Example Based Video Filters

Mihir Jain, Sreekanth Vempati, Chandrika Pulla and C. V. Jawahar
Center for Visual Information Technology
International Institute of Information Technology
Hyderabad, INDIA
{mihir@research., v_sreekanth@research., chandrika@research., jawahar@}iiit.ac.in

ABSTRACT

Many of the successful multimedia retrieval systems focus on developing efficient and effective video retrieval solutions with the help of appropriate index structures. In these systems, the *query* is an example video and the retrieved results are similar video clips which are available *apriori* in the database. In this paper, we address a complementary problem of filtering a video stream based on a set of given examples. By filtering, we mean to detect, accept or reject the part of a video stream matching any of the given examples. This requires matching of example videos with the on-line video stream. Since the concepts of interest could be complex, we avoid explicit learning of a representation from the example videos to characterize the visual event present in the examples. We model the problem as simultaneous on-line spotting of multiple examples in a video stream. We employ a vocabulary *trie* for the filtering purpose and demonstrate the applicability of the technique in a variety of situations.

Categories and Subject Descriptors

H.3.1 [**Content Analysis and Indexing**]: Indexing methods; H.3.3 [**Information Search and Retrieval**]: Clustering, Information filtering; I.4 [**Computing Methodologies**]: Image Processing and Computer Vision—*Applications*

General Terms

Algorithms, Experimentation, Performance

Keywords

Video filtering, Indexing and retrieval, Vocabulary Trie, Copy Detection

1. INTRODUCTION

In this paper, we address the problem of *video filtering*, the online process of identifying video segments from a continuous stream of videos, which are similar to a given set of examples. Traditional filtering approaches were aimed at accepting or rejecting video information by manipulating the video in spatial or frequency domain [29]. They extensively used the mathematical models related to signal processing for filtering (say low-pass or high-pass) by manipulating pixel values. However, our approach to filtering is more "semantic" in nature, where we would like to filter video segments based on the visual content present in them. In this work, the visual content to filter, is implicitly characterized by a set of examples available *apriori*. Like many other semantic tasks (eg. content-based retrieval), our method also depends on the matching of videos.

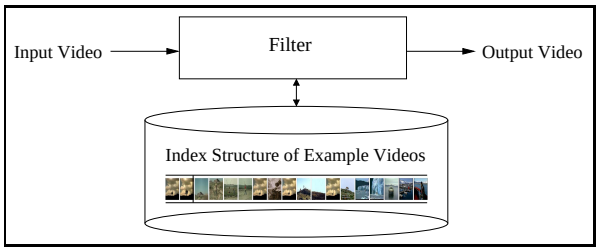


Figure 1: Overview of the Example-based Video Filtering

Example-based content-level processing of multimedia, has been popular in video and image retrieval literature [22, 27, 30]. The focus has been on identifying appropriate descriptors [30] and developing scalable systems which enable efficient retrieval from millions of images or video key-frames [22, 27]. There has also been significant interest in characterizing and recognizing activities and semantic concepts from video examples [15]. This class of algorithms, first learn to characterize the events from training data by computing a classifier, and then apply the learned concepts in new situations.

However, many concepts of practical interest are not easy to represent and learn. For example, the concept of violence is a difficult concept to characterize, even for the state-of-the-art machine learning algorithms. One may also come across categories like commercials in video streams which have high within class variance and relatively small inter-class variance. On the other hand, many of these concepts can be described with the help of examples. This allows us to model the problem as *simultaneous spotting* in a video stream. This approach could meet the immediate requirement of filtering the video stream based on the visual content.

In many practical situations, a human is present within the loop of a video processing system. For example, a human operator is often associated with surveillance video processing for initiating actions based on the video content. In such cases, *on-line spotting* of relevant information from a video sequence can be of immense

CIVR '09, July 8-10, 2009 Santorini, GR

help. We demonstrate that this is feasible even when a robust recognition of the specific concept is probably impossible.

We model the problem of video filtering in a manner complimentary to that of video retrieval. We begin with a set of examples (traditional "queries") which are indexed in the database. The larger video collection, which needs to be processed, is unseen during the offline indexing phase. The video collection is processed on-line, to identify the concepts represented by the given set of examples. In a way, what we are interested is in spotting rather than retrieving. Traditional retrieval systems focus on scalability to large databases for efficiency in retrieval. Our focus is on enhancing the throughput of the system and making the algorithm capable of simultaneous spotting of multiple examples. Our formulation also effectively utilizes the sequence information of the video stream, rather than treating it as a set of frames.

The notion of video filtering is highly related to image and video retrieval. Some of the related directions of research are briefly reviewed in Section 2. Basic idea of example based video filtering is presented in Figure 1. On-line video is matched with a set of indexed examples for locating their possible presence. Trees (eg. KD Tree, Vocabulary Tree) have been popular for indexing larger video/image collections and processing smaller queries. We use a *Trie* data structure to index the examples and processing the video stream. Details of our indexing scheme is presented in Section 3. We, then demonstrate the application of the video filtering for three different tasks in Section 4. We conclude the paper in Section 5 by describing some of the future extensions.

2. RELATED WORK

Most of the content based image and video retrieval systems identify similar objects to a given query [7]. Both query and database objects are represented with the help of a set of feature descriptors. Earlier approaches used color, texture and shape descriptors computed globally or locally to describe the visual content of the images. This has been successful in retrieving images with concepts which are rather weak, (for example, "images with red flowers" or "scene of a sun-set next to water"). With this initial success, the focus shifted to retrieving specific objects (under widely varying imaging conditions) or object categories. Invariant description of interest points and patches have been the key to the success in these situations.

Image and video retrieval has been successfully attempted for retrieving objects of interest invariant to scale, orientation and illumination [22, 27, 32] in diverse multimedia collections. These methods primarily addressed the scalability issue towards indexing in large databases. The videos are represented by their key-frames, which in turn are described as a bag-of-interest-regions. Features describing regions-of-interest are quantized using K-means or hierarchical K-means, in an offline phase to build a visual-vocabulary for the given data set. The video collection is then indexed against this visual vocabulary. Once indexed, the database can retrieve videos corresponding to "short" queries, such as a (part of) an image or key-frame selected by the user. Another set of works focuses on building efficient indexing schemes for multimedia collections. Successful examples include LSH [11], min-hash [3, 4], pyramid match hashing [9], vocabulary forest [32], etc. Vocabulary tree has been used for efficiently indexing and retrieving large number of images [22]. A hierarchical partitioning of the feature space makes the quantization efficient. Also the retrieval and ranking of documents are simultaneously achieved by traversing the tree.

Focus of most of these approaches has been on indexing large amount of multimedia data to efficiently search within the given collection. However, on-line structures for indexing video streams has received very little attention. One of the related problem which received some interest in recent past is that of adapting the index structure with changes in visual content. In this direction, Yeh *et al*. [32] extended the notion of vocabulary tree to vocabulary forest while making the indexing process applicable to dynamic environments. Yan *et al.* [31] performed content based copy detection over streaming videos. In this paper, we aim at defining a trie-based architecture for content-based processing of video streams. Our trie based solution allows simultaneous matching of multiple examples.

An important requirement in on-line processing is to retrieve video clips of interest as and when they arrive. This is similar to the concept of *keyword-spotting* popular in speech processing and document image retrieval [24]. Keyword spotting methods locate the possible occurrence of the query word by matching with every possible words in the database. In the case of document retrieval, words are often segmented first and indexed using a set of appropriate features. However such methods are not directly applicable for video data, due to the difficulty of characterizing the visual content corresponding to each concept.

Content-based filtering of images and videos are attempted in literature for applications like adult content detection [8, 34], removal of commercials [5, 28], event detection [20], copy detection [17] etc. Most of these methods formulate this problem as an object/scene recognition or detection by using an appropriate classifier in the right feature space. For example, the filters aimed at removal of adult content or detection of fire formulate the problem in an appropriate color space [20, 33]. In general, example video frames are used in an offline situation to learn the right model or a classifier. Then the new unseen video frames are classified using the learnt model/classifier. Accept and reject filters used for commercial removal also employ similar techniques. Colombo *et al.* [5] attempt to characterize the commercials with the help of low-level features and classify the video segments *into categories*. With the category of commercials becoming more and more diverse, such classification models in simple feature spaces are found to be insufficient. Content-based copy detection (CBCD) techniques have received significant attention in recent years [12, 16]. Focus of research has been on defining the right set of descriptors which are invariant to the allowable set of transformations [17]. There has also been significant concern about the computational complexity of this class of algorithms because of the practical applications in video sharing systems.

In this work, we would like to retrieve concepts from streaming videos, based on the similarity of a video sub-sequence with one of the given examples. This similarity has to be efficiently computed for each given example, for each incoming frame. To address this, we use the *Trie* data structure. Trie has been extensively used as an index structure in the area of string matching [26]. Trie is a suffix tree representation which can be used to find the strings that are exactly or approximately matched to a given query string. Tries offer text searches (exact or approximate) with costs which are independent of the size of the document being searched. By posing the problem of indexing multimedia data for video processing, similar to indexing in document search, the Trie can be used for video indexing [2, 23]. We describe our Trie construction in the following section.

3. VOCABULARY TRIE

Popular video retrieval systems aim at indexing large quantities of images and videos, and serving a small set of queries while being deployed on the field. Focus has been on the efficiency in retrieval and scalability to large video databases. These formulations typically employs trees [22], hashes [9] or inverted indices [27] for

the indexing of the visual data. Our objective is to process (filter) large amount of videos with the help of an index structure which is built out of *a relatively small set of example videos*. The indexing scheme that we require should be capable of

1. indexing relatively small number of examples available *apriori*
2. processing of large amount of *unseen* videos
3. avoiding explicit segmentation of video stream for matching with example videos and
4. employing any generic comparison scheme for comparing frames/sequences.

We achieve these objectives with the help of a trie data structure. Tries are ordered tree data structures popular for a number of tasks related to information retrieval [14]. They are useful for matching, based on some similarity measure, for sequences of symbols in a language. A path from the root to a leaf represents a symbol sequence inserted into a trie, during the indexing. The leaf nodes store the identifiers of symbol sequences. An example of trie is shown in Figure 2. Tries get constructed from a sequence of alphabets. An alphabet can be a scalar or vector or even a set representation. When trie is used for detection in an on-line setting, the stream of data gets matched/aligned with the sequence of nodes, and any successful termination at the leaf is treated as a valid detection. Tries are also used for approximate string matching [26]. Importantly, tries are not sensitive to the curse of dimensionality problems which is a challenge in multimedia computing.

A video (or even a key frame) can be represented as sequence of symbols or visual words and indexed into a Trie structure. This is made possible by the quantization of the visual data to produce a finite set of alphabets from a given video sequence [22, 27]. A set of videos to be indexed results in an appropriate vocabulary (words) and define the problem space. Traditional quantization schemes employ K-Means or its variants for the quantization and vocabulary construction. The resulting symbols/alphabets are formed by representing quantization cells or clusters in the feature space.

In our case, number and diversity of examples could be significantly smaller than the total amount of video that trie needs to process. In such cases, adding negative examples into the quantization step allows one to control the detection (false positive and false negative) rates. When the examples are diverse enough, influence of the negative examples seems to be negligible. Since the trie is represented in terms of index of clusters, representation is independent of the dimensionality of the feature space, as is the case in any bag of words representation.

There are two basic problems in formulating the on-line video processing problem using Trie: (i) representation of video sequences with the help of discrete symbols (ii) computing similarities of two video frames.

3.1 Representation and Matching of Videos

It is intuitive to use a temporal representation for videos, unlike the popular representation as a set of key-frames [27], which is not suitable for on-line processing of videos. Let us consider a simple representation. A video frame is represented as the average color of the frame and video clip is represented as a sequence of such color descriptors. Such a frame-level representation could be sensitive to the temporal sampling/segmentation process. One could also represent the averaged color over a set of consecutive frames (overlapping or non-overlapping) as another measure for the description. For many practical applications, a simple representation

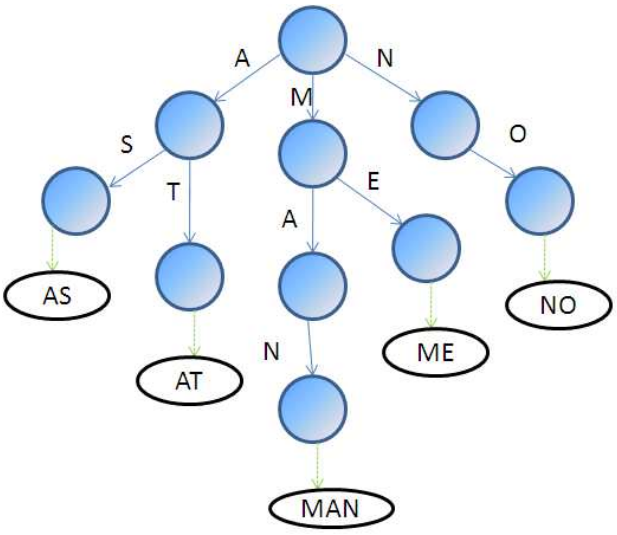


Figure 2: Example Trie for set of words

based on color could be quite insufficient. One could also think of representing the video frame(s) with the help of a set of interest points and their representations such as SIFT for matching and detection.

We represent the video at frame level using the features suitable for the given task. The feature space is quantized into K bins using features extracted from a limited set of training data, using a clustering algorithm. Each feature is then indexed to the closest quantized bin, each frame then represented as a set of these quantization indexes. The sequence of the frame features is used in the trie construction and look-up.

Exact matching of two words or bag of words for detecting identical content could be relatively straightforward with any reasonably invariant representation. In many practical situations, one is interested in matching which allows partial and inexact matches of two representation of words. When the alphabets are described by a set of interest point descriptors, one could define a matching score based on the cardinality of intersection of the representations in the video stream and in the trie. Such a similarity score was used earlier in [3].

3.2 Trie Construction

The given set of example videos are indexed in a trie. Vocabulary trie construction from example videos, is pictorially shown in Figure 3. During the construction phase, the trie is incrementally built from each example. The common prefix sub-sequences are aligned for those examples which have similar frames to begin with.

The trie has a height h and a breadth b. Each frame of an example video occurs at different depths from the node. Hence, the height of the Trie is the length of the longest example video. Each example video constitutes a path from the root to a leaf of the Trie. The leaf is labeled with the concept of the example. Example videos share the nodes corresponding to "similar" frames at the same depth. The total number of leaves in the trie is the number of given examples, N. In the worst case, each example will constitute a distinct path from the root to the leaf. In this case, the storage complexity would be $O(h.N)$ and the time for building the trie would be $O(N^2)$ requiring only the first frame to be compared with the previously built trie. The ideal case is a balanced trie, with equal breadth b

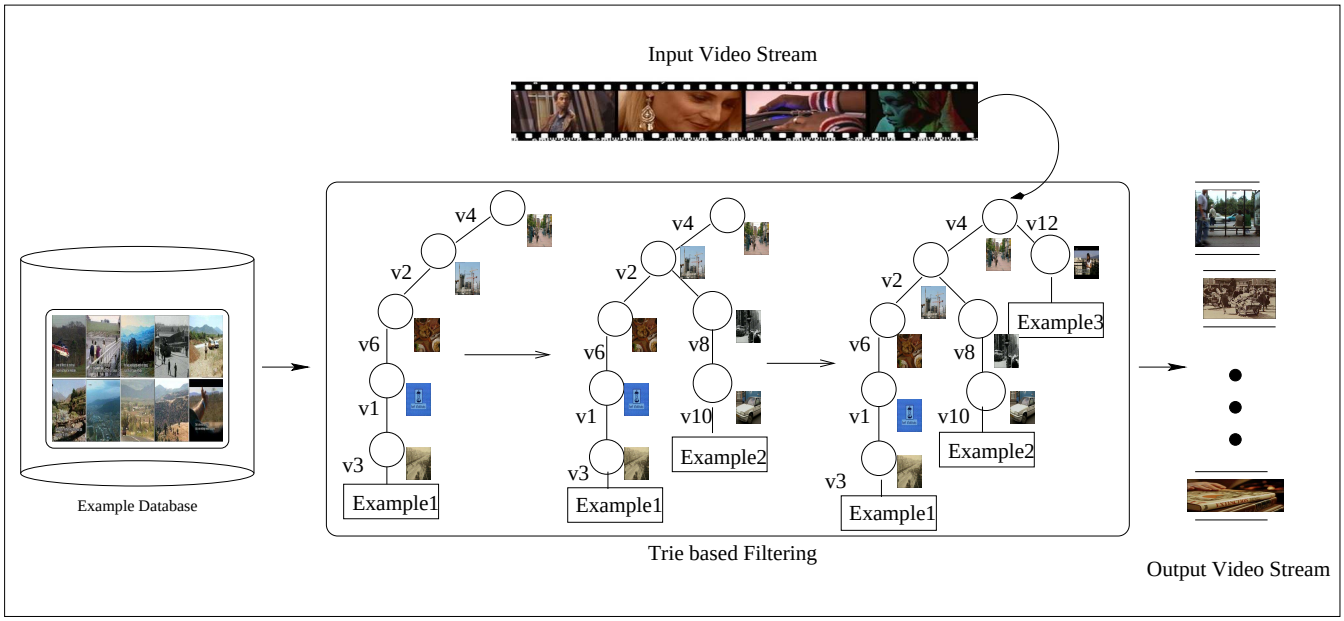


Figure 3: Building a Vocabulary trie for video sequences and using it for processing the input video stream.

at all depths. The storage complexity in the ideal case would be $O(h.b)$ ($b << N$), while the time complexity would be $O(h.b.N)$, since each frame is matched with b nodes at each depth.

Each edge of the vocabulary trie is a symbol. An input sequence of words takes the path along the edge, symbol corresponding to which is most similar to it and the similarity is above a certain threshold. During detection, each frame is checked for a possible match with any of the nodes at $d = 1$. Whenever there is a match, the subsequent frames are matched down the vocabulary trie, and so on. If the sequence of frames from the on-line video terminates in a leaf node, the appropriate concept is said to have been detected.

Algorithm **1** summarizes the trie construction and detection process. During the off-line phase, examples are inserted into the database. During the on-line phase, the trie allows fast processing of the given video sequences.

Such a trie can introduce a latency equal to the maximum length among the example videos, in the worst case. Matching in trie is efficient, since only a few set of nodes will be evaluated for most frames. Such a sequential matching, in general, favor's lesser false positives. However, the matching threshold can be varied to control the detection rates, depending on the application.

The score/matching performance of a video sequence depends on (i) the length of the sub-sequence which it matches, normalized with respect to the length of possible paths in the trie which has this as a sub-path (ii) the quality of match of each of the alphabets/symbols. (iii) Number of tries which generates warnings/ detections. Score of the query video is equal to the number of the frames that match with the examples in database. If the score is above a threshold θ, then the video is blocked.

4. APPLICATIONS

We now demonstrate the application of vocabulary trie on a spectrum of situations. We start by demonstrating the applicability of this method to the detection and removal of a set of *apriori* known commercials from a broadcast video stream. The task is to detect the possible presence of a sequence of video frames which are identical or highly similar to those available in the database. In the second application, we address the problem of detecting copies of videos where a larger set of transformations are possible [17]. Our method allows the detection of copies of multiple videos in a single pass (processing cycle). We then demonstrate the applicability of vocabulary trie in situations where relatively complex concepts of human activity, is spotted in images and videos.

4.1 Commercial Removal

Removal of commercials (or a set of example videos) help in segmenting, summarizing, storing and processing of broadcast videos [19, 28]. They are also an integral part of information retrieval systems designed for broadcast videos. Identification of the examples could be done either manually or with the help of audio-visual clues. Given a set of commercials, we index them into a vocabulary trie in the offline phase and use it for detecting the presence of similar video segments from the "test" videos. During indexing, we extract color histogram features and build an associated vocabulary by clustering them using K-Means, to 500 clusters. The visual words (or the cluster indices) are then used to construct the trie.

The trie is tested over a video sequence of 300 hours duration (or approximately 300 GB in MPEG) captured from 10 different broadcast news channels. We detect the possible presence of a commercial in this video sequence in about a second (excluding the feature extraction time). The false positive rate of detecting the commercials is about 28%. The false positive rate could be reduced further by using more complex and discriminative features (see the next sub-section). Our method scales to large number of commercials without any significant loss in computational efficiency or the precision as demonstrated in Figure 5. The exact time requirement depends on the percentage of commercials in the video sequence. In our case, commercials occupied 16% time of the video duration.

To further evaluate the performance of the vocabulary trie on detection of commercials, we manually ground truth-ed a database of

Algorithm 1 Vocabulary Trie

Trie-Construction In the offline phase, trie is constructed from example video sequences for the given examples: $\mathcal{V}_1, \mathcal{V}_2 \ldots$

- Initialize an empty trie. For the given examples $i = 1, 2, ...$
- Find the longest prefix sequence which is common to the trie and the i th example video. When a mismatch takes place in the sequence, initiate a new path resulting in termination of the leaf node labeled with this example.

Online-Detection In the on line phase, video stream is processed for the possible presence of the examples.

- For the given sequence of words, pass through the trie until either we get a leaf node or no path is available.
- If we reach the leaf node, return back success with the detail of the example and the location from where the possible sequence started.

20Hrs with 250 commercials. In addition to the label, start and end frames were also annotated. Some example frames from the commercial videos used can be seen in the Figure 4. The detection performance of the commercials depends on various parameters. We use F-score as evaluation measure, which takes both the precision and the recall into account. It is defined as

$$F = \frac{2 * (precision * recall)}{(precision + recall)}$$

In Figure 6(a), we demonstrate the effect of length of commercial on the detection rate. In general, it is observed that longer the duration of the commercial, better the detection rate. For this experiment, we have used the number of clusters (visual words) to be 500. Number of visual words used for representation of the video sequence also affect the detection rates. In Figure 6 (b), we demonstrate the effect of number of clusters on the detection rate. With increase in number of clusters, the detection rate also increases.

Many practical situations for video filters require controlling of the false positive/false negative rates depending on the application. As mentioned in the previous section Vocabulary Trie allows flexibility in design, and thereby parameters which can directly affect these rates. We vary the length of the example and query videos by grouping p consecutive frames together and obtain the word corresponding to the mean of their feature descriptors. We demonstrate the variation of false-negative rate with p in Figure 6(c). We can observe that false-negative rate increases with temporal quantization parameter p.

Thus, it can be seen that the vocabulary trie allows efficient and scalable spotting of commercials in a video stream with significant amount of flexibility on false positives/false negatives.

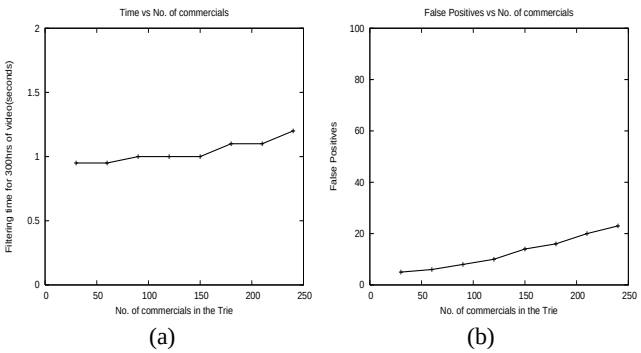
Figure 4: Example frames from the Commercial Videos used

4.2 Content Based Copy Detection (CBCD)

Content-based copy detection has received significant attention in recent years due to its immediate practical applications [13, 17]. On-line CBCD [31] is becoming an important problem, to filter duplicates in multimedia collections. The vocabulary trie approach is directly applicable to the problem of on line CBCD.

Popular methods for CBCD extract a small number of pertinent features (called signatures or fingerprints) from images or a video stream and then match them with the database according to a dedicated voting function [17]. An important requirement which has come to existence in this problem is the capability to detect (or match) possible copies of multiple video clips with minimal computational overhead. There are two important steps in solving this problem: (i) efficient methods for similarity computation (ii) detection of copies by accumulating the similarity scores. State-of-the-art methods focus on solving the first part efficiently. Our method is also capable of addressing the scalability in number of videos to be matched as demonstrated in the last section.

In the CBCD setting, one needs to allow larger amount of variability for defining duplicates. A copy could be a video clip which is modified in appearance (eg. color, contrast), geometry (eg.resize, cropping) or re-capturing (eg. perspective effects, overlaid text) etc. [18]. To accommodate these variabilities, we use SIFT [21] and SURF [1] feature descriptors computed over interest points to describe the frames. The visual vocabulary is built using hierarchical K-Means algorithm. In most situations, vocabulary is constructed by quantizing the feature descriptors obtained from example videos. In our case, Trie is supposed to function on similar examples as well as large number of non-example situations. Thus we tried introducing feature descriptors from non-example videos while quantizing. While clustering we weigh the distance from non-example videos by α, a measure of importance. We build the trie with a symbol/alphabet represented frames, which converts a video into a sequence of sets (bags) of visual words. Given two



(a) (b)

Figure 5: Scalability of Trie for detecting commercials in broadcast TV. (a) Time Vs No. of commercials and (b) False positives Vs No. of commercials. One can observe the scalability of the system to large number of examples

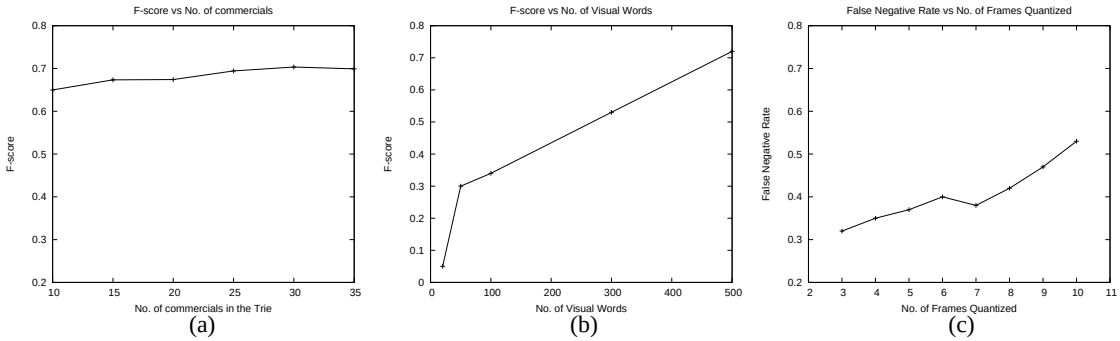


Figure 6: Effect of (a) duration of commercials, (b) number of visual words on F-score and (c) Temporal quantization parameter p on false-negative rate

elements of the sequence, A and B, we define the similarity as:

$$Sim(A,B) = \frac{|A \cap B|}{|A \cup B|} \quad (1)$$

In our first experiment on CBCD, we compare the performance of different features and trie parameters for a set of 1000 video clips. Original video clips were obtained from broadcast news channels. Video clips were manipulated by blurring, adding noise, cropping, resizing, gamma correction etc. We use average precision for performance evaluation as used in most of the CBCD tasks [12].

We compare the performance of the above two features and Trie parameters, and present the results in Table 1. The input video stream is formed by 100 transformed example videos and videos not present in the database which constitute a total of 46K frames when sub-sampled at the rate of $2fps$. Times reported do not include the time taken for feature extraction. It can be observed that the performance in general improves with Vocabulary size, K. Results are also not much affected by increasing the number of examples, N, to build the trie though the time of processing increases.

In a generic video filtering situation, there are other practical challenges. Such as, (i) If a symbol matching fails somewhere or (ii) when sequence in the query is a copy of some sub-sequence of an example, a naive implementation of the video filter could fail. We address these problems using the following methods:

Mean Score Decision to traverse further at any node in the trie will have to depend on the scores of symbol matching done from the root to the current node. Therefore, we keep a threshold on the mean of these scores to make the sequence matching robust to any rare symbol matching failure. We also keep a threshold, F, on the number of frames matched to detect the sequence as a copy.

Figure 8: Examples of original and transformed video frames of Muscle data-set

Forest of Tries To deal with the case (ii), we build a forest of N tries numbered from 1 to N, each of maximum depth D. For building the forest of tries any example from the database is first inserted in the trie 1, after inserting D frames we move to next trie and so on. Finally we get $N = \lceil L/D \rceil$ tries, where L is the length of longest example in the database.

While processing the query video stream we initially start from trie 1. If any mismatch happens after starting from the i^{th} trie, then we again start from the $(i+1)^{th}$ trie and continue until a sequence from the query is accepted or we reach the last trie. We move to the first trie when a mismatch occurs in the last trie or a sequence is accepted. By this we ensure that we don't miss any sub-sequence of $length \geq F + D$ in the query (assuming that when correct frames are compared they do match). This is because we can miss a maximum of D initial frames of any example when a forest of depth D is used. D (can vary from 1 to L) acts as a trade-off parameter between performance and time which can be observed in our

	Vocabulary Size (N=210)			Number of Examples (K=10^4)		
Feature	K	Average Precision	Time (secs)	N	Average Precision	Time (secs)
SIFT	9^4	0.7273	59	100	0.7907	28
	10^4	0.7778	62	150	0.7799	44
	11^4	0.8007	64	210	0.7778	62
SURF	9^4	0.7236	40	100	0.7633	20
	10^4	0.7656	42	150	0.7647	30
	11^4	0.7509	42	210	0.7656	42

Table 1: Performance of Trie for copy detection

	Mean Score		Trie Forest		
Feature	Average Precision	Time (secs)	D	Average Precision	Time (secs)
SIFT	0.8182	19	50	0.9011	86
			100	0.9011	51
			200	0.9011	26
SURF	0.8012	9	50	0.9011	35
			100	0.8182	21
			200	0.8012	11

Table 2: Results of Copy Detection on MUSCLE data-set

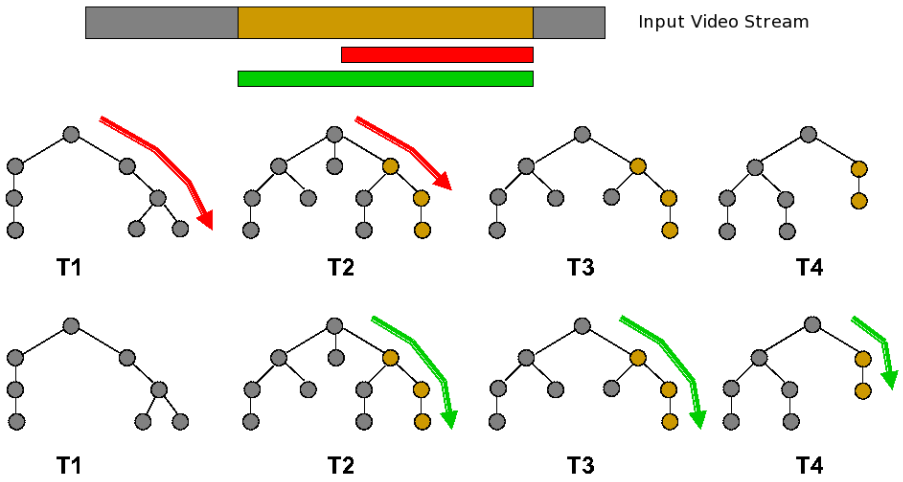


Figure 7: Processing a query with forest of tries: The top row shows that a mismatch occurs when we start searching from trie $T1$. The bottom row shows that a copy of sub-sequence of an example can be detected by starting from the next trie.

next experiment.

An example of how forest of tries work is shown in Figure 7, the brown part of the input video stream is a copy of sub-sequence of an example video in the database (brown colored nodes in the forest). When processing starts from trie $T1$ it leads to mismatch as shown by red path. The actual copy of sub-sequence of an example is found when we search by starting from the trie next to $T1$, i.e., $T2$, as shown by green path. Detecting copy of such sub-sequences of examples is not possible with a single trie.

For our second experiment we use MUSCLE-VCD-2007 database [18]. This database is composed of about one hundred hours of videos spread over 101 different files and it's ST1 query set is composed of 15 videos of total length of about two and a half hours. Out of these 15 videos, 10 are transformed from some video in the database and rest five are not from the database. Some examples of original and transformed frames from Muscle data-set are shown in Figure 8.

We use the ST1 query set as our database and join 101 videos from MUSCLE database to form a 100 hour input video stream. This is according to our objective of filtering large amount of videos with the help of trie which is built out of a relatively small set of example videos. Feature descriptors computed over interest points of frames from 15 videos of the database (sub-sampled at the rate of $0.5fps$) are quantized into 10K visual words and a Trie or a Forest of tries is built as explained above.

Results of this experiment using Trie and Forest of tries are shown in table 2. We can see the improvement in the performance by using Forest of tries. Performance improves by decreasing D in case of Forest of Tries at the expense of time. We can observe in the table that it improves for SURF and remains constant for SIFT. The above experiments show how our approach provides an efficient and accurate solution to the problem of CBCD.

4.3 Activity Spotting

We now show the applicability of our method in a more complex problem of spotting the actions which plays a crucial role in video surveillance and monitoring. This task is more semantic in nature. As any action is an ordered sequence of poses, a trie data structure can be employed to index and capture the temporal pattern of the actions. This can be achieved by representing each pose in an action using an appropriate feature.

We demonstrate the utility of the proposed architecture for the task of human activity spotting. Influenced by the recent success of Histogram of Oriented Gradients(HoG) in human/object detection tasks[6, 10], we represent each frame (pose) of an action with a HoG descriptor.

Given any video, we initially localize the moving person, by subtracting previous frame of the video from current frame. Figure. 9 shows the frames of an action in the first row and the corresponding regions of localized person in the bottom row. We extract HoG descriptors from the localized regions of the moving persons in example video frames and build a vocabulary by clustering these features using K-Means algorithm. The dimension of HoG descriptor is kept constant even if the size of the localized window changes. We achieve this by choosing appropriate cells per block, number of overlapping blocks per row and column, number of bins. Any video can now be represented with the sequence of cluster indices. We refer these cluster indices as pose words.

A trie is built from example videos of different actions performed by different people. We have used videos containing "Running" and "Hand Waving" actions performed by different people from KTH data-set [25]. When two different people perform the same action, length of the action period may differ, also the sequence of pose words may not be exactly aligned. Therefore, we use approximate matching in the trie by allowing some mismatches [26].

Test video stream is prepared by inserting 35 video sequences of the above actions at different places in a video stream with no human activity. These video sequences inserted in test video are not present in the database. We are able to detect 31 video sequences correctly with 6 false positives and 28 without any false positives. The average precision obtained is 0.812.

For all the three applications described here, we obtain quantitative results (detection rates) which are comparable to the state of the art. However, our detection architecture is computationally attractive since it demands only minimum number of comparisons for the spotting purpose. Our trie based filtering scheme (i) is able to process the video stream in an on-line manner (ii) it can support a variety of features (traditional color/texture descriptors as well as the invariant interest point descriptors) (iii) is also compat-

Figure 9: Localization of moving person in a video of running action

ible with the visual bag of words models for describing the visual content. (iv) avoids learning an explicit representation for characterizing the visual content of the examples. Thus, the proposed trie based video filtering scheme makes the content-level processing of video streams feasible in an on-line manner.

5. CONCLUSIONS

In this paper, we have addressed the problem of video stream filtering given a set of example videos. We have proposed a trie based architecture for on-line processing of the video stream. This architecture allows simultaneous spotting (or matching) of example videos in a stream of video frames. We have shown the applicability of a vocabulary trie in a variety of situations relating to the example based video filtering problems.

Processing of on-line video sequences for information extraction and data mining has many significant applications in video retrieval. We are working towards designing appropriate processing (indexing, matching, ranking) architectures for information retrieval tasks from broadcast and other similar on-line video streams. One of the challenges in obtaining real-time solutions to the on-line processing tasks is the computational efforts required for feature extraction and matching. Our proposed architecture is highly parallelizable and a GPU based implementation can speed up the solution significantly.

6. REFERENCES

[1] H. Bay, T. Tutelaars, and L. V. Gool. SURF: Speeded up robust features. In *ECCV*, 2006.

[2] A. P. Berman and L. G. Shapiro. Efficient content-based retrieval: Experimental results. In *Workshop on Content-Based Access of Image and Video Libraries*, 1999.

[3] O. Chum, J. Philbin, M. Isard, and A. Zisserman. Scalable near identical image and shot detection. In *CIVR*, 2007.

[4] O. Chum, J. Philbin, and A. Zisserman. Near duplicate image detection: min-hash and tf-idf weighting. In *BMVC*, 2008.

[5] C. Colombo, A. D. Bimbo, and P. Pala. Retrieval of commercials by video semantics. In *CVPR*, 1998.

[6] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*. IEEE Computer Society, 2005.

[7] R. Datta, D. Joshi, J. Li, and J. Z. Wang. Image retrieval: Ideas, influences, and trends of the new age. *ACM Comput. Surv.*, 40(2), 2008.

[8] M. M. Fleck, D. A. Forsyth, and C. Bregler. Finding naked people. In *ECCV*, 1996.

[9] K. Grauman and T. Darrell. Pyramid match hashing: Sub-linear time indexing over partial correspondences. In *CVPR*, 2007.

[10] K. Hatun and P. Duygulu. Pose sentences A new representation for action recognition using sequence of pose words. In *ICPR*, 2008.

[11] A. Joly and O. Buisson. A posteriori multi-probe locality sensitive hashing. In *ACM Multimedia*, 2008.

[12] A. Joly, O. Buisson, and C. Frlicot. Content-based copy detection using distortion-based probabilistic similarity search. *IEEE Transactions on Multimedia*, 2007.

[13] A. Joly, C. Frelicot, and O. Buisson. Robust content-based video copy identification in a large reference database. In *CIVR*, 2003.

[14] S. Karthik and C.V.Jawahar. Efficient region based indexing and retrieval for images with elastic bucket tries. In *ICPR*, 2006.

[15] I. Laptev, M. MarszaÅĆek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008.

[16] J. Law-To, O. Buisson, V. Gouet-Brunet, and N. Boujemaa. Robust voting algorithm based on labels of behavior for video copy detection. In *ACM Multimedia*, 2006.

[17] J. Law-To, L. Chen, A. Joly, I. Laptev, O. Buisson, V. Gouet-Brunet, N. Boujemaa, and F. Stentiford. Video copy detection: a comparative study. In *CIVR*, July 2007.

[18] J. Law-To, A. Joly, and N. Boujemaa. Muscle-vcd-2007: a live benchmark for video copy detection, 2007. http://www-rocq.inria.fr/imedia/civr-bench/.

[19] K. Lienhart. On the detection and recognition of television commercials. In *ICMCS*, 1997.

[20] C.-B. Liu and N. Ahuja. Vision based fire detection. In *ICPR*, 2004.

[21] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60:91–110, 2004.

[22] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *CVPR*, 2006.

[23] S. Park and K.-H. Hyun. Trie for similarity matching in large video databases. *Inf. Syst*, 29(8):641–652, 2004.

[24] T. Rath and R. Manmatha. Word spotting for historical documents. *IJDAR*, pages 139 – 152, 2007.

[25] C. Schüldt, I. Laptev, and B. Caputo. Recognizing human actions: A local svm approach. In *ICPR*, pages 32–36, 2004.

[26] H. Shang and T. Merrettai. Tries for approximate string matching. *IEEE Transactions On Knowledge And Data Engineering*, 1996.

[27] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *ICCV*, page 1470, 2003.

[28] Sung-Hwan-Lee, Won-Young-Yoo, and Young-Suk-Yoon. Real-time monitoring system for tv commercials using video features. In *ICEC*, 2006.

[29] A. M. Tekalp. *Digital Video Processing*. Prentice-Hall, 1995.

[30] A. Torralba, R. Fergus, and Y. Weiss. Small codes and large databases for recognition. In *CVPR*, 2008.

[31] Y. Yan, B. C. Ooi, and A. Zhou. Continuous content-based copy detection over streaming videos. In *ICDE*, 2008.

[32] T. Yeh, J. Lee, and T. Darrell. Adaptive vocabulary forests for dynamic indexing and category learning. In *ICCV*, 2007.

[33] H. Zheng and M. Daoudi. Blocking adult images based on statistical skin detection. *Electronic Letters on Computer Vision and Image Analysis, Vol4 No2*, December 2004.

[34] H. Zheng, H. Liu, and M. Daoudi. Blocking objectionable images: adult images and harmful symbols. In *ICME*, 2004.