International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 13, Number 12 (2020), pp. 5142-5148

A Content-Based Retrieval Model with Combinational Features and Indexing for Distributed Video Objects

Nithya Kaliaperumal, Akansha Das, Vijayakumar Balakrishnan

Computer Science Department, Birla Institute of Technology and Science, Pilani, Dubai Campus, UAE.

ORCID: 0000-0002-1722-2633 (Vijayakumar B.)

Abstract

A content-based video retrieval system (CBVR) provides a convenient mechanism for searching a large collection of video objects. These objects are dynamically generated by a wide user community over the internet and also by various video hosting servers. Due to the rapid growth of multimedia content originating from multiple sources, conventional video retrieval methods based on keywords are not adequate in meeting the technical challenges. This led to the development of efficient video retrieval mechanisms. A CBVR model using combinational features is formulated and its implementation is discussed in this paper. The model considers a combination of four features namely, texture, color, binary tree partitioning and local binary patterns for indexing and retrieval of distributed video objects. A *user* can initiate video-based queries from any given site. The distributed query execution engine parses the queries and sets them up for execution at one or more video repository sites. The search process computes the similarities between the input query video and the video objects in the repositories using Chi-squared distance. The matched keyframes and their ranks are maintained in a dictionary at each site. Finally, the user is presented with the matched and similar output for the given input query video. The current work provides a flexible and scalable mechanism for users at large, who mainly work on video retrieval systems.

Keywords: Content-based video retrieval, Distributed objects, Feature extraction, Indexing, Query execution

List of Abbreviations: BPT: Binary Partition Tree, CBVR: Content-Based Video Retrieval, CIE: International Commission on Illumination, COMB_FETS: Combinational Features, DQEE: Distributed Query Execution Engine, HSV: Hue Saturation Value, LBP: Local Binary Patterns, MPEG: Moving Picture Experts Group

I. INTRODUCTION

We live in an era, where vast amounts of data are collected from everything around as at an astonishing pace. Most of these data are in the form of multimedia content. Huge amounts of images, audios and videos are being created every day. All these data are very easily available at reasonably inexpensive prices due to an affordable multimedia acquisition sources such as mobile phones, digital cameras, webcams, internet, graphic tools and other advanced devices that require a large storage capacity. In recent years, due to trends and high demand, multimedia storage has grown and the cost for storing multimedia data has become cheaper. This has led to a vast amount of multimedia content generation such as text, audios, images, videos, animations and other interactive contents floating all over the internet.

In the present days, the users prefer videos for illustration and understanding purposes. The typical applications include video lectures, products demonstration and installation. It is estimated that the web constitutes around 85% of the multimedia content. Nearly 300 hours of video are uploaded to YouTube every minute, Facebook users generate more than 8 billion video views per day and Google's Nest reports more than 100 hours of surveillance video uploads per minute. This poses a real challenge in handing these voluminous video data.

The efficient storage and access of multimedia data means, we can avoid the customary manual process of searching and retrieving them based on metadata. The retrieval system can be made automatic and powerful by facilitating content-based search [16]. Traditional textual based video retrieval technique faced some drawbacks in the process of indexing and retrieving the videos from large databases. The process has been time consuming as it takes long time to analyze the metadata and manually assign a textual data with it. It also involves a different insight about a certain video and hence the results may not match the user's expectations, since it is a subjective process. In such a system, the user needs to use metadata and annotations similar to what is present in the video database, while querying for a video. So, an adequate knowledge about the large database is needed as a prerequisite. These challenges have paved the way for the next generation video retrieval system. Videos are very rich in visual content like color, texture, shape and motion, these can be used to resolve the drawbacks of the traditional retrieval system [18]. These retrieval systems that function on the content rather than the metadata of the video are called as content-based retrieval systems.

Nowadays, video processing techniques are deployed in a variety of applications including Multimedia based Learning and education, data visualization and interpretation, publications, digital libraries, broadcasting and entertainment. This paper proposes a framework for storing and retrieving large volume of distributed video objects, in an efficient manner.

II. RELATED WORK

With the development of the internet technologies and increasing demand for video processing applications, it is essential to develop efficient search and retrieval techniques for

video objects stored at one or more sites. The video fingerprinting technique, involving centroid of the gradient orientations, has been dealt in detail [1] for the search process in a video database. It is based on the pairwise independence and robustness of video processes such as changing frame size and lossy compression. The WordNet and MPEG-7 techniques are combined to form Visual Ontology [2] that describes a structure for the visual information. The structure links the classes between visual and general concepts, to make a visual ontology.

The storytelling structure of the video parts include key-frames, scenes and semantically related stories [11]. These parts provide the annotations of the visual content for browsing and retrieving the desired videos. Snoek et al. [3] proposed three models to select an essential detector from a multimedia thesaurus. The models include components for text matching, ontology querying and semantic visual querying and they facilitate an automatic video retrieval process.

The schemes based on sketches and strings are discussed in [5], for analyzing and indexing the trajectory matching process of videos. This process samples a set of control points from each trajectory and retrieves the selected videos from the databases. The concept-based video retrieval uses Markov Random Field approach. It generates rich textual descriptions of related video concepts, from the web source. A weighted query concept is also applied here for indexing, ranking and retrieving the resultant videos [8]. The dimensionality reduction, using Principal Component Analysis, significantly improves the retrieval of relevant videos from the database, for the given input query [7]. The key frames are down sampled to an appropriate size for extracting the features. Rassem et al. [17] developed a method using CLBP and LBP variant for image categorization. It improved the photometric invariance and its discriminating property based on its texture feature. Video Compression using content and feature coding, has been performed using a visual retrieval method [12]. This involves interactions between the feature descriptors and visual contents. The Rate Accuracy Optimization balances the coding rate and retrieval performance.

A framework for video retrieval system using image queries has been dealt with by Araujo and Girod [13]. Here, the scalability is compared with the baseline system using the key frames and the segmentation is handled by creating the Fisher vectors. Another framework by Jaspers et al. [19] uses computer vision techniques. It includes components to examine the video, split the content into objects, create the metadata and search the relevant video. Zhang et al. [14] defined a visual similarity function for video queries using top-k images in the database. They used a combination of Convolutional Neural Networks and Bag of Visual Word techniques, to extract the information from the video frames. In addition, a Visual Weighted Inverted Indexing scheme has been implemented to improve the accuracy of video retrieval process. Word2VisualVec [15], a deep neural network technique considers a predictive model for video captions using visual-audio representation. It uses multi-scale sentence vectorization and multi-layer perceptron properties for retrieving the captions.

A content-based video retrieval system has been developed to reduce the retrieval time for the appropriate video from a video database [4]. This system deploys an indexing structure, that can be categorized by an image search engine. The clustering and filtering techniques are used in building the indexes. Mokhtarian Farzin et al. [6] discuss a content-based video retrieval model, to extract the corners of video frames and track them using video databases. To extract the corners, they followed the multi-scale corner detector method in the pre-processing phase. The closest corners of the query objects are extracted and tracked in both forward and backward directions, to locate the positions of the similar objects in each video frame. Cedillo Hernandez A et al. [9] discuss a content-based video retrieval system involving three stages. They include processing the DC image of I-frames to extract the key-frames, identifying the region of interest of key-frames and estimating the Euclidean distance to retrieve the relevant videos from the database. Another content-based image retrieval method [10] relates few features of images like color, texture and color coherence vector. The Manhattan distance is computed to estimate the cosine similarity between the two matching images.

The existing literature on content-based video retrieval systems focused on video structure analysis such as key frame extraction, segmentation, calculating similarity measure and video indexing. The current work discussed in this paper uses a combination of features such as color, texture, local binary patterns, edge detection and object detection for content-based video retrieval from video databases, stored at one or more sites. The rest of this paper organized as follows. Sections I and II briefly review the CBVR system. Section III explains Model Description. Section IV and V present the Query Execution and Algorithm. Section VI and VII discuss the Search Process and Implementation. Lastly, Section VIII concludes the paper.

III. MODEL DESCRIPTION

The content-based video retrieval at each site in Fig.1. includes components for Key Frames Extraction and Storage for the input video objects, Indexing using combinational features, Video Query Processing and Similarity Matching.

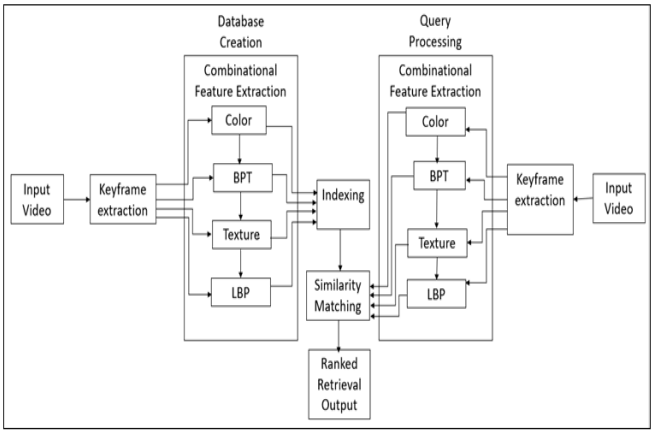

Fig. 1. Work flow for content-based video retrieval at each site.

III. I Indexing using Combinational Features

For indexing process, we quantify the dataset (video objects) by creating the feature descriptors for the appropriate keyframes, which are selected from distributed video objects.

A video object consists of a collection of frames. The frame rate as recorded by the camera, determines the display quality of the video. The video object consists a sequence of similar frames. A threshold value is specified to estimate the accuracy requirement of motion estimation between the successive frames. The key frames are extracted and stored in a Key Frames Repository. The steps are shown as follows:

Step 1: Read a video object file

Step 2: Set a frame threshold value. This can be definable by the user. In the current work, it is set as 18000.

Step 3: Generate the frames per second.

Step 4: Compare the current frame with the previous frame. If their motion vector exceeds the frame threshold value, these two frames are considered as key frames and store them in Key Frames Repository. If not, the current frame is skipped.

Step 5: Repeat steps 3 and 4 till the last frame has been compared with its previous frame.

III. II. Color Histogram

Color descriptors of each extracted keyframe includes local and global descriptors. Global descriptors specify the overall color content of the keyframe and local descriptors, provide the information about the specific regions in the keyframe. In the current work, the color spaces considered are HSV and CIELUV. HSV space is obtained by non-linear transformation of the RGB space. CIELUV is a transformation of CIE XYZ color space. It expresses color by three numerical values, where L* represents lightness while U* and V* are the chromaticity or color values.

Dividing Frame into Regions: The color feature vector is extracted for the objects in video repository as well as the query video at any given site. The color histogram for each region is created with a bin size of 140 to have detailed description of data points.

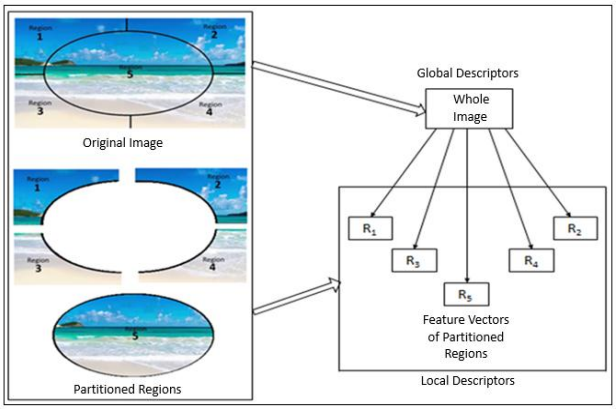

Fig. 2. Partitioning of input keyframe into distinct regions.

The key frame is divided into five regions as shown in Fig. 2 and the feature vectors for each region are stored in the index file. They include four corner regions and one elliptical region in the center. The regions help in the localization of the keyframe, thus being able to represent the blue sky in regions 1 and 2 (on top), brown sand and white beach in regions 3 and 4 (at the bottom) and the whole scenery in region 5 (the center of the image). The color histograms are normalized to obtain the scale invariance, for the verification of keyframe with similar features. Later, a comparison is made to measure the similarity between the corresponding parts of the query video keyframes and the keyframes of the video repository.

III. III. Binary Partitioning Tree

Binary Partitioning Tree is used to represent regions of a keyframe. The root node of the tree represents the entire keyframe. The keyframe is segmented to form smaller regions using a safe color cube, where the quantization of the image takes place. An image will be quantized by considering groups of pixels together. For each region, the mean values for the color & area are calculated and appended to a feature vector. This way, the feature vector for the entire keyframe can be formed and then compared with the feature vector of another keyframe to determine similarity between them.

III. IV. Texture Feature

Texture features are aimed at capturing the repetitive and granularity patterns present in a keyframe. Texture features can be extracted from the input image by using wavelet transformation. The process includes analyzing a keyframe by decomposing it into series of sub-band images. These sub-band images can be considered as one form of wavelet coefficients, where the texture features are computed. The steps are shown as follows:

Step 1: Convert the color scheme of the keyframe to greyscale.

Step 2: Each frame is divided into 8x8 equal regions.

Step 3: Wavelet decomposition is applied on each region.

Step 4: The Variance and Mean are calculated for the 4 arrays that are generated after performing wavelet transformation.

Step 5: Append the mean and variance vectors of each region into the feature vector array.

Step 6: Repeat step 5 for all regions in the keyframe.

Step 7: Normalize the feature vector array.

III. V. Local Binary Pattern

Local Binary Pattern works on 2D texture analysis. The basic functionality of LBP is to summarize the local structure in a keyframe by thresholding the neighborhood of each pixel. The result is formed as a binary number. Consider a pixel as the center and threshold its neighbors against it. If the intensity of the center pixel is greater than or equal to its neighboring pixel,

then the neighboring pixel is assigned a value of 1 or else it is assigned a value of 0. The binary number for each pixel will look like a binary pattern like11001111 and so on. In a 3x3 pixels matrix, there will be 8 surrounding pixels for any center pixel and there will be 2^8 possible binary patterns, called as Local Binary Patterns.

IV. QUERY EXECUTION

The distributed query execution engine for the content-based video retrieval system is shown in Fig. 3.

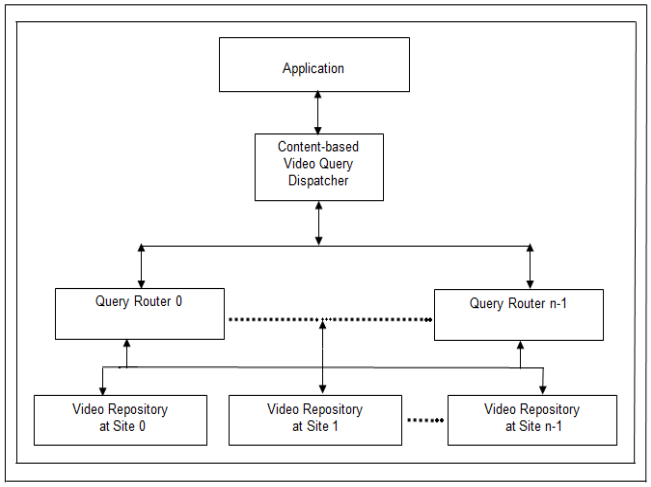

Fig. 3. Distributed Query Execution Engine.

V. ALGORITHM: COMB_FETS

The algorithm COMB_FETS works in two stages, such as INDEXER and QUERY_PROCESSOR as shown in Fig. 4 and Fig. 6.

V. I. INDEXER

The Indexer stage selects each video object from a site's video repository and extracts the keyframes. Each of the feature vectors namely color, BPT, texture and LBP are extracted from the keyframes and stored in a separate index file.

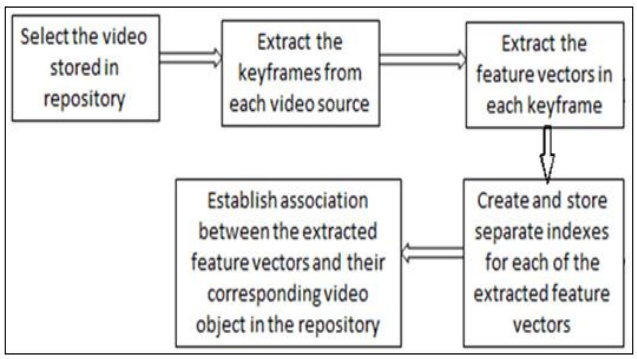

Fig. 4. Work Flow for Indexer at each site.

```
Input: VObj, Kfrms
       // VObj: VideoObject from VideoRepository VRep, Kfrms: Keyframes
Output: IDx: The Combinational FeatureVectors stored in the index
1:  begin
2:  open VRep
3:    for each VObj in VRep do                    // Video Objects in Video Repository
4:      Initialize (VObj, Kfrm);
5:        for each Kfrm in each VObj do           // Keyframes in Video Objects
6:              load the Kfrm into memory;
7:              if Kfrm in VObj = null then
8:              Kfrm = keyframes in VObj;
9:              save Kfrm to memory;
10:             for each FVec in each Kfrm do     // Combinational Feature Vectors in Keyframe
11:                     load the FVec into memory;
12:                     if Color FeatureVector in Kfrm = null then
13:                         CFVec = FVec in Kfrm;      // Extracted Values of Color
14:                         save CFVec to memory;
15:                     if BPT FeatureVector in Kfrm = null then
16:                         BPFVec = FVec in Kfrm;     // Extracted Values of BPT
17:                         save BPFVec to memory;
18:                     if Texture FeatureVector TFVec in Kfrm = null then
19:                         TFVec = FVec in Kfrm;      // Extracted Values of Texture
20:                         save TFVec to memory;
21:                     if LBP FeatureVector in Kfrm = null then
22:                         LBFVec = FVec in Kfrm;     // Extracted Values of LBP
23:                         save LBFVec to memory;
24:                         IDx = Index (Values (CFVec, BPFVec, TFVec, LBFVec));
25:                         save IDx to memory;        // Main Index File
26:                     end if
27:             end for
28:             return IDx
29:         end for
30:         close VObj;
31:   end for
32:   close VRep;
33: end
```

Fig. 5. Psudocode for the INDEXER at each site.

For similarity ranking, the feature vectors are compared between input video query object and the objects in the site's video repository.

The psudocode for the INDEXER is shown in Fig. 5. It's actions can be described as follows:

- The INDEXER is connected to the database creation component.
- It accepts each input video object in the site's repository, extracts and stores the keyframes in another subdirectory.
- It loops over every keyframe in each video object, extracts the combinational feature vectors such as CFVec, BPFVec, TFVec & LBFVec and builds entries for each feature vector in the main index file.
- The computing time for this process is O(K) where K indicates the total number of keyframes present in the site's video repository.

V. II. Query Processor

The input query video object given from the query initiation site is selected first. This is followed by extraction of key frames and combinational feature vectors. These feature vectors are compared for similarity matching and retrieval with the index file present at any given site.

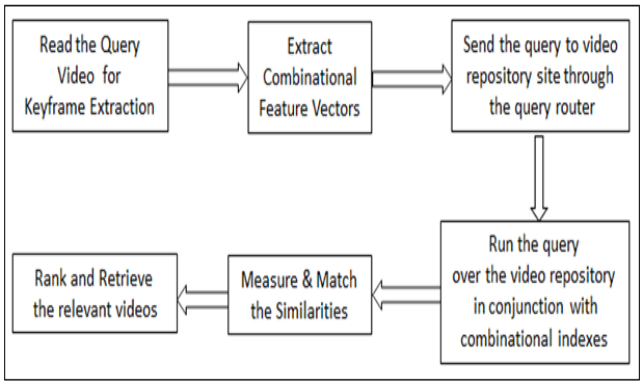

Fig. 6. Work Flow for Video Query Processor at each site.

The psudocode for the QUERY_PROCESSOR is shown in Fig. 7. It's actions can be described as follows:

- The QUERY_PROCESSOR accepts a single query video object as input, from an user at the query initiation site and parses it.
- The keyframes for the query video object are extracted and stored in another subdirectory.
- It loops over each keyframe in the query video object, extracts the combinational feature vectors such as CFVec, BPFVec, TFVec & LBFVec and compares with the main index file for similarity match process. The computing time for this process is O(Q) where Q indicates the total number of keyframes present in the input query video object.

```
Input: QVid; Query Video
Output: mtchlist; A list of matched videos objects          // Ranked Videos
 1: begin
 2:     Query (QVid);                                       // Query Video
 3:     Initialize (QVid, QKfrm);
 4:     for each QKfrm in QVid do                           // Keyframes in Query Video
 5:         load the QKfrm into memory;
 6:         if QKfrm in QVid = null then
 7:         QKfrm = Keyframe in QVid;
 8:         save all QKfrm of QVid to memory;
 9:             for each QFVec in each QKfrm do // Combinational Feature Vectors in QVid Keyframes
10:                 load the QFVec into memory;
11:                 if Color FeatureVector in QKfrm = null then
12:                     CQFVec = QFVec in QKfrm;     // Extracted Values of Color
13:                     save CQFVec to memory;
14:                 if BPT FeatureVector in QKfrm = null then
15:                     BPQFVec = QFVec in QKfrm;    // Extracted Values of BPT
16:                     save BPQFVec to memory;
17:                 if Texture FeatureVector in QKfrm = null then
18:                     TQFVec = QFVec in QKfrm;     // Extracted Values of Texture
19:                     save TQFVec to memory;
20:                 if LBP FeatureVector in QKfrm = null then
21:                     LBQFVec = QFVec in QKfrm;    // Extracted Values of LBP
22:                     save LBQFVec to memory;
23:                     QFVec = (Extractor(CFVec, BPQFVec, TFVec, LBQFVec));
24:                     save QFVec to memory;
25:                 end if
26:             end for
27:         end if
28:     end for
29: load IDx to measure similarities
30: for each Values in IDx do
31:     mtchlist.append (IDx.compare(Values, QFVec)); // Matching Process of Query Video
32: end for
33: return mtchlist;
34: end
```

Fig. 7. QUERY_PROCESSOR algorithm at each site.

VI. SEARCHING, RANKING AND RETRIEVING

VI. I. Search Process

The search process computes the similarities or distances between the input query video and the video objects stored at one or more sites. The features such as color, BPT, texture, and LBP are used together in their similarity match process involving keyframes. Accordingly, the keyframes of the matched video objects are returned in the ranked order. For similarity measure, Chi-squared distance is used to compare the distance between the corresponding feature vector pairs - index feature values and query feature values. A smaller value for Chi-squared distance implies a more closer match. Here, x_u is the first feature vector, x_v is the second feature vector and N is the number of elements in the feature vectors. Then the distance measure [20] is computed as given in equation 1.

$$d(X_u, X_v) = \frac{1}{2} \sum_{n=1}^{N} \frac{[\, x_u(n) - x_v(n)]^2}{x_u(n) + x_v(n)} \quad \ldots (1)$$

VI. II. Video Ranking Algorithm

Input: Keyframes of Query Video

Output: Keyframes of matched video objects in the ranked order

1. For each Keyframe of the query video:-
   a. Perform a lookup for atmost K similar keyframes of the video objects repository, at one or more relevant sites. /* Here K is set as 30 */
   b. The corresponding video objects for the matched key frames are identified in the above mentioned sites.
   c. Maintain a Dictionary of matched keyframes for the video objects present at each site. The dictionary entry stores three attributes, <keyframe, video-object,, rank>.
   d. The rank is computed as follows: Out of the 30 similar keyframes, the first keyframe is the most similar one and the last keyframe is the least similar one The rank is updated in the dictionary dynamically during the match process.
2. The dictionary finally gives the ranked list of similar videos, at the relevant sites.

VII. IMPLEMENTATION

The algorithm COMB_FETS has been implemented using Python. The test data consists of approximately 870 video objects with total size 1.6 GB. They are stored in *mp4* format in a separate video repository. These objects pertain to natural sceneries like oceans and surroundings, forests, sports and gadgets. The play duration varies between 7 to 10 minutes.

Fig. 8. Query Video 9000.mp4.

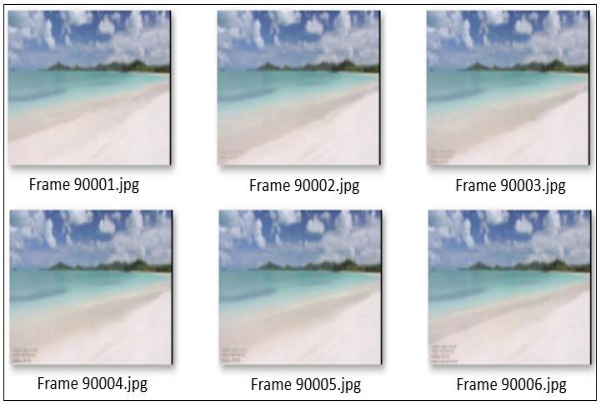

Fig. 9. Keyframes of Query Video.

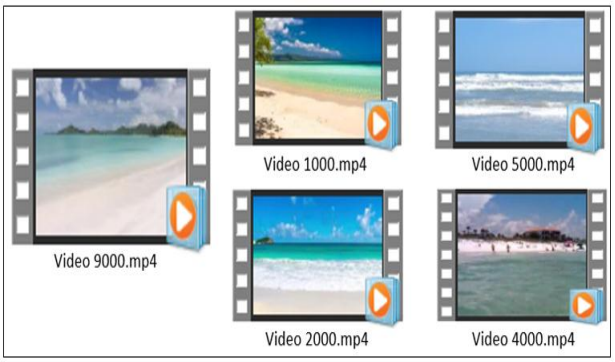

Fig. 10. Matched and Similar Results for the Input Query.

The Test Scenario involves an input video clip given as user input as shown in Fig. 8. This video relates to a seashore and its vicinity. The extracted six keyframes are shown in Fig. 9. The matched and similar results returned are shown as shown in Fig. 10.

The size of the video objects is varied between 15.7 to 112.7 MB respectively, during the successive runs in the search process. The average data transfer rate at local site is 6 Gbps and 20 Mbps at the remote site.

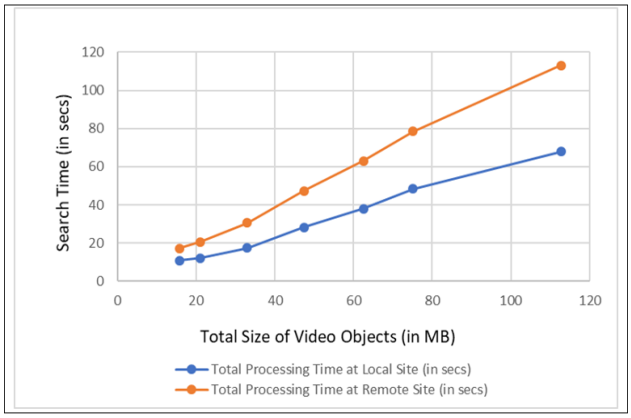

Fig.11. Search Time for the Input Query at Local and Remote Sites.

The search time for the input query is shown in Fig. 11. It includes the time taken for extraction of the combinational features and lookup in indexes. The search time at the local site varies between 10.92 to 67.94 seconds during the runs. The search time for the remote site varies between 17.2 to 113.02 seconds, during the same runs. As the data transfer rate is higher at local site, its search time is faster when compare to the remote site.

VIII. CONCLUSION AND FUTURE WORK

A content-based video retrieval model using combinational features has been dealt with in detail, in this paper. It involved the feature attributes such as texture, color, binary partition tree and local binary patterns for indexing and retrieval of distributed video objects. Chi-squared distance has been used as a similarity measure between the feature vectors from the input query video and the videos in the repository. The individual sites store the matched keyframes and their ranks as dictionary entries. Finally, the user from the query initiation site can access the matched and similar output for the given input query video. The current work offers flexibility, scalability and easy deployment in content-based video retrieval applications such as academic search and web information retrieval. The present work can be extended further by including deep learning mechanism to analyze the visual imagery.

ACKNOWLEDGMENT

The authors would like to acknowledge BITS PILANI DUBAI CAMPUS, Dubai, UAE for providing the experimental facilities.

REFERENCES

[1] S. Lee and C.D. Yoo, C.D, “Robust video fingerprinting for content-based video identification,” IEEE Transactions on Circuits and Systems for Video Technology, 18(7), pp.983-988, 2008.

[2] L. Hollink, M. Worring and A.T. Schreiber, "Building a visual ontology for video retrieval," In Proceedings of the

13th annual ACM international conference on Multimedia, pp. 479-482. ACM, 2005.

[3] C.G. Snoek, B. Huurnink, L. Hollink, M. De Rijke, G. Schreiber and M. Worring, "Adding semantics to detectors for video retrieval," IEEE Transactions on multimedia 9, no. 5: 975-986, 2007.

[4] Y. Hiwatari, K. Fushikida and H. Waki, "An index structure for content-based retrieval from a video database," In Proceedings Third International Conference on Computational Intelligence and Multimedia Applications. ICCIMA'99 (Cat. No. PR00300), pp. 268-272. IEEE, 1999.

[5] J.W. Hsieh, S.L. Yu and Y.S. Chen, "Motion-based video retrieval by trajectory matching," IEEE Transactions on Circuits and Systems for Video Technology 16, no. 3: 396-409, 2006.

[6] F. Mokhtarian and F. Mohanna, "Content-based video database retrieval through robust corner tracking," In 2002 IEEE Workshop on Multimedia Signal Processing, pp. 224-228. IEEE, 2002.

[7] U.S. Pacharaney, P.S., Salankar and S. Mandalapu, "Dimensionality reduction for fast and accurate video search and retrieval in a large-scale database," In Nirma University International Conference on Engineering (NUiCONE), pp. 1-9. IEEE, 2013.

[8] J. Dalton, J. Allan and P. Mirajkar, "Zero-shot video retrieval using content and concepts," In Proceedings of the 22nd ACM international conference on Information & Knowledge Management, pp. 1857-1860. ACM, 2013.

[9] A. Cedillo-Hernandez, M. Cedillo-Hernandez, F. García-Ugalde, M. Nakano-Miyatake and H. Pérez-Meana, "An efficient content-based video retrieval for large databases," In 2015 International Conference on Mechatronics, Electronics and Automotive Engineering (ICMEAE), pp. 15-19. IEEE, 2015.

[10] C. Zhang and L. Huang, "Content-Based Image Retrieval Using Multiple Features," Journal of computing and information technology 22, no. LISS (2014): 1-10, 2014.

[11] L. Baraldi, C. Grana and R. Cucchiara, "Neuralstory: an interactive multimedia system for video indexing and re-use," In Proceedings of the 15th International Workshop on Content-Based Multimedia Indexing, p. 21. ACM, 2017.

[12] X. Zhang, S. Ma, S. Wang, X. Zhang, H. Sun and W. Gao, "A joint compression scheme of video feature descriptors and visual content," IEEE Transactions on Image Processing, 26(2), 633-647, 2016.

[13] A. Araujo and B. Girod, "Large-scale video retrieval using image queries," IEEE transactions on circuits and systems for video technology, 28(6), pp.1406-1420, 2017.

[14] C. Zhang, Y. Lin, L. Zhu, A. Liu, Z. Zhang and F. Huang, "CNN-VWII: An efficient approach for large-scale video retrieval by image queries," Pattern Recognition Letters, 123, pp.82-88, 2019.

[15] J. Dong, X. Li and C.G. Snoek, "Predicting visual features from text for image and video caption retrieval," IEEE Transactions on Multimedia, 20(12), pp.3377-3388, 2018.

[16] https://www.researchgate.net/publication/284148891_Content_based_Video_Retrieval_Systems_Methods_Techniques_Trends_and_Challenges.

[17] T.H. Rassem, B.E. Khoo, N.M. Makbol and A.A. Alsewari, "Multi-Scale Colour Completed Local Binary Patterns for Scene and Event Sport Image Categorisation," IAENG International Journal of Computer Science, 44(2), 2017.

[18] http://www.ijaceeonline.com/VOL2ISS6/IJACEE_FPV2I6P5.pdf.

[19] E. G. Jaspers, R.G. Wijnhoven, A.H.R. Albers, X. Desurmont, M. Barais, J. Hamaide and B. Lienard, "Candela-Storage, Analysis and Retrieval of Video Content in Distributed Systems: Real-Time Video Surveillance and Retrieval," In 2005 IEEE International Conference on Multimedia and Expo, (pp. 1553-1556). IEEE, 2005.

[20] J. Naik, S. Doyle, A. Basavanhally, S. Ganesan, M.D. Feldman, J.E. Tomaszewski and A. Madabhushi, "A boosted distance metric: application to content-based image retrieval and classification of digitized histopathology," ISOP, In Medical Imaging, CAD Vol. 7260, p. 72603F, 2009.