DETECTION AND TRACKING OF REPEATED SEQUENCES IN VIDEOS

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTER ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

TOLGA CAN

August, 2007

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assist. Prof. Dr. Pınar Duygulu Şahin(Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Fazlı Can

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. A. Aydın Alatan

Approved for the Institute of Engineering and Science:

Prof. Dr. Mehmet B. Baray
Director of the Institute

ABSTRACT

DETECTION AND TRACKING OF REPEATED SEQUENCES IN VIDEOS

TOLGA CAN
M.S. in Computer Engineering
Supervisor: Assist. Prof. Dr. Pınar Duygulu Şahin
August, 2007

In this thesis, we propose a new method to search different instances of a video sequence inside a long video. The proposed method is robust to view point and illumination changes which may occur since the sequences are captured in different times with different cameras, and to the differences in the order and the number of frames in the sequences which may occur due to editing. The algorithm does not require any query to be given for searching, and finds all repeating video sequences inside a long video in a fully automatic way. First, the frames in a video are ranked according to their similarity on the distribution of salient points and colour values. Then, a tree based approach is used to seek for the repetitions of a video sequence if there is any. These repeating sequences are pruned for more accurate results in the last step.

Results are provided on two full length feature movies, Run Lola Run and Groundhog Day, on commercials of TRECVID 2004 news video corpus and on dataset created for CIVR Copy Detection Showcase 2007. In these experiments, we obtain %93 precision values for CIVR2007 Copy Detection Showcase dataset and exceed %80 precision values for other sets.

Keywords: copy detection, media tracking, story tracking.

ÖZET

TEKRAR EDEN SIRALILARIN BELİRLENMESİ VE TAKİBİ

TOLGA CAN
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Yöneticisi: Assist. Prof. Dr. Pınar Duygulu Şahin
Ağustos, 2007

Bu tez çalışmasında, bir video parçasının daha büyük videolarda aranması için yeni bir yöntem sunuyoruz. Sunulan yöntem sıralıların farklı zamanlarda yada farklı kameralarla çekilmesinden dolayı ortaya cıkabilecek olan görüş açısı ve aydınlanma değişimlerine ve bunlara ek olarak sıralılardaki film karelerinin sıra ve sayı değişimlerine karşı gürbüzdür. Bizim algoritmamız için sorgu gerekmemekte ve verilen medyadaki bütün tekrarlı tamamiyle otomatik olarak bulmaktadır. İlk olarak medyadaki film kareleri içn renk bilgilerine ve anahtar noktaların dağılımına dayanarak benzer film kareleri bulunmaktadır. Bunu ardından, medyadaki tekrarlar bir ağaç yapısı kullanılarak aranmaktadır. Son olarak da bu tekrar eden sıralılar daha doğru sonuçlar elde edilmesi için sadeleştirilmektedir.

Bu çalışmanın deneyleri iki adet filmde ”Run Lola Run” ve ”Groundhog Day”, TRECVID 2004 verisindeki reklamlarda ve CIVR'in Kopye Takibi için verdiği veritabanında yapılmıstır. Bu deneylerde kopya tanımada %90'nın üzerinde, diğer datasetlerinde ise %80 üzerinde doğruluk değerleri elde edilmiştir.

Anahtar sözcükler: hikaye takibi, kopya yakalama, medya takibi.

Acknowledgement

I would like to express my gratitude to my supervisor Assist. Prof. Dr. Pınar Duygulu Şahin who has guided me throughout my studies.

I would like to express my special thanks to Prof. Dr. Fazlı Can and Assoc. Prof. Dr. A. Aydın Alatan for showing keen interest to the subject matter and accepting to read and review the thesis.

I am thankful to RETINA Learning and Vision Group for their comments and suggestions on my studies.

I am thankful to TUBITAK for their financial support on my studies.

My greatest gratitude is to my family, (my father (Abdulkadir CAN), my mother (Nafiye CAN), my sister (Tuba (CAN) Erdoğan) and new member (Bülent Erdoğan)) who are the reason of the stage where I am standing now.

Finally, special thanks to you ....

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

While there is a growing amount of digital videos available in many sources, the current research on video retrieval does not go beyond image retrieval and discards the temporal information which makes videos distinct from images.

In searching for videos, most of the current systems either use textual information provided in the form of manual annotations or speech transcript text; visual information extracted from video frames or key-frames; or simple combination of both [48]. In all cases, the results are provided in the form of a single shot or a collection of shots. However, video shots are not independent from each other and the valuable information is available with a sequence of shots rather than with individual shots.

We argue that, for a video retrieval system to be distinct from an image retrieval system, it is important to search for video sequences rather than to search for individual shots. We approach to the problem similar to the Query Based Example (QBE) approach in image retrieval, and aim to find similar video sequences based on their visual representation.

Detection of similar sequences is important since it helps better indexing and summarisation, and also reduction of huge amount of data by eliminating the repetitions.However, there are important issues to be considered: (i) the signal distortions due to digitisation or encoding, and different frame rates, (ii) variations in the order and the number of frames due to the editing of the sequences, (iii) dissimilarity of video frames due to view point and lighting changes. Based on these issues, we divide the application domains which require detection of identical or similar video sequences into three groups.

Copy detection: Growing volumes of broadcast videos shared among different media resulted in a new requirement: detection and tracing of copies or duplicates. Detecting copies of videos is very important for copyright issues but difficult when the amount of data is large, resulting in a new challenge for Content Based Copy Detection [3]. In CIVR2007, an organisation is held to explore and compare different ways for dealing with the copy detection challenge since growing amount of digital media brings search of copies to a new critical issue. The assumption in copy detection is that the videos are distorted due to digitisation, encoding or transformations [22, 12].

Media tracking: Tracking a piece of media which is used in different times or sources is important. For example, companies want to monitor TV commercials to ensure that the commercials are broad-casted properly and to track competitor's commercials for planning their marketing strategies [11, 7]. Another example is the tracking of news stories in a single channel. It is common for the news channels to re-use the material as the related story develops by slightly editing the videos by removing or adding material [34, 5]. In both cases, the repeated video sequences may have slight variations due to editing.

Story tracking: It is common for the important events to be captured with different cameras. In this case, although the event or more generally the story is the same, the dotage's may be different since the camera positions and parameters may differ. Also, the footage may be edited differently to represent different perspectives. Similarly, in some movies, such as "Run Lola Run" and "Groundhog day" some portion of the story repeats several times with different footage. In these cases, both the lighting conditions and view point of the camera may change resulting in large variations in the video sequences corresponding to the same instance.

Common part of media tracking, story tracking and copy detection is that all require finding similar parts in videos which can be considered as repeated sequences. Most of the existing approaches firstly detect similar frames (near-duplicates) in media data then find repeating sequences. Approaches, that use near-duplicates, make use of a hard threshold to find similar frames. However, finding a hard threshold that is applicable to all databases is almost impossible. Also, those approaches discard temporal information of sequences.

Another problem in existing approaches is that most of them need a query. However, query can not be supplied in most of the cases. For example, how can we find the news in TV broadcast data which appears the most frequent? If we need a query, we should find all news in broadcast and use each of them as a query in one of the existing method. However, this approach is inefficient. But if there exists any query free approach, we can find the most important news, which is the one that is repeated more than others, by examining news repetition counts.

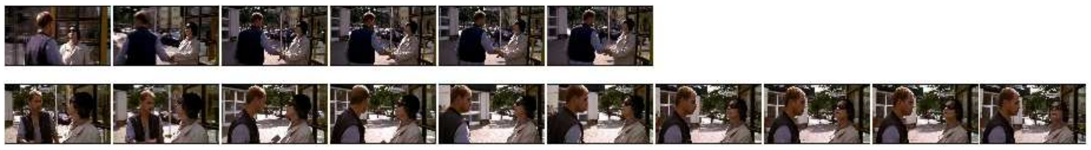


Figure 1.1: An example sequence taken with different camera angles.

Another problem, which the existing approaches have difficulty in solving is the variations in the sequences due to camera differences. While a repeated video sequence is captured with the same single source in copy detection and media tracking problems, resulting in almost identical duplicates, in story tracking there are multiple sources causing largely varied sequences. For example, although two sequences in Figure 1.1 corresponds to the same story, the frames inside the stories are very different (another example can be seen in Figure 1.2). Therefore, finding the similar frames for story tracking is more challenging and difficult to solve by using global features which are heavily experimented in finding similar video frames for copy detection and media tracking.

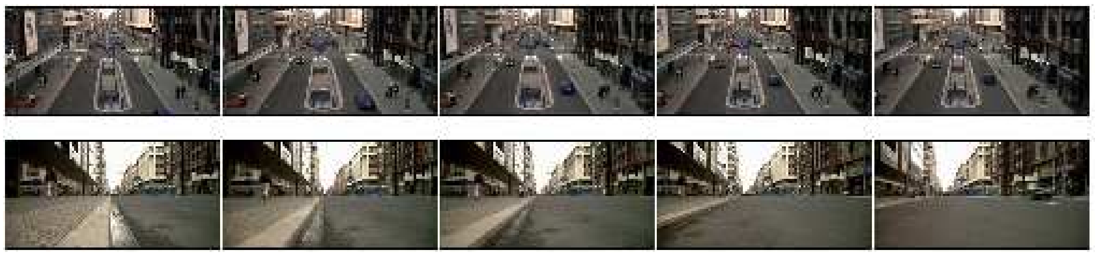


Figure 1.2: Repeating sequences with large differences. These two sequences are seen different since they are taken with different viewpoints. Our algorithm can detect this sequence since our features are viewpoint independent.

A novel approach is required to overcome all these problems mentioned above. It should make use of temporal order knowledge of sequences instead of using just single frame similarities. It should be able to work without a query for media and story tracking. It should also be able to handle differences due to viewpoint and/or illumination changes. In this study, we propose an approach that has all these capabilities.

1.2 Overview of Proposed Method

In this study, we propose a method to search for repeated video sequences using the temporal characteristics of videos. The proposed solution does not require a query and finds all sequences which are repeated more than once. It is invariant to viewpoint and illumination changes since the frames are represented by salient features together with global features. It does not put hard limit for finding the near duplicates but instead use a list of candidate similar frames which are then pruned using the temporal information. All these characteristics result in a method that is applicable to all story tracking, media tracking and copy detection.

The proposed approach creates a tree for each frame in the video. A set of similar frames are placed in different branches and a separate candidate sequence is created for each branch. The temporal information is coded in creating the

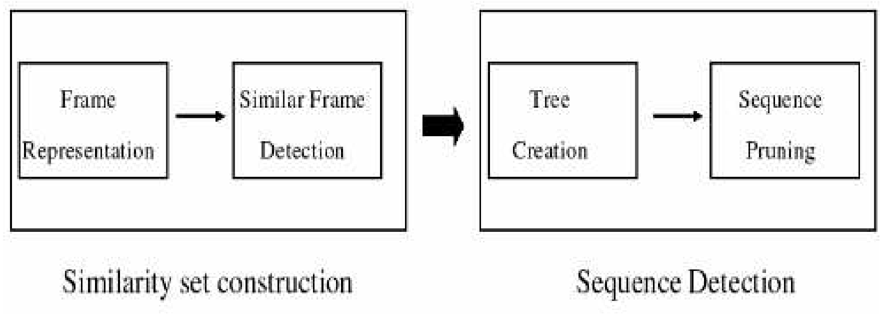


Figure 1.3: System Overview

paths from root to the leaves for each branch. The individual trees for each frame are then pruned to obtain the frames which are part of real sequences. As seen in Figure 1.3, the overall system is composed four steps :

- Similarity Set Construction

  Frame Representation

  Similar Frame Detection

- Sequence Detection

  Tree Creation

  Sequence Pruning

Frame Representation : Since the performance of finding similar video sequences highly depends on finding similar frames, a good representation of frames is a crucial step in our approach. In this study, together with the colour features which codes the global characteristics of the frames, we also make use of salient points which are proven to be helpful in matching images under illumination and viewpoint changes [42, 37, 24]. Details of this part can be found in Chapter 4.

Similar Frame Detection : A good representation is important to find similar frames and this affects performance of finding similar video sequences. In addition to a good representation, an efficient and effective similarity definition is needed. By our similarity definition, we try to find a list of candidate similar frames. This is done by locating a jump position on similarity values. These similarity values are linear combination of similarities that are calculated based on colour features and features extracted from salient points.

Tree Creation : In our tree-based approach, each branch from root to leaves corresponds to a sequence and root of tree corresponds to starting position of the sequence. Since a repeated sequence can start at any frame in the video, a separate tree is created for each frame in video. This makes our approach query free. While creating trees, similar frames are placed to suitable nodes based on some constraints that are described in Section 3.2. Also these constraints make use of temporal order of videos. Detail of Tree Creation is given in Chapter 3.

Sequence Pruning : In our tree structure, each path from root to leaves is considered as a sequence. However, because of insufficient similar sets, there can be false alarms in these sequences. These false alarms are eliminated by a pruning step which checks consistencies of sequences for consecutive frames and applies a one-to-one check to find false alarms. As a result of this pruning step, real sequences are detected. Details of these false alarms and elimination steps can be found in Chapter 3.

The experiments are carried out on the movie “Run Lola Run”, “Groundhog Day”, CIVR copy detection showcase data and on commercials of TRECVID 2004 data.

1.3 Organisation of the Thesis

Chapter 2 gives related studies about the subject, where the approaches are explained in a comparative manner with the proposed method.

Chapter 3 of the thesis introduces tree based approach for sequence detection that is main part of proposed approach. Tree Creation and Sequence Pruning are discussed in this chapter.

Chapter 4 gives details about frames representations based on features and also overviews similar set construction methodology. Distance measures for features are listed and discussed in the same chapter.

The experimental results are presented in Chapter 5.

Chapter 6 reviews the result of this thesis and outlines future research directions on this subject.

Chapter 2

Literature Overview

Recently, several approaches are proposed for finding similar video sequences due to the requirement in this direction parallel to the growing amount of digital videos. Most of these studies deal with copy detection and media tracking and a few of them deal with story tracking. In the following sections, we briefly discuss these related studies.

2.1 Copy Detection

Most of the studies on copy detection focus on signal distortions. They do not cope well with display formats. In addition to this, current studies make use of similarities of single frames by discarding temporal information of videos. In this section, details of current studies on copy detection are given.

Kim et. al [22] proposed an algorithm to detect copies of a video clip based on sequence matching. They used both spatial and temporal similarities between sequences. Spatial similarity is based on 2*2 grid intensity averages. Distance among sequences are calculated by using intensity averages and temporal signatures of sequences.

Similarity measure calculation can significantly affect copy detection results.

Arun et. al. [12], compare several image distance measures, Histogram Intersection, Hausdroff Distance, Local Edge Descriptors and Invariant Moments in their experiments. Their dataset contains exact copies and they propose that local edge descriptors followed by the partial Hausdorff Distance gives the best result. Julien et. al [20] use a voting function based on use of the signal description, the contextual information and the combination of relevant labels. Instead of using SIFT descriptors, they propose a new descriptors based on 4 different spatial positions around interest points in 5 directions. 20-dimensional feature vector is extracted for each keypoint that are extracted by Harris detector. Their approach is more logical than using global features to detect sequences. Keypoints can give more accurate results to describe an image compared to global features.

Ke et. al. [21] propose at method to find copyrighted images and detect forged images. Their approach is based on locality-sensitive hashing [9, 15] and distinctive local descriptors. Since global and local statistics suffer from transformations, low recall and low precision, they used distinctive local descriptors. They need to index all descriptors and Although they used PCA_SIFT, 90 MB memory space is needed to index 1 thousands if database contains more than 200 thousands then they need 18GB memory for indexing. This is not a feasible memory for a system.

Most of the copy detection algorithms use string matching techniques as in [10]. Guimares et. al. propose a method based on the fastest algorithm of exact string matching, the Boyer-Moore-Hoorspool (BMH). They allow some small differences between two correspondent frames by adding a threshold to BMH. Also they modify shifts after a mismatch by allowing smaller distance to move the query pattern to the next alignment verification. Their new algorithm is faster than Longest Common Substring method but they are using some thresholds to find similarities.

2.2 Media Tracking

Media tracking is the problem of keeping track of particular video usage. For example, detection and tracking commercials and/or tracking of news in different channels. Media tracking can be challenging because of editing. This can change number of frames and also orders in sequences. In this section, details of current studies are given.

Arun et. al [11], propose a method for media tracking. They create an index table by using keyframes colours and gradients. To search a media, they first extract keyframes of segments in videos, encode these keyframes by using features and find similars by using previously created index table. Duygulu et. al. [5] track news events by finding the duplicate video sequences and identifying the matching logos. They use both visual cues of keyframes and textual information of shot transcriptions.

Gauch et. al. [8], uses repeated characteristics of commercials to detect and track commercials in videos. As a first step, they extract extract shot boundaries, fades, cuts and dissolves by using RGB colour histograms and some thresholds to find temporal video segmentation. After this step, they use a hash table based on colour moments for frames. They detect sequences by using this hash table and voting scheme. They apply a filtering based on number of frames, relative lengths of shots and mean colour moment of each shot. By using video sequence classification, they can classify sequences as commercial or non-commercials.

Naturel et. al. [39] propose a method based on signatures generated from DCT of frames and hashing. First of all, shots are extracted from videos. For each shot a signature is calculated based on frames in that shot using DCT coefficients of frames. A hash table is created based on these signatures and used to find repeated sequences. For a query signature, all candidate shots are found in the hash table and a similarity value is calculated between candidate shot and query shot. Then sequence is detected based on this distance and a threshold value.

In [53], Zhao et. al. proposes a method to find near-duplicate keyframes based on local interest points (LIP) and PCA-SIFT. Local interest points makes proposed approach robust to affine transformations. Also by PCA-SIFT features vectors are less dimensional and this makes ranking more efficient. They used one-to-one symmetric (OOS) matching to find rankings. OOS matching rustle up the matching to be more effective since a LIP match can be a real match if it is the nearest neighbour in for both LIPs. They only find near duplicates and does not extend this approach to find sequences. In [40], Ngo et. al. proposes a very similar method again based on OOS matching, and LIP_IS indexing structure. Also they used transitivity property of near duplicate keyframes are used for effective detection. This transitivity property is a simple method as if A is similar to B and B is similar to C, then A is similar to C. Their method is based on thresholds and applied to single keyframes. This approach can be problematic while dealing with sequences in case of missing and edited frames.

LIP and PCA_SIFT approach and other features such as wavelet texture, colour moments are compared in [6] by Jiang et. al. They state that LIP based features are effective for semantic detection and video retrieval and complementary to traditional colour/textures features. Also their experiments give better results when combination of LIP, colour moment and wavelet features are used.

Sivic et. al. in [47], propose a method for object and scene retrieval, that finds all occurrences of a user outlined object in a video. They use affine covariant regions and SIFT descriptors to identify objects. A visual vocabulary is created using k-means. Term Frequency Inverse Term Frequency (tfidf) vector is calculated for user outlined objects by using a visual vocabulary. At retrieval stage, frames are ranked by using normalised scalar product of tfidf vectors.

Results of visual vocabulary based methods can change easily by using different approaches to create visual vocabulary. These changes are examined in depth [42]. They compare Bag-of-Features approach for classification. Classification result can change depending on sampling strategy for keypoint detectors, visual vocabulary size and method used to define images based on visual vocabularies. Although, they report that random sampling gives better results for their classification results, we use affine co-variant regions since they are more useful and effective in our case.

Visual vocabulary technique is very effective but it does not use colour information and spatial layout of features. Lazebnik et. al. [26] propose a method to recognise scene categories based on global geometric correspondence. They repeatedly subdivide the image and compute histograms of local features. This method is not robust against geometric changes since it compares histograms of by one-to-one correspondence and subdividing image avert features to be robust against geometric changes.

Graph based approaches are widely used for image matching. Jiang et. al. [18], propose a graph based method for image matching. They divide each image into parts and a create a bipartite graph among two images. Similarity measure between two parts of different image is calculated by histogram intersection. If this similarity value is greater than a threshold value, an edge is created between these two parts with a weight equal to similarity value. Graph based similarity value is the sum of weights of edges in graph. Also they used Gaussian-like function to take spatial information into account. Resulting similarity value is mean of these two similarity values. Their approach is not robust to zoom-in zoom-out or transformations since they used parts of images separately.

Another usage of graph based method is clip retrieval. Peng et. al. [45, 43] propose a method maximal and optimal matching in graph theory [51, 33, 30] for matching, ranking and retrieval of video clips. They used visual similarity, granularity, interference factor and temporal order of shots to find video clips. They try their approach on a dataset containing 190 minutes video clips and it is not clear how their approach behaves if there are missing or additional frames in video clips. Additional or missing frames can affect interference factor too much. This can result in ignoring similar video clips. Also Peng et. al [44] used same approach for audio clip retrieval. Instead of using visual similarity, acoustical similarity (MFCC) is used. Same deficiencies occurs for this approach too. Audio clips are so short and missing or additional parts in clips can be problematic in retrieval.

2.3 Story Tracking

Story talking is the problem of tracking same topic that is taken with different camera positions and/or in different time. There is a few number of studies on story tracking and all these studies are based on textual information. In this section, details of these studies are given.

Yang et. al. [52], proposed a method to detect near duplicate of text documents. They used instance level constrained clustering based on constraints such as, must-link, semi-link, family-link since Bag-of-words and fingerprint are not sufficient for near duplicates. SHA1 is used to find exact copies, all document is considered as a string and according to hash function similar documents are found. Copy that has more number of references is considered as a reference copy. In this approach, if a false reference exists in the database more than real reference then real reference can not be found.

Ide et. al. [14] propose a topic tracking method based on textual information extracted from news videos. According to sentences in text, similarities between news are found based cosine measure between the keyword vectors. Based on some threshold values, a hierchical tree is created. In this tree, relations and timestamps of news are considered. Topics can be tracked based on this hierchical tree.

2.3.1 Text Applications

In text applications, the similar problems (copy detection and plagiarism) exist. There are more studies on these subjects compared to copy detection, media tracking and story tracking studies on videos. In order to give a brief information about these studies, we will give studies on text applications.

Chowdhury et al. [2] proposes an algorithm, I-Match, to find a solution to text similarity. Instead of relying on strict parsing, I-Match makes use of collection statistics to identify which terms should be used for comparison. An inverse document frequency (idf) weight is determined for each term. The idf for each term is defined by a logarithmic function of N and n, where N is the number of documents in the collection and n is the number of documents containing the given term. The usefulness of terms for duplicate detection is found by using idf collection statistics. I-Match hinges on the premise that removal of very infrequent terms or very common terms results in good document representations for identifying duplicates.

Hoad et. al. [13] propose a method to identify documents that originate from same source by synthesing many variants of fingerprinting into a common framework. In the proposed method, ranking is extended by developing a new identity measure, and explore variants of the ngerprinting method. Their identity measure is based on the occurrences of similar words in documents. These occurrence numbers should be similar in similar documents. Documents with this property will have higher rankings while different number of word occurrences are penalised.

Chapter 3

Sequence Detection

Repeated sequence detection is a challenging issue that can be applied to copy detection, media tracking and story tracking. Most of the existing approaches on these approaches discard temporal order of videos, use near-duplicate frames and require a query.

In the following, details of sequence detection is given assuming that for each frame, the similar sets are provided as described in Section 4.3.

Unless otherwise stated, distances are based on chronological order of frames for following subsections. For example, if a frame is shown at time 10 and second one is shown at 20 then distance will be 10 seconds.

3.1 Problem Definition

Our goal is to find repeated sequences. We have a media and a part of this is repeated in the same media. We need to find that repeated part first then we can locate its repetitions. If we consider media as a string, each sub-string with different lenghts can be a repeated part candidate. So that, sub-string matching technique should be applied to all sub-strings to find their repetitions if exist.

Let's discuss how we can find these in a simple way. The simplest way is using a string matching technique. In string matching, we first compare the first character of query string with the first character of target string (Figure 3.1(a)). If they do not match, second character of target string is compared with first character of query string (Figure 3.1(b)). This comparison is done up to a match is found (Figure 3.1(c)). When this match is found, consecutive characters in query string and target string are compared (Figure 3.1(d) 3.1(e) 3.1(f)). If all these characters match, we can conclude that target string contains a match.

How can we adapt this solution to our problem? Assume that our media is a sequence of frames as $\{ f_1, f_2 \dots f_n\}$ and a subset of this media $S_i=\{f_i \dots f_{i+m}\}$ is its repetition. In a simple way, we can use a sub-string matching technique to find this repetition. At the beginning, we try to find a similar frame of f_i in the media. If f_k is similar to f_i then we check similarities of consecutive frames in the sequence and media. If frames $\{f_{i+1} \dots f_{i+m}\}$ are similar to the frames $\{f_{k+1} \dots f_{k+m-1}\}$ respectively, then we can conclude that $\{f_i \dots f_{i+m}\}$ is a repeated sequence of $\{f_k \dots f_{k+m}\}$. This is the simplest algorithm to find repeated sequence and summarised in Algortihm 1.

Algorithm 1 Rabin Karp string search algorithm [50].

```
procedure $NaiveSearch(str[1 \dots n], sub[l \dots l+m])$
1: for i from 1 to n do
2:     for j from 1 to m do
3:         if $str[i+j-1]$ is not equal to $sub[j]$ then
4:             jump to next iteration of outer loop
5:         end if
6:     end for
7:     return $i$
8: end for
9: return not found
```

In Algorithm 1, $str[1 \dots n]$ is the original string where sub-string is searched on and $sub[l \dots l+m]$ is sub-string to search. (m $\prec$ n)

Algorithm 1 can work for some cases but it is not efficient since first frame of S_i ($sub[1]$), is compared with almost all media. This significantly slows down the procedure. Another drawback of Algorithm 1 is that it needs a query part,

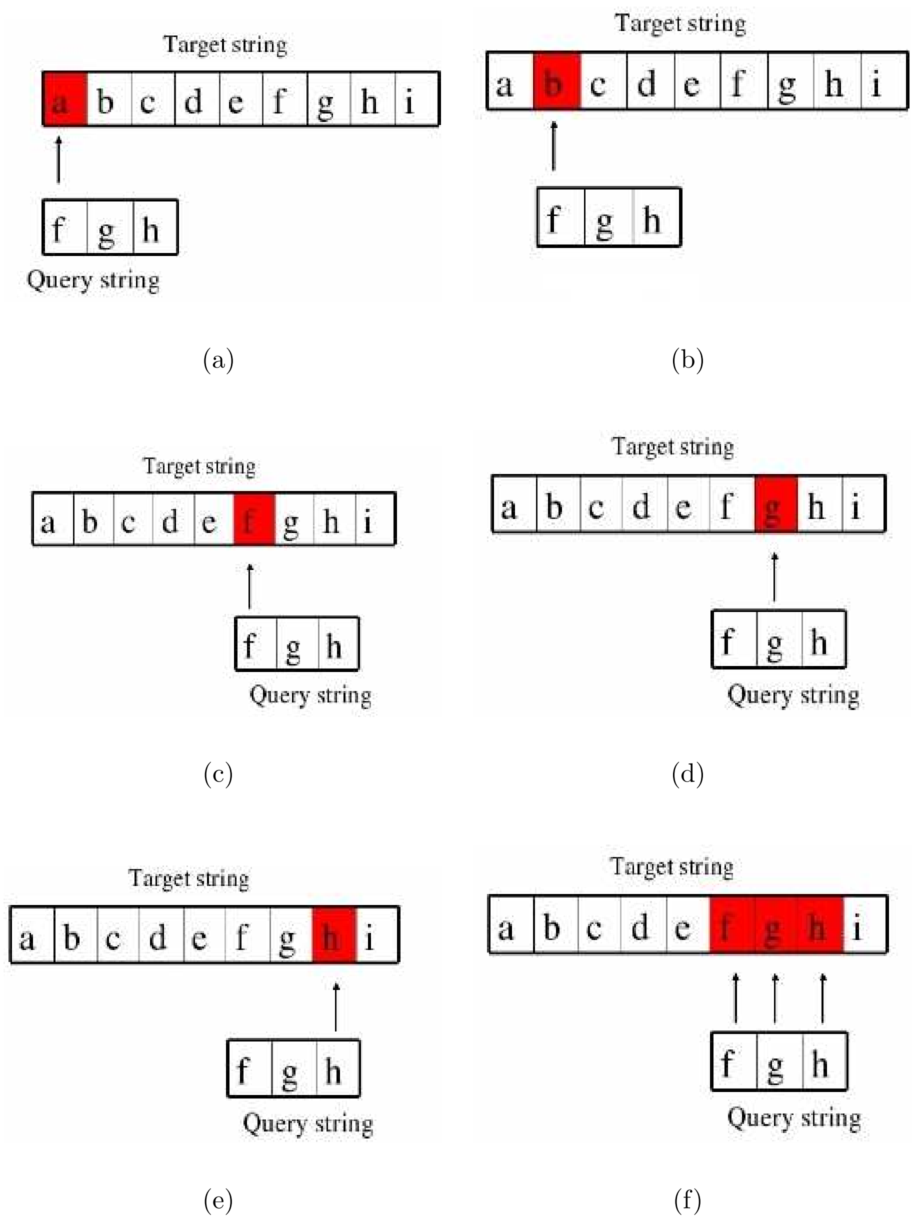


Figure 3.1: Substring matching algorithm demonstration.

$sub[l \ldots l+m]$. We want to propose a query free method so that sub-string part, S_i=$\{f_i \ldots f_{i+m}\}$ mentioned above, will be all subsets of real media with different lengths and different starting positions. Then if we want to apply string matching method without a query, Algorithm 1 should be extended to Algorithm 2.

Algorithm 2 A query free method to find repeating sequences by using string matching.

```
procedure $SequenceFinder(media = f_1 \ldots f_n)$
1: for $start$ from 1 to n do
2:     for $len$ from 1 to n-$start$ do
3:         $NaiveSearch(media[1, n], media[start, start + len])$
4:     end for
5: end for
```

In Algorithm 2, *media* is the media that is searched for repeating sequences. *start* and *end* are starting position and length of sub-string to be searched, respectively.

Algorithm 2 has three main drawbacks. The first one is that each frame should be considered as a starting position. Second one is that each length should be considered. These two drawbacks make string matching too complex to apply our problem. In our approach, we propose a solution to these problems by using a tree-based approach which codes both the similarity and order information to enhance simple string matching idea as seen in Figure 3.2. In our tree-based approach, a tree is constructed for each frame in the media. Each level in these trees is created by using similarity information and this limits number of starting positions. Also, levels from root to leaf node are created by using temporal orders and this enhance string matching algorithm by limiting number of possible lengths. The last drawback of string matching is that in string matching there is an exact definition for similarity. However, in our case, there is no exact similarity definition. We also propose a solution for similarity definition of frames. New string matching algorithm adapted to our problem is given in Figure 3.

In Algorithm 3, $f_{current}$ is the current frame in media to be considered as a start position of a sequence. $sf_{current}$ is the frames in similar list of $f_{current}$. $sf_{current+len}$ is the end position of sequence. Although Algorithm 3 has the same

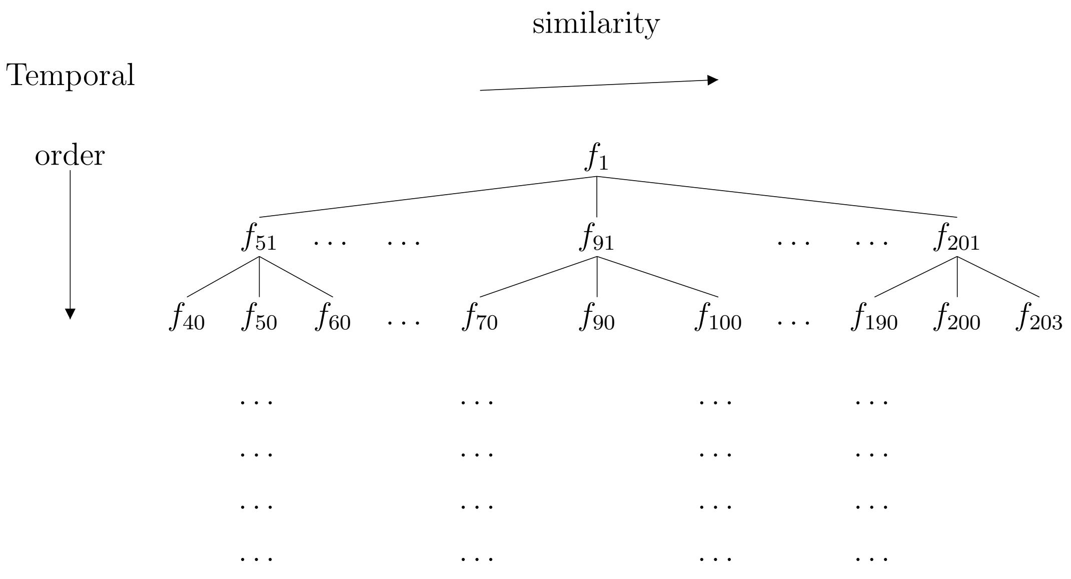


Figure 3.2: Tree structure created by our system.

complexity with Algorithm 2, second and fifth lines of Algorithm 3 reduces time complexity significantly. Second line reduce number of start positions and fifth line reduces possible substring lengths to search.

The proposed approach does not require any query sequence to be given, and finds all repeating sequences automatically. This is performed by building a separate tree for each frame f_i in the video to find the candidate repeating sequences for a sequence starting from frame f_i. By this way, we do not need to try all subsets of media as a repeated sequence candidate. In our tree based approach, each path from root to leaf is considered as a sequence candidate. If a frame does not belong to a repeating sequence, then no candidate sequences will be produced by the tree and the frame will be marked as a non-sequence frame. Otherwise, the candidate sequences will further be examined in the pruning step to check whether the sequence is also approved with the sequences produced by the neighbouring frames.

In the following, assuming that the similarity of frames are given, we will present our tree based approach to find candidate sequences and our pruning strategy to find final sequences.

Algorithm 3 Enhanced substring matching algorithm.

```
procedure $SequenceFinder(media = f_1 \dots f_n)$
1: for each frame $f_{current}$ in media do
2:    if $f_{current}$ has similar frames then
3:       for each similar frame $sf_{current}$ in similars($f_{current}$) do
4:          for $len$ from 1 to n-$sf_{current}$ do
5:             if $sf_{current+len}$ satisfy temporal order then
6:                $NaiveSearch(media[1, n], media[sf_{current}, sf_{current+len}])$
7:             else
8:                break
9:             end if
10:         end for
11:      end for
12:   end if
13: end for
```

3.2 Tree Creation

The main idea of our method is that the frames of a sequence are repeated with similar periods. That is, if i^{th} frame of a sequence repeats with period T then, $(i+1)^{th}$ frame of the sequence should also repeats with the same period T. That is, if a sequence is represented by a list of frames as $S_i = \{f_1, f_2, ..., f_n\}$, then the repetition of that sequence after T frames should be represented by a list of frames as $S_j = \{f_{T+1}, f_{T+2}, ..., f_{T+n}\}$. (Here f_i corresponds to the i^{th} frame in the video). This means that if there are T frames between the first frame of sequence S_i and first frame of sequence S_j, than a similar distance should appear for all the other frames of the sequences. We consider each frame in media as a possible starting position of a sequence. We place first frame of possible sequence to the root of tree. Then all similar frames of this frame are placed to first level. By this way, each similar frame creates a new branch. These branches are considered as sequence candidates at the end of tree creation.

For example, the two sequences given in Figure 3.3 are ideal case repeating sequences with T=100. There is no missing frames or order change. Assume we have only these sequences and all frames except frames in these sequences are black frames. In such a case, the similarity of frames will be as in Figure 3.4.

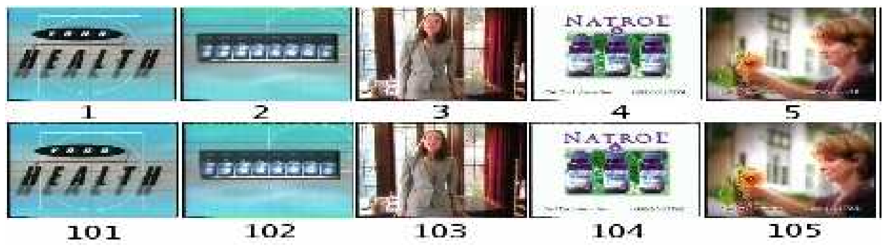


Figure 3.3: An example sequence from TRECVID dataset. First row represents real sequence S_i and second row represents repeating sequence S_j. Numeric values under frames are frames numbers in time domain.

Our tree based approach will create a separate tree for each frame in the list. Let's consider the creation of tree for frame f_1. Our tree based approach gets f_1 and its list of similar frames first, which is f_{101}. Then places f_1 to root and $f_{101}(f_1)$ ($f_{101}(f_1)$ corresponds to frame f_{101} coming from similar set of f_1) to first level, assuming that f_1 is first frame of S_i and f_{101} is the first frame of S_j. A tree branch is created as seen in Figure 3.5(a).

$$\begin{aligned} f_1 &\rightarrow f_{101} \\ f_2 &\rightarrow f_{102} \\ f_3 &\rightarrow f_{103} \\ f_4 &\rightarrow f_{104} \\ f_5 &\rightarrow f_{105} \end{aligned}$$

Figure 3.4: Ranking sample for a simple set given in Figure 3.3.

Then list of similar frames for the next frame, f_2, is taken. Our tree based approach assumes that f_2 is second frame of S_i and second frame of S_j must be present in similar set of f_2. In this case, f_2 has one similar frame, f_{102}, so that $f_{102}(f_2)$ is placed to tree as seen in Figure 3.5(b).

By the same assumption, third frame of S_i, f_3, must be similar to third frame of S_j, f_{103}. In our case, this assumption holds and f_{103} is in the similar set for f_3. So that, $f_{103}(f_3)$ is placed as a child to $f_{102}(f2)$ and resulting tree can be seen in Figure 3.5(c).

Tree is basically created by inserting rankings of f_{i+n} to the level n of tree of f_i and a resulting tree is created from rankings given in Figure 3.5(d).

$$\begin{tikzcd}
f_1 \arrow[d, dash] \\
f_{101}(f_1)
\end{tikzcd}$$

(a)

$$\begin{tikzcd}
f_1 \arrow[d, dash] \\
f_{101}(f_1) \arrow[d, dash] \\
f_{102}(f_2)
\end{tikzcd}$$

(b)

$$\begin{tikzcd}
f_1 \arrow[d, dash] \\
f_{101}(f_1) \arrow[d, dash] \\
f_{102}(f_2) \arrow[d, dash] \\
f_{103}(f_3)
\end{tikzcd}$$

(c)

$$\begin{tikzcd}
f_1 \arrow[d, dash] \\
f_{101}(f_1) \arrow[d, dash] \\
f_{102}(f_2) \arrow[d, dash] \\
f_{103}(f_3) \arrow[d, dash] \\
f_{104}(f_4) \arrow[d, dash] \\
f_{105}(f_5)
\end{tikzcd}$$

(d)

Figure 3.5: Tree construction examples based on similar frames given in Figure 3.4 ($f_{101}(f_1)$ corresponds to frame f_{101} coming from similar set of f_1).

If we sum up, resulting algorithm can be given as in Algortihm 4.

Algorithm 4 Algorithm for creating trees.

```
procedure $TreeCreation()$
1: for each frame $f_i$ in media do
2:     $tree.setRoot(f_i)$
3:     $f_{current} \leftarrow f_i$
4:     while stop criteria is not satisfied do
5:         $f_{current} \leftarrow getNextFrameOf(f_{current})$
6:         for each similar frame $sf_{current}$ in similars($f_{current}$) do
7:             $addToSuitableParents(sf_{current}, root)$
8:         end for
9:     end while
10: end for
```

In Algorithm 4, $f_{current}$ is the current frame, similars($f_{current}$) is the similar frames for a frame $f_{current}$. $tree$ corresponds to tree that is created for current frame f_i. $sf_{current}$s are the frames in the similar set of $f_{current}$.

A separate tree is needed to be created for each frame f_i to check consistencies of sequences and also to make our approach query free since a repeated sequence can start at any position. At the beginning, f_i (frame for which tree is created) is placed to root. Then similar frames for f_i is taken. In our approach, addition to root and other nodes are considered different. Only frames that are similar to root frame (f_i in this case) are added to first level (children of root). But in inner nodes, a new child can be added to any place if constraints, described in following sections, are satisfied. So that, similar frames of f_i are inserted as first level by $tree.createFirstLevel()$.

Root of tree and first level are created. After these, tree is created by inserting new frames coming from similar sets of consecutive frames of root frame, f_i. In the first step, similar set of next frame of f_i is taken. Each frame in similar set is considered as a new node candidate and current tree is traversed to find a parent (or parents) for this new node candidate. By the same way, similar frames of f_{i+n} are tried to be placed to the n^{th} level of the tree.

Up to now, we have considered that sequences are repeated with exact periods, T, and next frame for current frame f_i is considered as f_{i+1}. However, in real cases, sequences are not repeated with exact periods because of broadcast, shot boundary detection method or editing. That is why, vicinage should be considered while taking next frame of current frame. Reasons of and solution to this problem will be discussed in the next section.

3.2.1 Period Constraint

One of the problems in sequence detection is related with sequence lengths and orders of frames in sequences. An example of this problem can be seen in Figure 3.6. In this example, frames are different in original sequence and its repetition because of shot boundary detection method and/or editing. In order to overcome this problem, we modify our simplest tree based approach given in Algorithm 4 as will be explained in the following.

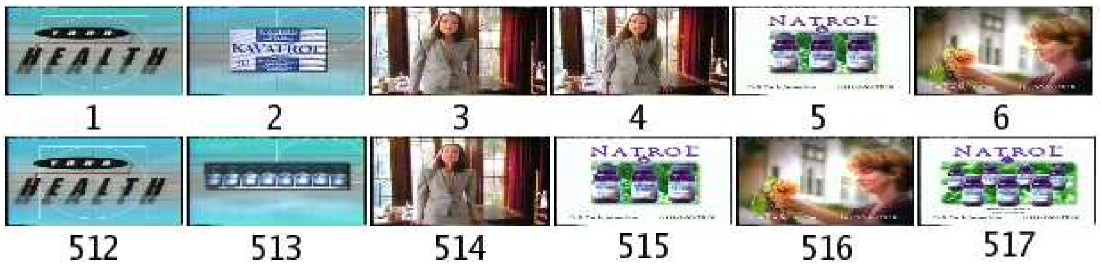


Figure 3.6: An example of unequal **Period Constraint** from TRECVID dataset. First row represents real sequence and second row represents repeating sequence. Numbers below frames are frames numbers in time domain. Although first frame in first row satisfies **Period Constraint** with 511 frames, last frame in the same row satisfies with 510 frames.

Time period, T, mentioned before, is valid for perfect cases. However, due to missing or additional frames, and since the order of frames may slightly change from one sequence to another, the strict period T between frames of S_i and S_j is not satisfied in real situations. Instead, we modify the constraint by adding a neighbourhood information. We assume that two similar frames could be the corresponding frames in the repeating sequences if they are placed with distances $T \pm \delta$ where δ is a small number (see Figure 3.6). We call this **Period Constraint**. This constraint mainly states that frames of S_j must be repeated with a difference of $T \pm \delta$ with the corresponding frames in S_i.

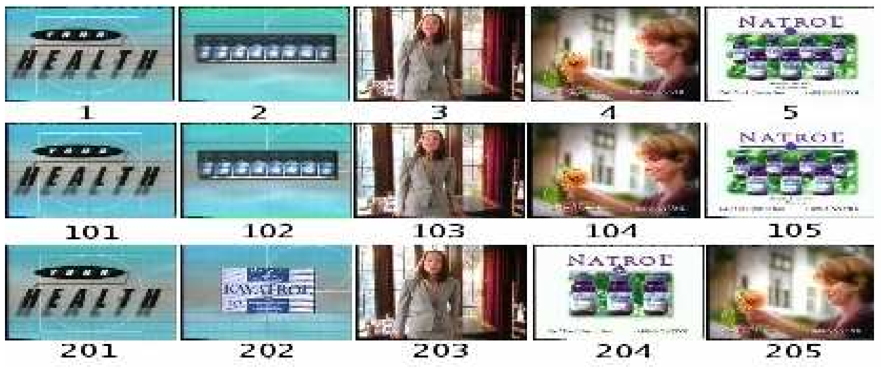


Figure 3.7: An example sequence which repetitions of sequences are different from TRECVID dataset. First row represents real sequence and second and third row represent repeating sequences. Numbers below frames are frames numbers in time domain.

Let's again work on an example. In Figure 3.7, there is one more sequence S_k in addition to sequences S_i and S_j in Figure 3.3. We can see severity of **Period Constraint** in these sequences. Most probably rankings for these 15 frames will be like in Figure 3.8.

$$\begin{array}{l} f_1 \rightarrow f_{101}, f_{201} \\ f_2 \rightarrow f_{102}, f_{202} \\ f_3 \rightarrow f_{103}, f_{203} \\ f_4 \rightarrow f_{104}, f_{205} \\ f_5 \rightarrow f_{105}, f_{204} \end{array}$$

Figure 3.8: Ranking sample for a simple set given in Figure 3.7.

For all three sequences, repetition period, T, is same up to fourth frame. So that tree is created in same way by one difference, this time our tree has two branches. We obtain a tree as seen in Figure 3.9(a).

However, when we check similar frames of f_4, we see that repetition period, T, is not valid for f_{205}. But as stated before in this section, we extend T value to $T \pm \delta$ to tolerate this kind of inconsistencies. The most important criteria to add a new node to tree is to find a node that satisfy **Period Constraint**. In current tree, node $f_{203}(f_3)$ satisfy constraint for f_{205} and $f_{205}(f_4)$ is added to tree. After that $f_{204}(f_5)$ is added in same way and resulting tree becomes as seen in Figure 3.9(b).

$$\begin{tikzcd}
f_1 \arrow[d, dash] \arrow[dr, dash] & \\
f_{101}(f_1) \arrow[d, dash] & f_{201}(f_1) \arrow[d, dash] \\
f_{102}(f_2) \arrow[d, dash] & f_{202}(f_2) \arrow[d, dash] \\
f_{103}(f_3) & f_{203}(f_3)
\end{tikzcd}$$

(a)

$$\begin{tikzcd}
f_1 \arrow[d, dash] \arrow[dr, dash] & \\
f_{101}(f_1) \arrow[d, dash] & f_{201}(f_1) \arrow[d, dash] \\
f_{102}(f_2) \arrow[d, dash] & f_{202}(f_2) \arrow[d, dash] \\
f_{103}(f_3) \arrow[d, dash] & f_{203}(f_3) \arrow[d, dash] \\
f_{104}(f_4) \arrow[d, dash] & f_{205}(f_4) \arrow[d, dash] \\
f_{105}(f_5) & f_{204}(f_5)
\end{tikzcd}$$

(b)

Figure 3.9: Tree construction examples using **Period Constraint**.

Period Constraint is checked in *addToSuitableParents* (line 7 of Algorithm 4). This procedure searches all current tree and new child candidate is added to parents that satisfy **Period Constraint**. Algorithm for this procedure is given in Algorithm 5.

Algorithm 5 Algorithm to find suitable parents for a new child candidate.

```
procedure $addToSuitableParents(sf_{current}, curRoot)$
1: $dist = getDistance(curRoot, sf_{current})$
2: if $dist$ is in range $[\pm\delta]$ then
3:     $candidateparent$
4: end if
5: if $curRoot$ has children then
6:     for each $child_i$ of $curRoot$ do
7:         $findSuitableParOf(sf_{current}, child_i)$
8:     end for
9: end if
```

In Algorithm 5, *root* is root of tree, *curRoot* is current root while traversing tree. *dist* corresponds to time distance between *curRoot*'s frame and $sf_{current}$.

addToSuitableParents is a recursive algorithm that traverse all tree to find suitable parents for $sf_{current}$. For each node in the current tree, distance is calculated between new child and current root node. If this distance value satisfies **Period Constraint**, current root is considered as a candidate parent. A new child node can be added to several places in the tree if they all satisfy **Period Constraint**.

3.2.2 Self-Similarity Constraint

Period Constraint allows our tree-based approach to be flexible while placing new nodes to tree by a period T. However, in some cases, frames in same sequence can be similar to each other, as seen in Figure 3.10. This kind of self similarities can result in dividing sequences into several parts. For this reason, we define another constraint called **Self-Similarity Constraint**.

Self-Similarity Constraint is simply related with chronological distances between the frame for which the tree is created, f_i and similar frame f_j. If this distance is smaller than a value γ, then this will be used as a clue to show that similar frame f_j is also in the same sequence. This is a crucial step to prevent creating false sequences since there can be some sequences which frames in a sequence are similar to each other. For example, some commercials can have similar frames in the sequence as seen in Figure 3.10. If this constraint is not satisfied, a real sequence can be divided into several repeated sequences according to its length.

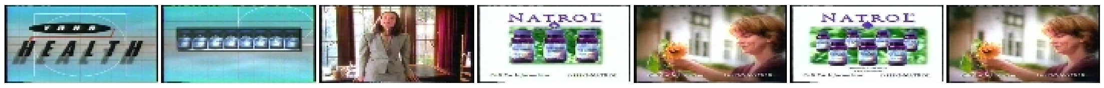


Figure 3.10: An example for **Position Constraint** from TRECVID dataset. Frames in sequence are very similar to each other so that frames belonging to this sequence will have higher ranking for each other and sequence will be divided to several parts if **Position Constraint** is not regarded.

In Figure 3.10, it is obvious that 4^{th} frame is similar to 6^{th} frame and 5^{th} frame is similar to 7^{th} frame in the sequence. So that, they exist in their similar sets interchangeably. If we consider those frames as similar frames, we end up divided sequences. Our tree-based approach overcomes this problem by **Self-Similarity Constraint**.

TreeCreation routine (Algorithm 4)is modified to consider **Self-Similarity Constraint** as in Algorithm 6. At this routine, distance between new child candidate and root frame is calculated. If this distance does not satisfy **Self-Similarity Constraint**, tree is not traversed to find suitable parents for new child candidate.

In Algorithm 6, *dist* is the distance between current root and $sf_{current}$. Other parameters are same with Algorithm 4.

Algorithm 6 differs from Algortihm 4 in lines [7,8]. This part checks **Self Similarity Constraint**. At the beginning of traversal, distance between $sf_{current}$ and root of tree is calculated. If this distance is in the range of $\pm\gamma$, then new child candidate is considered in the same sequence with root frame. As a result of this, tree is not traversed to place $sf_{current}$ to the current tree.

Algorithm 6 Algorithm for creating trees.

```
procedure $TreeCreation()$
 1: for each frame $f_i$ in media do
 2:     $tree.setRoot(f_i)$
 3:     $f_{current} \leftarrow f_i$
 4:     while stop criteria is not satisfied do
 5:         $f_{current} \leftarrow getNextFrameOf(f_{current})$
 6:         for each similar frame $sf_{current}$ in similars($f_{current}$) do
 7:             $dist = getDistance(root, sf_{current})$
 8:             if $dist$ is not in the range $[-\gamma, \gamma]$ then
 9:                 $addToSuitableParents(sf_{current}, root)$
10:             end if
11:         end for
12:     end while
13: end for
```

3.2.3 Closest Parent Constraint

Up to now, we eliminate some similar frames by **Self-Similarity Constraint** and find suitable nodes for new child by traversing current tree under **Period Constraint**. However, there can be several nodes that satisfy **Period Constraint**. If new child is added to all these nodes, tree starts to grow exponentially after a certain point and it is almost impossible to handle such a tree. So that, some parent candidates should be eliminated although they satisfy **Period Constraint**.

While placing new child to tree, we mainly consider not to break sequences and also we consider tree size, so that new child should be added to the most closest parent(s). For example; we have a tree given in Figure 3.11(a) and new child candidate is f_{104}. If current tree (Figure 3.11(a)) is traversed under previously defined constraints, $f_{107}(f_2)$ $f_{101}(f_1)$ and $f_{99}(f_1)$ are considered as new parents. However, distance between $f_{99}(f_1)$ and $f_{104}(f_3)$ is greater than others. So that, $f_{104}(f_3)$ should be added to $f_{101}(f_1)$ and $f_{107}(f_2)$ to satisfy sequence integrity. As a result of this, new tree becomes as seen in Figure 3.11(b).

As a next step suppose that we need to add $f_{103}(f_4)$. Nodes $f_{101}(f_1)$, $f_{107}(f_2)$ and both of $f_{104}(f_3)$s satisfy **Period Constraint** in current tree. In our approach, a node can be added to many places but if a new node is added to many places by

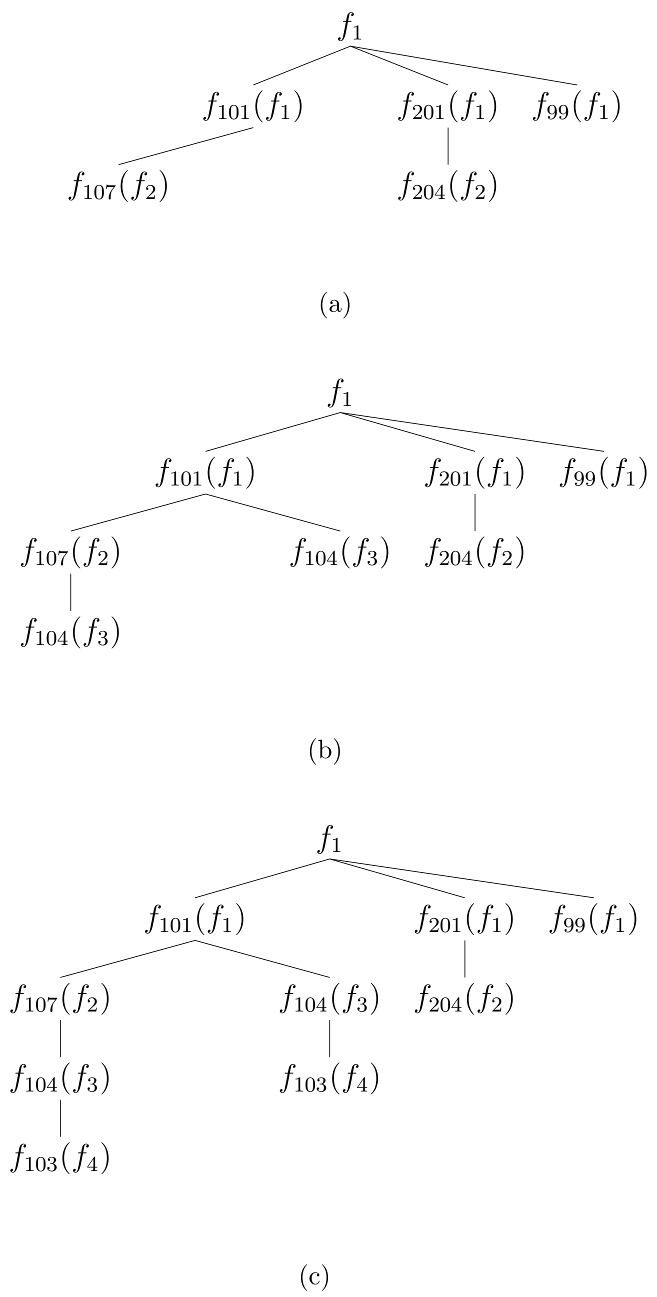


Figure 3.11: Tree construction examples for **Closest-Parent Constraint**.

just **Period Constraint** then tree start to grow exponentially and it is impossible to handle. In above tree, $f_{103}(f_4)$ must be only added to nodes $f_{104}(f_3)$ to satisfy sequence integrity under another constraint **Closest Parent Constraint**.

We need to consider to reduce gaps in sequences while adding new nodes to tree and also sequences should not be divided into pieces and integrity should be concerned. **Closest Parent Constraint** mainly deals with finding closest parent to a new child in terms of time distance. There is no threshold for this constraint, instead of that, we search for the closest parent.

If the above three constraints are satisfied for any node in the tree, we consider this node as a parent node. By this approach, we allow to add new nodes to multiple positions. Also this multiple addition helps our method to overcome missing or edited frames in sequences.

The $addToSuitableParents$ routine is modified to check the **Closest Parent Constraint** as given in Algorithm 7.

Algorithm 7 Algorithm to find suitable parents for a new child candidate with all constraints.

```
procedure $addToSuitableParents(sf_{current}, curRoot)$
1:  $dist = getDistance(curRoot, sf_{current})$
2:  if $dist$ is in range $[\pm\delta]$ then
3:      if $dist == minDist$ then
4:          $additional\ candidate\ parent$
5:      else if $dist < minDist$ then
6:          $minDist \leftarrow dist$
7:          $new\ candidate\ parent$
8:      end if
9:  end if
10: if $curRoot$ has children then
11:     for each $child_i$ of $curRoot$ do
12:         $findSuitableParOf(sf_{current}, child_i)$
13:     end for
14: end if
```

Note that, in addition to variables used in In Algorithm 5, $curRoot$ is current root while traversing tree. $root$ is root of current tree. $dist$ is the distance between current root and new child candidate.

minDist stores smallest distance between new child candidate and all nodes in tree.

Closest Parent Candidate is checked in lines [10,16] of Algorithm 7. At the beginning, distance between current root and new child candidate is compared with *minDist*. If they are equal then we conclude that current root is a parent candidate in addition to previously found ones. If *dist* is smaller than *minDist* then previously found parent candidates are discarded and current root is considered the only parent candidate since current root is closer than previously found ones. This helps to satisfy integrity of sequences in addition to reducing number of sequence candidates produced by trees.

3.2.4 Stopping Criteria

If we sum up all the steps described above, the tree is constructed for frame f_i as follows. First, the frame f_i is placed as the root node (level 0). The nodes in the first level corresponds to the frames which are similar to frame f_i. Then, for each node in the first level, the corresponding children nodes are constructed in the second level, for the frames which are similar to the $(i+2)^{th}$ frame and appearing within a distance δ from their parents. In general, the $(n+1)^{th}$ level of the tree is constructed such that the nodes in that level corresponds to the frames which are similar to the $(i+n)^{th}$ frame in the sequence and placed with distance δ from the frames in their parent nodes. Each path from root to leaf node is considered as a sequence candidate.

In some level n, we may not be able to insert nodes to some paths since the constraints are not satisfied. However, that does not disallow adding new nodes in the next levels. This approach is important for dealing with missing frames. Here the choice of δ is important since large values corresponds to allowing large gaps which usually does not happen in sequences, and small numbers cannot deal with small number of missing frames.

This tree based approach described above has two problems. First of all, if

$f_1 \rightarrow f_{101}, f_{266}, f_{201}, f_{250}, f_{301}, f_{55}, f_{104}$
$f_2 \rightarrow f_{102}, f_{202}, f_{302}, f_{56}, f_{104}, f_{252}, f_{58}$
$f_3 \rightarrow f_{204}, f_{303}, f_{254}, f_{264}, f_{252}$
$f_4 \rightarrow f_{262}, f_{90}, f_{278}, f_{304}, f_{206}, f_{104}$
$f_5 \rightarrow f_{15}, f_{105}, f_{205}, f_{76}, f_{305}, f_{58}$
$f_6 \rightarrow f_{15}, f_{207}, f_{306}, f_{120}, f_{600}$
$f_7 \rightarrow f_{307}, f_{25}, f_{107}, f_{400}, f_{209}, f_{352}$

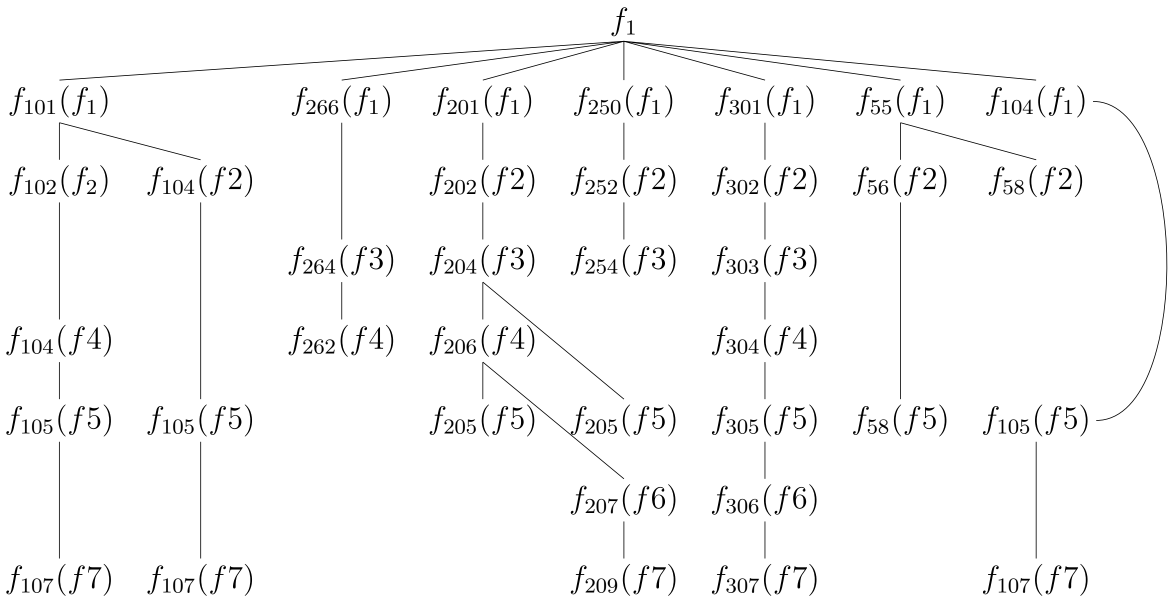


Figure 3.12: An example of tree creation. Above lines give ranking for consecutive 7 frames. In the tree representation, subscripts represents from which each image is coming from.

the video has N frames then each frame will have N-1 similar images, listing all the similar frames causes a huge tree which is impossible to handle. Even the **Period Constraint** not sufficient to reduce the number of nodes, since for each node it will limit the number of children nodes with only 2δ. In Section 4.3 we discuss methods to limit the number of similar images differently for each frame.

The second problem is that, in the current form, there is no condition to stop the tree for growing and therefore for each path in the order of $N - i$ frames should be investigated to be added for the i^{th} frame in the video. However, note that as mentioned above, for some levels it is possible not to add any node to some paths since the similar frames do not satisfy the distance constraint. We use this fact and stop investigating the paths if for consecutive σ levels it is not possible to add any new node to those paths. In the experiments we choose σ as testing different values for detests.

The above steps are applied to each frame in the video, and for each frame the paths with lengths more than η are selected as candidate sequences. These candidate sequences are further pruned to see whether they are consistent with the candidate sequences found for the neighbouring frames.

The approach is simulated on an example given in Figure 3.12. Here, assuming that we have a sequence including the frames from f_1 to f_7, we would like to find the candidate repeating sequences. The figure shows the tree construction for frame f1. We assume that, it has a list of similar frames which are f_{101}, f_{266}, f_{201}, f_{301}, f_{250}, f_{55} and f_{104} in the ranked order. In the first level, these similar frames to f_1 are placed. Then, in the second level, we insert the similar frames of f_2 which is the second frame of original sequence. Similar frames of f_2 are f_{102}, f_{202}, f_{302} f_{56}, f_{104}, f_{252} and f_{58}. Obeying the **Period Constraint**, these frames can only be added as the children to the nodes in the first level if they are in a δ neighbourhood. Here, if we choose δ as 7, all similar frames of f_2 passed **Period Constraint** test and one suitable parent is found for each similar frame. This means that all constraints (**Self-Similarity Constraint**, **Period Constraint** and **Closest Parent Constraint**) are satisfied. 2 child nodes are added to $f_{101}(f_1)$ and $f_{55}(f_1)$. These 2 child adding process produces 2 branches for $f_{101}(f_1)$ and $f_{55}(f_1)$. In the next level, we consider rankings for f_3. All ranking frames passes **Self-Similarity Constraint** and $f_{264}(f_3)$, $f_{204}(f_3)$, $f_{303}(f_3)$, and $f_{254}(f_3)$ are bind to tree since **Period Constraint** is satisfied and **Closest Parent Constraint** is passed by only one node for each frame. In this level, $f_{252}(f_3)$ is not added since it is already added by previous frame in the sequence. Addition of similar frames of f_4 is same as f_3. In this step, $f_{104}(f_4)$ is added by one missing level. This helps our system to tolerate missing frames in sequences. $f_{105}(f_5)$ and $f_{205}(f_5)$ are added to multiple positions since both positions satisfy **Closest Parent Constraint** with same distance. Same as f4, $f_{105}(f_5)$ is added as a second level to $f_{104}(f_1)$. When ranking for f_6 is traced, f_{600} and f_{120} can not satisfy **Period Constraint** for current tree. Also f_{15} can

not satisfy **Self-Similarity Constraint**. So that only $f_{207}(f_6)$ and $f_{306}(f6)$ are added to tree by satisfying all tree constraints. $f_{207}(f_6)$ is added to an inner node, this means that most probably $f_{205}(f_5)$ is a false ranking for f_5 or sequence order is changed in repetition. For the last frame f_7 of sequence, f_25 can not be added because of **Self-Similarity Constraint** and f_{400} and f_{352} can not be added to tree since they can not pass **Period Constraint**. As a result of this tree creation simulation, following sequence candidates are found :
f_{101}, f_{102}, f_{104}, f_{105}, f_{107}
f_{101}, f_{104}, f_{105}, f_{f107}
f_{266}, f_{264}, f_{262}
f_{201}, f_{202}, f_{204}, f_{205}
f_{201}, f_{202}, f_{204}, f_{206}, f_{205}
f_{201}, f_{202}, f_{204}, f_{206}, f_{207}, f_{209}
f_{301}, f_{302}, f_{303}, f_{304}, f_{305}, f_{306}, f_{307}
f_{250}, f_{252}, f_{254}
f_{55}, f_{56}, f_{58}
f_{104}, f_{105}, f_{107}
Since we do not consider sequences shorter than η (η=2) as a sequence [f_{55}, f_{58}] is not a sequence candidate in our system. Real sequences are detected after pruning step if any exist. If a real sequence is found by pruning step, sequence from f_1 to f_7 is considered as an original sequence since we start with f_1 to create our tree and last frame that adds new node to our tree is f_7.

3.3 Pruning Sequences

We consider each path from root to leaf as a candidate sequence. However, in the set of candidate sequences, there are also many false alarms which are needed to be eliminated.

There are two types of false alarms. First type is the ones which are actually sub-sequences of a longer sequence. This type of false alarms occur since for a sequence with length L, there are L-2 sub-sequences with starting and ending

$$\begin{gathered}
f_1 \rightarrow f_{101}, f_{102}, f_{103}, f_{104}, f_{105} \\
f_1 \rightarrow f_{251}, f_{252}, f_{253}, f_{254}, f_{255} \\
f_2 \rightarrow f_{102}, f_{103}, f_{104}, f_{105} \\
f_3 \rightarrow f_{103}, f_{104}, f_{105}
\end{gathered}$$

Figure 3.13: Tree results for consecutive 3 frames.

frames $[i, (i + L)], [(i + 1), (i + L)], \ldots, [(i + L - 2), (i + L)]$ if we let all sequences with length greater than 3 to be candidate sequences.

Second type of false alarms are the ones that are not actually sequences but decided as sequences. Since our definition of repeating sequences requires similarity of consecutive frames, because of the insufficiency of the feature representations, two sequences may be very similar when the visual features are considered but actually may not even be sequences by themselves.

These two type of false alarms require different solutions. For the first type, we track sequence candidates for consecutive frames and try to find sequence's actual starting and ending positions. If candidate sequence is not repeated inside the other sequences found for the neighbouring frames, then that candidate sequence is labelled as a false alarm. Among the candidate sequences which repeat in the other sub-sequences the longest one with the farthest starting and ending points are taken as the final sequence, and the others are eliminated.

In Figure 3.13, there are tree results for consecutive 3 frames. Two sequence candidates are created for f_1 and one sequence candidates for f_2 and f_3. From this example, we can see that sub-sequences of first sequence candidate, $\{f_{101}, \ldots, f_{105}\}$, is repeated for f_2 and f_3. So that, it is concluded that $\{f_{101}, \ldots, f_{105}\}$ is not a first type of false alarm given above. But second sequence candidate, $\{f_{251}, \ldots, f_{255}\}$, is not repeated for other frames. That is why, it is considered as a first type false alarm and removed from sequence candidate list.

To eliminate the second type of false alarms, which are more commonly encountered, we apply a **one-to-one match** constraint. We require that two sequences S_i and S_j to be repeated sequences, both S_j should be found in the candidate list of S_i, and also S_i should be found in the candidate list of S_j. Note

$$\begin{array}{l}\{f_1, f_5\}, \{f_{101}, f_{105}\} \\ \{f_{101}, f_{105}\}, \{f_1, f_5\} \\ \{f_{201}, f_{205}\}, \{f_1, f_5\}\end{array}$$

Figure 3.14: Example sequence candidates coming from tree creation where first type of false alarms are eliminated. First double represent start and end position of original sequences. Second double represent start and end position of repeating sequences.

that, since the similar sets are different for each frame, in the case of false alarms it is unlikely to have the candidate sequences in both direction to be constructed. As expected, one-to-one constraint largely reduces the number of false alarms.

There are 3 sequences doubles (original sequence and its repetition) in Figure 3.14. When we check first row, there are two sequences S_i (original sequence) that starts at f_1 and ends at f_5 and its repetition S_j that starts at f_{101} and ends at f_{105}. In this case, according to our one-to-one match approach, we expect to find another sequence double which original sequence is S_j and its repetition is S_i as second row of Figure 3.14. We conclude that first two rows are correct sequences. When last row, $\{f_{201}, f_{205}\}, \{f_1, f_5\}$, is checked, we see that its repetition, $\{f_1, f_5\}$, does not have a repetition as $\{f_{201}, f_{205}\}$. So that, $\{f_{201}, f_{205}\}, \{f_1, f_5\}$ is decided as a second type false alarm.

False alarms need different solutions. We propose a two pass algorithm to eliminate first type of false alarms. In the first pass, we get all sequence candidates coming from *treeCreation* process and find consistent sequences, that are repetitive for consecutive frames. First pass is done by Algortihm 8.

In Algorithm 8, *AllTrees* corresponds to trees created by *createTree* routine. $tree_i$ is element of *AllTrees* and $sequenceCandidate_j$ is a path from leaf to node in one tree. $sequenceCandidate_j{+}k$ represents sub-sequence of $sequenceCandidate_j$ starting at k.

SequenceFinder routine gets all trees created by *createTree* routine. Then for each tree, each *sequenceCandidate* is checked whether it exists in trees of consecutive frames. A consistent sequence must occur in all these trees with some

Algorithm 8 Algorithm for finding sequence candidates from created trees.

```
procedure $SequenceFinder()$
1: $AllTrees$ = all trees are created by createTree routine
2: for each $tree_i$ $AllTree$ do
3:    for each $sequenceCandidate_j$ in $tree_i$ do
4:       for $k$ in range $[1, length(sequenceCandidate_j)]$ do
5:          if $sequenceCandidate_j + k$ does not exist in $tree_{i+k}$ then
6:             break;
7:          end if
8:       end for
9:       if $k < length(sequenceCandidate_j)$ then
10:         $not\_a\_sequence\_candidate$
11:      else
12:         $a\_sequence\_candidate$
13:      end if
14:   end for
15: end for
```

frame order change. Our assumption on this point is that, we have sequence S_i starting at frame f_i and repeated at f_j as S_j. S_j must occur in tree created for f_i and sub-sequence of S_j must exist in tree created for f_{i+1}. By this assumption, we eliminate sequence candidates as follows. For example, if $tree_i$ contains a sequence candidate $sequenceCandidate_j$, then $tree_{i+k}$ must contain a sequence candidate $sequenceCandidate_j + k$ that is a sub-sequence of $sequenceCandidate_j$. However, because of missing or additional frames, it is possible that $tree_{i+k}$ does not contain any sub-sequence of $sequenceCandidate_j$. In this case, trees up to $tree_{i+k+\beta}$ are searched for same sub-sequence of $sequenceCandidate_j$, $sequenceCandidate_j + k$. β value is limited by sequence length since in long sequences can have more missing frames than shorter sequences.

A real sequence can exist with different orders in sequence candidates since in our tree approach, a node can be added to multiple positions. This results in multiple repetitions of same sequence at the end of *SequenceFinder* routine with different orders. In fact, this routine eliminates first type of false alarm. Actual start and end location of sequences are found by analysing these repeated sequences.

Up to this end, we eliminate first kind of false alarms. However we have 2 types of false alarms. Also sequences resulting from Algorithm 8 can be divided versions of real sequences. For these reasons $PruneSequences$ has mainly two functionalities :

- Merging divided sequences
- Eliminating second type of false alarms

Merging divided sequences : Real long sequences can be divided into smaller parts because of missing frames or insufficient ranking results. These are not considered as false alarms by our system but we combine these divided sequences into one long sequence. First of all, we get all sequences coming from $SequenceFinder$ routine. For each sequence $sequence_i$ we compare start and end position of other sequences to find any sequence that has an intersection with $sequence_i$ in time domain. If we find a $sequence_j$ that has an intersection with $sequence_i$ than we update start and end time of $sequence_i$ and remove $sequence_j$ from sequence list. As a result of this, we obtain merged sequences.

Eliminating second type of false alarms : As stated in Section 3.3, we have two types false alarms.After merging divided sequences, we can eliminate second type of false alarms. Second type is eliminated by one-to-one match in $PruneSequences$ routine. For each sequence, we have an original start $stat$ and end position $orEnd$ and a repetition start $repStart$ and end $repEnd$ position. If a sequence exists starting at $orStart_x$ and repeated at $repStart_y$ then another sequence must be found starting at $orStart_y$ and repeated at $repStart_x$. By this assumption we search all sequences for each sequence and try to find a match. If a match sequence $sequence_j$ is found for $sequence_i$ then we conclude that $sequence_i$ is a real sequence. If no match exists $sequence_i$ is considered as a false alarm.

Above two functionalities are performed by $PruneSequences$ algorithm given in Algorithm 9.

In Algorithm 9, $AllSequences$ corresponds to sequences coming from $SequenceFinder$ routine. $sequence_i$ and $sequence_j$ are sequences in $AllSequence$.

Algorithm 9 Algorithm for pruning sequences.

```
procedure $PruneSequences$
 1: $AllSequences$ = all sequences coming from $SequenceFinder$ routine
 2: for each $sequence_i$ in $AllSequences$ do
 3:   for each $sequence_j$ in $AllSequences$ do
 4:     if intersectionof($sequence_i$, $sequence_j$) $\geq 0$ then
 5:       update $sequence_i$
 6:       remove $sequence_j$
 7:     end if
 8:   end for
 9: end for
10: for each $sequence_i$ in $AllSequences$ do
11:   for each $sequence_j$ in $AllSequences$ do
12:     if $sequence_{j_{repStart}}$ is in the range [$sequence_{i_{orStart}} \pm \beta$] AND
        $sequence_{j_{repEnd}}$ is in the range [$sequence_{i_{orEnd}} \pm \beta$] then
13:       $a\_real\_sequence$
14:     end if
15:   end for
16:   if No Match for $sequence_i$ then
17:     $not\_a\_real\_sequence$
18:   end if
19: end for
```

Subscripts $orStart$, $orEnd$, $repStart$ and $repEnd$ are start and end position of original sequence and repetitions, respectively.

$PruneSequences$ algorithm gets output of $SequenceFinder$ routine and firstly, it merge divided sequence lines between 2 and 9. This is basically done by comparing sequences intersections in time domain. Then these merged sequences are further analysed for eliminating second type of false alarms (one-to-one check) lines between 10 and 19. As a result of this algorithm, we obtain repeated sequences for given media.

Chapter 4

Frame Representations

Up to now, we assumed that the similar frames for a frame f_i is given. However, definition of frame-wise similarity is another problem in finding similar sequences.

In this chapter, we describe creation of these similar sets in three steps:

- Keyframe Selection
- Keyframe Representation
- Similar Frame Detection

The most challenging problem in sequence detection is changes in camera positions or illumination. Especially in media tracking and story tracking, this kind of problems occurs more often. This can be the result of either camera position change or shot boundary detection as seen in Figure 4.1. This problem can affect similarity results and therefore our tree based approach since precision of our tree based approach depends on similarity results. This chapter introduces generation of representations for frames in videos.

At the beginning, shot boundaries are extracted from videos, based on RGB histograms and Canny Histograms. Then keypoint descriptors are extracted from the most representative frame of each shot, keyframes. Those keypoint descriptors

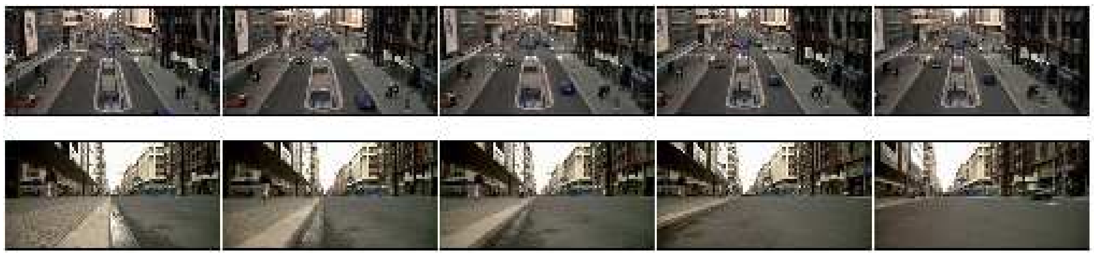


Figure 4.1: Repeating sequences with large differences. These two sequences are so different since they are taken with different viewpoints. Our algorithm can detect this sequence since our features are viewpoint independent.

are vector quantised by k-means and visual terms are created. Each keyframe is defined as a combination of these visual terms. Representation of keyframes is completed by HSV statistics extraction. These steps are defined in the following sections in detail.

4.1 Keyframe Selection

Up to now, we have described our algorithm on frames of media. However, a media is composed of shots and frames in each shot are similar to each other. That is why, using representative frames of shots instead of all frames of shots improves time complexity of our tree based approach without loosing any knowledge about media.

Our method needs to detect shot boundaries to represent videos as efficient as possible by using less number of keyframes. There are several approaches for shot boundary detection. Koumaras et. al. [23] propose a method based on discrete cosine transform. Liu et al. [29] propose a shot boundary detection method based on temporal statistics using eigenspace updating method. In this method, the histogram of the current frame is compared with eigenspace model learnt from previous frames. Shot boundary is detected when model does not fit to the current frame well. Jeong et. al. proposes a method based on frame differences and histogram differences in [17]. In their approach, each frame is divided into MxM grids and intensity difference of consecutive frames are calculated as the first step. If this difference value is between two values based on two thresholds then their histograms difference is calculated and shot boundary detection is detected based on another threshold.

Although shot boundary detection can affect results of sequence detection significantly, we decided to use a simple method based on RGB histogram differences and Canny Edge Histogram differences since our system does not depend on keyframe extraction technique and can tolerate differences that are results of keyframe extraction.

Our shot boundary detection algorithm starts with extracting RGB and Canny histograms of first frame in the media. We extract histograms for consecutive frames and find Euclidean Distance between consecutive frames histograms. We obtain L_2 distances based on histograms at the end of this function for consecutive frames.

The most important problem in keyframe extraction is locating correct shot boundaries. Histogram differences in a shot must be small, in general case, by the assumption that frames in a shot are almost identical. Then by the same assumption, these differences must have a local maxima value on shot boundaries. However, this assumption does not hold in all cases. In addition to that, finding local maximas can give false alarms because in histogram differences, there can be lots of local maximas in addition to shot boundaries. We need to eliminate some local maxima points to obtain correct and sufficient shot boundaries. This can be done by smoothing histogram differences. By smoothing, we can guarantee that some small differences are eliminated and big differences corresponding to shot boundaries becomes more obvious.

As mentioned above, histogram differences must be local maxima values. These local maxima values can be extracted by finding jumps. After finding local maxima points, we find shot boundaries by getting intersection of local maxima points at RGB histogram differences and Canny Edge Histogram differences. By this intersection, we extract points where both colour and edge of frames differs in media. As a result, we obtain shot boundaries of the given video.

Each shot contains different number of frames. There are lots of approaches to choose a representative keyframe from frames of shots such as, mean frame, last frame, first frame or median frame. We choose first frame of each shot as a keyframe. Each shot is represented by one keyframe by shot boundary detection.

4.1.1 Keyframe Extraction

Each video contains almost 25-30 frames per second in general so that a one hour long film contains more than 100,000 frames. This number can be feasible for most of the applications but most of these keyframes are the same or very similar. We can remove such frames without loosing any knowledge about general structure of videos. Instead of using repetition of same frames several times, we can discard these repetitions and use the one as a representative frame, keyframe, among similar ones.

We need to find shot boundaries for keyframe extraction. There are several approaches for shot boundary detection given in [17], [16], [29]. We use a similar approach as used in [17] since none of the methods can give exact results for shot boundary detection.

First step in keyframe extraction is finding shot boundaries. First frame or last frame or median of all frames in a shot can be used as a keyframe for that shot. We use first keyframe of each shot as a keyframe.

We try to use a method independent of thresholds. Our shot boundary detection algorithm works based on colour and edge histogram differences. This is a two-pass algorithm.

In the first pass, we calculate colour and edge histogram differences for consecutive frames. We know that at shot boundaries these difference values must have higher values than surrounding differences, some kind of peaks. We use both colour and edge histograms because using only one of them can give lots of false alarms for shot boundaries. Common peak positions in difference values are taken as shot boundaries but there are too many peak locations in raw difference

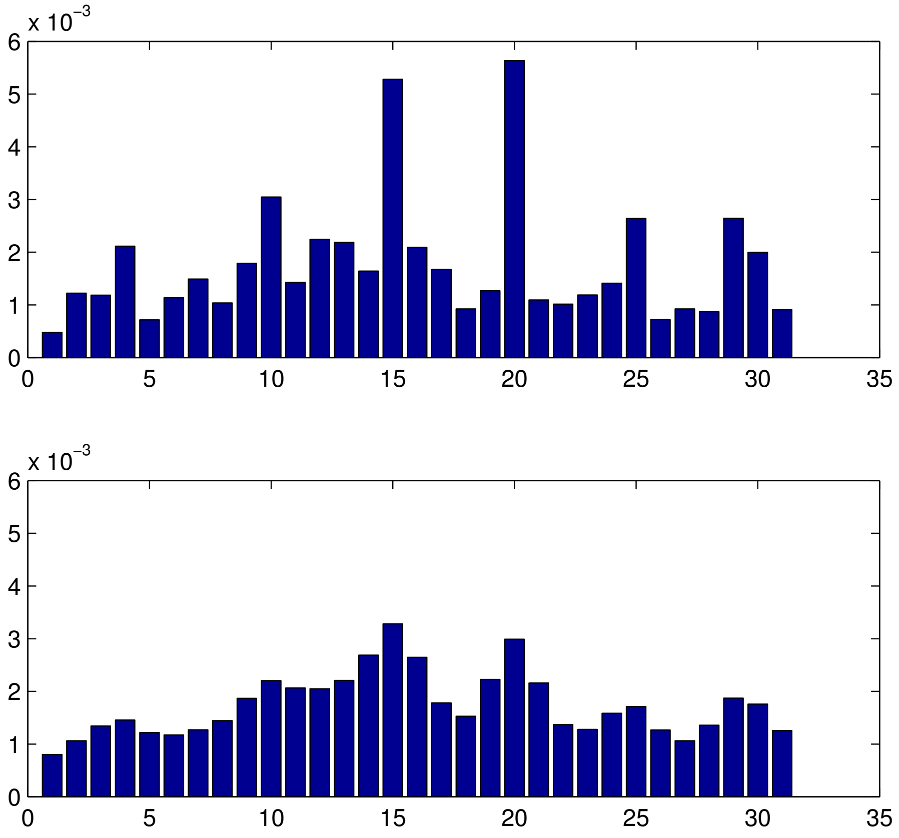


Figure 4.2: Canny Edge Histogram differences (above) for consecutive images and its filtered version(below).

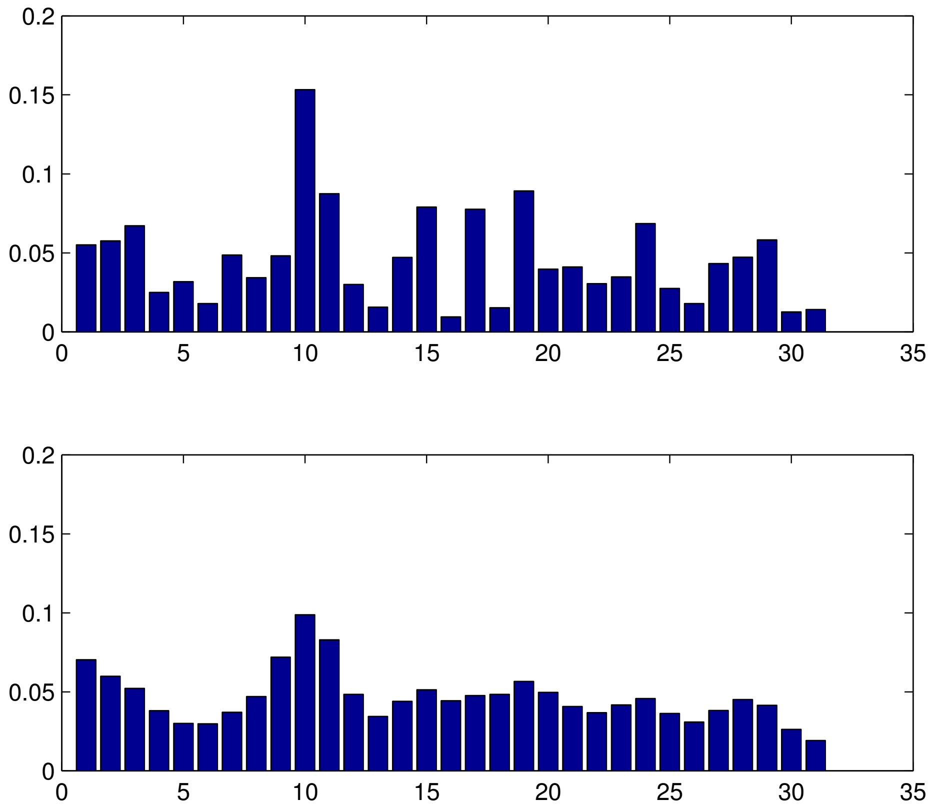


Figure 4.3: Colour Histograms differences (above) for consecutive images and its filtered version(below).

values. We need to remove some small peaks without using a threshold value.

We decided to apply a smoothing filter to difference values. This helps us to remove some small differences. Results of this filtering can be seen at Figure 4.2 and Figure 4.3. We can find peak positions, where difference values start to decrease just after an increase, after these smoothing step. Common peak position in edge and colour histograms are considered as shot boundaries as a result of first pass.

In the second-pass, we extract keyframes according to shot boundaries. We consider first frame of each shot as a keyframe.

4.2 Keyframe Representation

Recently, it has been shown that the viewpoint changes and transformations can be successfully handled by use of salient points, key points. In this study, we follow up the same approach and represent frames with salient points which are robust to affine transformations, illumination changes and viewpoint changes.

4.2.1 Detection of Visual Points

Most well-known approaches to represent an image is using either local feature or global features or both. However, some points in images are more robust to scale and illumination changes. Hence, using descriptors of these points is more meaningful to identify an image. These kind of local descriptors are mostly used in object recognition, image matching and scene classification [47, 42, 37, 26].

Detection of local interest points recently attracts researchers attention [38, 36, 31, 27, 19]. Among these descriptors, we mainly focuses on Lowe's Difference-of-Gaussian (DoG) detector and Mikolajczyk's Harris Affine detector.

Lowe followed four major stages in order to detect keypoints and generate descriptors [31]. These steps are :

- scale-space extrema detection,
- keypoint localisation
- orientation assignment
- keypoint descriptors extraction

In the first step, local maxima and minima points are extracted in different scales by using Difference-of-Gaussian (DoG). These points are considered as keypoints candidates. In second step, keypoints are selected based on measures of their stability and localised with a detailed model determining location and scale. Then, an orientation is assigned to each keypoint based on gradient directions. Finally, local image gradients are computed at specified scale and region around each keypoint. This is used as keypoint descriptor referred as Scale Invariant Feature Transform (SIFT).

Mikolajczyk *et al.* extended Harris-Laplace detector to deal with significant affine transformations [36]. Harris multi-scale detector are used for detection of keypoints, where an initial location and scale for the keypoint is assigned simultaneously. To obtain the shape adaptation matrix for each interest point, the second moment descriptor is computed with automatically selected integration and derivation scale, where integration scale is the extrema over scale of normalised derivatives and derivation scale is the maximum of normalised isotropy. Consequently, the points of an image that are invariant to affine transformations are acquired.

We have used two types of regions, Shape Adapted regions and Maximally Stable regions [47]. Shape adapted regions are extracted by elliptical curve adaptation and corresponds to corner like structures [1, 28, 35, 46]. Maximally Stable regions are areas where the area is approximately stationary as the intensity threshold is varied [32].

We observe that both DoG and Harris Affine detector detects important parts to describe an image. Figure 4.4, shows Lowe Sift Points and Harris Affine points for same image.

In our study, we mainly used Harris Affine detector since it is proposed that Harris Affine gives better result than DoG detector and also LoG is an approximation to DoG.

4.2.2 Description of Visual Points

In [37], it is claimed that SIFT based descriptors perform best compared to other descriptors. SIFT based descriptors are extracted by first computing orientation and magnitude around each interest point. then a 8 bin histogram is created for 4X4 sub-region around interest point resulting a 128 dimensional vector for descriptor. Figure 4.5 shows this creation process.

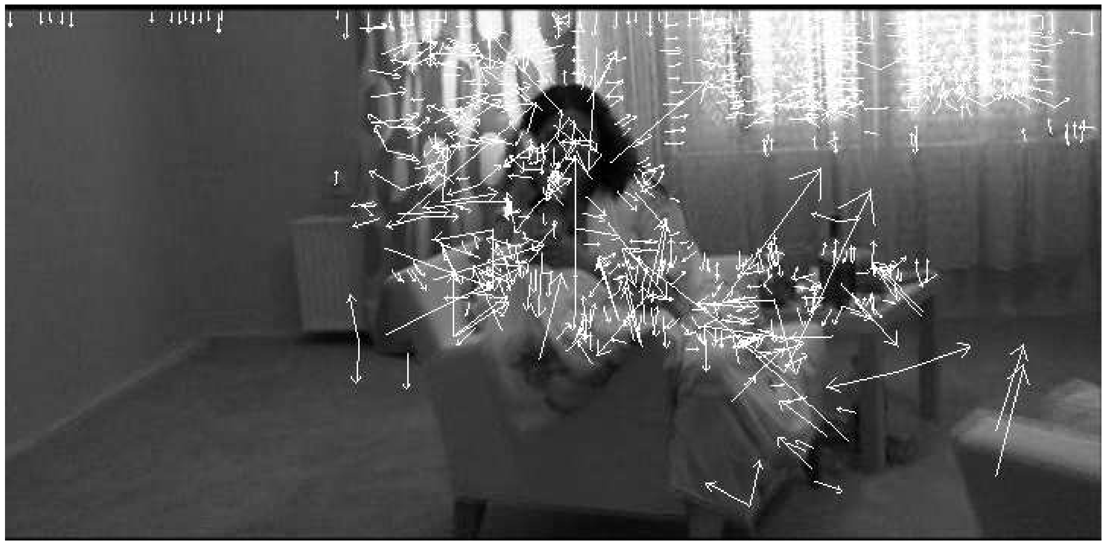
We also add 5 more dimension ,coming from affine transformation, to 128 dimensional descriptor vector and obtain a 133 vector. First two dimensions are location of interest point and other three dimensions are parameters of affine transformation. By this way, we make use of location and shape information of detected regions.

Following section describes creation of visual terms that is the next step after interest point detection and extraction detectors of these points.

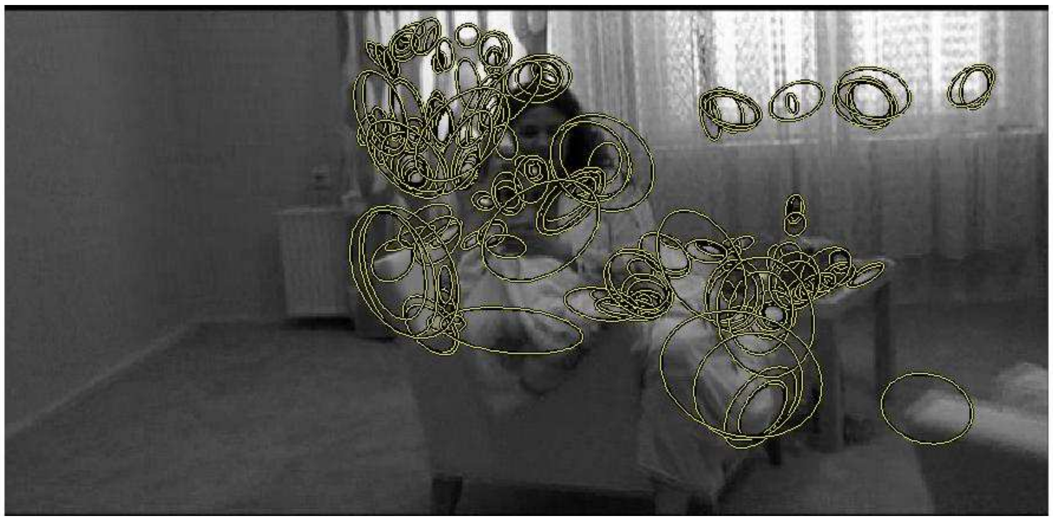
4.2.3 Visterm Generation

Local interest points and descriptors are extracted and visual terms are needed to be generated. Visual terms are used to define images so that, visual terms are very essential for image representation.

Each keyframe has different numbers of keypoint descriptor ranging from 0 to 4000. Some methods are proposed to find matches between keypoint descriptor



(a)



(b)

Figure 4.4: Keypoints of same image detected by DoG (a) and Harris Affine detector (b).

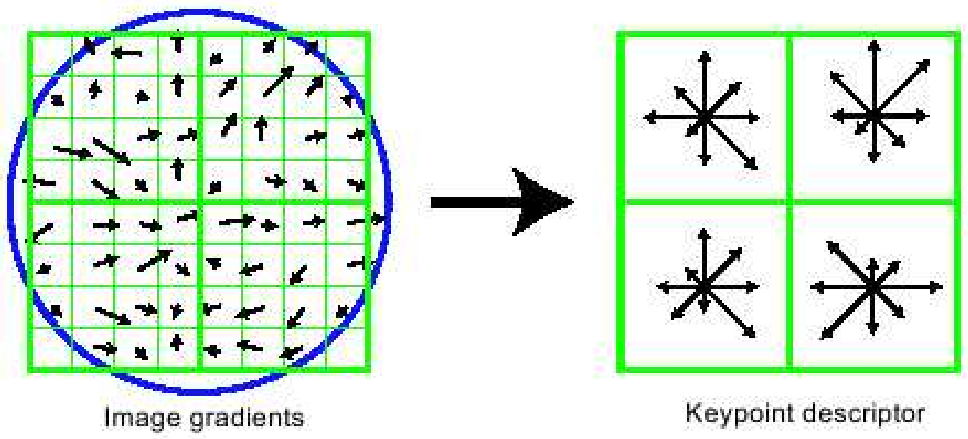


Figure 4.5: On the left, magnitude and orientation are given around a local interest point. On the right, summarised version of gradient over 4x4 sub-region is given which correspond to summation of magnitudes in same direction on that region [31].

of different keyframes such as Bag of Features *BoF* [41, 42, 31] and one-to-one keypoint descriptor match. We use *BoF* method to define keyframes since numbers of keypoint descriptors are so many and it is not applicable to find one-to-one keypoint descriptor in large databases. We also tested one-to-one keypoint descriptor match on DoG SIFT descriptors and results are given in Chapter 5.

BoF approach treats images as documents with each image is described by visual terms histograms. Visual Terms, *Visterms*, are similar to words in a document. Each image is considered as a combination of *Visterms* just like a document is a combination of words. We need to find *Visterms* as representative as possible. General approach to determine correct *Visterms* is quantizing all keypoint descriptor by k-means and considering each cluster centroids.

4.2.4 Representations Based on Visterms

We can consider our media database as a library after finding *Visterms*. Most popular and well-known technique to represent a book is using *Term Frequency Inverse Document Frequency*, *tfidf*, method. We use similar method to represent images in media database since our media database is similar to library by

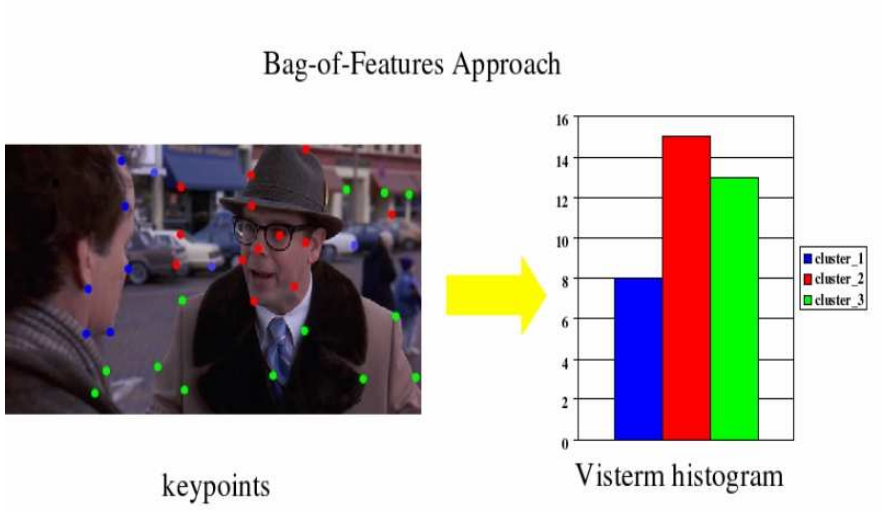


Figure 4.6: Visterm Histogram

$Visterms$.

Each bin in $tfidf$ presentation is the product of two terms, $term-frequency$ and $inverse-document-frequency$. For example, a document is represented by a k dimensional $tfidf$ vector $(t_1, t_2, \ldots, t_k)$, where $term-frequency$ of image i is :

$$tf_i = \frac{n_{id}}{n_d} \tag{4.1}$$

and $inverse-document-frequency$ is

$$idf_i = \log \frac{N}{n_i} \tag{4.2}$$

and i^{th} term, t_i is the product of tf_i and idf_i

$$tfidf = \frac{n_{id}}{n_d} \log \frac{N}{n_i} \tag{4.3}$$

where, n_{id} is number of occurrences of term i in document d, n_d is total number of terms in document d, N is the total number of documents in database and

n_i is the number of documents in database containing term i.

Colour information of images can not be discarded since keypoint descriptors are extracted from gray-scale images. We add colour information to representation of images by making use of HSV statistics. Following section describes representation of images by HSV statistics.

4.2.5 Representation Based On HSV Statistics

Although SIFT descriptors extracted from Maximally Stable and Shape Adapted regions are important for allowing view point and illumination changes, we noticed that they are not sufficient to correctly capture the similarities, and we also incorporate the colour information.

We use 5x7 grid HSV statistics to represent colour. Each frame is divided into 5x7 grids and mean and standard deviation of each band is calculated for each grid. We obtain 210-dimensional vector for each frame for colour data.

4.3 Similar Frame Detection

For the sequence detection algorithm to be successful, it is very important to capture the similarities between frames correctly. In our ranking approach, firstly each frame is described by features given in this chapter. Then dissimilarity values are calculated based on each feature separately and combined by weighted sum. By sorting resulting dissimilarity values, similar frames are found.

Most of the mentioned systems make use of a hard threshold to find duplicates of the frames and then sequences are found assuming that they contain the same set of frames. However, there are two problems with such approaches: first, it is difficult to define a single general threshold applicable to different characteristics of large number of frames; second, some frames can be missed due to wrongly selected thresholds causing gaps in the sequences. We propose a threshold free approach to find similar frames by using jump positions in dissimilarity values.

Our contribution to find ranking is that, in our ranking strategy, there is no hard threshold to find similar frames. Thresholds are used to find near-duplicates frames in most of the existing approaches. We overcome this problem by finding jump positions in dissimilarity values.

This chapter overviews features to find ranking, distance calculations for different features used, method to combine distances and jump position extraction.

4.3.1 Features

To reduce number of false sequences, trees must be created as precise as possible. Effective similarity sets are needed for precise tree creations. Hence, frames should be represented correctly.

We use SIFT descriptors extracted from affine co-variant regions to be robust to view point and illumination changes, but also incorporate the colour information in the form of HSV statistics extracted from fixed sized grids since colour is also a valuable information in most of the cases.

We detect two types of viewpoint co-variant regions for each frame as used in [47]. First one, called as Shape Adapted Region, is constructed by elliptical shape adaptation around an interest point. Second one, called as Maximally Stable Region, is constructed by selecting areas from an intensity watershed image segmentation.

Although SIFT descriptors extracted from Maximally Stable and Shape Adapted regions are important for allowing view point and illumination changes, we noticed that they are not sufficient to correctly capture the similarities, and we also incorporate the colour information.

We use 5x7 grid HSV statistics to represent colour. Each frame is divided into

5x7 grids and mean and standard deviation of each band is calculated for each grid. We obtain 210-dimensional vector for each frame for colour data.

4.3.2 Distance Calculation

In Information Retrieval and approaches that uses *BoF* method, Cosine Distance is used to find similarities documents or frames. Also in computer vision, L_2 Distance is mostly used to calculate distances based on HSV statistics.

In this study, features can be classified into two types based on distance calculation. First type is features for which distance calculation is based on cosine distance, and other type is features that distance calculation is based on L_2 distance.

Cosine Distance is the arc-cosine angle of angle between two vectors and finds the similarity of vectors based on directions. The similarity of frames based on SIFT descriptors of the salient regions are found by Cosine Distance of tfidf vectors. For example, we have two $tfidf$ vectors t_i and t_j, Cosine Distance between t_i and t_j (which we refer D_1) is calculated as follows :

$$D_1 = 1 - \frac{\sum_{i=1}^{n} t_i \times t_j}{\left(\sum_{j=1}^{n} {t_i}^2\right)^{1/2} \left(\sum_{j=1}^{n} {t_j}^2\right)^{1/2}} \tag{4.4}$$

L_2 distance is the line length between two points in space. The similarity of two frames based on HSV statistics is calculated by L_2 Distance. L_2 Distance between two vectors P and Q (which we refer D_2) is calculated as follows :

$$D_2 = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2} \tag{4.5}$$

4.3.3 Combining Distances

Neither SIFT descriptor based similarity values nor HSV statistics based similarity values are perfect. Also none of them has higher priority. For these reasons, we decided to combine these similarity values by equal weights and obtain a single distance value as

$$D(f_i, f_j) = D_1(f_i, f_j) + D_2(f_i, f_j) \tag{4.6}$$

for each frame pair (f_i, f_j). Note that the distances are normalised before combination.

4.3.4 Extracting Jump Positions

Our main contribution to ranking is that we do not use a hard threshold value on dissimilarity values to limit dissimilar frames. Main aim to find rankings is to accept only near-duplicate frames by eliminating others. Most existing approaches in near-duplicate detection make use of a hard threshold value. This kind of thresholds are not meaningful since near-duplicate frames have smaller similarity values than other frames. By this assumption, if dissimilarities of a frame is sorted, a jump value is encountered at the position of last near-duplicate frame. We make use of this jump value and discard other frames.

To find jump position, for each frame f_i, first we rank all the images according to distance D, and then seek for a jump in the distances to separate the similar instances from the others. We eliminate the different frames by using peaks on similarity values. We apply a filter to sharpen peaks and find the maximum peak to ignore different frames. When the peak is at a number less than 10, we take 10 as the peak position. This approach allows us to reduce the number of similar frames without loosing the correct ones as seen in Figure 4.7.

In Figure 4.8, distances between query frame and frames in the similar set is given. Our filtering approach discards distances of first two most similar frames because of convolution. So that, maximum jump position is found after the 7^{th} image in similar set.

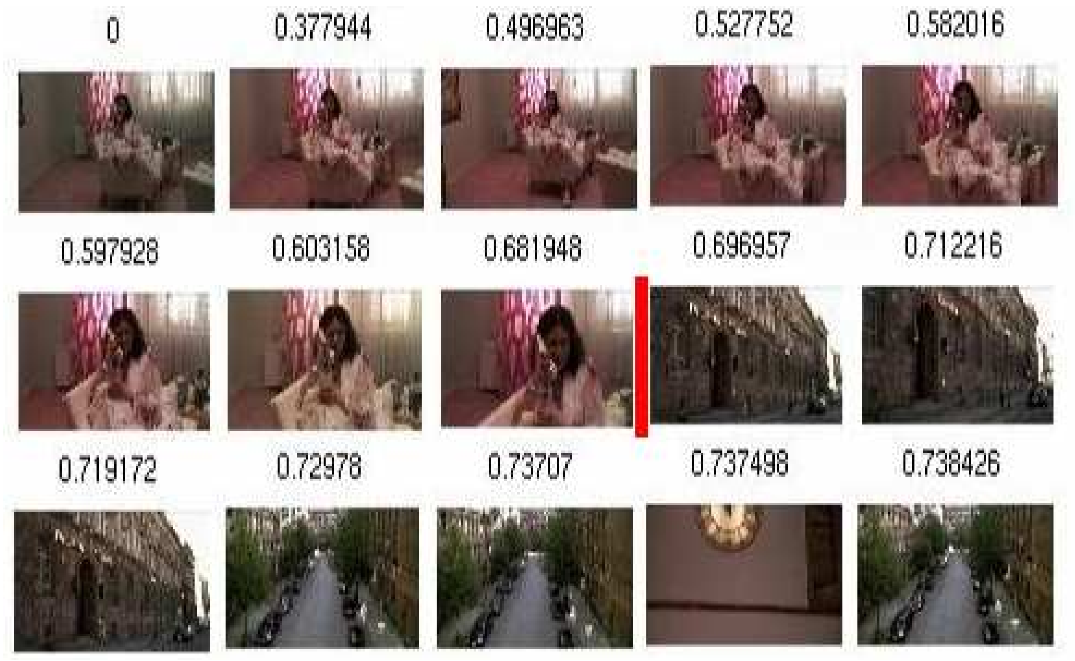


Figure 4.7: Ranking results for a query image. First image is query image and others are most similar images. Titles of images show their dissimilarity values. In our approach we find peak position after 7^{th} image and take first 10 image as similar to query image.

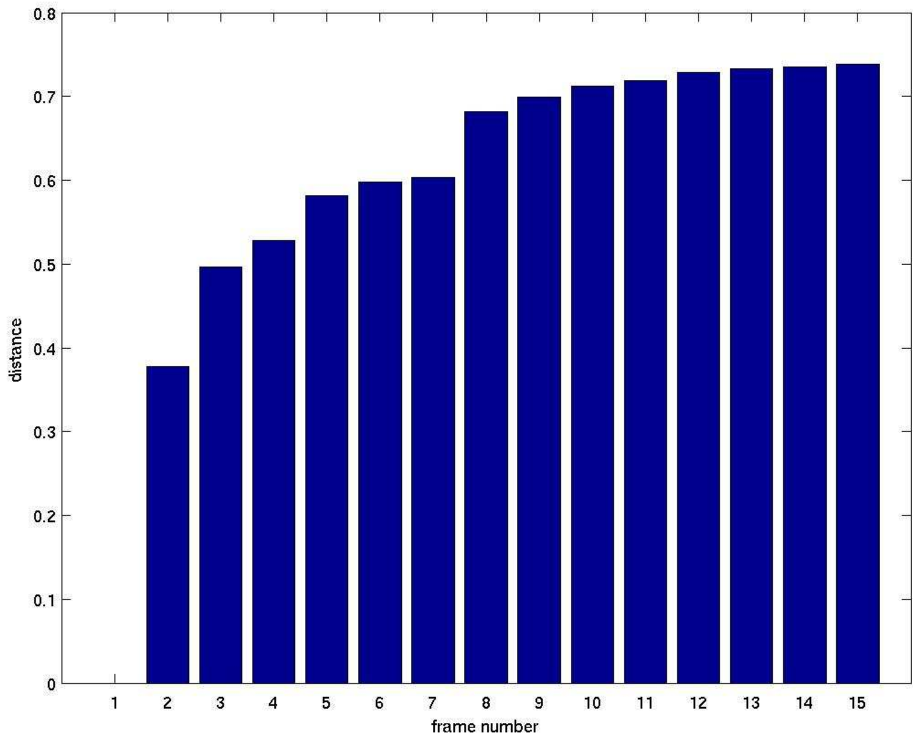


Figure 4.8: Bar graph of distances obtained from Figure 4.7.

Chapter 5

Experiments

Experiments are carried out on CIVR2007 Copy Detection Showcase dataset, two full length movies, "Run Lola Run" and "Groundhog Day" and commercials of TRECVID news corpus. In all datasets, repetitions are not exactly same but similar. There are some illumination changes, scene changes and also transformations. We use keyframes instead of frames, which also resulted in different sequences. Keyframes are provided by NIST for TRECVID corpus for commercial dataset. For the movies and CIVR2007 Copy Detection Showcase dataset, we extract the keyframes using our approach.

In the following, first characteristics of the experimental datasets are given. Then results on each dataset is reported followed by detailed evaluations based on different aspects.

5.1 Datasets

We try to solve copy detection,media tracking and story tracking problems. We have three different datasets since for each problem different dataset is needed. In the following sections, detailed information about datasets will be given.

5.1.1 CIVR Copy Detection Dataset

As mentioned before, in ACM International Conference on Image and Video Retrieval 2007 (CIVR2007), a session is held to compare and explore existing Copy Detection methods. For this session, a dataset is published. It contains 101 videos (about 35GB, 80 hours) that are collected from different sources These videos' durations are ranging from 5 minutes to 2 hours. This dataset is processed and more than 220,000 keyframes are obtained and more than 80,000,000 keypoints are extracted.

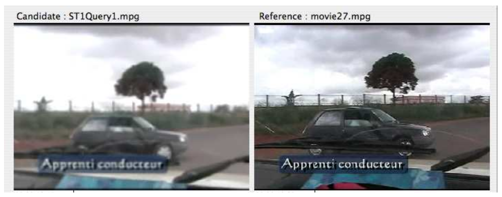
For Copy Detection session, another 15 query videos are supplied. Again these query videos are collected from different sources and durations are ranging from 5 minutes to 20 minutes. These queries contain several transformations as seen in Figure 5.1. Because of these transformations and size of dataset (35GB), CIVR is the most challenging dataset. Some of these query videos are copies of videos in database.

We need to change some parts of our system for this dataset. Since for copy detection a query is needed, we adapt our system to get a query. Then we need to reduce running time of our algorithm. We create an index structure by grouping frames in each movie in the database by using cosine measures to mean $tf - idf$ vector of each movie. By using this indexing structure, sequences are detected according to query clip. If detected repeated sequences give a %75 or more coverage in the target movie, we conclude that target movie is a copy of query clip.

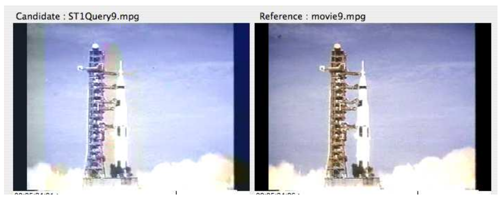
We use this dataset to test our method for copy detection and also compare our results with other methods.

5.1.2 Commercials

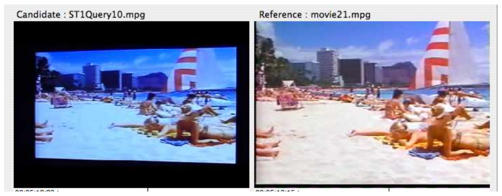
Another dataset used in the experiments is the broadcast news videos provided by NIST for TRECVID video retrieval evaluation competition [49]. It consists of



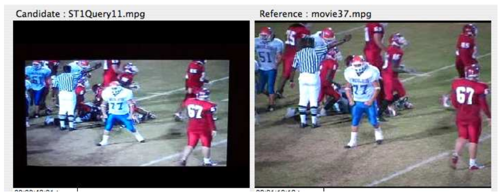
(a)



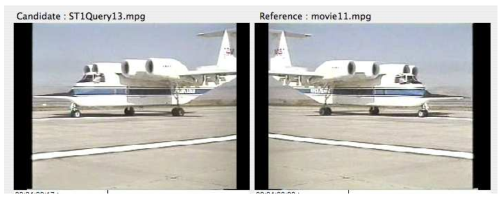
(b)



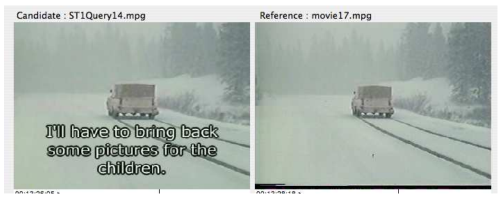
(c)



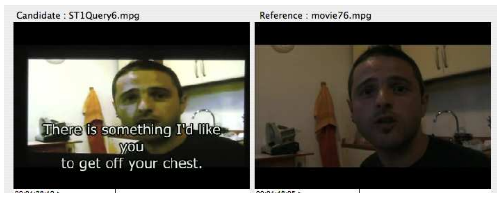
(d)



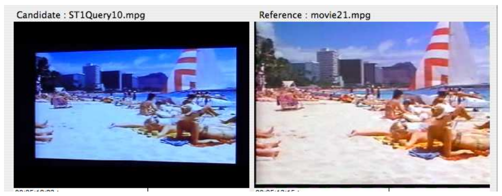
(e)


Candidate : ST1Query14.mpg
Reference : movie17.mpg
I'll have to bring back some pictures for the children.

(f)


Candidate : ST1Query6.mpg
Reference : movie76.mpg
There is something I'd like you to get off your chest.

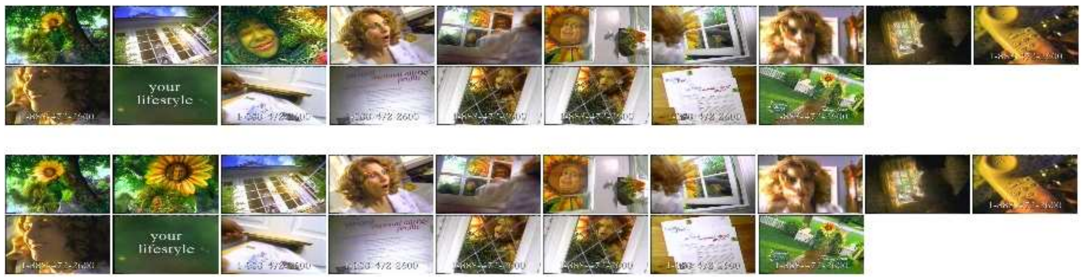
(g)



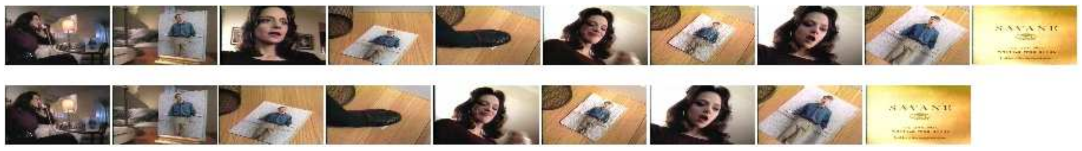
(h)

Figure 5.1: Transformations used in CIVR2007 Copy Detection dataset. These transformations are blur (**(a)**), change in phases of colour (analogical noise) (**(b)**), camcording (**(c)**), camcording with angle (**(d)**), flip (**(e)**), zoom and subtitle (**(f)**), camcording and subtitle (**(g)**) and camcording with an angle(**(h)**), [4].

50 videos. Shot boundaries and keyframes are provided by NIST. Some example sequences can be seen in Figure 5.2. As seen in these examples, although shot boundaries are so precise in this dataset, number of frames and frame orders are different because of editing.



(a)



(b)

Figure 5.2: Repeating sequences extracted from commercial dataset. **(a)** has frame order change and **(b)** has different number of frames in sequences.

We choose keyframes of commercials from keyframes of 50 videos (1647 frames). This set is used for story tracking problem.

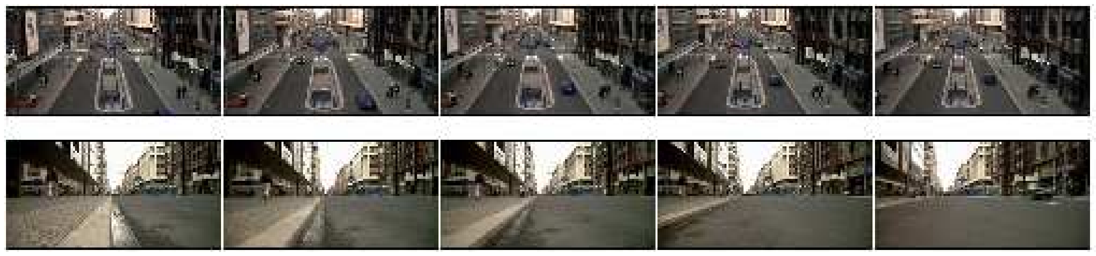
5.1.3 Movies

One of the problem that we try to solve is story tracking. We tested our method on two feature films "Run Lola Run" and "Groundhog Day". These two films are also used in [47] since they contain several repetitions.

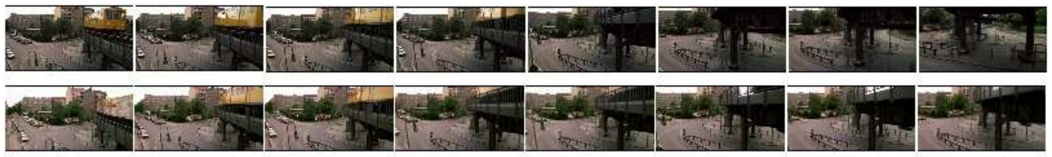
5.1.3.1 Run Lola Run

"Run Lola Run" is directed by Tom Tykwer in 1998. It is a 1.5 hour long mpeg4 encoded film. It contains 30 frames per second and totally 162,000 frames. Each frame size is 720x352 pixels. In this movie, an event is repeated three times during movie. Event starts with a phone talk of Lola (leading character in movie) and ends with shot of Lola. This part is repeated with different camera positions and also some scene changes that change the result of event.

Viewpoint and illumination changes make this dataset challenging. Also since keyframes are extracted by our method, keyframes in sequences and repetitions are not precise as in commercial dataset. This makes similar set construction more difficult. Examples of repeating sequences can be seen in Figure 5.3.



(a)



(b)

Figure 5.3: Repeating sequences extracted from "Run Lola Run" movie. **(a)** is taken with different camera positions and **(b)** has differences because of shot boundary detection.

We used "Run Lola Run" in three different ways. In the first one, we extract keyframes by our shot boundary detection algorithm. This gives us 5922 frames. This set is used to run overall algorithm. In the second one, keyframes are extracted with equal distances and 13601 frames are obtained. This set is used to find the effect of keyframe extraction part on tree-based approach. Last set is created by reducing first set. It contains 1077 frames and is used for testing.

5.1.3.2 Groundhog Day

"Groundhog Day" is directed by Harold Ramis in 1993. It is similar to "Run Lola Run" and again 1.5 hour long mpeg4 formatted film. It contains 30 frames per second and totally 162,000 frames. Each frame size is 720x352 pixels. "Groundhog Day" gives one day of a news reporter in three different ways. Each repetition starts with the same scene but repetitions of the day differ by camera positions and illumination changes.

Same as "Run Lola Run" dataset, this dataset has viewpoint and illumination changes. Also in this set, illumination changes are more obvious. In addition to these, our keyframe extraction method can give different number of frames for original sequence and repetitions. Some example sequences can be seen in Figure 5.4.

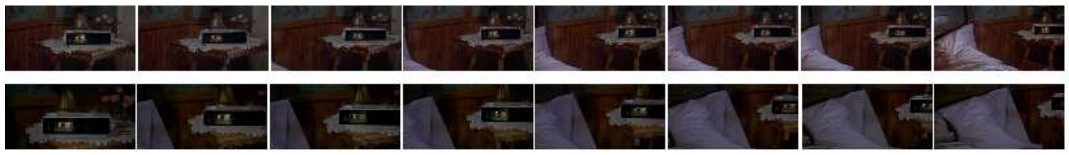
We extract keyframes of "Groundhog Day" by using our shot boundary detection and 7570 frames are obtained. This set is used to test overall tree-based algorithm.

5.2 Results on Datasets

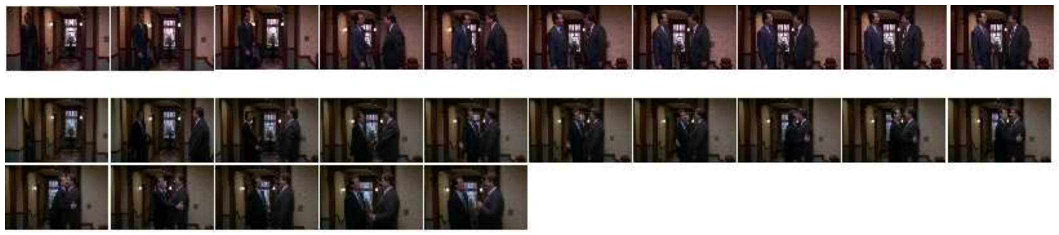
In the following sections, results on CIVR Copy Detection Dataset, commercial dataset and movies are given.

5.2.1 CIVR Copy Detection Dataset

Results obtained in CIVR2007 Copy Detection Showcase are given in Table 5.1. We perform the same with the best in competition, ViCopT that is created by



(a)



(b)

Figure 5.4: Repeating sequences extracted from "Run Lola Run" movie. **(a)** has differences because of zooming and **(b)** has different number of frames. These scene changes are because of shot boundary detection.

INRIA, France. ViCopt uses Harris point of interest and a 20-dimensional signature is created for each point. After that, copies are detected by using trajectories of keypoints (More details can be found in [25]).

Among 15 queries, we have only missed one of them. It is created by cam cording and subtitles as seen in Figure 5.5. In this case, because of subtitles, queries have additional keypoints compared with similar frame in target video. This changes visterm histograms and distances between similars so that similar sets. That is why, our method can not detect this copy.

CIVR2007 Copy Detection Dataset contains different type of transformations. Number of missed queries according to these transformations are given in Figure 5.6. As seen this figure, camcording transformation with subtitle is the most difficult transformation to detect.

Table 5.1: Results on CIVR Copy Detection Dataset

Team	Precision
OUR APPROACH	**0.93**
ViCopT	0.93
Advestigo	0.86
IBM-1	0.86
IBM-3	0.80
IBM-2	0.73
City University of Hong Kong	0.66
Chineese Acedemy of Sciences	0.53
Chineese Acedemy of Sciences-2	0.46

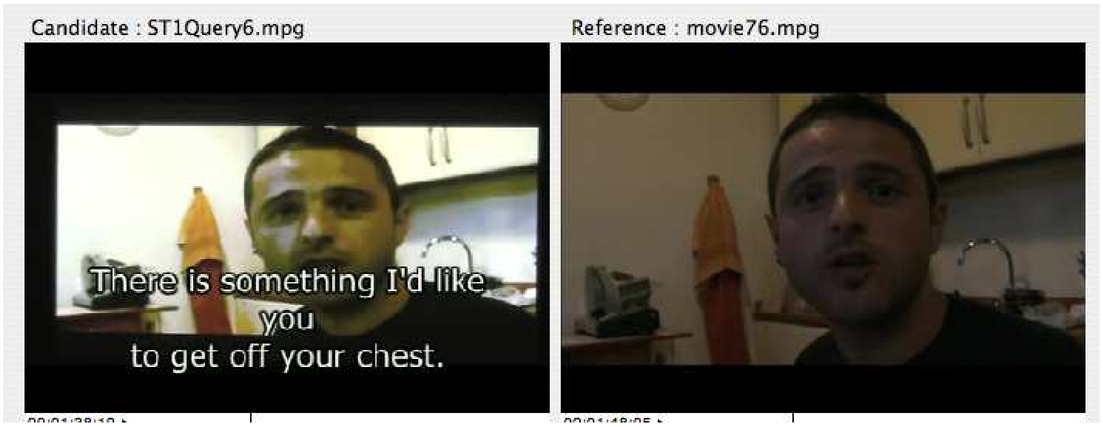


Figure 5.5: Two frames from missed query in CIVR2007 Copy Detection Showcase. Query frame is on the left and target frame is on the right. [4]

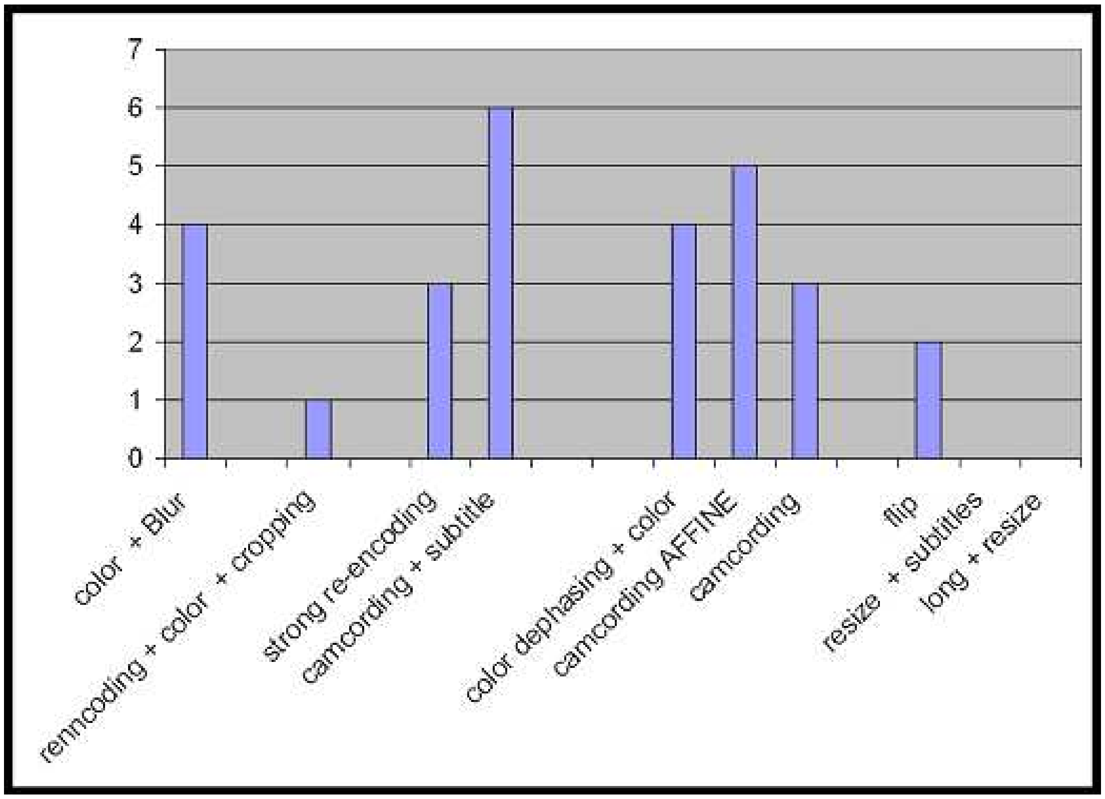


Figure 5.6: . Histograms of missed transformations in CIVR2007 participant results.[4]

5.2.2 Commercials

Finding sequences in commercial dataset is much easier than others since commercials have colourful structure and keyframes are extracted more precisely. Some example sequences extracted from commercial dataset are shown in Figure 5.7 and Figure 5.8.

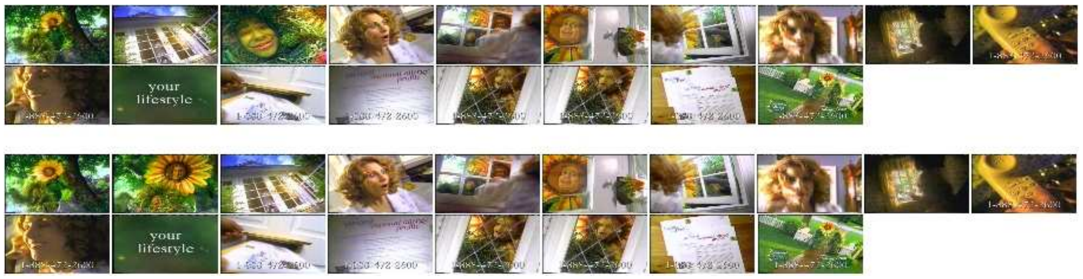


Figure 5.7: An example sequence from TRECVID dataset. Second and third frames are different in real sequence and repetition.

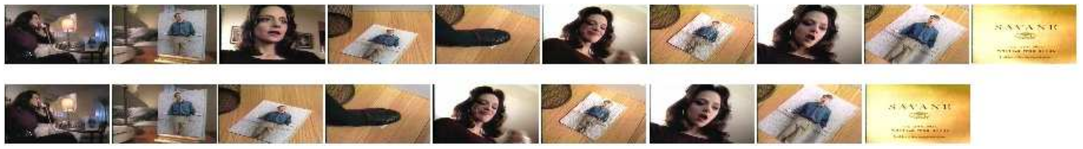


Figure 5.8: An example sequence from TRECVID dataset. Number of keyframes is not same for two sequences.

As seen in the Figure 5.7 and Figure 5.8, the proposed method is able to capture the differences in the number and order of frames in the sequences. These sequences are found by using combination of HSV statistics and SIFT descriptors extracted from Maximally Stable and Shape Adapted regions.

We have tested 3 features on commercial dataset and results are given in Table 5.2.

Table 5.2: Sequence Detection Results for commercial dataset.

Method	Precision.	Recall.
SIFT Descriptor	0.92	0.71
HSV Statistics	0.98	0.76
Combination	0.91	0.74

In Table 5.2, it is seen that HSV statistics gives better performance. However this is not valid for other datasets used in our experiments. Main reason is that commercials have distinctive colours and it can be sufficient for finding similar sets. In addition to that, this can be a result of encoding of TRECVID data. It is mpeg1 encoded and extracting SIFT descriptors from mpeg1 encoding can give insufficient results.

5.2.3 Movies

Both movies ”Run Lola Run” and ”Groundhog Day” have almost same characteristics. So that, precision results on these movies are so close to each other. Sequences are repeated with camera position and/or illumination changes. Results of ”Run Lola Run” and ”Groundhog Day” are given in the following sections.

5.2.3.1 Run Lola Run

”Run Lola Run” contains more challenging sequences than commercial dataset. Some example sequences are given in Figure 5.9, Figure 5.10, Figure 5.11, Figure 5.12, Figure 5.13 and Figure 5.14.

Figure 5.9, Figure 5.10 are the easiest sequences in ”Run Lola Run” movie. Although keyframes in sequences are different because of keyframe extraction method, number of keyframes in sequences are almost same.

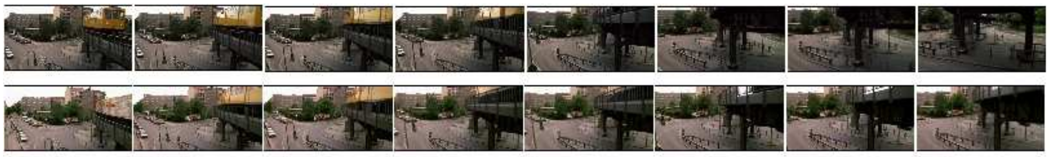


Figure 5.9: An example sequence. Number of frames in sequences are same but frames are not exactly same.

Sequences in Figure 5.11, Figure 5.12 and Figure 5.13 have different number of keyframes in sequences and also because of viewpoint change keyframes in sequences are significantly different.

Figure 5.10: An example of slightly different sequences.

Figure 5.11: An example sequence found by SIFT and HSV combination. Number of frames and frames in sequences are different.

Figure 5.12: An example sequence taken with different camera angles.

Figure 5.13: An example sequence that contains different keyframes because of camera position and keyframe extraction method.

In Figure 5.14, longest sequence detected in ”Run Lola Run” is given In this example, keyframes are different because of zooming and also number of keyframes of similar parts are different.

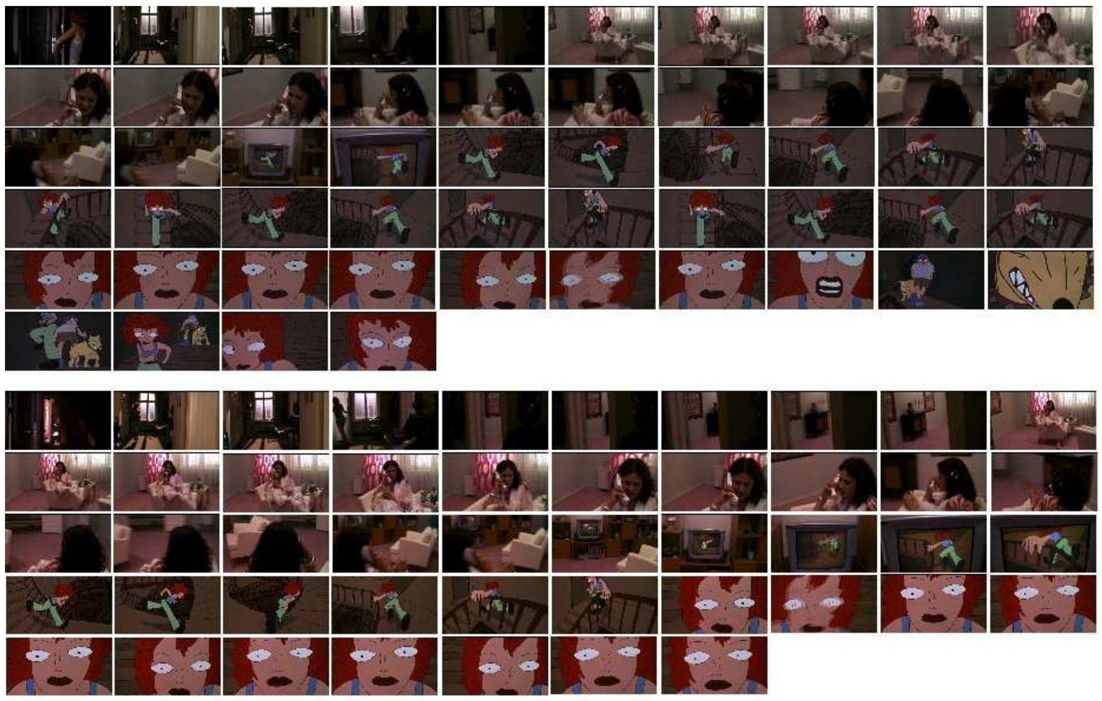


Figure 5.14: Longest sequence found by our method.

As shown in all figures, the proposed method is able to capture both the differences in the frames due to viewpoint and illumination changes and the differences in the number and order of frames in the sequences.

Sequences in commercial dataset can be found by using only HSV statistics since they have distinctive colourful structure. If we apply this to ”Run Lola Run” movie, we obtain false alarms as seen in Figure 5.15. This sequence is extracted by using only HSV statistics to represent frames. In this false alarm, original sequence and repetition is both too dark and HSV statistics can give these keyframes similar.

Another false sequence is given in Figure 5.16. These sequences are extracted by using only SIFT descriptors. As seen, this kind of false alarms can be discarded by using colour information since one sequence is gray and other is colourful. So

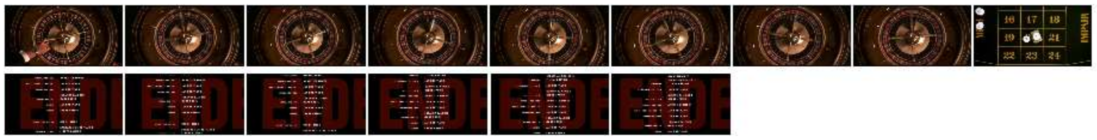


Figure 5.15: A false alarm sequence from TRECVID dataset extracted by using HSV statistics only.

that, we make use of combination of HSV statistics and SIFT descriptors to represent frames.

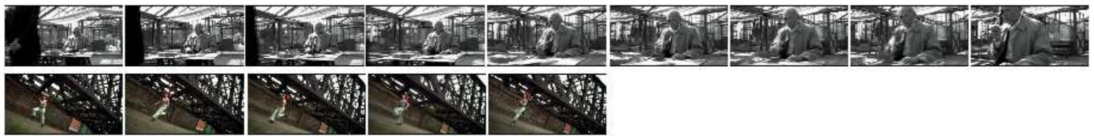


Figure 5.16: A false alarm sequence from TRECVID dataset extracted by using HSV statistics only.

We have conducted three types of experiments on ”Run Lola Run”. For the first experiment, keyframes of ”Run Lola Run” are extracted by using our keyframe extraction method and 5922 keyframes are obtained. In this set, it is almost impossible to create ground truth since boundaries of sequences are not exact. So that we give only precision values as seen in Table 5.3.

Table 5.3: Sequence Detection precision values on Run Lola Run movie.

Method	**Correct Det.**	**False Det.**	**Precision**
SIFT Descriptor	89	19	0.82
HSV Statistics	55	18	0.75
Combination	105	13	0.89

As expected, combination of HSV statistics and SIFT descriptors performs better than single usage of these features. This is mostly because of characteristics of ”Run Lola Run”. There are illumination and viewpoints changes and these changes can not be tolerated by using SIFT descriptors.

For the second experiment, we subsample previously extracted 5922 keyframes and obtain a set containing 1077 keyframes. We create a truth set for this set. We detect sequences by using three different features to represented frames. These are SIFT descriptors extracted from MS and SA regions, HSV statistics and combination of these features. Since boundaries of sequences are not exact, a detected sequence is considered as a correct sequence if it is detected by ± 5 neighbourhood of ground truth set. If a sequence is divided into parts they are considered as a false sequence for recall calculation but considered as a correct sequence for precision calculation. Results on this dataset is given in Table 5.4.

Table 5.4: Sequence detection precision, recall values, number of divided sequences and number of detected sequences for similar sets obtained by using different features. MS stands for descriptors extracted from Maximally Stable regions and SA stands for descriptors extracted from Shape Adapted regions. Lowe Sift stands for descriptors extracted from regions detected by DoG.

Method	**Sequence Count**	**Divided Count**	**Precision**	**Recall**
Lowe SIFT	52	24	0.8269	0.3725
HSV Stat.	46	18	0.7173	0.2941
Lowe SIFT and HSV Stat.	44	13	0.7727	0.4117
MS and SA	47	11	0.8298	0.5490
Combination	48	0	0.9583	0.9019

According to Table 5.4, combination of SIFT descriptors on MS and SA regions and HSV statistics give better performance. This is an expected result since "Run Lola Run" has viewpoint and/or illumination changes. Also descriptors extracted from MS and SA regions performs better than descriptors extracted from Lowe keypoints (DoG) since MS and SA regions are more robust to affine transformations and this makes MS and SA regions more robust to viewpoint changes compared to DoG. Also, because of illumination and viewpoint changes HSV statistics gives the worst performance on this set.

Figure 5.17 shows the distribution of sequence lenghts. We can see that our tree-based approach can miss some short sequences. The reason is that, our similar set construction method can miss some similar frames and when this is encountered in short sequences, repetition of sequence can be shorter than real sequence and sequence lenghts becomes inconsistent. This much differences in number of keyframes can be tolerated and repetition can be missed.

It is obvious that sequences can be found easily and more complete with a

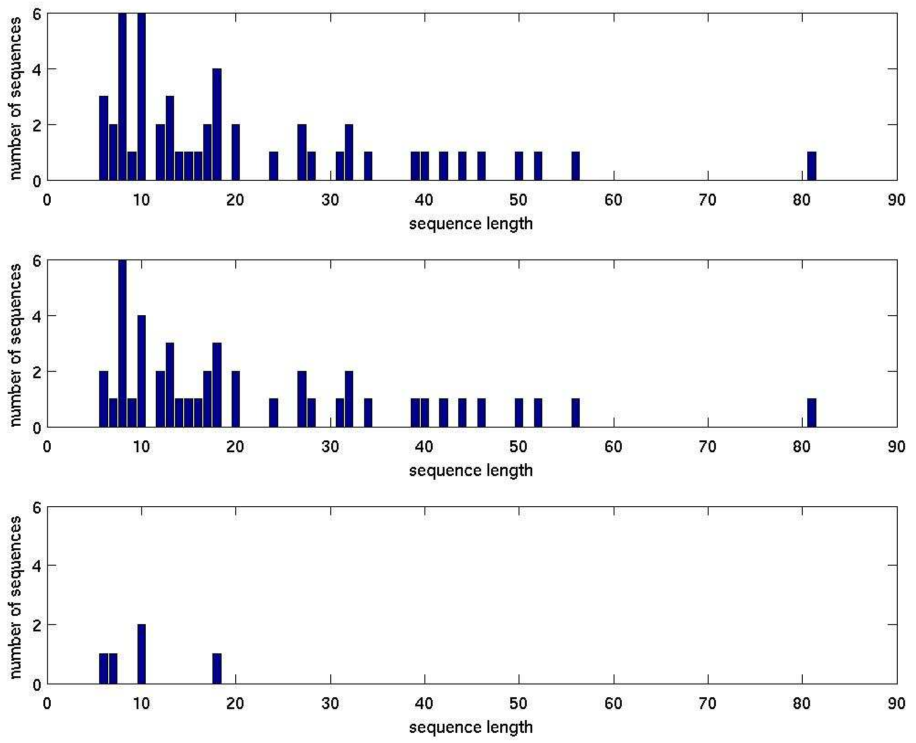


Figure 5.17: Sequence lengths for original sequences, detected sequences and missed sequences. **Top row** shows number of sequences, **middle row** shows number of correctly detected sequences and **bottom row** gives number of false sequences according to their lenghts. x axis show sequence lengths and y axis show number of sequences on that length.

robust shot boundary method. However, our tree-based approach can tolerate differences caused by shot boundary detection method. That is why, we use a simple method described above. But we also have tested another keyframe extraction method. In this experiment, instead of finding shot boundaries, we use three frames in each second as a keyframe. By this way, we obtain 13,601 keyframes from "Run Lola Run" movie. We detect sequences on this set and obtain %100 precision value. In this set, sequences are repeated with more similar frames than the other "Run Lola Run" sets. This show that keyframe extraction method has significant influence on sequence detection.

5.2.3.2 Groundhog Day

Similar to "Run Lola Run", there is no exact sequence boundaries in "Groundhog Day"and keyframes in sequences are different because of viewpoint and/or illumination change and also keyframe extraction.

Some example sequences for "Groundhog Day" can be seen in Figure 5.18, Figure 5.19 and Figure 5.20.

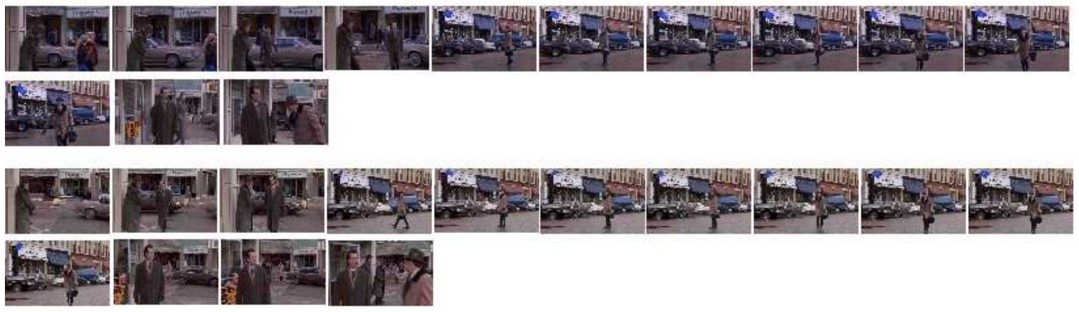


Figure 5.18: An example sequence from Groundhog Day. Number of frames in sequences are same but frames are not exactly same.

Sequences in Figure 5.18 have same number of frames. However, keyframes are different. This difference can be a result of zooming or keyframe extraction. Sequences in Figure 5.20 and Figure 5.19 have different number of frames in

Figure 5.19: Example repeating sequence from "Groundhog Day" that has differences because of zooming.

Figure 5.20: Repeating sequence extracted from "Run Lola Run" movie. that has different number of frames and illumination change. These scene changes are because of shot boundary detection.

sequences and also illumination changes. Proposed method can detect these sequences since our tree based approach can tolerate keyframe numbers in sequences and SIFT features can tolerate illumination and viewpoint changes.

We can not create a ground truth for this set since it contains 7570 keyframes and does not have exact sequence boundaries. That is why, we can only give precision values on detected sequences. We detect sequences by using three different features to represent frames. These are SIFT descriptors extracted from MS and SA regions, HSV statistics and combination of these features. Results are given in Table 5.5.

Table 5.5: Sequence Detection Results for Groundhog Day.

Method	**Correct Det.**	**False Det.**	**Precision**
SIFT Descriptor	109	18	0.83
HSV Statistics	126	39	0.69
Combination	109	22	0.79

We expect to get higher performance from combination of two features when we consider results obtained from "Run Lola Run". However, we get higher performance from SIFT descriptors. ”Groundhog Day” has more illumination changes compared to ”Run Lola Run” so that HSV statistics can not contribute to results as expected in this case. Also because of these changes, SIFT performs better than HSV statistics.

5.3 Comparison with Original

We have also compared our similar set detection strategy with the original SIFT matching technique. As a first step, we try to find jump positions on SIFT matching results and this approach gives us high precision values. However, several sequences are missed. Then we choose to get top 10, 20, 30, 40 frames that have higher matching values. In these tries, again several sequences are missed and after a point precision value starts to decrease significantly. Also we see that time complexity of SIFT matching is too high. Results can be seen in Figure 5.6.

Table 5.6: Sequence detection precision and recall values on similar sets produced by SIFT matching on ”Run Lola Run” dataset that contains 1077 frames.

Method	**Sequence Count**	**Divided Count**	**Precision**	**Recall**
Jump position	14	10	0.8500	0.0392
Top 10 matches	30	17	0.8000	0.1372
Top 20 matches	41	17	0.7073	0.2352
Top 30 matches	46	16	0.7045	0.2941
Top 40 matches	75	17	0.4666	0.3333

In Table 5.6, top 40 matches give higher recall value. However, its precision is very low compared to other methods and also most of the sequences are divided. In addition to this, SIFT matching proposed by David Lowe. is too complex. Similar set construction for 1077 frames takes almost 3 weeks. When we compare SIFT matching results with other results on same dataset (Table 5.4), SIFT matching is not the best choice to create similar sets.

5.4 Weighted Combination Results

We use weighted combination to combine similar sets created by using HSV statistics and SIFT descriptors extracted from MS and SA regions. We also tested effects of weights on this combinations. Results can be seen in Table 5.7.

Table 5.7: Sequence detection precision and recall values on "Run Lola Run" small set for combination with different weights. **Detected** corresponds to detected sequence count and **Divided** corresponds to number of divided sequences.

	Weights for Cosine Measure								
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Detected	48	58	49	48	48	43	49	51	39
Divided	12	18	6	5	0	5	7	16	7
Precision	0.72	0.72	0.79	0.89	0.95	0.83	0.85	0.84	0.82
Recall	0.45	0.47	0.64	0.74	0.90	0.60	0.68	0.52	0.49

According to Table 5.7, we can conclude that equal weights performs better than other combinations. Also if weight of cosine measure is greater than 0.5 precision values do not change so much but recall values decrease. This shows that SIFT descriptors are more descriptor compared to HSV statistics.

5.5 Parameter Testing

In our approach, there are three constraints, **Period Constraint, Self-Similarity Constraint** and **Closest-Parent Constraint**, and one **Stopping Criteria**. We have parameters for only three of them, **Self-Similarity Constraint, Period Constraint** and **Stopping Criteria**. We have tested all these parameters on small set created from "Run Lola Run". First, we tested **Self-Similarity Constraint** versus **Stooping Criteria**. Precision and recall values are given in Table 5.8 and Table 5.9, respectively. Then we tested 5 values for **Period Constraint** as given in Table 5.10.

When we check the precision values given in Table 5.8, it is seen that **Self-Similarity Constraint** and **Stopping Criteria** do not affect precision or recall

Table 5.8: Precision values for parameter testing on Run Lola Run movie.

		Self-Similarity Constraint (γ)						
		10	20	30	40	50	60	70
Stopping Criteria (σ)	2	0.86	0.90	0.91	0.91	0.9167	0.91	0.91
	3	0.84	0.92	0.91	0.91	0.91	0.91	0.91
	4	0.83	0.94	0.95	0.95	0.95	0.95	0.95
	5	0.83	0.94	0.95	0.95	0.95	0.95	0.95
	6	0.83	0.94	0.95	0.95	0.95	0.95	0.95

Table 5.9: Recall values for parameter testing on Run Lola Run movie.

		Self-Similarity Constraint (γ)						
		10	20	30	40	50	60	70
Stopping Criteria (σ)	2	0.64	0.76	0.72	0.72	0.72	0.72	0.72
	3	0.74	0.86	0.82	0.84	0.84	0.84	0.84
	4	0.72	0.90	0.88	0.90	0.90	0.90	0.90
	5	0.72	0.90	0.88	0.90	0.90	0.90	0.90
	6	0.72	0.90	0.88	0.90	0.90	0.90	0.90

Table 5.10: Precision, recall values and divided sequence counts for testing **Period Constraint** δ parameter.

		Precision	**Recall**	**Divided Count**
Period Constraint (δ)	1	0.8254	0.3333	34
	2	0.8750	0.3921	35
	3	0.9482	0.6666	21
	4	0.9583	0.9019	0
	5	0.6500	0.4313	4

values after a certain value. However, they affect running time of algorithm significantly. According to Table 5.8, we can conclude that **Stopping Criteria** can be defined as any value greater than 4. Also we have used **Stopping Criteria** as 4 and **Self-Similarity Constraint** as 50 for **Period Constraint** test.

In **Period Constraint** test, which is given in Table 5.10, we see that the best result is obtained when δ is set to 4. This value is related with characteristic of dataset. If there are big gaps or frame order changes in sequences, **Period Constraint** (δ) can be set to higher values. For this set, we can say that there is no gap or frame order change in sequences greater than 4.

5.5.1 Complexity Analysis

We did not analyse prepossessing part of our algorithm since it is done only once. However, we give some numeric values on feature extraction in Table 5.11. From these values, we can conclude that although SIFT descriptors are more descriptive than HSV statistics, its time complexity is higher that extracting HSV statistics. Prepossessing part includes shot boundary and keyframe extraction, local interest points and descriptor extraction, visual term preparation and tfidf calculation. After reprocessing step, our tree-based approach creates a tree for each image in our dataset. Sequences are found according to created trees. If a frame is a member of a sequence length m and maximum number of similar frames for one frame is d then running time for creation of one tree is $O(m * d * log(d * m))$. If frame is not a member of a sequence, tree creation takes $O(d * log(d))$. In our approach, we create a tree for each frame so that running time of tree creation is $O(N * m * d * log(m * d))$, where N is the number of images in our database, d is the maximum number of similar images for one image and m is the length of longest sequence. Our method's space complexity is O($m * d$) since we need to store only one tree of a keyframe at a time.

However, note that while N is in the order of 6000, d is in the order of 50 and m is in the order of 15. For Run Lola Run which contains 5922 keyframes, the algorithm run on a P4 1 GH machine in 1148 seconds.

Table 5.11: Running times of feature extractions for 1000 frames.(Lowe regions corresponds to regions extracted by using DoG.)

Feature	**Time (seconds)**
Detection of SA regions	320
Detection of MS regions	50
Detection of Lowe regions	750
Extraction of Descriptors	1000
Extraction of HSV statistics	50

Chapter 6

Conclusion and Future Work

6.1 Summary and Discussion

In this study, we propose a method to search for the similar instances of a sequence inside a long video. Unlike most of the current studies on video copy detection and media tracking, the proposed method is robust to view point and illumination changes which may occur since the sequences are captured in different times with different cameras, and to the differences in the order and the number of frames in the sequences which may appear due to editing. The algorithm does not require any query to be given for searching, and finds all repeating video sequences inside a long video or a video collection in a fully automatic way.

Our algorithm considers both the temporal order of videos and also the similarities of keyframes. We only use visual information extracted from keyframes. Features used in our approach are chosen to be robust to viewpoint and illumination changes. Our method is independent of features extracted from keyframes and similar set construction technique. These parts can be changed with any method.

Experiments are carried out on two feature movies, commercials of TRECVID 2004 and CIVR2007 Copy Detection Showcase dataset. For copy detection, we can achieve %93 precision rate, for media tracking, we obtain %98 precision rate and for story tracking, we can achieve a precision rate higher than %80.

According to results given Chapter 5, we can conclude that features to represent frames should be chosen according to characteristics of datasets. If we know that datasets include mostly commercials then HSV statistics or other colour features can be used. This also reduce running time of algorithm. However, if datasets contain viewpoint or illumination changes then SIFT descriptors should be chosen to represent frames since SIFT descriptors are robust to viewpoint and illumination changes. However, using SIFT descriptors always give adequate results. Combination of HSV statistics and SIFT descriptors can give better results on datasets that have viewpoint changes and less illumination changes as ”Run Lola Run”.

Proposed tree-based approach is mainly affected by features used to find similar sets. That is why, we compare five features as given in Section 5.1.3.1. In this experiment, combination of SIFT descriptors extracted from Maximally Stable regions and Shape Adapted regions and HSV statistics performs better than other features for ”Run Lola Run” movie. However, if time complexity is considered HSV statistics can be used to represent frames since it also gives close results to combination and its time complexity is lower than others.

In our experiments, we see that SIFT descriptors extracted by Laplacian-of-Gaussian (Maximally Stable and Shape Adapted Regions) performs better than descriptors extracted from Difference-of-Gaussian (Lowe SIFT detector). This is mostly the result of affine transformations. Since LoG is robust to affine transformations it performs better in case of viewpoint changes.

In parameter testing, we see that parameters used in our tree-based approach does not affect precision and recall values after a certain value. They only affect running time.

Keyframe selection is very important to detect sequences. However, their running time is also important. According to our experiments, we can say that our sequence detection algorithm can tolerate differences caused by our shot boundary detection algorithm. Also our algorithm is not too complex. That is why, our shot boundary detection algorithm can be used instead of a complicated one.

Results on CIVR2007 Copy Detection Showcase shows that our method is comparable with existing copy detection methods. Our method performs the same with the best of participants. We can achieve %93 precision rate for this dataset. .

The experimental studies show that, the algorithm is successful in media tracking, specifically commercial tracking, in story tracking which is a more difficult task and also in copy detection.

6.2 Future Work

Proposed system is independent of features used for similar detection. We have used representations based on $Visterm$s and HSV statistics. However, extracting especially SIFT descriptors is time consuming. In future studies, frame representation technique can be changed or enhanced with methods that run more quick than $Visterm$s and also represent frames better than our method.

Existing method can be improved by better indexing on movies of dataset for Copy Detection. By this way, instead of searching for copies of query clip in all dataset, only a small set can be searched. Indexing based on $Visterm$ distribution of movies in dataset can be used in future studies.

Our shot boundary detection method is a heuristic approach. It is obvious that if same keyframes can be extracted from all shots of videos, similar sets can be detected easily and pruned. In future studies, our shot boundary detection method can be changed with more complex algorithms to extract exact shot boundaries.

Using keyframes reduces number of frames in videos. However, some information is lost by keyframes. Our approach can be extended to use all frames of videos. In this case, representation of frames should be changed or improved by indexing since SIFT representation for all frames in video produces huge data that is almost impossible to handle without indexing.

In this study, we discard audio knowledge of videos. This can be an important clue to detect repeating sequences. In future studies, features extracted from audio data such as mel frequency cepstral coefficient (MFCC) can be added to represent frames by considering time complexity of algorithm. to

Our method can be applied to news tracking. In future studies, news can be detected by using topic start and end time of news topics and defining a new similarity values for news frames. In news, durations of same news can change significantly in different channels so that constraints in our system should be revised.

Bibliography

[1] A. Baumberg. Reliable Feature Matching Across Widely Separated Views. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 774–781, Hilton Head Island, SC, USA, 2000.

[2] A. Chowdhury, O. Frieder, D. Grossman, and M. C. McCabe. Collection Statistics for Fast Duplicate Document Detection. *ACM Trans. Inf. Syst.*, 20(2):171–191, 2002.

[3] CIVR. Video Copy Detection Evaluation Showcase http://staff.science.uva.nl/ civr2007/videocopy.php, 2007.

[4] CIVR2007 Copy Detectin Showcase http://www-rocq.inria.fr/imedia/civr-bench/st1query.html.

[5] P. Duygulu, J.-Y. Pan, and D. Forsyth. Towards Auto-Documentary: Tracking the Evolution. In *Proceedings of ACM Multimedia*, pages 820–827, New York City, NY, USA, 2004.

[6] Y. gang Jiang, X. Wei, C.-W. Ngo, H.-K. Tan, W. Zhao, and X. Wu. Modeling Local Interest Points for Semantic Detection and Video Search at TRECVID 2006. In *Proceedings of (VIDEO) TREC 2006*, 2006.

[7] J. Gauch and A. Shivadas. Finding and Identifying Unknown Commercials Using Repeated Video Sequence Detection. In *Computer Vision and Image Understanding (CVIU)*, 2006.

[8] J. M. Gauch and A. Shivadas. Identification of New Commercials Using Repeated Video Sequence Detection. In *International Conference on Image Processing*, 2005.

[9] A. Gionis, P. Indyk, and R. Motwani. Similarity Search in High Dimensions via Hashing. In *The VLDB Journal*, pages 518–529, 1999.

[10] S. J. F. Guimaraes, R. K. R. Coelho, and A. Torres. Counting of Video Xlip Repetitions Using A Modified BMH Algorrithm : Preliminary Results. In *Multimedia and Expo, 2006 IEEE International Conference on*, pages 1065–1068, 2004.

[11] A. Hampapur and R. Bolle. Feature Based Indexing For Media Tracking. In *International Conference on Multimedia and Expo*, New York, NY, USA, 2000.

[12] A. Hampapur and R. Bolle. Comparison of Distance Measures For Video Copy Detection. In *International Conference on Multimedia and Expo*, New York, NY, USA, 2001.

[13] T. C. Hoad and J. Zobel. Methods for Identifying Versioned and Plagiarized Documents. *J. Am. Soc. Inf. Sci. Technol.*, 54(3):203–215, 2003.

[14] I. Ide, H. Mo, N. Katayama, and S. Satoh. Exploiting Topic Thread Structures in a News Video Archive for the Semi-Automatic Generation of Video Summaries. In *Multimedia and Expo, 2006 IEEE International Conference on*, pages 1473–1476, 2006.

[15] P. Indyk and R. Motwani. Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. In *Proc. of 30th STOC*, pages 604–613, 1998.

[16] G. Jaffre, P. Joly, and S. Haidar. The SAMOVA Shot Boundary Detection for TRECVID Evaluation 2004. In *TREC Video Retrieval Evaluation Workshop*, pages 179–183, Gaithersburg, Maryland, USA, 2004.

[17] I.-S. Jeong and O.-J. Kwon. Video Shot Boundary Detection Using Relative Difference Between Frames. *Optical Engineering*, 42(3):604–605, 2003.

[18] H. Jiang and C.-W. Ngo. Graph Based Image Matching. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, 2004.

[19] A. Joly. New Local Descriptors Based on Dissociated Dipoles. In *Proceedings of the International Conference on Image and Video Retrieval*, 2007.

[20] V. G.-B. Julien Law-to, Olivier Buisson and N. Boujemaa. Robust Voting Algortihm Based on Labels of Behavior For Video Copy Detection. In *Proceedings of the ACM Multimedia*, 2006.

[21] Y. Ke, R. Sukthankar, and L. Huston. Efficient Near-duplicate Detection and Sub-image Retrieval. In *ACM Multimedia*, 2004.

[22] C. Kim and B. Vasudev. Spatiotemporal Sequence Matching for Efficient Video Copy Detection. In *IEEE Transactions on Circuits and Systems for Video Technology*, 2005.

[23] H. Koumaras, G. Gardikis, G. Xilouris, E. Pallis, and A. Kourtis. Shot Boundary Detection Without Threshold Parameters. In *Journal of Electronic Imaging*, volume 15, 2006.

[24] 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, MM'06*, Santa Barbara, USA, October 2006.

[25] J. Law-To, O. Buisson, V. Gouet-Brunet, and N. Boujemaa. Robust Voting Algorithm Based on Labels of Behavior for Video Copy Detection. In *MULTIMEDIA '06: Proceedings of the 14th annual ACM international conference on Multimedia*, pages 835–844, New York, NY, USA, 2006. ACM Press.

[26] S. Lazebnik, C. Schmid, and J. Ponce. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2169–2178, Washington, DC, USA, 2006. IEEE Computer Society.

[27] T. Lindeberg. Feature Detection with Automatic Scale Selection. *International Journal of Computer Vision*, 30(2):77–116, 1998.

[28] T. Lindeberg and J. Garding. Shape-adapted smoothing in estimation of 3-d depth cues from affine distortions of local 2-d brightness structure. In *Proc 3rd European Conference on Computer Vision, ECCV'94*, pages 389–400, 1994.

[29] X. Liu and T. Chen. Shot Boundary Detection Using Temporal Statistics Modeling. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, (ICASSP '02).*, volume 4, pages 3389–3392, 2002.

[30] L. Lovasz and M. D. Plummer. *Matching Theory.* Elsevier Science Ltd, 1986.

[31] D. G. Lowe. Distinctive Image Features From Scale-Invariant Keypoints. In *International Journal of Computer Vision*, volume 60, pages 91–110, 2004.

[32] J. Matas, O. Chum, U. Martin, and T. Pajdla. Robust Wide Baseline Stereo from Maximally Stable Extremal Regions. In *Proceedings of the British Machine Vision Conference*, volume 1, pages 384–393, London, 2002.

[33] J. A. McHugh. *Algorithmic Graph Theory.* Prentice Hall, 1990.

[34] J. Miadowicz, J. Gauch, and A. Shivadas. Story Tracking in Video News Broadcasts. In *ACM Transactions on Multimedia Computing Communications and Applications*, 2005.

[35] K. Mikolajczyk and C. Schmid. An Affine Invariant Interest Point Detector. In *ECCV (1)*, pages 128–142, 2002.

[36] K. Mikolajczyk and C. Schmid. Scale and Affine Invariant Interest Point Detectors. *International Journal of Computer Vision*, 60(1):63–86, 2004.

[37] K. Mikolajczyk and C. Schmid. A Performance Evaluation of Local Descriptors. *IEEE Transactions on Pattern Analysis Machine Intelligence*, 27(10):1615–1630, 2005.

[38] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. V. Gool. A Comparison of Affine Region Detectors. *International Journal of Computer Vision*, 65(1/2):43–72, 2005.

[39] X. Naturel and P. Gros. A Fast Shot Matching Strategy for Detecting Duplicate Sequences in a Television Stream. In *CVDB '05: Proceedings of the 2nd international workshop on Computer vision meets databases*, pages 21–27, New York, NY, USA, 2005. ACM Press.

[40] C.-W. Ngo, W.-L. Zhao, and Y.-G. Jiang. Fast Tracking of Near-Duplicate Keyframes in Broadcast Domain with Transitivity Propagation. In *ACM International Conference on Multimedia*, 2006.

[41] D. Nister and H. Stewenius. Scalable Recognition with a Vocabulary Tree. In *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2161–2168, Washington, DC, USA, 2006. IEEE Computer Society.

[42] E. Nowak, F. Jurie, and B. Triggs. Sampling Strategies for Bag-of-Features Image Classification. In *European Conference on Computer Vision*. Springer, 2006.

[43] Y. Peng and C.-W. Ngo. Clip-based Similarity Measure for Hierarchical Video Retrieval. In *MIR '04: Proceedings of the 6th ACM SIGMM international workshop on Multimedia information retrieval*, pages 53–60, New York, NY, USA, 2004. ACM Press.

[44] Y. Peng, C.-W. Ngo, C. Fang, X. Chen, and J. Xiao. Audio Similarity Measure by Graph Modeling and Matching. In *MULTIMEDIA '06: Proceedings of the 14th annual ACM international conference on Multimedia*, pages 603–606, New York, NY, USA, 2006. ACM Press.

[45] Y.-X. Peng, C.-W. Ngo, Q.-J. Dong, Z.-M. Guo, and J.-G. Xiao. Video Clip Retrieval by Maximal Matching and Optimal Matching in Graph Theory. In *ICME '03: Proceedings of the 2003 International Conference on Multimedia and Expo*, volume 2, pages 317–320, Washington, DC, USA, 2003. IEEE Computer Society.

[46] F. Schaffalitzky and A. Zisserman. Multi-view Matching for Unordered Image Sets, or "How Do I Organize My Holiday Snaps?". In *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part I*, pages 414–431, London, UK, 2002. Springer-Verlag.

[47] J. Sivic and A. Zisserman. Video Google: A Text Retrieval Approach to Object Matching in Videos. In *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*, page 1470, Washington, DC, USA, 2003. IEEE Computer Society.

[48] C. G. Snoek and M. Worring. Multimodal Video Indexing: A Review of the State-of-the-Art. In *Multimedia Tools and Applications*, 2005.

[49] Trec Video Retrieval Evaluation
http://www-nlpir.nist.gov/projects/trecvid/.

[50] Wikimedia Foundation Inc,
http://en.wikipedia.org/wiki/rabin-karp_string_search_algorithm.

[51] W. S. Xiao. Graph Theory and Its Algorithms. In *Beijing Aviation Industrial Press*, 1993.

[52] H. Yang and J. Callan. Near-duplicate Detection by Instance-level Constrained Clustering. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 421–428, New York, NY, USA, 2006. ACM Press.

[53] W.-L. Zhao, C.-W. Ngo, H.-K. Tan, and X. Wu. Near-Duplicate Keyframe Identification with Interest Point Matching and Pattern Learning. In *IEEE Transactions on Multimedia*, 2007.