recent research work in content-based video search engines in the application of indexing and classification has been massively increasing in the past few years due to the growth and demand of multimedia transmission and use over the world wide web. As video streaming and sharing became massively appealing on commercial use of entertainment, advertisement, and news over the Internet, which became a dominant field. Accordingly, video search engines and video content analysis of video files have got attention in huge numbers of studies and researches in the past two decades. Artificial intelligence techniques with data science as deep-learning, machine-learning, data mining, and big data analysis attracted researchers to apply in their studies in this field. However, the hardest challenges currently facing all research is indexing and retrieving video information in the shortest time possible. This is due to the substantial search demands for videos and video files over the Internet. Unfortunately, almost all video search engines over the web are either text-based or image-based queries, as it's a less complicated and resource-consuming way to search for video files over the Internet, in which users describe a title, surrounding text, tags, narration text, or sometimes comments around the video rather than the actual video content.

Content-based video indexing is the process of organizing tagging and extracting feature vectors from the actual content of the video sequence file in a way that can be retrieved fast and accurately. Automating the process of indexing is essential for the reduction of computation and elimination of tedious procedures. Most of the content-based video indexing systems in numerous studies use either low- or high-level features or both in their feature vectors. As an example, low-level features include colors, histograms, textures, etc., extracted from key-frames, high-level features include semantics, key-objects/concepts, scale-invariant feature transform (SIFT), etc. However, the larger the indexed features vector gets, the better becomes the accuracy of accessing and retrieving the video record. On the other hand, a tradeoff between complexity and accuracy will appear as the features vector increases with more features. This requires an optimum representation of features extracted from the video sequence in the index repository in order to reduce computation and complexity for indexing and retrieving procedures. This is considered one of the major challenges facing video indexing systems, to build an optimum representation for the video record in a content-based video index database.

This paper is organized into five sections, second section will discuss the related work in content-based video indexing and retrieval search engines. Section three will show the method and techniques used in this research. The fourth section is all about experiments and results, showing the new dataset and the process of acquiring videos, extracting features, and indexing records. Finally, the fifth section with conclusions, future experiments, and research work.

2. Related Work

Content-based video indexing and tagging in major video streaming web services are still manually grasped based on surrounding text tied to a video file which is provided mainly by the video uploader. However, this exposes video contents for piracy and misuse by providing the same video file with different information using a different language or phrases. Accordingly, content-based video indexing

offers a transparent solution for video archives and databases. However, time consumption for both indexing and retrieval is an important matter depending on feature vectors and the procedures of features extraction complexity, along with video classification and matching algorithms, that need to be optimal as possible. In [1], the authors introduced an exhaustive review regarding content-based video search engines in terms of indexing and retrieval with a survey of state-of-the-artwork and their applications, accompanied by a novel approach and feature vector proposal.

Furthermore, there has been numerous and available non-commercial public datasets serving indexing and retrieval for content-based video search engines, equipped with pre-extracted features and semantic objects, such as in [2], Ortiz, et al, introduced a dataset prepared with 113 videos for movie trailers covering 145 faces recognized acting members in all videos of the dataset applying representation-based classifiers using sparse of mean sequence. However, face recognition is very computationally expensive and requires excessive training on a preset database of acting members, which is very difficult to prepare, another problem is that it is suitable only for movie video genres. Moreover, other segment-leveling annotation datasets like human motion database HMDB-51 [3] and human actions classes UCF-101 [4] providing annotation to multiple categories signifying different human activities. Yet, it also computationally exhaustive and relevant only to videos of genres involving humans. Although, THUMOS [5] provides much more rich annotations with localization and temporal information.

Furthermore, a set of yearly competitions called TRECVID [6] specialized mainly on content-based video indexing and retrieving, introducing numerous and different video datasets, for example, they hosted a 1000 videos local test dataset annotated and bounded with 10 classes of bounding boxes. Still, some of the videos may not encompass a bounding box.

Some visual object tracking datasets found include the visual object tracking VOT and the multiple target tracking MOT [7,8], yearly challenges which are small and accurately chosen to provide various solutions for common difficult object tracking problems such as size and illumination variation or occlusions. Other temporal localized and segment level annotated datasets such Sports-1M, consist of one million videos of sports varieties [9,10]. Additional datasets were found captured from vehicles driving in a city, containing around 350 thousand bounding boxes of pedestrian annotations, called Caltech Pedestrian Detection [11]. ImageNet dataset [12], with records of more than five thousand four hundred objects detected in videos.

Many YouTube-based datasets were found such as YouTube-8M [13], a dataset with 8 million YouTube acquired video files, containing a very large frames level automatically annotated YouTube videos with a newly introduced deep network used for generating class labels to thousands of frames and entities; YouTube-Objects [14],[15], a dataset with hundreds of frames extracted from YouTube and annotated using few hundred bounding boxes. However, all datasets seem to lack the combinational significance for objects in relation to the video records acquired, focusing only on the problem of video annotation.

3. Methodology & Technique

Video streams and video files are built of subsets of scenes, furtherly divided into shots, and finally, into static frames of fixed images, a video shot is a group of frames recorded without interference for a nonstop single-camera activity [1].

Let us consider I as a set for video records stored in an index of a content-based video search engine which was processed and feature vector extracted from a set of web streaming video files E , with each video is represented by URL and video id number. The goal is to gather a video index that stores video records representing each video in E stored and classified in a data structure that contains video ID,

URL, keyframes, and feature vector, which are processed and extracted from each crawled video stream and classified and stored accordingly in the index or video dataset.

As the video index, I become larger in scale and number of video records, it grows into a more expensive computation search procedure over matching query videos against all video records in the index. To reduce the search cost, two levels of hierarchy of video classification were used according to residual genres and objects related to the extracted video features vectors, this helps to allocate video records into a structural easy to access manner. The first level of classification for video records is object/concept aggregation, where video records are classified according to the extracted vector of objects/concepts from each shot in the video sequence based on the highest extracted object category appearing on the objects/concepts vector, calculating maximum occurring object function $f(c_i)$ on the set of objects/concepts vector $C_v = \{c_i, \dots, c_n\}$ extracted from a crawled video before indexing.

$$\mathbf{arg\ max}_{c_i \in C_v} f(c_i) := |\{c_i \in C_v \mid \forall x \in C_v : f(x) \leq f(c_i)\}| \quad (1)$$

Where x is an ordered set and points c_i for which $f(c_i)$ achieves its largest value.

The second level of classification is statistical shots aggregation, where furtherly classifying video records inside each object category according to the number of video shots in each video sequence into category groups of 100s, (i.e. grouping video records from 1 to 100 shots in one group, then 101 to 200, ...etc.) Figure 1 shows the designed index data structure.

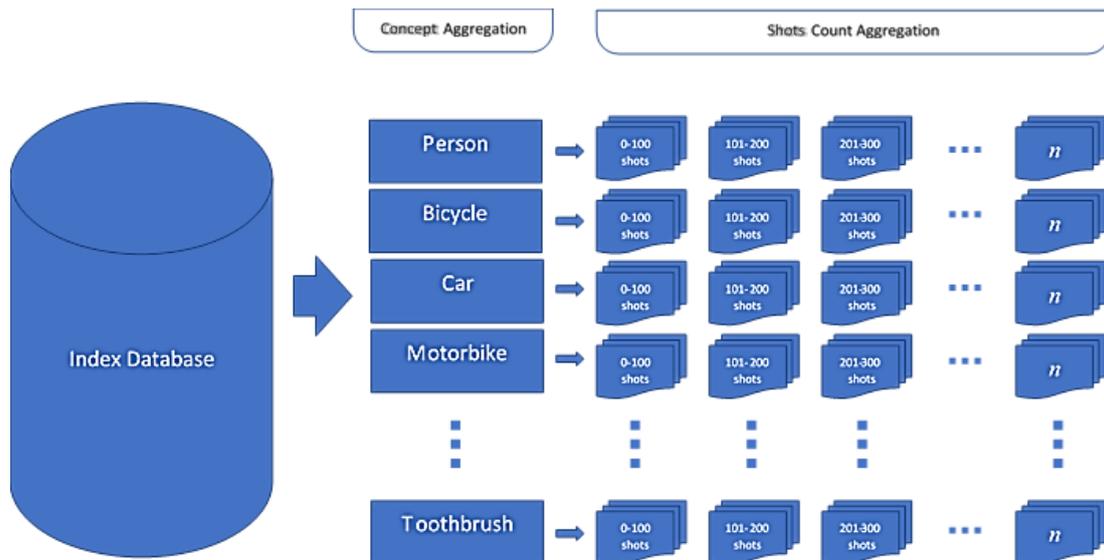


Figure 1: Index Data Structure

Finally, the process of searching the index, or matching two video records comparing feature vectors, will go through two levels of search, concepts/objects and the number of shots search. The objects/concepts search level, will search for and extract the class of interest L , then the number of shots search level will retrieve S from L . Lastly, a video record of the match would be found in S .

The following sections will present further video classification methods for indexing the content-based video dataset.

3.1. Content-Based Video Indexing System

Video streaming web services such as YouTube, which is recognized as one of the largest resources of Internet videos known to mankind. Topics covered in YouTube are with unknown depth, multi-lingual, incomplete information, no classification nor categorization for most videos, and other challenges regarding video copyright misuse.

Figure 2 illustrates the steps of crawling and indexing video files using root URLs. Each video file indexing goes through three stages before storing as a record in the video index, video file preprocessing, feature extraction, and video matching classification. Moreover, after the URL of a video file is being fetched, it goes through a duplication check to see whether it's stored in the index. If the test returns negative, the video file will be downloaded for further video preprocessing. Content-aware scene detection is then applied to extract the shot boundaries tied to the video stream. Afterward, the stage of feature extraction commences to extract both statistical and temporal features, and keyframes from each shot to use later for the key-objects or concepts feature extraction, to shape the feature vector to pass for the next and last stage. The third stage performs combinational matching criteria for the video's feature vector to determine similar video records amongst the entire video records index. This process is essential for video aggregation and classification to store video records in an orderly fashion for fast future retrieval. The following text will discuss each stage in further detail.

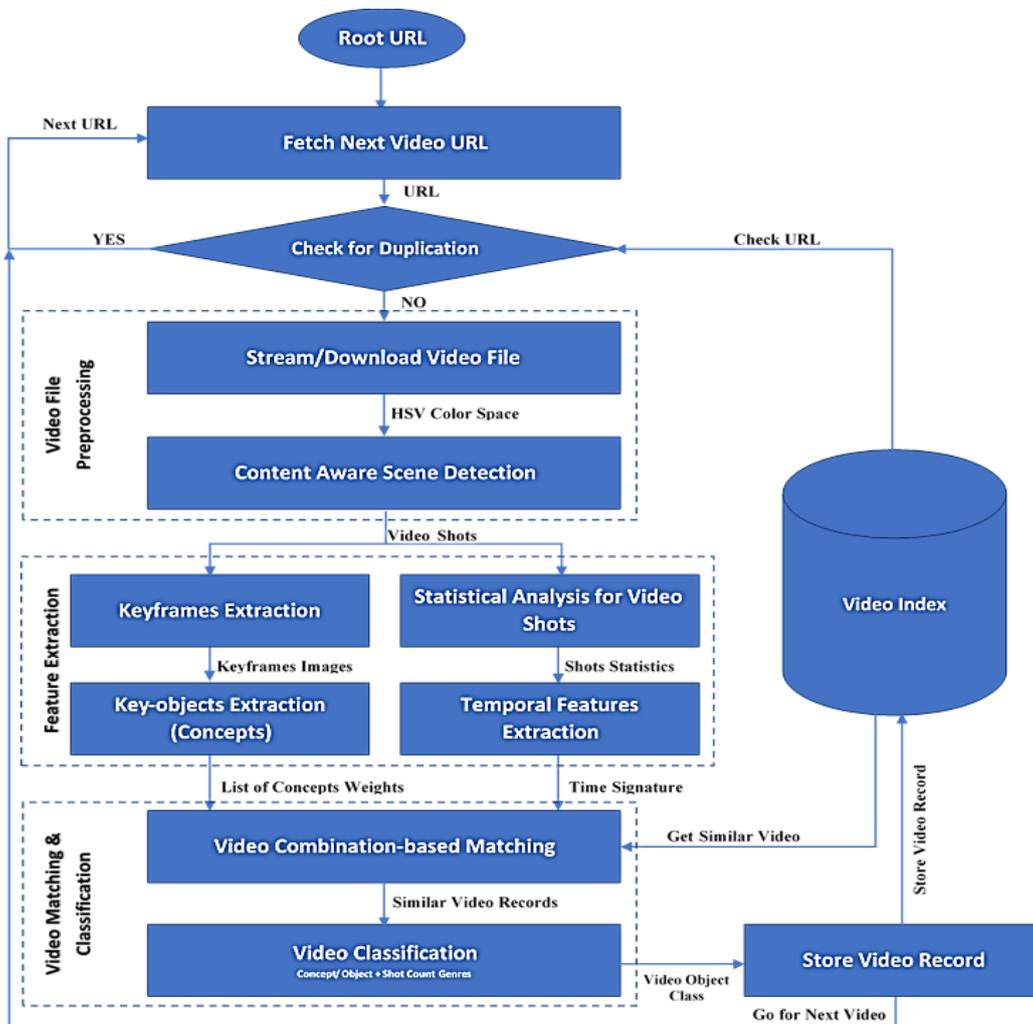


Figure 2: Content-Based Video Indexing System (Crawler).

3.2. Video Preprocessing

Extracting the contents and features automatically from video files requires numerous technologies, which is a result of complex and rich contents encoded within these videos. In addition, video contents have a substantial structure of visual, auidial, and lingual content like images, tunes/music, narration, and more.

3.2.1 Content-Aware Video Shot Detection

A shot boundary can be found between two adjacent shots transition, and according to Zhang et al [16], can be categorized into four major types hard cuts, fades, wipes, and dissolves. The first resembles the strict and complete transformation of the shot, the rest signifies gradual change with more than one frame located in the shot boundary. Figure 3 shows the steps of content-aware shot boundary detection.

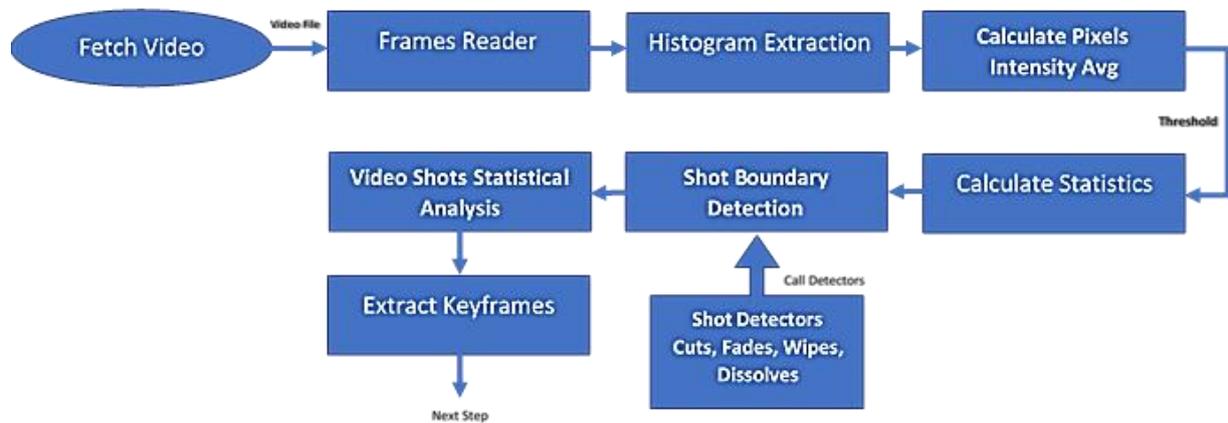


Figure 3: Content-Aware Shot Boundary Detection

After the video file is being fetched, checked for duplications, and downloaded, the next step is to scan and segment the video sequence into scenes and shots to fatherly extract features. A histogram-based detection algorithm has been used via the mean of intensity difference between pixels to detect scene change amongst two sequential frames. However, the scope of this study does not target shot boundary detection nor scene segmentation as a primary scope, which led to studying and evaluating tools that employ scene and shot boundary detection through histogram-based algorithms, some advanced video processing tools have been found as PySceneDetect [17], which is an opensource library, using python as the main programing language, and supports threshold variety, scene segmentation, and shots extraction. Furthermore, FFprobe and FFmpeg [18], both use C language and share the same functionality features with PySceneDetect. In addition, Almousa, et al [19], examined and evaluated several histogram-based shot boundary detection tools including PySceneDetect, FFprobe, and FFmpeg according to performance and speed by applying several threshold values, and histogram variance-based computation for scene scores was used, which computes the pixels' average intensity difference between sequential frames in the video sequence. They found that FFmpeg, PySceneDetect, and FFprobe are ordered respectively according to execution speed based on the various values used for threshold. Accuracy on the other hand showed that FFmpeg and FFprobe performed better than

PySceneDetect comparing results to a gold standard that was based on human judgment. Therefore, they concluded that FFmpeg performed superiorly in both speed and accuracy within the video dataset used applying thresholds of 10 and 20 percent.

In this study, all mentioned tools have been applied and evaluated in relation to speed, accuracy, and extracted features, using a preset value threshold, all against this study's video dataset crosswise a great portion of YouTube downloaded video files. It has been found that the FFmpeg performed best in terms of speed, accuracy, and feature extraction of video shots' temporal vector and keyframes, using a fixed threshold of 5.7% for the procedure. Section 4.2 consists of further details of conducted experiments and the results achieved.

3.3. Features Extraction

After video segmentation and shots detection complete, the process of extracting features from video shots comes next. Most of the work in the field of content-based video indexing and retrieval systems categorizes features into two main types, low- and high-level features. Low-level features represent visual and statistical features directly extracted from keyframes and video shoots, which is less costly in indexing, and yet retrieval is more expensive and complex. On the other hand, high-level features are semantic-based features requiring previous training and knowledge to extract features for the indexing process, still, benefits retrieval as it requires less complex and less costly matching measures. This study introduces a new feature vector including both categories representing the video.

3.3.1. Keyframes Extraction

To extract visual or semantic feature vectors from a given video, a keyframes extraction process is needed. Moreover, keyframes is an abstract representation for video data in a form of a set of fixed images that one or more represent a video shot. Furthermore, video shots contain numerous and redundant frames which are vividly reduced to one or two keyframes that signify the video shot briefly. The keyframes extraction process is crucial for searching video files faster and downsizing the cost of processing frame by frame. Keyframes are also efficient in retrieval and accessing indexed video records quickly using matching against summarized versions of the video files using keyframes as a very condensed visual content representation for a long video sequence file. However, keyframes must be chosen with a maximal representation of the video shot's visual content with avoiding repetition. Recent work shows that keyframes extraction techniques are divided into six main categories. The first is sequential techniques which take frames consequently ordered after a chosen keyframe, comparing them until capturing a various frame, which will be chosen as the next keyframe. The second technique is global comparisons of keyframes which are dependent on frame variations in a video shot relying on minimizing a predefined objective function to dissect and distribute keyframes, as Evan temporal variance, maximum coverage, minimum correlation, and minimum reconstruction error. The third technique is called reference frame, which creates a reference frame firstly and then comparing it against the rest of the video shot's frames in order to extract keyframes. Forth technique is clustering, which uses an algorithm to first cluster shots and then picks keyframes located near the cluster center. The fifth technique called curve simplification which uses a point in the feature space that represents a frame in a shot, and all of the points are consecutively connected to make a direction curve that will be used to locate a set of points representing the outline of the curve. The last and sixth technique is the

objects or events approach, which guarantees the extraction of keyframes that contains objects or events data from the video shot [1].

However, this study as mentioned before uses a histogram-based approach for shot segmentation and boundary detection as in many related studies and research work [20,21], all of which handles distribution information of pixels values not taking into account their location in the frame image. If the gray-levels values in which $\{1, 2, 3, \dots, g\}$, a frame $f \in F = \{f_1, \dots, N\}$ in the video shot is said to have a histogram H_f , with g gray-levels dimensional vector, can have the percentage of pixels computed with function $H_f(i)$ where i is the gray-level. As for colored frame images signifying the possible color number with g^3 with the histogram represented by the number of the same components. Thus, a histogram-based discontinuity function SB can be put at each point in the simplest form of:

$$SB(f_k, f_{k+c}) = \sum_{i=0}^g \frac{|H_{f_k}(i) - H_{f_{k+c}}(i)|^2}{H_{f_{k+c}}(i)} \quad (2)$$

Where c is an offset parameter must be priorly set, and it needs to be a tradeoff and balanced between either comparing two sequential frames and detecting rapid changes as in cut transitions using $c = 1$ or simply is set to $c > 1$ for smother transitions detection on which changes is proven between frames with the certain distance apart. However, the function also enhances the single shot's frames difference which helps in keyframes extraction.

The next step is to extract keyframes by first converting video shots into a sequence of fixed frame images. Subsequently, extracting histograms for each frame image in the sequence. A principal component analysis PCA function is then applied to find eigenvectors of the frames' histograms with maximum eigenvalues to project the data in a subspace of fewer or equal dimensions. Lastly, a K-means clustering algorithm is applied to the projected histogram data to extract K centroids that will be used to select keyframes closer to the largest clusters found. Moreover, the user may alter the number of keyframes extracted per shot, or the system will set it to less than or equal to 3 per video shot based on size and number of shots per video sequence. Table 1 shows a sample of keyframes set extracted from a trailer video with 88 shots and 264 keyframes extracted.

Table 1: Example of Keyframes Extraction from "The Extra Man" (trailer on YouTube)

No	Shot Length	Keyframes		
13	0.918			
14	1.001			
15	1.043			
23	1.418			
24	0.667			
25	0.751			
26	1.210			
33	0.792			
34	0.709			
62	1.877			

3.3.2. Video Shots Temporal Relations Analysis and Extraction

Temporal relations are one of the ontology techniques used to solve content-based video indexing and retrieval problems, by detecting temporal relations between shots extracted from a video sequence, depending on the fact that shots individually have temporal relations linking them with other adjacent shots in the video sequence. Sequence alignment is known as one of the greatest bioinformatics fields, that properly exemplify the relations between DNA or RNA proteins and their sequence arrangements to recognize similarities between regions of the protein chains [22,23]. However, some computer vision research adopted sequence alignment problems using multimode data with high dimensions applying alignment between videos with untrimmed sequences and video sequences text abiding actions [24].

Another approach used for temporal relations analysis is term-based similarity measurement/function which is a quantification function for measuring similarity between two objects/vectors and reflects the intimacy degree of the targeted object/vector corresponding to features distinguishing clusters indexed in the dataset. Before the clustering process exploits, similarity measurement/function must be chosen and applied [25]. Selecting a proper similarity measurement/function is vital for relational analysis, which in this case a temporal relational analysis and matching.

3.3.3. Key-Objects Extraction (Concepts)

This study chose You Only Look Once YOLOv3 [26] as the main object detector, which was trained on the dataset library of Microsoft COCO: Common Objects in Context [27] with 80 different object types plus a “null object” type which annotates key-frames from video shots that don’t contain any extracted objects, all based on a TensorFlow framework [28]. This setup was introduced by ImageAI [29], a state-of-the-art computer vision open source library built on Python offering pre-trained models mined from

ImageNet-1000, recognizing 1000 dissimilar objects, implemented on ResNet, Inception v3, DenseNet, and SqueezeNet. It offers object detection trained on the COCO dataset, delivering detection, location, and identification of 80 different objects based on RetinaNet, YOLOv3, and TinyYOLOv3 models, the following text demonstrates the issues and performance using this setup in creating the dataset, and the following sections will show the mechanism and results.

3.4. Comparative Similarity Measurements

The similarity amongst vectors is based upon a contiguous chain of terms (in this case shots time intervals and objects/concepts annotations) that can be found in both vectors. Several similarity measurement algorithms are used to calculate similarities between vectors. Addressing some of them in the following text.

Jaccard similarity coefficient algorithm: measures similarities between two vectors by calculating the number of common or shared terms divided by unique terms number in both vectors [30]. Jaccard similarity JS is calculated using the following formula:

$$JS(T_a, T_b) = \frac{|T_a \cap T_b|}{|T_a \cup T_b|} = \frac{|T_a \cap T_b|}{|T_a| + |T_b| - |T_a \cap T_b|}, 0 \leq JS(T_a, T_b) \leq 1 \quad (3)$$

Where T_a and T_b are the shots' time intervals vectors in comparison.

Cosine similarity algorithm: measures similarities between two vectors by calculating the angle in which the two vectors imply in the dot matrix space measuring θ angles if both vectors are similar to each other, and less similar inversely proportional to the angle degree, the larger the angle the lesser the similarity score. Cosine similarity is vector length independent, where the two compared vectors are not necessarily to be equal in length making it common in high dimensional space vectors [31]. The following formula represents cosine similarity CS between two vectors of shot time intervals T_a and T_b with N and M terms respectively:

$$CS = \frac{\sum_{i=0}^N (T_a)_i \cdot \sum_{k=0}^M (T_b)_k}{\sqrt{\sum_{i=0}^N (T_a)_i^2} \cdot \sqrt{\sum_{k=0}^M (T_b)_k^2}} \quad (4)$$

Euclidean distance similarity algorithm: a standard measurement metric for geometrical dilemmas using the distance between two points in the compared vectors and easy to measure and extracted from 2D or 3D space. It is also used widely in clustering problems as comparing vectors of shot time intervals, and k -means algorithm for distance measuring [32]. It measures the distance between two vectors T_a and T_b with N terms using the following formula:

$$Ed(T_a, T_b) = \sqrt{\sum_{i=0}^N |(T_a)_i - (T_b)_i|^2} \quad (5)$$

The distance is inversely proportional to the similarity, which means the two vectors are less similar when the distance increases. However, Euclidean distance must obtain vectors with the same length for all terms, which makes it difficult for vectors to have a difference in one or more (less than 10) shots.

Minkowski distance similarity algorithm: is a normed vector space metric that generalizes both Euclidean and Manhattan distances algorithms. The following formula shows the similarity measurement Md using Minkowski distance between two shot time intervals vectors T_a and T_b with N terms:

$$Md(T_a, T_b) = \sqrt[p]{\sum_{i=0}^N |(T_a)_i - (T_b)_i|^p}, \quad p > 0 \quad (6)$$

Where p is the Minkowski distance order and can be adjusted to any value other than 1, 2, and ∞ which represents Manhattan distance, Euclidean distance, and Chebyshev distance respectively. Furthermore, Minkowski distance is like Euclidean distance metric measuring vector similarity inversely proportional to the computed distance. Some work using Minkowski distance as a similarity measure are found in [33,34].

Pearson similarity algorithm: also known as bivariate correlation, measuring the linear correlation between two vectors of shot time intervals T_a and T_b , and defined as the covariance for the two vectors T_a , T_b with N terms divided by the standard deviation [35]. The result is a range between -1 and $+1$, where -1 is considered fully nonsimilar, $+1$ is exactly similar, and 0 means no linear correlation found. Pearson similarity PS is given by the following formula:

$$PS(T_a, T_b) = \frac{Cov(T_a, T_b)}{\sigma_{T_a} \cdot \sigma_{T_b}} = \frac{\sum_{i=0}^N ((T_a)_i - \bar{a}) \cdot ((T_b)_i - \bar{b})}{\sqrt{\sum_{i=0}^N ((T_a)_i - \bar{a})^2} \cdot \sqrt{\sum_{i=0}^N ((T_b)_i - \bar{b})^2}} \quad (7)$$

Where \bar{a} and \bar{b} are the means of vector T_a and T_b respectively.

Overlap similarity algorithm: known also as Szymkiewicz-Simpson coefficient a metric for vector similarity measurement that deliberate similarity between two compared vectors if one vector is a subset of the other, and related to the Jaccard index measuring the overlap similarity OS between two vectors T_a and T_b with size N by dividing the intersection size by the minimum term of the two vectors as follows:

$$OS(T_a, T_b) = \frac{|T_a \cap T_b|}{\min(|T_a|, |T_b|)} \quad (8)$$

The overlap coefficient value is 1 if the vector T_a is a subset of vector T_b or the other way around. Some related work on overlap similarity algorithm found in [36,37].

4. Experiments & Results

4.1. Dataset & Indexing

The current study's purpose is to build a reliable crawler for a content-based video index to fetch and analyze video files using suitable YouTube videos. Thus, a new dataset was created for evaluating the proposed video retrieval system using a total of 1088 videos from 4 different genres of videos, one of them is a 475 videos dataset called the movie trailers dataset [38] as a sample for movies. The dataset covers the movie list, YouTube ids, movie genres, and other metadata. Three other famous playlists

from YouTube with large numbers were indexed, first is the “Top Tracks - Children's Music” playlist [39], under the music category offered by Best of YouTube section with more than 500 videos, crawled only 246 videos from the music videos in the playlist. Second the playlist “Just good news” offered by BBC News Channel on YouTube with more than 500 videos [40], acquiring only 206 news videos. Lastly the playlist “Popular Videos – Sports” harvested from Best of YouTube [41], Sports category with more than 200 videos, indexed only 134 sports videos. In addition, 27 videos are added randomly from different genders from YouTube videos. Each URL with YouTube id was compiled and fetched to be first checked for duplication and then downloaded for processing and indexing.

Lastly, no pre-defined queries or any information regarding video files gathered from the YouTube platform was embedded in the dataset without being learned and processed through the indexing system. Furthermore, the dataset is not serving the problem of video annotation as in datasets mentioned earlier, TRECVID, YouTube-8M, YouTube-Objects, ...etc. However, temporal data, annotation, and object tagging were used as a combinational signification for video records learned from a collection of combinational strategy training.

4.2. Video Combination-based Matching & Classification

In the process of searching for queries using two-phased temporal and key-objects video combination-based matching, an imperative comparison similarity metrics is used to evaluate both similarity percentages which are compared to a threshold for each phase, deciding whether acquired/crawled video features are in match with an existing record in the dataset or not. This work employed three different metrics for similarity matching, cosine, Jaccard, and Minkowski similarities, from which elected the best two (cosine and Jaccard) to use in the proposed video matching algorithm. An experiment was conducted to evaluate matching threshold and similarity metrics, to find an optimal matching threshold for both phases of the matching process.

A new dataset has been created to evaluate this study's newly proposed search engine system in terms of indexing and retrieve videos located and scattered on public video providing services on the world wide web. The approach succeeded to acquire and index using the developed crawling system, a total of 1088 videos all of which are located on YouTube's public video database. For each of the acquired video files the steps of the following procedure are performed:

First step: applying FFmpeg shot boundary detection tool to detect shot boundaries and 3 keyframes from each shot (one from first tertiary, second from the middle tertiary, and 3ed form last tertiary of the shot length), number of shots (Ns), and set of shot length, called STF or Shots Temporals Feature (Tv). As mentioned earlier, the goal is to split any given video sequence into individual shots at shot boundaries transition locations along the video sequence. FFmpeg uses a histogram-based detection algorithm that calculates the average intensity of pixels amongst two sequential frames. However, an accurate and involuntary threshold selection value is a necessity to the shot boundary detection procedure due to video contents diversity and shots boundaries transitions variety. Consequently, to find an optimal threshold value, a great portion of video sequences have been analyzed automatically to estimate a threshold for each video separately with the aid of FFmpeg and PySceneDetect tools. Initially, the conducted experiment tested for all the changes in the scenes using a primary threshold value of $\geq 1\%$ on both tools, raising the threshold value from 1% to 30% determining scene match variation in-between, and then calculating the average value for the threshold of all detected scene match variation in the threshold rage to compute the optimal threshold value for the experimental video dataset. Table 2 shows the shot detection performance processes using various threshold values divided into groups of 5 percent apart, against 15 random video files per group from the video dataset.

Table 2: Shots Detection Recall and Precision Performance

Threshold %	Shots Statistics		FFmpeg		PySceneDetect		Average	
	No. of Frames	Detected Shots	Best Recall	Best Precision	Best Recall	Best Precision	Avg Recall	Avg Precision
1 – 5 %	935,380	552	1.0	1.0	1.0	1.0	1.0	1.0
6 – 10 %	1,176,608	688	0.994	1.0	1.0	1.0	0.997	1.0
11 – 15 %	1,082,479	632	0.988	0.991	0.996	1.0	0.992	0.9955
16 – 20 %	1,023,649	598	0.964	0.988	0.994	0.989	0.979	0.9885
21 – 25 %	1,362,498	843	0.866	0.875	0.952	0.966	0.909	0.921
26 – 30 %	1,249,361	773	0.762	0.822	0.895	0.903	0.829	0.863

Moreover, shot detection performance is measured in many research studies [42],[43],[44], as the defined performance of the shot detectors applying two parameters recall and precision rates, both depending on the fake or false detections, the missed detections, and the number of detected shots, as shown in the following equations:

$$R(v) = \frac{S_d}{S_d + S_m} \quad (9)$$

$$P(v) = \frac{S_d}{S_d + S_f} \quad (10)$$

Where v is a video with S number of shots, S_d is the detected shots using the shot detectors, S_m and S_f are missed and false detected shots respectively.

Henceforth, an optimum threshold value for the experimental video dataset used in this research is found ranging from 3.6 to 5.7 percent average using both FFmpeg and PySceneDetect shot boundary detection tools.

Second step: after STF set and the number of shots multiplied by 3 keyframes $3N_s$ are extracted, ImageAI is applied on the $3N_s$ keyframe images to extract multiple vectors of object sets all of which constructed out of the 81 object type names in the Microsoft COCO object types, also an extra object called “null object” has been used to indicate the no object extraction from the keyframe image, and lastly by calculating only the maximum object’s percentage probability function $f(o_i)$ out of the set of objects $O_{kf} = \{o_i, \dots, o_n\}$ extracted from each image keyframe:

$$f(O_i) = \begin{cases} 0, & \text{if } o_i \text{ is a null object} \\ \arg \max_{o_i \in O_{kf}} f(o_i) := |\{o_i \in O_{kf} \mid \forall x \in O_{kf} : f(x) \leq f(o_i)\}| & \end{cases} \quad (11)$$

Where x is an ordered set and points o_i for which $f(o_i)$ achieves its largest value. Furthermore, three types of objects related features sets are then extracted according to this criterion as follows:

- A set of object names called Plain Object Names Feature (PONF), with $3N_s$ objects and contains only the text names of each extracted o .
- A set of object weights called Multiple Concepts Weights Feature (MCWF), with $3N_s$ concepts, with each shot has 3 concepts one for each keyframe.
- A set of objects weights sum per shot called Cumulative Concept Weights Feature (CCWF), where concept weights in the 3 keyframes associated with each shot are summed together to

give only one concept weight tagging this particular shot, which means only N_s concepts are extracted per video file.

- A set of object weights called Single Concepts Weights Features (SCWF), where only the maximum weight concept is selected from the lists of objects in each of the 3 frames per shot.

Third step: comparative similarity measurement and matching each video's extracted feature sets against all previously-stored records in the dataset to check for similar video records. If found, the new feature sets and video URL is appended to the same record in the dataset, otherwise, a new record is created for the new video feature sets, keyframes, URL, genre, and the number of shots.

Furthermore, an experiment was conducted based on the comparative similarity measurements explained in section 3.4 to evaluate and select similarity metrics suitable for this study according to accuracy for evaluating the similarity between two vectors of shot time intervals extracted from two video sequences. A video from the movie trailer dataset was used to be a gold standard, comparing it with five video replicas, four of them found and downloaded from the YouTube video database, and the fifth is an undirect recording or camcording for the video captured using a screen and a simple camcorder to capture the video. In addition, two extra-resolution edited videos were used, one is 320×240 , and 1980×1080 ; two more videos are speed edited, one is edited in fast speed with 125%, the other is edited with 75% slow motion speed. The last is horizontally reversed frames video to evaluate the temporal combination feature vector against this kind of misusing approaches of copyrighted materials and video content plagiarism. Table 3 shows the results of comparisons of four algorithms Jaccard, Cosine, Euclidean distance, and Minkowski distance-vector similarity metrics showing the supremacy of cosine similarity as a reliable metric followed by Minkowski distance similarity metric using order of Minkowski $p = 3$. This study indulges cosine similarity metric to be used in comparing and matching temporal combination features vectors before indexing.

Table 3: combinations similarity metrics comparison of time intervals against a gold standard for "The Extra Man Trailer".

Videos 1 to 4 are replicas for the same title trailers on YouTube, a webcam recording, 4 resolution and speed edited, and horizontal reversed frame videos.

Similarity Algorithms	YouTube Replicas				Camcorder	Resolution Edit		Speed Edit		Reversed Frame Image
	Vid 1	Vid 2	Vid 3	Vid 4		320×240	1980×1080	75% Slow	150% Fast	
Jaccard	0.689	0.0	0.0116	0.106	0.0	1.0	1.0	0.0	0.0135	0.0123
Cosine	0.7	0.897	0.635	0.753	0.752	0.93	1.0	0.791	1.0	0.928
Euclidean Distance	12.34	11.42	13.47	11.64	11.236	7.51	0.0	12.35	3.423	6.442
Minkowski Distance	7.988	7.191	8.528	7.873	6.96	5.678	0.0	8.021	1.977	4.614

Fourth step: is classification, if 3ed step reveals that the crawled video is a new video record, thus a new record is generated in the dataset based on the dominant number of objects extracted from the video using PONF feature set as described in section 3 and illustrated in Figure 2 in this paper.

However, as the experiment showed, the dominant name of object aggregation in the dataset was "person", which made 70.2% of the records, followed by object name "null object" with 27.2%, making a total of 91.8% of the video records, this means that only 8.2% of the other objects representing the rest of the videos. According to E. Real, et al [45], concluded the object class name "person" and object absence or "NONE" was the most dominant objects traced in an approximation of 380 thousand 19 seconds YouTube videos.

Therefore, during the experiment, an alteration for the classification algorithm took place to reorder concepts aggregation criteria to consider the second maximum occurring concept/object name in the PONF from the other 79 if the first are one of the 2 dominant concepts "person" and "null object".

However, the algorithm only considers “person” and “null object” if and only if the other 79 concepts maximum representation are less than two occurrences of all shots’ object list in the video record. Table 4 shows the concept-shot aggregation of the dataset according to concepts/objects and the number of shots of 1088 video records. Moreover, 1088 video records were classified with only 58 concepts aggregation result, 23 remaining categories all of which appeared as objects in fewer numbers of video records in the dataset in other words 23 object classes didn’t appear enough to meet classification criteria for concepts aggregation. Additionally, it has been recorded that the highest concept aggregation was the object name class “car” with 211 videos, followed by the “person” aggregation class with 120 videos, and the “null object” with 91 videos.

Furthermore, the second step of aggregation number of shots had 552 videos less than 100 shots, 363 videos between 101 to 200 shots, 43 videos between 201 to 300 shots, 36 videos between 301 to 400 shots, 29 videos between 401 to 500 shots, 36 videos between 500 to 1000 shots, 12 videos between 1001 and 2000 shots, 10 videos between 2001 to 3000 shots, and lastly 7 videos in the range between 3001 to 4000 shots, including one video record with the maximum number of shots, that reached 3603 shots. Figure 4 shows a representation of the number of video records against categories regarding the number of shots aggregations in the dataset.

Finally, the purpose for collection and creation for this 1088 video records dataset is merely evaluating the proposed search engine’s content-based video retrieval system which will include more than 100 queries by example video files from 4 genders of videos (Movie Trailers, Music Videos, News Videos, Sports Videos), each of which has 25 video queries divided into 5 types of video groups with 5 videos set representing each group: original videos, dimension edit, speed edit, webcam, and flipped sets.

Table 4: video classification according to concept/object and number of shots aggregation classes. Only 58 concepts/objects aggregations are recorded, and the maximum number of shots recorded was 3603 shots

Concepts/Object Aggregation	# videos	Shots Aggregation								
		0-100	101-200	201-300	301-400	401-500	501-1000	1001-2000	2001-3000	3001-4000
airplane	32	10	10	2	5	1	1	1	1	1
apple	15	3	1	3	2	2	1	1	1	1
backpack	17	1	2	3	3	4	1	1	1	1
banana	11	3	1	1	1	1	1	1	1	1
baseball bat	6	1	1	1	1	1	0	0	1	0
bear	15	8	1	1	2	1	1	1	1	0
bed	28	10	6	1	2	2	4	1	1	1
bench	11	5	2	1	1	1	0	0	1	0
bicycle	21	5	7	1	3	1	1	1	1	1
bird	59	25	23	1	1	2	5	1	1	0
boat	10	4	4	2	0	0	0	0	0	0
book	7	6	1	0	0	0	0	0	0	0
bottle	9	4	4	0	0	0	1	0	0	0
bowl	3	2	1	0	0	0	0	0	0	0
bus	8	5	3	0	0	0	0	0	0	0
cake	22	18	3	0	0	1	0	0	0	0
car	211	53	143	9	1	2	2	1	0	0
cat	11	6	5	0	0	0	0	0	0	0
cell phone	9	5	3	1	0	0	0	0	0	0
chair	21	11	9	0	1	0	0	0	0	0
clock	10	6	2	2	0	0	0	0	0	0
couch	5	5	0	0	0	0	0	0	0	0
cow	7	5	1	0	1	0	0	0	0	0
cup	24	14	9	1	0	0	0	0	0	0
dog	25	16	9	0	0	0	0	0	0	0
donut	1	1	0	0	0	0	0	0	0	0
elephant	7	3	4	0	0	0	0	0	0	0
frisbee	6	4	1	0	0	1	0	0	0	0
horse	29	11	13	1	1	0	2	1	0	0
keyboard	2	2	0	0	0	0	0	0	0	0
laptop	6	3	3	0	0	0	0	0	0	0
microwave	1	1	0	0	0	0	0	0	0	0
motorcycle	8	3	3	0	0	2	0	0	0	0
null object	91	89	1	0	1	0	0	0	0	0
orange	3	3	0	0	0	0	0	0	0	0
oven	1	1	0	0	0	0	0	0	0	0
parking meter	2	1	1	0	0	0	0	0	0	0
person	120	101	14	2	3	0	0	0	0	0
pizza	2	2	0	0	0	0	0	0	0	0
potted plant	7	6	1	0	0	0	0	0	0	0
scissors	2	1	1	0	0	0	0	0	0	0
sheep	1	1	0	0	0	0	0	0	0	0
sink	1	1	0	0	0	0	0	0	0	0
skateboard	1	1	0	0	0	0	0	0	0	0
sports ball	56	18	7	8	3	6	12	1	1	0
stop sign	3	2	1	0	0	0	0	0	0	0
surfboard	4	3	0	0	1	0	0	0	0	0
teddy bear	13	8	4	0	0	0	0	1	0	0
tie	25	16	9	0	0	0	0	0	0	0
toilet	2	1	1	0	0	0	0	0	0	0
toothbrush	1	1	0	0	0	0	0	0	0	0
traffic light	5	3	2	0	0	0	0	0	0	0
train	9	5	3	1	0	0	0	0	0	0
truck	28	11	16	0	0	0	1	0	0	0
tv	47	13	25	1	3	1	3	0	0	1
umbrella	4	3	1	0	0	0	0	0	0	0
vase	2	1	1	0	0	0	0	0	0	0
wine glass	1	1	0	0	0	0	0	0	0	0
Totals	1088	552	363	43	36	29	36	12	10	7

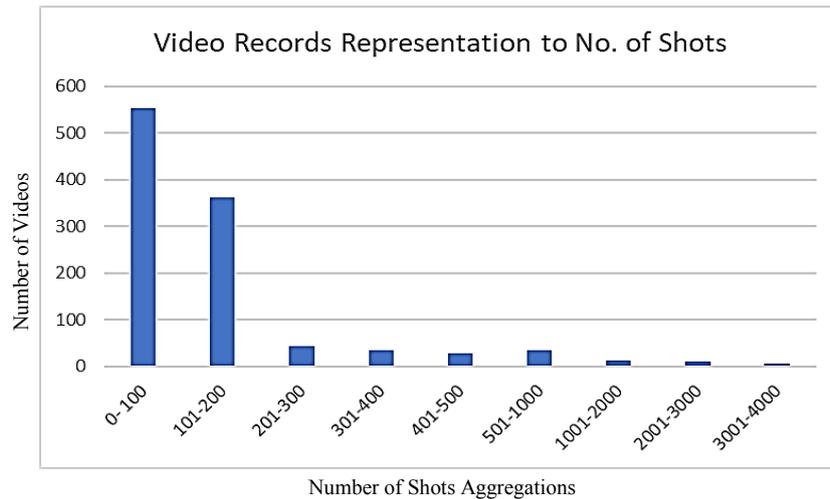


Figure 4: Number of shots aggregation classes representation with the number of video records in each class.

However, the dataset as described is only for evaluation and assessment of the search engine, this means it's not a video annotation method nor for learning and data mining videos for concepts, objects, or

information except for cataloging, indexing, and matching for video files over the web. This means that the proposed method offers a solution for the search problem for any video with multiple shots that have been learned and indexed through the proposed process but not for any video that has not been learned.

5. Conclusions & Future Work

In this paper, a new method for content-based video search engine has been introduced sided with the creation of a large video dataset, for evaluation of the developed retrieval algorithm. This dataset contains an index of 1088 video records gathered from the YouTube public video streaming service. The objective is to enhance the field of web-based video search engines using content-based similarity matching in a large-scale video index that represents video files especially public video streaming services providing services and video databases over the world wide web. Accordingly, there were two core challenges with content-based video search engines that were signified. First, is pre-processing video files extracting features to build a large video content-related index gathering a collection of different video genres with record representation for each video of minimal accommodations and maximal retrieval accuracy, using a lessening computational feature vector and time-efficient state-of-the-art algorithms. Secondly, providing a copyright violation detection manifesto by tracing misused public video uploads, by relating them to original copyrighted video versions removing the obstacles of language, video time/size editing, bootleg recording, and/or flipped frame video material copying. This study processed and stored more than 65 hours of videos, 113502 shots, and 338502 keyframes in more than 1088 videos. This will enable further evaluation for the retrieval system as a final part of the newly proposed content-based video search engine. Expecting that this approach will empower traditional text-based video commercial and non-commercial search engines in providing a more transparent and accurate query-by-example searching tool for users.

This work also provided on the new dataset, extensive testing and comparison of shot-boundary detection and extraction tools, object extraction detectors, and similarity metrics comparative analysis, all of which demonstrated the efficiency of using in the experimental models. In addition, explored and gathered various representations of video categories and genders to ensure a qualitative evaluation based on outcomes and content analysis for the retrieval system approach.

For future work, ensuring that a competitive performance compared with recent work and state-of-the-art complex approaches, focusing on both precision and speed/accuracy enhancement for content-based video retrieval from a large dataset problem. This will demonstrate that this study's minimal representation feature vector relying on low processing shot-based features is stronger and more accurate than frame-by-frame-based approaches and other expensive and sophisticated techniques.

Finally, this study illustrated the compensations of the dataset for performing video retrieval faster and more accurately by using combination-based matching joint with two level-based classifications of video records affording objects- and event-based classification with numerous subevents regarding several shots in each video record. The conducted experiments showed efficiency in querying time using the matching algorithm against a minimized classified subset inside a very large-scaled dataset.

References

1. A. S. Adly, I. Hegazy, T. Elarif, and M. S. Abdelwahab, "Issues and Challenges for Content-Based Video Search Engines: A Survey," in *IEEE Proceedings - 2020 21st International Arab Conference on Information Technology (ACIT)*, 2020, pp. 1–18.

2. E. G. Ortiz, A. Wright, and M. Shah, "Face recognition in movie trailers via mean sequence sparse representation-based classification," in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2013, pp. 3531–3538.
3. H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, "HMDB: A Large Video Database for Human Motion Recognition Terms of Use Creative Commons Attribution-Noncommercial-Share Alike 3.0 HMDB: A Large Video Database for Human Motion Recognition," IEEE, 2011.
4. K. Soomro, A. R. Zamir, and M. Shah, "UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild," CoRR, abs/1212.0402, vol. abs/1212.0, 2012.
5. H. Idrees et al., "The THUMOS challenge on action recognition for videos 'in the wild,'" Comput. Vis. Image Underst., vol. 155, pp. 1–23, 2017.
6. G. Awad et al., "TRECVID 2016: Evaluating video search, video event detection, localization, and hyperlinking," in 2016 TREC Video Retrieval Evaluation, TRECVID 2016, 2016.
7. D. Mishra and J. Matas, "The Visual Object Tracking VOT2017 Challenge Results The Visual Object Tracking VOT2017 challenge results," Icvc, no. November 2017, 2019.
8. P. Dendorfer et al., "MOTChallenge: A Benchmark for Single-camera Multiple Target Tracking," Int. J. Comput. Vis., Dec. 2020.
9. B. SravyaPranati, D. Suma, C. ManjuLatha, and S. Putheti, "Large-Scale Video Classification with Convolutional Neural Networks," 2021, pp. 689–695.
10. A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale Video Classification with Convolutional Neural Networks," in CVPR, 2014.
11. P. Dollar, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: A benchmark," in In Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference, 2010, pp. 304–311.
12. O. Russakovsky et al., "ImageNet Large Scale Visual Recognition Challenge," Int. J. Comput. Vis., vol. 115, no. 3, pp. 211–252, 2015.
13. S. Abu-El-Haija et al., "YouTube-8M: A Large-Scale Video Classification Benchmark," arxiv.org, 2016.
14. A. Prest, C. Leistner, J. Civera, C. Schmid, and V. Ferrari, "Learning object class detectors from weakly annotated video," in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2012, pp. 3282–3289.
15. A. Prest, C. Leistner, J. Civera, C. Schmid, and V. Ferrari, "YouTube-Objects Dataset: A large-scale database of object videos from YouTube," 2012. [Online]. Available: <https://data.vision.ee.ethz.ch/cvl/youtube-objects/>. [Accessed: 15-Nov-2020].
16. H. J. Zhang, A. Kankanhalli, and S. W. Smoliar, "Automatic partitioning of full-motion video," Multimed. Syst., vol. 1, no. 1, pp. 10–28, 1993.
17. B. Castellano, "PySceneDetect." [Online]. Available: <https://pyscenedetect.readthedocs.io/en/latest/>. [Accessed: 04-May-2019].
18. "FFmpeg." [Online]. Available: <https://ffmpeg.org/>. [Accessed: 04-May-2019].
19. M. Almousa, R. Benlamri, and R. Khoury, "NLP-Enriched Automatic Video Segmentation," in International Conference on Multimedia Computing and Systems -Proceedings, 2018.
20. R. Hannane, A. Elboushaki, K. Afdel, P. Naghabhushan, and M. Javed, "An efficient method for video shot boundary detection and keyframe extraction using SIFT-point distribution histogram," Int. J. Multimed. Inf. Retr., vol. 5, no. 2, pp. 89–104, Jun. 2016.
21. O. Küçüktonç, U. Güdükbay, and Ö. Ulusoy, "Fuzzy color histogram-based video segmentation," Comput. Vis. Image Underst., vol. 114, no. 1, pp. 125–134, Jan. 2010.
22. S. F. Altschul et al., "Gapped BLAST and PSI-BLAST: A new generation of protein database search programs," Nucleic Acids Research. 1997.

23. K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask R-CNN,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 2, 2020.
24. V. T. Chasanis, A. C. Likas, and N. P. Galatsanos, “Scene detection in videos using shot clustering and sequence alignment,” *IEEE Trans. Multimed.*, vol. 11, no. 1, pp. 89–100, 2009.
25. H. Chim and X. Deng, “Efficient phrase-based document similarity for clustering,” *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 9, 2008.
26. J. Redmon and A. Farhadi, “YOLOv3: An Incremental Improvement,” Apr. 2018.
27. T. Y. Lin et al., “Microsoft COCO: Common objects in context,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2014, vol. 8693 LNCS, no. PART 5, pp. 740–755.
28. M. Abadi et al., “TensorFlow: A system for large-scale machine learning,” in *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016*, 2016, pp. 265–283.
29. J. O. Moses, “ImageAI, an open source python library built to empower developers to build applications and systems with self-contained computer vision capabilities,” 2018. [Online]. Available: <https://github.com/OlafenwaMoses/ImageAI>.
30. P. A. Zeitschrift, S. N. Band, P. Link, and E. Dienst, “Étude comparative de la distribution florale dans une portion des Alpes et du Jura,” *Bull. la Société Vaudoise des Sci. Nat.*, vol. 37, 2013.
31. C. Plattel, “Distributed and Incremental Clustering using Shared Nearest Neighbours,” 2014.
32. C. Henderson and E. Izquierdo, “Robust feature matching in the wild,” in *Proceedings of the 2015 Science and Information Conference, SAI 2015*, 2015, pp. 628–637.
33. N. Iyer, S. Jayanti, K. Lou, Y. Kalyanaraman, and K. Ramani, “Three-dimensional shape searching: State-of-the-art review and future trends,” in *CAD Computer Aided Design*, 2005, vol. 37, no. 5 SPEC.ISS., pp. 509–530.
34. R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin, “Matching 3D models with shape distributions,” in *Proceedings - International Conference on Shape Modeling and Applications, SMI 2001*, 2001, pp. 154–166.
35. D. G. Bonett and T. A. Wright, “Sample size requirements for estimating Pearson, Kendall and Spearman correlations,” *Psychometrika*, vol. 65, no. 1, pp. 23–28, 2000.
36. A. Cimmino and R. Corchuelo, “A hybrid genetic-bootstrapping approach to link resources in the web of data,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2018, vol. 10870 LNAI, pp. 145–157.
37. Y. Echevarría, L. Sánchez, and C. Blanco, *Hybrid Artificial Intelligent Systems*, vol. 9648. Cham: Springer International Publishing, 2016.
38. A. Tadimari, “2010-2014 American Movie Trailers Dataset,” 2015. [Online]. Available: <https://github.com/tadarsh/movie-trailers-dataset>.
39. YouTube, “Top Tracks - Children’s Music YouTube Playlist.” [Online]. Available: <https://youtube.com/playlist?list=PLFgquLnL59akkUoiohCP9AxkPBjGgjaDE>. [Accessed: 11-Jul-2020].
40. BBC News Channel, “Just good news - YouTube Playlist.” [Online]. Available: https://youtube.com/playlist?list=PLS3XGZxi7cBU5Yo_P26cjihXLN-k3w246. [Accessed: 05-Jul-2020].
41. YouTube, “Popular Videos – Sports YouTube Playlist.” [Online]. Available: <https://youtube.com/playlist?list=PL8fVUTBmJhHJmpP7sLb9JfLtdwCmYX9xC>. [Accessed: 19-Jul-2020].
42. J. S. Boreczky, “Comparison of video shot boundary detection techniques,” *J. Electron. Imaging*, 1996.

43. U. Gargi, R. Kasturi, and S. Antani, "Performance characterization and comparison of video indexing algorithms," in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1998.
44. V. Kobla, D. DeMenthon, and D. S. Doermann, "<title>Special-effect edit detection using VideoTrails: a comparison with existing techniques</title>," in Storage and Retrieval for Image and Video Databases VII, 1998.
45. E. Real et al., "YouTube-BoundingBoxes: A Large High-Precision Human-Annotated Data Set for Object Detection in Video," openaccess.thecvf.com. [Online]. Available: <https://research.google.com/youtube-bb>. [Accessed: 18-Dec-2020].