



## **Video Indexing Based on Scene Features**

**By**

**Mazen Ghazi Alwadi**

**Advisor:**

**Dr. Mohammad Al-Jarrah**

**Co-Advisor**

**Dr. Abdel-Karim Al-Tamimi**

**Program: Master of Science in Computer Engineering/**

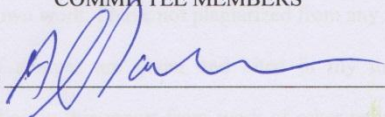
**Industrial Automation**

## Video Indexing Based on Scene Features

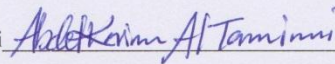
By

Mazen Ghazi Alwadi

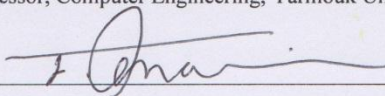
### COMMITTEE MEMBERS

Dr. Mohammad Al-Jarrah  Chairman

Associate Professor, Computer Engineering, Yarmouk University

Dr. Abdel-Karim Al-Tamimi  Member

Associate Professor, Computer Engineering, Yarmouk University

Prof. Faruq Al-Omari  Member

Professor, Computer Engineering, Yarmouk University

Dr. Ahmed Mousa  External Member

Associate Professor, Computer Engineering, Yarmouk University

May 2017

II

Declaration

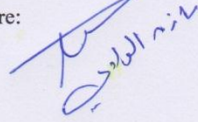
## Dedication

## Declaration

I am **Mazen Alwadi**, recognize what plagiarism is and I hereby declare that this thesis, which is submitted to the department of **Computer Engineering at Hijawi Faculty for Engineering Technology**, for the partial fulfillment of the requirements for the degree of Master of Science, is my own work. I have not plagiarized from any sources. All references and acknowledgments of sources are given and cited in my thesis. Each significant contribution to and quotation in this report from work of other people has been attributed and referenced.

Name: Mazen Ghazi Alwadi

Signature:



## Dedication

This thesis is dedicated to everyone who helped me to finish this, especially my family.

## Acknowledgments

First of all, I would like to thank my advisors Dr. Mohammad Al-Jarrah and Dr. Abdel-Karim Al-Tamimi, for their endless support and great patience.

I would like also to take this opportunity to express my gratitude to all who gave me the possibility to complete this thesis and present it in this form.

I would like to express my love and gratitude to my beloved family for their understanding and endless love throughout the duration of my studies. Special thanks to my loving parents, who always inspired and guided me throughout my life.

## Table of Contents

ABSTRACT	
CHAPTER 1	
INTRODUCTION .....	1
1.1 Introduction.....	1
1.2 Motivation.....	3
1.3 Contributions.....	3
1.4 The objectives of the study .....	4
CHAPTER 2	
BACKGROUND AND LITERATURE REVIEW .....	5
2.1 Introduction.....	5
2.2 Frameworks and tools .....	5
2.3 KAZE.....	6
2.4 Locality sensitivity hashing .....	7
2.5 Literature review .....	9
CHAPTER 3	
METHODOLOGY.....	22
3.1 Introduction.....	22
3.2 Frames extraction.....	23
3.3 Features finder.....	24
3.4 Locality sensitivity hashing .....	25
CHAPTER 4	
IMPLEMENTATION .....	27
4.1 Implementation .....	27
CHAPTER 5	
EXPERIMENTAL RESULTS AND DISCUSSION .....	33
5.1 Introduction.....	33
5.2 Single frame hash experiments .....	33
5.3 Four-frame hash experiments.....	35
5.4 Single frame Vs Four-frame .....	36
CHAPTER 6	
CONCLUSIONS AND FUTURE WORK .....	38
6.1 Conclusions .....	38
6.2 Future work .....	38

## List of Figures

Figure 1: Video hierarchical structure and keyframe extraction, retrieved from [72] .....	7
Figure 2: Presented methodology flowchart .....	21
Figure 3: Examples of partial duplicate images [6]. .....	26
Figure 4: Frame extraction application .....	30
Figure 5: Sample video frames, results of the frames extraction process. ....	31
Figure 6: Features extraction application.....	32
Figure 7: Features extractor file menu options .....	32
Figure 8: A feature list of keypoints and their structure .....	34
Figure 9: Keypoint structure and detected features for each keypoint .....	34

## List of Tables

Table 1: Selected concepts numbers and their associated objects. ....	20
Table 2: Top 10 results, single frame queries for a video from the database. ....	38
Table 3: Top 10 results, single frame queries for a video belongs to the same concept but not hashed in the database. ....	38
Table 4: Top 10 results, four-frames queries for a video hashed in the database. ....	39
Table 5: Top 10 results, four frames queries for a video belongs to a concept but not in the database. ....	40
Table 6: Single frame VS four-frames KAZE, queries from the database. ....	40
Table 7: Single frame VS four-frames KAZE, same concept queries but not in the database. .....	41



## Abbreviations

AOS	Additive Operator Splitting
Brief	Vector of image features
DB	Database
FAST	Vector of image features
FTP	File transfer protocol
KAZE	Vector of image features
KB	Kilobytes
LSH	Locality sensitive hashing
MySQL	Open source database management system
ORB	Vector of image features
SIFT	Vector of image features
SURF	Vector of image features
TB	Terabytes
TRECVID	evaluation meetings focusing on a list of different information retrieval research areas in content-based retrieval
x32	32-bit build operating systems
x64	64-bit build operating systems

## **ABSTRACT**

### **Video Indexing Based on Scene Features**

**Mazen Alwadi, Master of science in Computer Engineering/Industrial Automation Systems, Department of Computer Engineering, Yarmouk University, 2017. (Supervisor: Dr. Mohammed Al-Jarrah. Co-Advisor: Dr. Abdel-Karim Al-Tamimi).**

The explosive increase of multimedia contents due to the recent technological advances that enabled the wide usage of mobile content creating devices and the presence of ubiquitous supporting infrastructures represented by broadband Internet and social networks, has emphasized the need for effective and automatic video indexing and retrieval systems.

In this thesis, we present a video indexing system based on the extracted video scene features using KAZE features descriptor. Scene features are the distinguishable characteristics of a video scene based on its visual attributes. To increase the efficiency of the mechanism of indexing and retrieving videos based on their extracted features, we use locality sensitive hashing (LSH) approach to condense scene features into manageable data pieces that can be used to identify video scenes. We represent each video of our 900 videos dataset with two feature sets. The first set represent each second with a single frame (i.e. the first frame), and the second set represent each second with four evenly spaced frames. We test our proposed systems by querying video scenes from videos that were previously indexed , in addition to new ones. Our testing results show the effectiveness and the accuracy of our approach as we managed to always retrieve the desired stored video among the top two results, and retrieved similar videos that belong to the same or related concepts in the rest of our top ten results. Similar outcomes were achieved when querying new videos, as our system managed to retrieve videos from the exact or related video concepts in the top two results.

# CHAPTER 1

## INTRODUCTION

### 1.1 Introduction

In the recent years, we have witnessed a huge demand on multimedia over the Internet, which in its turn led to an increase of audiovisual databases. This huge amount of multimedia data represents one of the main challenging researches in the field of automatic information processing particularly in the field of video indexing. Database professionals work to index their video contents in their video databases to ensure the effective use of the video data [1].

The most common technique for multimedia retrieval is the text-based approach [2], where the file is retrieved based on predefined keywords or tags. The problem in this approach is the insufficient descriptions of the video based on the assigned keywords or tags.

Videos have much richer content than individual images, huge amount of raw data and very little prior structure which makes it very difficult to index and retrieve videos [3]. In the past, video databases have been relatively small, where indexing and retrieval have been based on keywords assigned manually. With the dramatic growth of these databases, content-based video indexing and retrieval are required, based on the video analysis with minimum human participation. Generally, videos are structured according to a descending hierarchy of clips, scenes and frames. Video analysis aims to segment videos into simpler elements that hold a semantic content which can be used to accurately index the videos [4].

Another common technique is the feature based approach [2], where each video is assigned a tag, and a description vector based on low level features such as color, texture, shape...etc.

These features are complex to extract and they describe the video content although they are not intelligible to ordinary users.

Video data exhibit strong consistency along their temporal domain, which implies the scene is visually smooth and semantically coherent. This also implies that the relevant shots tend to gather in temporal neighborhoods or even appear next to each other consecutively. However, temporal consistency provides valuable contextual clues to video analysis and tasks retrieval. In most existing approaches, the relevance of a given shot is based on its content independently from the neighboring shots [17].

In this thesis, we extracted four frames per second from each video, and then we calculated the KAZE features for these frames [7]. Then we used Locality Sensitive Hashing (LSH) to produce a similarity hash values to group similar scenes, finally we used the Jaccard distance as a similarity measure for retrieval purposes [12].

The basic ideas behind LSH are to standardize and reduce data dimensionality. In our case, the number of detected features varies from one frame to another and varies from 30 to 50 keypoint per frame. LSH uses min-hash algorithm to classify and assign hashes into buckets where similar hashes will be assigned to the same bucket, then at query time, hashes to an existing bucket points to the nearest hashes [12]. Min-hashing converts the features vector into a single hash key vector by applying a predefined number of hash functions to each value in the features vector and store them in the hash key vector. Then the same functions will be applied to the next value, in the features vector and the result will be compared with the previous vector and the minimum value will be stored in the hash key vector [6].

KAZE is a novel method used for features detection and description in nonlinear scale spaces. KAZE detects scene features and represents them as a set of keypoints. The

keypoints consists of six components which are the coordinates of each keypoint with their sub-pixel accuracy, angle, class ID, octave, response, and size. To detect these features, KAZE uses Additive Operator Splitting (AOS) and first order image derivatives. [7]

On receiving a query, a similarity measure method is used to search for the candidate videos in accordance with the query. Video similarity measure plays an important role in video retrieval. Feature matching is the most direct measure of similarity between two videos which is the distance between the features of the corresponding videos. [4]

## **1.2 Motivation**

As mentioned before the increasing amount of publicly available media content make it necessary to have an automatic way to tag and classify video contents. According to YouTube statistics for 2017 [79], 300 hours of videos are uploaded to YouTube every minute. In order to access these databases contents effectively. In addition, video indexing was promoted by TREC Video Retrieval Evaluation as one of the tasks for the years 2010-2015. Our goal is to develop an effective method to index and retrieve videos based on scene features in order improve videos tagging and retrieval mechanisms.

## **1.3 Contribution**

In this thesis, we present a method to classify and tag video contents based on their scene features. This study focuses on combining various techniques of video segmentation, feature extraction... etc, in order to achieve a highly accurate video indexing tool.

## **1.4 The Objectives of the Study**

Automatic classification of video segments is a fundamental technology for video categorization, searching, filtering and browsing. The objective of the study is to develop a

method to automatically classify and retrieve videos using scene features which represents the visual contents or multimodal concepts, to help multimedia database professionals to effectively store and retrieve their videos in a way to guarantee the effective use of available multimedia data.

## CHAPTER 2

### BACKGROUND AND LITERATURE REVIEW

#### 2.1 Introduction

Video indexing is the process in which the video contents are analyzed in order to extract a signature to characterize the video content uniqueness and is abstract enough to capture useful similarities with other video contents. Research and development in the area of video indexing falls under the domain of multimedia content analysis. [8]

Indexing a large group of images has become an interesting challenge for many multimedia-related applications. There are currently several smartphone apps that allow their users to take a photo and search a database of stored images (e.g. Snaptell, Barnes and Noble and Google Goggles app). Databases vary in size, and they can conceivably reach billions of images [8]. The ultimate goal is to identify the stored image containing the objects in a query image. This kind of applications poses three main challenges: storage, computational cost, and recognition performance [8]. For example if we consider a database that contains a billion images, and stores 100KB per image, then we need to store 100 TB of data just for their features descriptions, and the task of searching for the nearest set of features in a database of 100TB of features would take a lot of time. In this thesis, we present an innovative approach to index videos based on their 2D features using KAZE [7]. The following two examples of research activities are particularly noteworthy, 1) Since 2001, the National Institute of Standards and Technology has been sponsoring the annual Text Retrieval Conference (TREC) Video Retrieval Evaluation (TRECVID) to promote the progress in video analysis and retrieval. Since 2003, TRECVID has been independent of TREC. TRECVID provides a large-scale test collection of videos, and dozens of

participants apply their content-based video retrieval algorithms to the collection [28-30].

2) The task to guarantee the video contents description is compatible which facilitates the development of fast and accurate video retrieval algorithms. The main standards for videos are the moving picture experts group (MPEG) and the TV-Anytime Standard [65].

There exist many investigations that adopt the MPEG-7 to extract features to classify video contents or to describe video objects in the compressed domain [66]. Videos usually contain a visual channel and an auditory channel. The available information from videos includes the following: 1) video metadata, which are tagged texts embedded in videos, usually including title, summary, date, actors, producer, broadcast duration, file size, video format, copyright, etc.; 2) audio information from the auditory channel; 3) transcripts: speech transcripts can be obtained by speech recognition and caption texts can be extracted using optical character recognition techniques; 4) visual information contained in the images themselves from the visual channel [67, 68]. If a video is embedded in a web page, there are usually web page texts associated with such a video. In this proposal, we focus on the visual contents of videos. The importance and the current focus on video indexing and retrieval in the research community were the motives to many survey papers, together with the publication years and topics [67].

Generally, each paper focuses on a subset of video indexing and retrieval topics. For example, Smeaton et al. [29] give a good review of video shot boundary detection during a seven years of the TRECVID activity. Snoek and Worring [69] present a detailed review of concept-based video retrieval. They emphasize video search using semantic concepts, and the evaluation of algorithms using TRECVID databases. Ren et al. [70] review the state of the art of spatiotemporal semantic information-based video retrieval. Schoeffmann et al. [71] give a good review of interfaces and applications of video browsing.



In general, videos are organized in a hierarchy of video clips, scenes, shots, and frames. As shown in figure (2). Video structure analysis aims at segmenting a video into a number of structural elements that have semantic content including shot boundary detection, key frame extraction and scene segmentation [34].

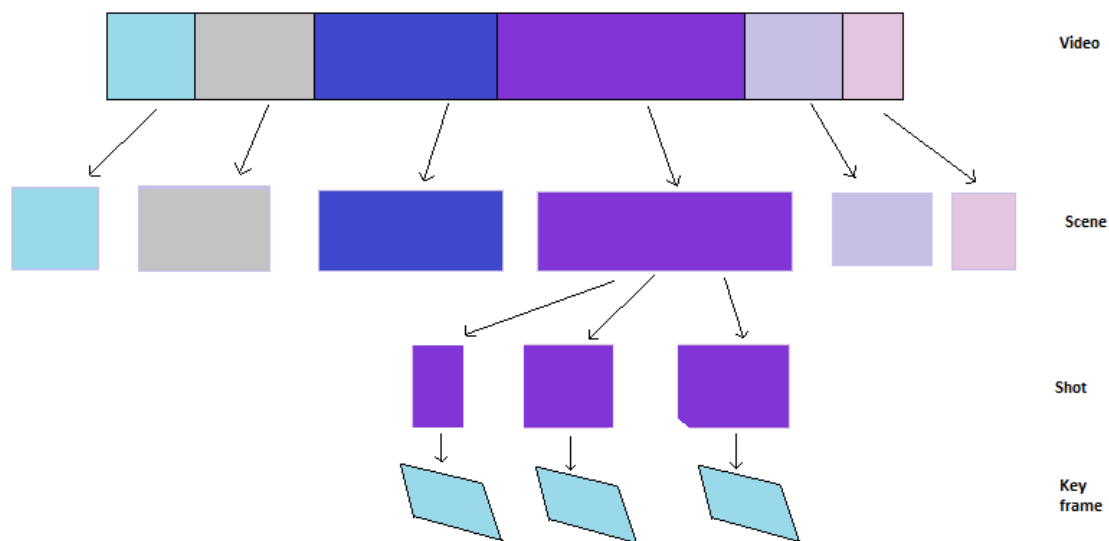


Figure 1: Video hierarchical structure and keyframe extraction, retrieved from [72]

## 2.2 Literature Review

Research in the recent years involved different approaches to help in the task of the exploration of semantic knowledge among shots for video indexing [17, 18, 19]. Yang and Hauptmann [31] used temporal consistency to sample informative examples to enhance online-learning detector accuracy; Naphade et al. [5] used inter-concept relations and temporal relationships to learn a probabilistic Bayesian network; QI et al. [19] used Gibbs random fields to integrate conceptual correlation with video feature for semantic annotation. Being successful in experiments with many concepts, these methods become impractical for applications with a large number of concepts due to the complexity inherent

in involving such relationships in the learning stage [1]. Moreover, these methods lack the flexibility when it comes to continually increasing the sizes of the training corpora [1].

Early systems, such as Informedia [20] and MARVEL multimedia analysis engine [21], proposed to combine the video modalities in order to enhance the video interpretation. The semantic pathfinder [22] explored different paths through three consecutive analysis steps which are: content analysis, style analysis, and context analysis. Other works focused on including spatio-temporal information into visual content processing in order to detect moving objects and events [23-25]. Some other alternatives have focused on using semantic knowledge to enhance the accuracy of concept detection [26].

Multimedia information indexing and retrieval [26] are required to describe, store, and organize multimedia information and to assist people in finding multimedia resources conveniently and quickly. Video is an important form of multimedia information which have the following characteristics: 1) much richer content than individual images; 2) huge amount of raw data; and 3) very little prior structure [27]. These characteristics increase the difficulty of indexing and retrieval of videos. In the past, video databases were relatively small, indexing and retrieval have been based on keywords assigned manually [1]. More recently, these databases have become much larger and content-based indexing and retrieval is required, based on the automatic analysis of videos with the minimum of human participation.

Content-based video indexing and retrieval have a wide range of applications such as quick browsing of video folders, analysis of visual electronic commerce (such as analysis of interest trends of users' selections and orderings, analysis of correlations between advertisements and their effects), remote instruction, digital museums, news event analysis [27], intelligent management of web videos (useful video search and harmful video

tracing), and video surveillance. This broad range of applications that motivates and ignites the interests of researchers worldwide [27].

As we mentioned earlier, videos are organized in a hierarchy structure of video clips, scenes, shots and frames. In the following we are going to discuss the various techniques for video segmentation including: shot boundary detection, key frame extraction and scene segmentation. Then we are going to discuss the video features including static key frame features, object features and motion features. After that we are going to discuss the video data mining and classification.

### **A. Shot Boundary Detection**

A shot is a consecutive sequence of frames captured by a camera action that takes place between start and stop operations, which mark the shot boundaries [33]. There are strong content correlations among frames in a shot. Therefore, shots are considered to be the fundamental units to organize the contents of video sequences and the primitives for higher level semantic annotation and retrieval tasks. Generally, shot boundaries are classified as a cut in which the transition between successive shots is abrupt and gradual transitions. Cut detection is easier than gradual transition detection [33].

The research on shot boundary detection has a long history, and several surveys exist on video shot boundary detection [34]. Methods for shot boundary detection usually start by extracting visual features from each frame, then measure similarities between frames using the extracted features, and finally, detecting shot boundaries between frames that are dissimilar [34]. The main three steps in shot boundary detection are: feature extraction, similarity measurement, and detection [34]. The features used for shot boundary detection include color histogram or block color histogram, edge change ratio, motion vectors,

together with more novel features such as scale invariant feature transform, corner points [35], information saliency map, etc. Color histograms have a good degree of robustness against small camera motions, but not against large camera motions, and they cannot differentiate the shots within the same scene [35]. Edge features are more invariant to illumination changes and motion than color histograms. Moreover, motion features can effectively handle the influence of object and camera motion. However, edge features and motion features as well as more complicated features cannot in general outperform the simple color histograms [34].

The second step required for shot boundary detection is done using the extracted features. Current similarity metrics for extracted feature vectors include the 1-norm cosine dissimilarity, the Euclidean distance, the histogram intersection, and the chi-squared similarity [36], as well as some novel similarity measures such as the earth mover's distance and mutual information [37].

The similarity measures include pair-wise similarity measures that measure the similarities between consecutive frames, and window similarity measures that measure similarities between frames within a window. Window based similarity measures incorporate contextual information to reduce the influence of local noises or disturbances, but they need more computation than the pair-wise similarity measures [73].

Shot boundaries can be detected using the measured similarities between frames. Current shot boundary detection approaches can be classified into threshold-based and statistical learning-based [73].

After discussing the shot boundary detection, we are going to discuss the key frame extraction techniques.

## **B. Key Frame Extraction**

Huge redundancies exist among the frames in the same shot due to the video nature and the frame rate; therefore, certain frames are selected as key frames [38], to represent the shot effectively. The extracted key frames supposed to contain as much important content of the shot as possible and reduce the redundancy as much as possible. The features used for key frame extraction include: colors (particularly the color histogram), edges, shapes, optical flow. MPEG-7 motion descriptors such as temporal motion intensity and spatial distribution of motion activity, MPEG discrete cosine coefficient and motion vectors, camera activity, and features derived from image variations caused by camera motion [39]. Current approaches to extract key frames are classified into six categories: sequential comparison-based, global comparison-based, reference frame-based, clustering based, curve simplification-based and object/event-based [74].

In the following section we are going to discuss the scene segmentation techniques, which is the last technique of video segmentation that we are going to discuss. Then we are going to discuss the static key frame features, object features and motion features.

## **C. Scene Segmentation**

Scene segmentation is also known as story unit segmentation. In general, a scene is an action in a single location and continuous time which results in contiguous shots that are coherent with a certain theme. Scenes have higher level semantics than shots. Scenes are identified or segmented out by grouping successive shots with similar content into a meaningful semantic unit. The grouping may be based on information from the audio track in the video, texts, or images. According to shot representation, scene segmentation

approaches can be classified into three categories: key frame-based, audio and visual information integration-based and background-based [76].

Extracting video features according to the video structural analysis is the base of video indexing and retrieval. Focusing on the visual features suitable for video indexing and retrieval which mainly include features of key frames, objects, and motions.

After discussing the video segmentation technique, we are going to talk about video features including the static features of key frames, object features and motion features.

#### **D. Static Features of Key Frames**

The key frames of a video characterize the video contents to some extent. Traditional image retrieval techniques can be applied to key frames to achieve video retrieval. The static key frame features useful for video indexing and retrieval are mainly classified as color-based, texture-based, and shape-based [31].

1) *Color-Based Features*: color-based features include color histograms, color moments, color correlograms, a mixture of Gaussian models, etc. According to [31], color-based features are the most effective features for video indexing and retrieval. In particular, color histogram and color moments are simple but efficient descriptors [31].

2) *Texture-Based Features*: Texture-based features are object surface-owned intrinsic visual features that are independent of color or intensity and reflect homogenous phenomena in images. They contain crucial information about the organization of object surfaces, as well as their correlations with the surrounding environment. Texture features include Tamura features, orientation features, simultaneous autoregressive models, wavelet transformation-based texture features, co-occurrence matrices, etc [31]. Amer *et al* [31] used a co-occurrence texture and Tamura features for the TRECVID-2003 video retrieval

task.

3) *Shape-Based Features*: Shape-based features that describe object shapes in the image can be extracted from object contours or regions. A common approach is to detect edges in images and then use the image histogram to describe the distribution of the edges.

After mentioning the features of the key frames, in the next section we are going to discuss the object features.

### **E. Object Features**

Object features are the features of the object which include the dominant color, size, texture, etc., of the image regions corresponding to the object shown in the video. These features can be used to retrieve videos that contain similar objects [40]. As an example, faces are unique and useful objects in many video retrieval systems. Sivic et al. [41] constructed a person retrieval system that is able to retrieve a ranked list of shots containing a particular person, given a query face in a shot. Le et al. [42] proposed a method to retrieve faces in broadcast news videos by integrating temporal information into facial intensity information. Texts in a video are extracted as one type of object to help understand video contents. Current algorithms focus on identifying specific types of objects rather than various objects in various scenes.

So far we have discussed the static features of key frames and object features, which are features that can be extracted from a single frame and represents the frame. In the next section we are going to discuss the motion features, which link multiple frames.

### **F. Motion Features**

Motion is the primary feature used to distinguish dynamic videos from still images. Motion information shows the content with some variations in time. Motion features are more of a

semantic concept than static frames or object features. Video motion includes camera motion which causes the motion in the background, and moving objects which causes the motion of the foreground. Thus, motion-based features for video retrieval can be divided into two categories: 1) camera-based and 2) object-based [43]. In camera-based features, different camera motions, such as “zooming in or out,” “panning left or right,” and “tilting up or down,” are measured and used for indexing purposes [43].

Video retrieval using only camera-based features has the limitation that they cannot describe motions of key objects. Object-based motion features have attracted much more interest in recent research work. Object-based motion features can be further classified into statistics-based, trajectory-based, and objects’ spatial relationships-based [43].

Video data mining, classification, and annotation rely heavily on the analysis of video structure and the features extraction. There are no boundaries between video data mining, video classification, and video annotation. As an example, the concept of video classification and the concept of video annotation are very similar. In this section, we review the basic concepts and approaches for video data mining, classification, and annotation. The annotation is the basis for the detection of video’s semantic concepts and the construction of semantic indices for videos.

After discussing the video segmentation techniques and the video features, we are going to discuss what are these techniques used for, which primarily are used in video data mining and classification. In the next section we are discussing the video data mining and classification.

### **G. Video Data Mining**

The task of video data mining is to find structural patterns of video contents, behavior



patterns of moving objects, content characteristics of a scene, event patterns, and their associations, and other video semantic knowledge using the extracted features, in order to achieve video intelligent applications, such as video retrieval. The choice of a strategy for video data mining depends on the application. Current strategies include object mining, special pattern detection, pattern discovery, video association mining, tendency mining and preference mining [44].

## **H. Video Classification**

The task of video classification is to find rules or knowledge from videos using extracted features or mined results and then assign the videos into predefined categories. Video classification is the key to increase video retrieval efficiency. The semantic gap between extracted formative information, such as shape, color, and texture, and an observer's interpretation of this information, makes content-based video classification very difficult. Video content includes semantic content and editing effects. Referring to [45], semantic content classification can be performed on three levels: video genres, video events, and objects in the video, where genres have rougher and wider detection range; and events and objects have thinner and limited detection range. In the following paragraphs, we are going to discuss edit effect classification, genre classification, event classification, and object classification.

1) *Edit Effect Classification*: Editing effects depend on the ways for editing videos, such as camera motion and the composition of scenes and shots. Editing effects are not a part of video content, but they affect the meaning of the video content; therefore, they may be used in video semantic classification. For instance, Ekin et al. [46] classified shots of soccer videos into long, in-field medium, close-up, and out-of-field views using cinematic features

and further detect events such as play, break, and replay.

2) *Video Genre Classification*: Video genre classification is the classification of videos into different genres such as “movie,” “news,” “sports,” and “cartoon.” Approaches to classify video genres can be classified into statistic-based, rule- or knowledge based, and machine learning-based [45].

3) *Event Classification*: An event can be defined as any human noticeable occurrence that has significance in video content representation. Each video can consist of a number of events, and each event can consist of a number of sub-events. To determine the classes of events in a video, is an important component of content-based video classification [45], and it is connected with event detection in video data mining.

4) *Object Classification*: Video object classification which is connected to object detection in video data mining is conceptually the lowest grade of video classification. The most common detected and classified object is the face [48]. Object detection often requires the extraction of structural features of objects and classification of these features. Prior knowledge about the video contents such as an object appearance is often used in the process of object feature extraction and classification.

Non-semantic-based video query types include query by example, query by sketch, and query by objects while semantic-based video query types include query by keywords and query by language [48].

Once video indices are obtained, video retrieval can be performed. On receiving a query a similarity measure method is used, based on the indices, to search for the candidate videos in accordance with the query. In the following we are going to review the query types.

1) *Query by Example*: This method uses a given query image or video to extract its features and retrieve similar videos by measuring the features similarity. Extracted features of the

key frames are suitable for this method, as the key frames of the provided query can be matched with the key frames of stored videos.

2) *Query by Sketch*: This method takes user drawn sketches as a query to represent the video they are looking for. Features extracted from the sketches are compared against the stored videos features. Hu et al. [52] proposed a method of query by sketch, where the sketch trajectories are matched to trajectories extracted from videos.

3) *Query by Objects*: This method allows the user to provide an image of an object as a query. Then, the system finds and returns all occurrences of the object in the video database [53]. In contrast with query by example and query by sketch, the search results of query by objects are not the videos, but the locations of the provided query object in the videos [53]. As an example, if we want to search for basketball related videos, we will provide an image of a basketball as the query.

4) *Query by Keywords*: This query represents the user's query by a set of keywords. Query by keywords is the most direct and the simplest type of queries, and it represents the video contents to some extent. Keywords can refer to video metadata, visual concepts, transcripts, etc [53]. As an example, if we need to find a specific song, we can use the song name, album name or the singer name as a query.

5) *Query by Natural Language*: It is the natural and direct way of making a query by describing the video contents using keywords. Aytar et al. [54] used semantic word similarity to retrieve the most relevant videos and rank them, given a search query specified in the natural language. The most difficult part of a natural language interface is the parsing of natural language and the acquisition of accurate semantics.

6) *Combination-Based Query*: This query combines different types of queries such as text-based queries and video example-based queries. The combination-based query is adaptable

to multi-model search. Kennedy et al. [55] developed a framework to automatically discover useful query classes by clustering queries in a training set according to the performance of various unimodal search methods.

Video similarity measures play an important role in content-based video retrieval. Video similarity measurement methods can be classified into four; which are: feature matching, text matching, ontology-based matching, and combination-based matching. The choice of method depends on the query type.

After discussing the query type we are going to describe the similarity measures. Video similarity measures play an important role in content based video retrieval. Methods to measure similarity can be classified into feature matching, text matching, ontology based matching, and combination-based matching.

1) *Feature Matching*: The most direct measure of similarity between two videos is the average distance between the features of the corresponding frames [56]. Query by example usually uses low-level feature matching to find relevant videos. However, video similarity can be considered in different levels of resolution or granularity [57]. According to different user' demands, static features of key frames [58], object features [59], and motion features [52] all can be used to measure video similarity.

2) *Text Matching*: Matching the name of each concept with query terms is the simplest way of finding the videos that satisfy the query. Snoek et al. [61] normalize both the descriptions of concepts and the query text and then compute the similarity between the query text and the text descriptions of concepts by using a vector space model. Finally, the concepts with the highest similarity are selected.

3) *Ontology-Based Matching*: This approach achieves similarity matching using the ontology between semantic concepts or semantic relations between keywords. Query

descriptions are enriched from knowledge sources, such as ontology of concepts or keywords [54]. Aytar *et. al* [54] utilize semantic word similarity measures to measure the similarity between text annotated videos and users queries. Videos are retrieved based on their relevance to a user-defined text queries.

4) *Combination-Based Matching*: This approach learns the combination strategies from a training collection to leverage semantic concepts, e.g., learning query-independent combination models as presented in [31] and query-class-dependent combination models [62, 63] proved to be useful for combination-based queries which adapts to multimodal searches.

This chapter reviewed the technologies and methods used previously for the purposes of video indexing and retrieval, in the next chapter we will discuss the methodology we used to create our approach of videos indexing and retrieval, and how we used the various techniques of frames extraction, features detection, minhashing, Jaccard similarity and LSH to achieve the theses goals.

## CHAPTER 3

### METHODOLOGY

#### 3.1 Introduction

In this thesis, we provide a video indexing system based on KAZE extracted video features [7]. We started by downloading the DB videos, then we extracted their videos frames as single images. Afterward, we extracted the frames features using KAZE. Then we calculated the minhashes for the features. Finally, we used LSH to classify the hashes into buckets using the Jaccard similarity measure to measure the similarity between hashes. After the LSH buckets were created, we used sample queries to test and validate our approach.

To achieve this, we selected the first thirty concepts from Columbia University video indexing dataset (FTP: dvm06.cs.columbia.edu) that have enough number of videos to train and to test our system as shown in Table 1. We downloaded the first thirty videos in each concept with a total of 900 videos for training purposes, and we downloaded an extra five videos per concept for testing purposes (totaling 150 videos).

Table 1: Selected concepts numbers and their associated objects.

Concept Number	Concept Name	Concept Number	Concept Name
Concept 1	Frame	Concept 16	Hip
Concept 2	Front	Concept 17	Hobby
Concept 3	Fruit	Concept 18	Home
Concept 4	Game	Concept 19	Hop
Concept 5	Garden	Concept 20	Horse
Concept 6	Garment	Concept 21	House
Concept 7	Girl	Concept 22	Ice
Concept 8	Glass	Concept 23	Ingredient
Concept 9	Grand	Concept 24	Instrument

Concept 10	Green	Concept 25	Japanese
Concept 11	Grill	Concept 26	Jump
Concept 12	Groom	Concept 27	Kid
Concept 13	Hair	Concept 28	Kitchen
Concept 14	Hand	Concept 29	Kitten
Concept 15	High	Concept 30	Landscape

As shown in figure (2), we started our work by downloading and gathering the DB videos and classifying them into concepts as mentioned in table 1.

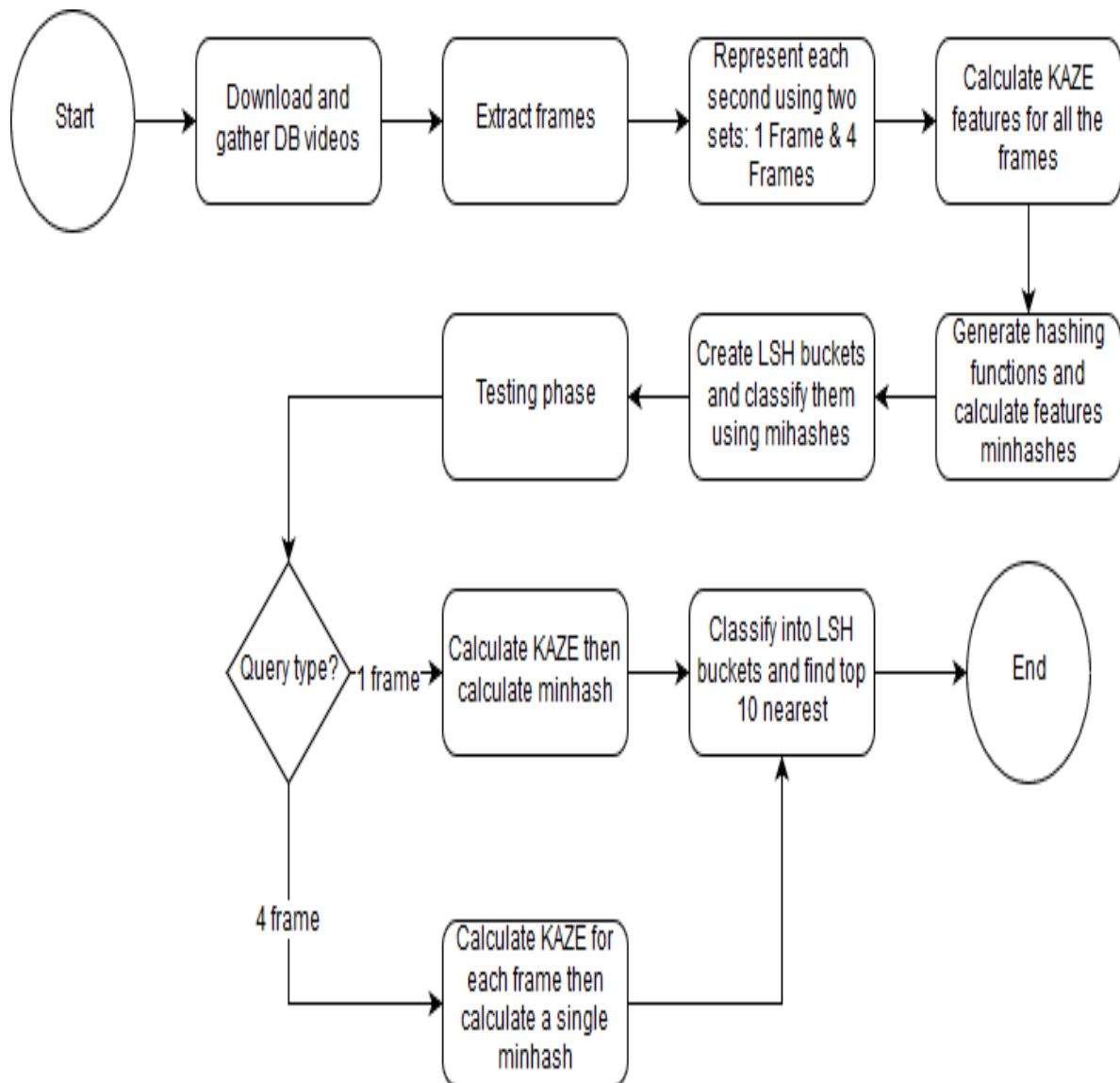


Figure 2: Presented methodology flowchart

Then we created a frames extraction application using the AForge.NET framework [10] and extracted the videos frames, then we classified the frames into two groups, the first group contains all the video frames while the second group contains four frames per second. After that we created the features detection application using EMGU.CV framework and calculated the features for the extracted frames of the DB videos and stored the features in DB tables. Then, we created min hash functions and calculated the min-hash values for the frames features and stored them in another DB table for comparison and retrieval. After the min hashes were created, we applied LSH to create buckets and assign DB hashes into buckets. Finally we used a query hashes for testing, the testing included two phases, in the first phase we used a single frame hashes for testing purposes while in the second phase we used four-frame hashes to obtain the results. We used queries from the DB and others from the same concepts but were not included in the DB for both phases. After describing our methodology in brief, in the following sections we are going to discuss the details of KAZE features, frames extraction, features finder, and LSH.

### **3.2 KAZE**

KAZE is a novel method used for features detection and description in nonlinear scale spaces [7]. KAZE detects the features and represents them as a set of keypoints. To compute the nonlinear scale space KAZE takes an approach similar to SIFT by discretizing the scale space in logarithmic steps arranged in a series of  $O$  octaves and  $S$  sub-levels. A difference worth mentioning that KAZE works without any down sampling for the octaves as done in SIFT. Then, it converts the set of discrete scale levels in pixel units to time units because nonlinear diffusion filtering is defined in time terms. In the case of the Gaussian scale space, the convolution of an image with a Gaussian of standard deviation  $q$  is



equivalent to filtering the image for some time. KAZE convolves the image with a Gaussian kernel to reduce the noise then it computes the image gradient histogram and obtains the contrast parameter in an automatic procedure. Then, given the contrast parameter and the set of evolution times it is straightforward to build the nonlinear scale space using the AOS schemes. [7]

For detecting the points of interest it computes the response of scale normalized determinant of the hessian at multiple scale levels which includes the second order horizontal, vertical and cross order derivatives. Then, it analyzes the detector response at different scale levels and searching for maxima in scale and spatial location. The position of the keypoint is estimated with sub-pixel accuracy. [11]

We described the KAZE features and how the extraction process is done, but we need to extract the video images before we can extract those features. In the next section we are going to describe the application we built to extract the frames and how the frames extraction process is done.

### **3.3 Frames Extraction**

After downloading the database videos we started the process of frames extraction, to extract the frames we built an application using AForge.NET libraries. The frames extractor connects to the video source file using a file reader and extracts the frames with the same quality of the source video file without the need of pausing or residing to screen snapshots. For our DB, the average frames per second for the videos were 24 and we chose to extract four frames per second from each video. The frame extractor extracts four frames per second from each video and stores them in a directory with the same name of the video file. The extracted frames are stored in BMP format. Extracted frames are then classified into

two groups, the first group contains all the extracted frames, and the second group contains four frames per second. Since our database contains thirty concepts and thirty video for each concept which makes it a total of 900 video files, and each video file length varies from 30 seconds to 11 minutes, our database of videos generated approximately one million frames.

When the frames extraction process is done, we can start with the features extraction process. In the following section we are going to discuss the application we built to extract the KAZE and ORB features.

### **3.4 Features Finder**

The features finder application is the application we created using the EMGU.CV libraries to detect KAZE and Orb features of the extracted frames. In our work, we have detected the mentioned features for all the frames. To extract the KAZE features of a video, we select the directory that contains the video frames and run the application, the features of each frame are represented as a list of keypoints. The detected keypoints are composed of the following:

- Key point Angle: computed orientation of the keypoint (-1 if not applicable); it's in [0,360) degrees and measured relative to image coordinate system, ie in clockwise.
- Key point class ID: object class (if the keypoints need to be clustered by an object they belong to)
- Key point octave: octave (pyramid layer) from which the keypoint has been extracted
- Key point sub-pixel coordinates.

- Key point response: the response by which the strongest keypoints have been selected. Can be used for the further sorting or subsampling.
- Key point size: diameter of the meaningful keypoint neighborhood.

After detecting the features for each frame, we had around 50 keypoints with 6 values for each keypoint and a number of frames that varies from 120 to 3200 frames per video; it would be problematic and time consuming process to brute force the video comparisons knowing that we have 36,000 to 960,000 different values for each video. To ease the process of video comparisons we used LSH. In the following section we are going to discuss the locality sensitive hashing and how we used it.

### 3.5 Locality Sensitive Hashing

Locality sensitive hashing is a hashing technique used to reduce the dimensionality of high-dimensional data. LSH hashes input items and assign them into buckets in a way so that similar items map to the same bucket. LSH differs from conventional cryptographic hash functions as it aims to maximize the probability of similar items hashing into similar hash values, which is known as collision [78]. In our work, we used minhashing to generate the hashes which are going to be used by LSH. Minhash is a standard hashing technique for discovering similar text documents or web pages [12]. Recently its variants have been successfully applied to discovering near duplicate images [13, 14]. In the min-hash algorithm a hash function is applied to all visual words in an image ignoring the location of these visual words, and the visual word with the minimum hash value is selected as a global descriptor of the given image. Min-hash is a hash function which maps a set  $I$  to some value  $v$ . A hash function is applied to each visual word in the set  $I$  and the word that has minimum hashed value is returned as the min-hash [15]. Assuming we have a set  $I$  of 100

words and we want to reduce them into a vector  $v$  of 10 values, we will need to create 10 different hashing functions, then we use the first word of the set  $I$  as input to the hashing functions, then we store the outcome of the hashing functions in the vector  $v$ . After we are done with the first word, we repeat the same process for the second word, and compare the outcome with the stored values in the vector  $v$  and keep the smaller value in the vector  $v$ . Unlike text documents which are usually represented by a set of words, images are characterized by their 2D structure. Objects within an image are often spatially localized in the image and exists strong geometric constraints among the visual words in an object.



Figure 3: Examples of partial duplicate images [6].

Figure (3) shows some examples where the objects and the duplicate regions are localized in the image.

Min-hash is a Locality Sensitive Hashing (LSH) scheme that approximates similarity between sets. When an image is represented as a set of visual words, the similarity between two images can be defined as the Jaccard similarity between the two corresponding sets of visual words which is simply the ratio of the intersection to the union of the two sets.

The brute-force approach is to use the min-hash and Jaccard similarity measure and

compare each hash in our DB with the query hash to find the most similar or the top 10 matches for the query hash, but doing this will consume a lot of time. Using Locality sensitive hashing will reduce this time dramatically. LSH takes the Min-hash values and hashes them into buckets so the similar hashes will hash into the same bucket. Buckets are organized into bands, with a number of videos per band. Videos in the same bucket in a band are then considered candidates for being similar. But before we use the LSH we need to calculate the Min-hashes for the DB features.

To calculate the minhashes we retrieve the frames features from the DB and recreate the keypoint structure to store the information, and then the hash functions are calculated once, and then applied to all keypoints. In our implementation, Universal Hashing is used in the Generate hash functions process. The generate hash functions process is passed the number of bits to represent the universe size which is the number of distinct features expected across all features. For each hash function two random numbers are used. The random function is seeded with the same value to generate the same hash functions. After the hash functions are generated the features are passed to the hash functions and every feature is hashed into a value and stored in a hash vector and compared with the next feature hash value while maintaining the minimum value as the hash value. In our work, we used 100 hash functions with a 4-byte integer for each value, but since our features are too long and there are a lot of things to be hashed, we noticed that these values will always converge to 0. Therefore we modified the hashing function to restart from the max value whenever a value reaches 0. Assuming we have a frame KAZE features of 48 points that we want to hash, the hashing will start by generating the 100 hash function and assigning the hash value of (2,147,483,647), which is the maximum value a 4-byte integer can hold; to 100 4-byte integers. Then it will hash the first keypoint of the feature with the first hash function

and compare the hashing result with the hash value vector; first integer and then it will store the minimum value in the hash vector. After that, it will keep repeating this until the first point is hashed with the 100 hash functions and compared to all 100 hash values vector. Finally it will repeat the same thing for all the keypoints in the selected frame features to be hashed.

After the min-hashes are calculated for the features in the DB, we used LSH to hash the similar hashes into buckets. Moreover to detect similar hashes, we used the Jaccard similarity measure [75].

The Jaccard similarity index can be used to represent similarity between two hashes. A value of 1 means they are identical, 0 means they are completely dissimilar, where values between 0 and 1 represent a degree of similarity. Jaccard index is defined as the size of the intersection of the two hashes divided by the size of the union.

We have described our methodology in the previous section, and how we managed to extract the frames, extract the frames features, Minhash the features, creating the LSH buckets, and using the Jaccard similarity measure to retrieve the results. In the next chapter we are going to discuss the implementation details of the applications we built for the mentioned purposes, the used framework and tools, the input and output formats.

## CHAPTER 4

### IMPLEMENTATION

In the previous chapter, we have discussed our methodology and how it works. In this chapter we are moving forward towards the implementation details of the described methodology. We are going to discuss the frameworks and tools used, the implementation details of the applications we built, and the input and output formats.

#### 4.1 Frameworks and Tools

AForge.NET is an open source C# framework designed for developers and researchers in the fields of computer vision and artificial intelligence – image processing, neural networks, genetic algorithms, fuzzy logic, machine learning, robotics, etc.[10]

AForge.NET libraries allow us to connect to video files and extract their frames with the same quality without the need to play the video, and screenshot the required frames. Through our work, we split the application into two apps as the AForge.NET framework projects needs to be built for x64 processors while the EMGU.CV is an x32 processor. The first is the frames extraction application which used AForge.NET, and the other one; the features calculation application which uses the EMGU.CV framework.

EMGU.CV as mentioned in their website is a cross platform .Net wrapper to the OpenCV image processing library. Allowing OpenCV functions to be called from .NET compatible languages. The wrapper can be compiled by Visual Studio, Xamarin Studio and Unity. It can run on Windows, Linux, Mac OS X, iOS, Android and Windows Phone. In our work, we used EMGU.CV libraries to build the application to calculate the Orb and KAZE features for the extracted frames. Having around one million frames extracted from the

selected videos will result in another one million features vector which needs a lot of storage space. MySQL is an open source relational database management system which allows us to create our DB and store all the information needed without the need for a license.

## 4.2 Implementation

In this chapter, we will describe the implementation of our work. The first thing we have done was to download and arrange the DB videos into concepts.

After we finished the downloading and classification of the video files we started the process of frames extraction. The average frame rate in the videos DB is 24 frames per second. Then, we extracted four frames to represent each second of each video. Then we classified the frames into two groups: the first group contains all the frames for each video, while the second group contains four frames representing every second of each video.

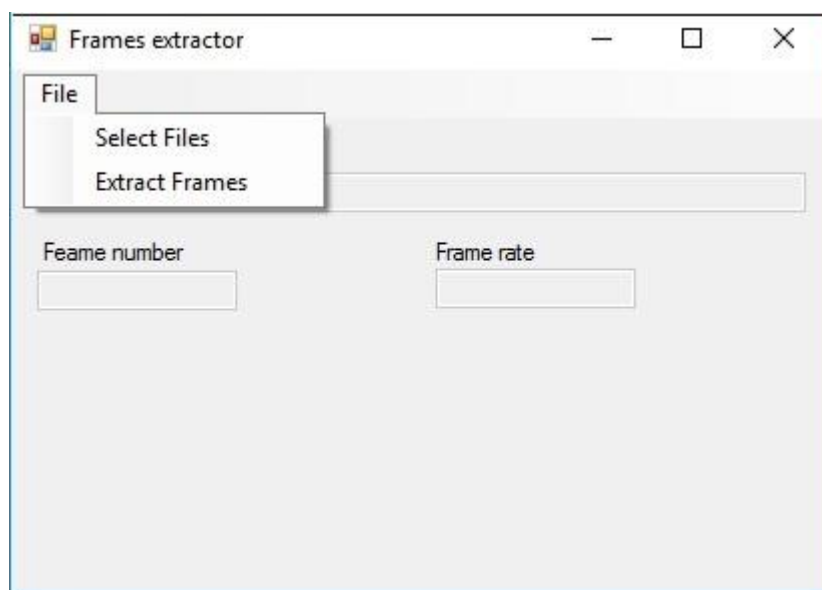


Figure 4: Frame extraction application

Figure (4) shows the developed frame extraction application. The file menu contains two options, the first option is “Select Files”, which opens an open file dialog box to select the



concept directory we want to extract its frames. The selected directory is navigated and the video files are counted by the application and the locations and names of the videos are stored in a list. Then using the second option “Extract Frames”, the application starts by the first video in the list and connects to the video using AForge.NET libraries and extract the frames of the video. When the application starts extracting a video frames it searches for a matching directory name and if does not exists it creates a directory with the same name in the same location and starts saving the frames in that directory.

The disabled text boxes in the application, shows the name of the video file which is been extracted, the current frame number, and the frame rate of the video.



Figure 5: Sample video frames, results of the frames extraction process.

Figure (5) shows a sample of a video frames after been extracted by the application, as we can see in the figure the frames are named according to their respective number as they appeared in the original video.

When the process of frames extraction was done we started the process of features extraction, to do so we built the features extraction application.

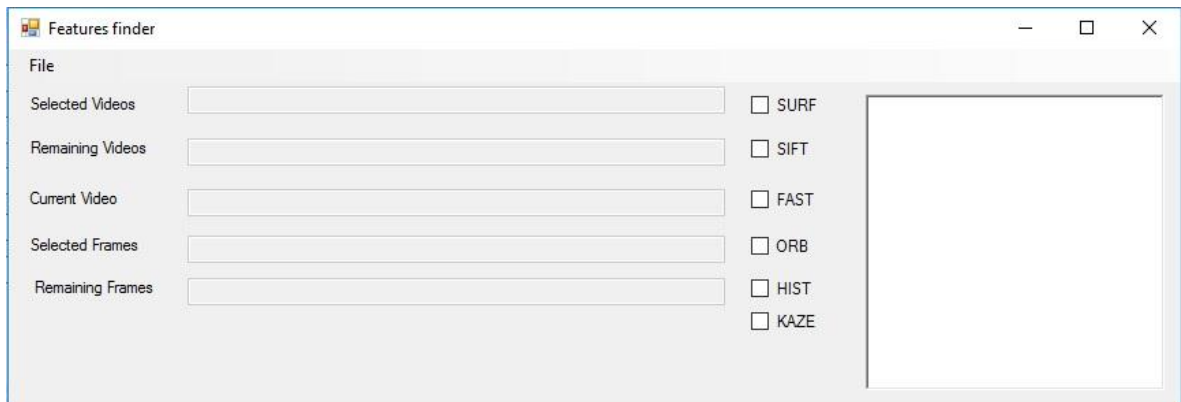


Figure 6: Features extraction application

Figure (6) shows the features extraction application. As we can see in the figure, the “Selected Videos” text box shows the number of videos selected to extract their features. While the “Remaining Videos” text box shows the number of remaining videos for the extracting process. “Current Video” text box shows the name of the video that its features are being extracted at the moment. “Selected Frames” text box shows the number of frames that were extracted for the selected video. ”Remaining Frames” text box shows the number of remaining frames of the selected videos that their frames features still need to be extracted. The text box located on the right, is for the results showing when we are testing, it shows the top 10 matches of the selected query and the concept of each one.

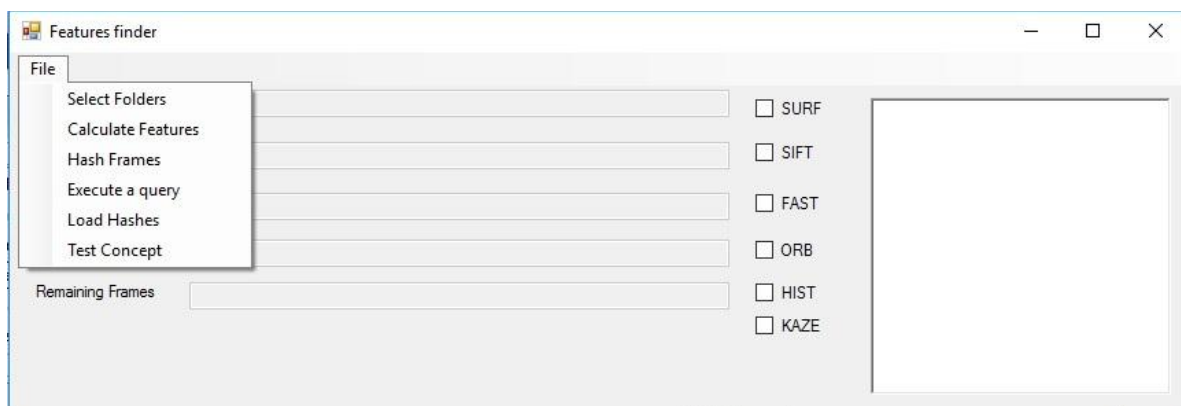


Figure 7: Features extractor file menu options

Figure (7) shows the “File” menu of the features extractor application. The menu contains six options which work as follows:

*Select Folders:* This option opens a “Select Folder Dialog” from which you can select a single director or multiple directories, the selected directories are then searched for video files and then added to a list to be processed.

*Calculate Features:* This option will go through the list of selected videos and select the frames of videos, and then it will calculate the features as selected from the right check boxes.

*Hash Frames:* Will calculate the minhashes of the selected videos frames and store them into the DB.

*Execute a query:* Allows us to select a query for testing.

*Load Hashes:* Will load the minhashes from the DB into the application for LSH buckets creation process.

*Test Concept:* Tests the query sample against the LSH buckets.

In our work we calculated the KAZE features for the DB videos.

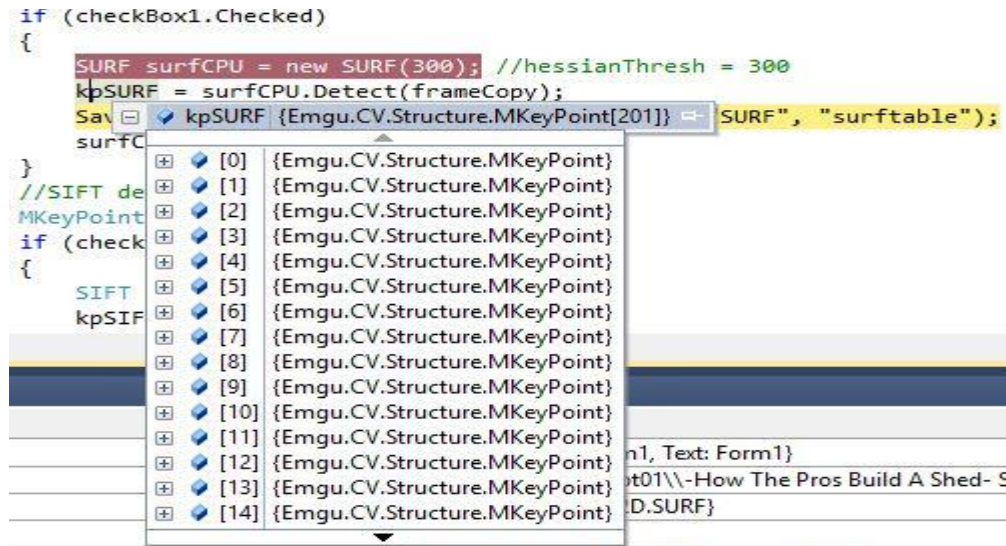


Figure 8: A feature list of keypoints and their structure

Figure (8) shows the detected SURF features of a sample frames, as we can see in the figure the features are represented as a list of keypoints.

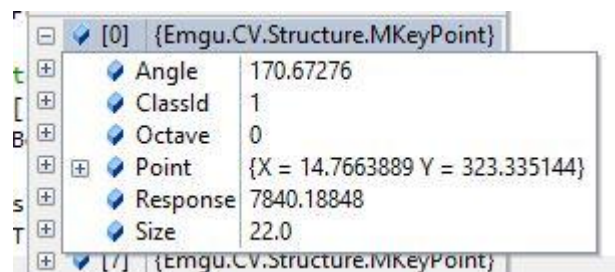


Figure 9: Keypoint structure and detected features for each keypoint

Figure (9) shows the structure of the keypoint which contains six values, to store the keypoints in the DB we serialized the keypoints list into a string separated by “;” between the keypoint values and “;” between each keypoint.

When we finished the features extraction of the DB videos, we moved to the minhashes generating phase. In order to generate the minhashes we used Universal hashing [77] to generate the hash functions which will be used to hash the features.

The hash functions are calculated once, and then applied to all DB features. In the used implementation, Universal hashing is used in the GenerateHashFunctions method. This

method is passed the number of bits to represent the universe size ( $u$ )- which is the number of distinct features expected across all features. For each hash function two random numbers are used. The random function is seeded with the same value to generate the same hash functions. After the hash functions are generated the features are passed to the hash functions and every feature is hashed into a value and stored in a hash vector and compared with the next feature hash value while maintaining the minimum value as the hash value. In our work, we used 100 hash functions with a 4-byte integer for each value. but since our features are too long and there are a lot of things to be hashed we noticed that the values will always converge to 0 so we modified the hashing function to restart from the max value whenever a value reaches 0. Assuming we have frame KAZE features of 48 points that we want to hash, the hashing will start by generating the 100 hash function and assigning the hash value of 2,147,483,467 to 100 4-byte integers, which is the largest number a 4-byte integer can hold. Then it will hash the first keypoint of the feature with the first hash function and compare the hashing result with the hash value vector first integer. And then store the minimum value in the hash vector, then it will keep repeating this until the first point is hashed with the 100 hash functions and compared to all 100 hash value vector. Finally, it will repeat the same thing for all the keypoints in the selected frame features to be hashed.

After the minhashes has been calculated for all the frames, we started the LSH buckets classification and creation process. In order to create the LSH buckets, we loaded the hashes into our developed application and classified them into buckets that contain similar hashes, the similarity measure we used to classify the hashes is the Jaccard similarity coefficient [75].

Jaccard similarity coefficient is defined as the size of the intersection divided by the union

of the two hashes. Jaccard values varies between 0 and 1, where a value of 0 means the hashes are completely different and a value of 1 means the hashes are identical.

Considering a hash A and a hash B, the Jaccard similarity coefficient equation will look like this:  $J = \text{Intersection (A, B)} / \text{Union (a, B)}$  [75]

After discussing the methodology and the implementation details, we will describe the tests and the results of our work in the next chapter.

## CHAPTER 5

### EXPERIMENTAL RESULTS AND DISCUSSION

In the previous chapter, we discussed the implementation details of the frames extractor and the feature finder applications. In this chapter we are going to present and discuss the tests and the results of our work. To test our results we used 10 queries to test each concept, the queries came from videos hashed into the database, and videos from the same concepts but not hashed into the database.

#### 5.1 Introduction

Our experiments are divided into two stages: in the first stage, we used the hashes of single frames to search for similar frames and videos search. In the second stage, we used the hashes of four frames to represent one second of the video; and compare the hash of a single second from the query with the hashes in the database, to retrieve the top ten matches for the query. Our test includes queries of videos that have been hashed into the database and other videos that belong to the same category but have not been hashed into the database. As we mentioned before the database contains 30 concepts and 30 videos for each concept which makes it a total of 900 videos. We used five videos that are already hashed for each concept test and another five videos that have not been hashed to obtain our results which make a total of 150 tests for in database videos and another 150 tests for out of the database videos.

#### 5.2 Single Frame Hash Experiments

In this stage, we started by loading the single frames hashes from the database and creating the LSH buckets and then selecting a video which has already been hashed to the database.

Table 2: Top 10 results, single frame queries for a video from the database.

Result	KAZE
1 <sup>st</sup>	100% Same video
2 <sup>nd</sup>	100% Same video
3 <sup>rd</sup>	67% Same video – 33% Same Concept
4 <sup>th</sup>	88% Same Concept- 12% Same video
5 <sup>th</sup>	100% Same Concept
6 <sup>th</sup>	100% Same Concept
7 <sup>th</sup>	100% Same Concept
8 <sup>th</sup>	94% Same Concept – 6% Others
9 <sup>th</sup>	83% Same Concept – 17% Others
10 <sup>th</sup>	76% Same Concept- 34% Others

Table 2 shows the results of the experiments for single frame hashes on queries that had been already hashed into the DB. The results show the results for the tests for each concept where the top 10 matches are considered. As the table shows, our application successfully identified the query video where the first two matches were pointing to the same video in 100% of the tests, and most of the top 10 matches belong to the same concept.

In the second test we used videos that belong to the same concept but were not hashed into the database. Concepts such as concept17 and concept 20 (Hobby- Horse), concept16 and concept19 (Hip-Hop), concept12 and concept13 (Groom- Hair), concept5 and concept10 (Green-Garden)...etc, May appear together in the same video but classified in a single concept due to the main theme of the video.

Table 3: Top 10 results, single frame queries for a video belongs to the same concept but not hashed in the database.

Result	KAZE
1 <sup>st</sup>	100% Same Concept
2 <sup>nd</sup>	100% Same Concept
3 <sup>rd</sup>	93% Same Concept –7% Others
4 <sup>th</sup>	85% Same Concept – 15% Others
5 <sup>th</sup>	76% Same Concept – 24% Others
6 <sup>th</sup>	69% Same Concept – 31% Others
7 <sup>th</sup>	61% Same Concept – 39% Others
8 <sup>th</sup>	57% Same Concept – 43% Others
9 <sup>th</sup>	52% Same Concept – 48% Others
10 <sup>th</sup>	46% Same Concept- 54% Others



In the second experiment, which is similar to the first one, we used videos that belong to the same concept but not hashed into the database. As Table 3 shows that the first two results were 100% of the same concept and the remaining top 10 most likely belong to the same concept. Table 1 shows the concept names and we can notice that many concepts may appear in the same video, which leads to these results. The result depends on the selected frame or hash as query, if the selected query shows a strong relation to a certain concept but classified in another concept because of the whole video context such results should be expected. As we know the concepts are classified as videos but after LSH is applied each hash is assigned to a different bucket which may contain similar hashes from other videos based on the dominant features in that single frame.

### 5.3 Four-Frame Hash Experiments

In this stage, we started by loading the four frames hashes from the database and creating the LSH buckets and then selecting a query which has already been hashed to the database.

Table 4: Top 10 results, four-frames queries for a video hashed in the database.

Result	KAZE
1 <sup>st</sup>	100% Same video
2 <sup>nd</sup>	100% Same video
3 <sup>rd</sup>	73% Same video – 27% Same Concept
4 <sup>th</sup>	55% Same Concept- 45% Same video
5 <sup>th</sup>	92% Same Concept – 8% Same video
6 <sup>th</sup>	100% Same Concept
7 <sup>th</sup>	100% Same Concept
8 <sup>th</sup>	88% Same Concept – 12% Others
9 <sup>th</sup>	81% Same Concept – 19% Others
10 <sup>th</sup>	77% Same Concept- 33% Others

Table 4 shows the results of the experiments for four-frame hashes on queries that had been

already hashed into the DB. The results show the results for the tests for each concept where the top 10 matches are considered. As the table shows, our application successfully identified the query video where the first two matches were pointing to the same video in 100% of the tests, and the top 10 matches belong to the same concept in most of the tests. In the second test we used videos that belong to the same concept but were not hashed into the database.

Table 5: Top 10 results, four frames queries for a video belongs to a concept but not in the database

Result	KAZE
1 <sup>st</sup>	100% Same Concept
2 <sup>nd</sup>	100% Same Concept
3 <sup>rd</sup>	100% Same Concept
4 <sup>th</sup>	91% Same Concept – 9% Others
5 <sup>th</sup>	86% Same Concept – 14% Others
6 <sup>th</sup>	81% Same Concept – 19% Others
7 <sup>th</sup>	77% Same Concept – 23% Others
8 <sup>th</sup>	68% Same Concept – 32% Others
9 <sup>th</sup>	59% Same Concept – 41% Others
10 <sup>th</sup>	51% Same Concept- 49% Others

Table (5) shows the results of the experiments for four-frame hashes on queries that had been already hashed into the DB. The results show the results for the tests for each concept where the top 10 matches are considered. As the table shows that the application successfully identified the first three matches from the same category in 100% of tests, and the top 10 matches belongs to the same concept in most of the cases.

#### 5.4 Single Frame Vs Four-Frame

As Table 6 shows, the results for Four-Frame in DB hashes has a better performance for the first seven matches which is due to the fact that the four frames represent the video in a better way than a single frame does.

Table 6: Single frame VS four-frames KAZE, queries from the database.

Result	Single Frame	Four-Frame
1 <sup>st</sup>	100% Same video	100% Same video
2 <sup>nd</sup>	100% Same video	100% Same video
3 <sup>rd</sup>	67% Same video – 33% Same	73% Same video – 27% Same Concept
4 <sup>th</sup>	88% Same Concept- 12% Same	55% Same Concept- 45% Same video
5 <sup>th</sup>	100% Same Concept	92% Same Concept – 8% Same video
6 <sup>th</sup>	100% Same Concept	100% Same Concept
7 <sup>th</sup>	100% Same Concept	100% Same Concept
8 <sup>th</sup>	94% Same Concept – 6% Others	88% Same Concept – 12% Others
9 <sup>th</sup>	83% Same Concept – 17% Others	81% Same Concept – 19% Others
10 <sup>th</sup>	76% Same Concept- 34% Others	77% Same Concept- 33% Others

But for the last three results, the single frame has a slightly better performance which can be explained by the regrouping process done by the LSH and assigning into buckets. The regrouping process starts by creating buckets and assigning the similar hashes into the same bucket, and as we described earlier; the database concepts have some similarities among them, so it is possible to have a hash which belongs to concept17 video but assigned into a concept 20 video. In the single frame case, it would be harder to be assigned into a different bucket because consecutive frames in the same video are almost identical.

Table 7: Single frame VS four-frames KAZE, same concept queries but not in the database.

Result	Single Frame	Four-Frame
1 <sup>st</sup>	100% Same Concept	100% Same Concept
2 <sup>nd</sup>	100% Same Concept	100% Same Concept
3 <sup>rd</sup>	93% Same Concept – 7% Others	100% Same Concept
4 <sup>th</sup>	85% Same Concept – 15% Others	91% Same Concept – 9% Others
5 <sup>th</sup>	76% Same Concept – 24% Others	86% Same Concept – 14% Others
6 <sup>th</sup>	69% Same Concept – 31% Others	81% Same Concept – 19% Others
7 <sup>th</sup>	61% Same Concept – 39% Others	77% Same Concept – 23% Others
8 <sup>th</sup>	57% Same Concept – 43% Others	68% Same Concept – 32% Others
9 <sup>th</sup>	52% Same Concept – 48% Others	59% Same Concept – 41% Others
10 <sup>th</sup>	46% Same Concept- 54% Others	51% Same Concept- 49% Others

As we can see in Table 7, the results for the Four-Frame were slightly better than the results of the single frame for videos that have not been hashed into the DB, which is expected as the Four-Frame represents the video better than the single frame and reduces the likelihood of retrieving a video from a different concept than the main object in the hashed query.

## CHAPTER 6

### CONCLUSIONS AND FUTURE WORK

#### 6.1 Conclusions

KAZE features has proved to be an effective descriptor to index and retrieve videos. In our single frame and four-frame queries for in DB and not in DB tests, we found that the four-frame sets representing video seconds achieve slightly better results in most of the cases. Moreover, we noticed that the mismatching that we had in our results always points to videos where the main object is presented but were not classified in the same concept. This is the result of the fact that the concepts may appear in different videos that belong to another concept which can be described by the intra-relation between the concepts.

Finally, we can say that our work correctly identified the query video for in DB hashes, and correctly identified the query concept for videos not present in DB hashes in 100% of the test cases in the top 2 matches.

#### 6.2 Future Work

We intend to improve our work by trying to add different features to descriptive features vector such, SURF, SIFT, FAST, ORB, Brief and Histogram. Adding such features may prove helpful to improve our results. Therefore, we suggest the following as future work:

- 1- Create features vectors composed of different extracted from various features extraction method to form vectors (e.g. SURF- KAZE, SIFT-KAZE) in order to evaluate which combination will perform better for indexing tasks.
- 2- Modify the used hashing functions in the Min-Hash approach to produce a better locality sensitive hashing for similar features vectors.

## REFERENCE

1. M.-F. Weng and Y.-Y. Chuang. "Multi-cue fusion for semantic video indexing." *Proceedings of the 16th ACM international conference on Multimedia*. ACM, 2008.
2. R. Datta, J. Li, and J. Z. Wang. "Content-based image retrieval: approaches and trends of the new age." *Proceedings of the 7th ACM SIGMM international workshop on Multimedia information retrieval*. ACM, 2005.
3. D. H. Patel. "Content based Video Retrieval: A Survey." *International Journal of Computer applications* 109.13 (2015): 1-5.
4. W. Hu, et al. "A survey on visual content-based video indexing and retrieval." *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 41.6 (2011): 797-819.
5. M. R. Naphade, I. V. Kozintsev, and T. S. Huang. "Factor graph framework for semantic video indexing." *IEEE Transactions on circuits and systems for video technology* 12.1 (2002): 40-52.
6. D. C. Lee, Q. Ke, and M. Isard. "Partition min-hash for partial duplicate image discovery." *European Conference on Computer Vision*. Springer Berlin Heidelberg, 2010.
7. P. F. Alcantarilla, A. Bartoli, and A.J. Davison. "KAZE features." *Computer Vision–ECCV 2012* (2012): 214-227.
8. M. Soleymani, et al. "Corpus development for affective video indexing." *IEEE Transactions on Multimedia* 16.4 (2014): 1075-1089.
9. M. Aly, M. Munich, and P. Perona. "Indexing in large scale image collections: Scaling properties and benchmark." *Applications of Computer Vision (WACV), 2011 IEEE Workshop on*. IEEE, 2011.
10. "Aforge.NET Framework". *Aforgenet.com*. N.p., 2017. Web. 11 Apr. 2017.
11. M. Brown, D. Lowe. "Invariant Features from Interest Point Groups." *BMVC*. Vol. 4. 2002.
12. A. Broder. "On the resemblance and containment of documents." *Compression and Complexity of*

Sequences 1997. Proceedings. IEEE, 1997.

13. O. Chum, et al. "Scalable near identical image and shot detection." Proceedings of the 6th ACM international conference on Image and video retrieval. ACM, 2007.
14. O. Chum, et al. "Near Duplicate Image Detection: min-Hash and tf-idf Weighting." BMVC. Vol. 810. 2008.
15. P. Indyk, and R. Motwani. "Approximate nearest neighbors: towards removing the curse of dimensionality." Proceedings of the thirtieth annual ACM symposium on Theory of computing. ACM, 1998.
16. H. R. Naphide, and T. S. Huang. "A probabilistic framework for semantic video indexing, filtering, and retrieval." IEEE Transactions on Multimedia 3.1 (2001): 141-151.
17. J. Yang, and A.G. Hauptmann. "Exploring temporal consistency for video analysis and retrieval." Proceedings of the 8th ACM international workshop on Multimedia information retrieval. ACM, 2006.
18. M. Koskela, and A. F. Smeaton. "An empirical study of inter-concept similarities in multimedia ontologies." Proceedings of the 6th ACM international conference on Image and video retrieval. ACM, 2007.
19. G. J. Qi, et al. "Correlative multi-label video annotation." Proceedings of the 15th ACM international conference on Multimedia. ACM, 2007.
20. A. Hauptmann, et al. Informedia at TRECVID 2003: Analyzing and searching broadcast news video. Carnegie-Mellon Univ Pittsburgh PA School of Computer Science, 2004.
21. A. Natsev, et al. "IBM multimedia search and retrieval system." Proceedings of the 6th ACM international conference on Image and video retrieval. ACM, 2007.
22. C. GM. Snoek, et al. "The semantic pathfinder: Using an authoring metaphor for generic multimedia indexing." IEEE Transactions on Pattern Analysis and Machine Intelligence 28.10 (2006): 1678-1689.

23. I. Laptev, et al. "Learning realistic human actions from movies." Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on. IEEE, 2008.
24. L. Wang, et al. "Improving bag-of-visual-words model with spatial-temporal correlation for video retrieval." Proceedings of the 21st ACM international conference on Information and knowledge management. ACM, 2012.
25. M. Zampoglou, et al. "From low-level features to semantic classes: Spatial and temporal descriptors for video indexing." Journal of Signal Processing Systems 61.1 (2010): 75-83.
26. M. S. Lew, et al. "Content-based multimedia information retrieval: State of the art and challenges." ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) 2.1 (2006): 1-19.
27. Y. Peng, and C.-W. Ngo. "Hot event detection and summarization by graph modeling and matching." International Conference on Image and Video Retrieval. Springer Berlin Heidelberg, 2005.
28. P. Over, et al. "TRECVID 2009-goals, tasks, data, evaluation mechanisms and metrics." (2010).
29. A. F. Smeaton, P. Over, and A. R. Doherty. "Video shot boundary detection: Seven years of TRECVID activity." Computer Vision and Image Understanding 114.4 (2010): 411-418.
30. A. F. Smeaton, P. Over, and W. Kraaij. "High-level feature detection from video in TRECVID: a 5-year retrospective of achievements." Multimedia content analysis. Springer US, 2009. 1-24.
31. A. Amir, et al. "IBM research TRECVID-2003 video retrieval system." NIST TRECVID-2003 (2003).
32. J. Fan, et al. "ClassView: hierarchical video shot classification, indexing, and accessing." IEEE Transactions on Multimedia 6.1 (2004): 70-86.
33. C. Yeo, et al. "A framework for sub-window shot detection." Multimedia Modelling Conference, 2005. MMM 2005. Proceedings of the 11th International. IEEE, 2005.
34. J. Yuan, et al. "A formal study of shot boundary detection." IEEE transactions on circuits and systems for

video technology 17.2 (2007): 168-186.

35. X. Gao, J. Li, and Y. Shi. "A video shot boundary detection algorithm based on feature tracking." *Rough Sets and Knowledge Technology* (2006): 651-658.
36. G. Camara-Chavez, et al. "Shot boundary detection by a hierarchical supervised approach." *Systems, Signals and Image Processing, 2007 and 6th EURASIP Conference focused on Speech and Image Processing, Multimedia Communications and Services. 14th International Workshop on. IEEE, 2007.*
37. L. Bai, et al. "Video shot boundary detection using petri-net." *Machine Learning and Cybernetics, 2008 International Conference on. Vol. 5. IEEE, 2008.*
38. K. W. Sze, K. M. Lam, and G. Qiu. "A new key frame representation for video segment retrieval." *IEEE transactions on circuits and systems for video technology* 15.9 (2005): 1148-1155.
39. B. Fauvet, et al. "A geometrical key-frame selection method exploiting dominant motion estimation in video." *International Conference on Image and Video Retrieval. Springer Berlin Heidelberg, 2004.*
40. R. Visser, N. Sebe, and E. Bakker. "Object recognition for video retrieval." *International Conference on Image and Video Retrieval. Springer Berlin Heidelberg, 2002.*
41. J. Sivic, M. Everingham, and A. Zisserman. "Person spotting: video shot retrieval for face sets." *International Conference on Image and Video Retrieval. Springer Berlin Heidelberg, 2005.*
42. D. D. Le, S. Satoh, and M. Houle. "Face retrieval in broadcasting news video by fusing temporal and intensity information." *International Conference on Image and Video Retrieval. Springer Berlin Heidelberg, 2006.*
43. M. S. Dao, F. G. B DeNatale, and A. Massa. "Video retrieval using video object-trajectory and edge potential function." *Intelligent Multimedia, Video and Speech Processing, 2004. Proceedings of 2004 International Symposium on. IEEE, 2004.*
44. T. Mei, et al. "Modeling and mining of users' capture intention for home videos." *IEEE transactions on*



multimedia 9.1 (2007): 66-77.

45. Y. Yuan, Research on video classification and retrieval. Diss. Ph. D. dissertation, School Electron. Inf. Eng., Xi'an Jiaotong Univ., Xi'an, China, 2003.
46. A. Ekin, A. M. Tekalp, and R. Mehrotra. "Automatic soccer video analysis and summarization." *IEEE Transactions on Image processing* 12.7 (2003): 796-807.
47. G. Lavee, E. Rivlin, and M. Rudzsky. "Understanding video events: A survey of methods for automatic interpretation of semantic occurrences in video." *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 39.5 (2009): 489-504.
48. J. S. Boreczky, and L. D. Wilcox. "A hidden Markov model framework for video segmentation using audio and image features." *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on.* Vol. 6. IEEE, 1998.
49. L. Yang, et al. "Multi-modality web video categorization." *Proceedings of the international workshop on Workshop on multimedia information retrieval.* ACM, 2007.
50. Q. -J. Qi, et al. "Correlative multi-label video annotation." *Proceedings of the 15th ACM international conference on Multimedia.* ACM, 2007.
51. L. Hollink, M. Worring, and A. T. Schreiber. "Building a visual ontology for video retrieval." *Proceedings of the 13th annual ACM international conference on Multimedia.* ACM, 2005.
52. W. Hu, et al. "Semantic-based surveillance video retrieval." *IEEE Transactions on image processing* 16.4 (2007): 1168-1181.
53. J. Sivic, and A. Zisserman. "Video Google: Efficient visual search of videos." *Toward category-level object recognition.* Springer Berlin Heidelberg, 2006. 127-144.
54. Y. Aytar, M. Shah, and J. Luo. "Utilizing semantic word similarity measures for video retrieval." *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on.* IEEE, 2008.

55. L. S. Kennedy, A. P. Natsev, and S. F. Chang. "Automatic discovery of query-class-dependent models for multimodal search." Proceedings of the 13th annual ACM international conference on Multimedia. ACM, 2005.
56. P. Browne, and A. F. Smeaton. "Video retrieval using dialogue, keyframe similarity and video objects." Image Processing, 2005. ICIP 2005. IEEE International Conference on. Vol. 3. IEEE, 2005.
57. R. Lienhart, "A system for effortless content annotation to unfold the semantics in videos." Proc. of the IEEE Workshop on Content-based Access of Image and Video Libraries (CBAIVL'00). 2000.
58. Y. Wu, Y. Zhuang, and Y. Pan. "Content-based video similarity model." Proceedings of the eighth ACM international conference on Multimedia. ACM, 2000.
59. A. Anjulan, and N. Canagarajah. "A unified framework for object retrieval and mining." IEEE Transactions on Circuits and Systems for Video technology 19.1 (2009): 63-76.
60. W. Hu, et al. "Semantic-based surveillance video retrieval." IEEE Transactions on image processing 16.4 (2007): 1168-1181.
61. C. G. M. Snoek, et al. "Adding semantics to detectors for video retrieval." IEEE Transactions on multimedia 9.5 (2007): 975-986.
62. R. Yan, J. Yang, and A. G. Hauptmann. "Learning query-class dependent weights in automatic video retrieval." Proceedings of the 12th annual ACM international conference on Multimedia. ACM, 2004.
63. R. Yan, and A. G. Hauptmann. "A review of text and image retrieval approaches for broadcast news video." Information Retrieval 10.4-5 (2007): 445-484.
64. S. Velusamy, et al. "SPSA based feature relevance estimation for video retrieval." Multimedia Signal Processing, 2008 IEEE 10th Workshop on. IEEE, 2008.
65. F. Pereira, A. Vetro, and T. Sikora. "Multimedia retrieval and delivery: Essential metadata challenges and standards." Proceedings of the IEEE 96.4 (2008): 721-744.

66. R. V. Babu, and K. R. Ramakrishnan. "Compressed domain video retrieval using object and global motion descriptors." *Multimedia Tools and Applications* 32.1 (2007): 93-113.
67. A. F. Smeaton, "Techniques used and open challenges to the analysis, indexing and retrieval of digital video." *Information Systems* 32.4 (2007): 545-559.
68. Y. Y. Chung, et al. "Content-based video retrieval system using wavelet transform." *WSEAS Transactions on Circuits and Systems* 6.2 (2007): 259-265.
69. C. G. M. Snoek, and M. Worring. "Concept-based video retrieval." *Foundations and Trends in Information Retrieval* 2.4 (2008): 215-322.
70. W. Ren, et al. "State-of-the-art on spatio-temporal information-based video retrieval." *Pattern Recognition* 42.2 (2009): 267-282.
71. K. Schoeffmann, et al. "Video browsing interfaces and applications: a review." *Journal of Photonics for Energy* (2010): 018004-018004.
72. M. Wang and H-J Zhang. "Video Content Structuring", Available Online: [http://www.scholarpedia.org/article/Video\\_Content\\_Structuring](http://www.scholarpedia.org/article/Video_Content_Structuring).
73. M. Cooper, T. Liu, and E. Rieffel. "Video segmentation via temporal pattern classification." *IEEE transactions on multimedia* 9.3 (2007): 610-618.
74. B. T. Truong, and S. Venkatesh. "Video abstraction: A systematic review and classification." *ACM transactions on multimedia computing, communications, and applications (TOMM)* 3.1 (2007): 3.
75. N. Grattan, "Jaccard Similarity Index for Measuring Document Similarity". Available Online : <https://nickgrattan.wordpress.com/2014/02/18/jaccard-similarity-index-for-measuring-document-similarity/>
76. R. Yan, and M. Naphade. "Semi-supervised cross feature learning for semantic concept detection in videos." *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. Vol. 1. IEEE, 2005.
77. N. Grattan. "MinHash for Document Fingerprinting in C#". Available Online: <https://nickgrattan.wordpress.com/2014/02/23/minhash-for-document-fingerprinting-in-c/>
78. A. Gionis, P. Indyk, and R. Motwani. "Similarity search in high dimensions via hashing." *VLDB*. Vol. 99. No. 6. 1999.
79. D. Donchev. "YouTube Statistics 2017". Available Online: <https://fortunelords.com/youtube-statistics/>

## الملخص

مازن الوادي, ماجستير هندسة حاسوب- أتمتة صناعية, قسم هندسة الحاسبات, جامعة اليرموك, 2017.

( المشرف الرئيسي: د. محمد الجراح. المشرف المساعد: د. عبدالكريم التميمي).

لقد أكدت الزيادة الكبيرة في محتويات الوسائط المتعددة الناتجة عن التطورات التكنولوجية الأخيرة التي سهلت من الاستخدام الواسع النطاق لأجهزة إنشاء المحتوى المتنقلة ووجود البنى التحتية الداعمة المنتشرة في كل مكان ممثلة في شبكات الإنترنت ذات السرعات العالية والشبكات الاجتماعية الحاجة إلى أنظمة فعالة و أوتوماتيكية لفهرسة و إسترجاع الفيديوهات .

في هذه الأطروحة، نقدم نظام لفهرسة الفيديو استنادا إلى ميزات مشاهد الفيديو المستخرجة باستخدام تقنية KAZE في وصف ميزات الصور. ميزات مشاهد الفيديو هي الخصائص المميزة لمشهد الفيديو استنادا إلى سماته البصرية. حرصا على زيادة كفاءة آلية الفهرسة واسترجاع الفيديو استنادا على الميزات المستخرجة، فلقد استخدمنا نهج الفهرسة المبني على طريقة LSH للعمل على تكثيف ميزات المشهد وتمثيلها بقطع بيانات صغيرة ليسهل التحكم فيها واستخدامها بفاعلية لتحديد مشاهد الفيديو. لقد قمنا بتمثيل كل مقطع فيديو تابع لمجموعة الفيديوهات المستخدمة البالغ عددها 900 فيديو بمجموعتين من الميزات. تمثل المجموعة الأولى كل ثانية من الفيديوهات بإطار واحد (الإطار الأول)، وتمثل المجموعة الثانية كل ثانية بأربعة إطارات متباعدة فيما بينها بالتساوي. تم اختبار نظامنا المقترح من خلال الاستعلام عن مشاهد فيديو من مقاطع الفيديو التي تمت فهرستها سابقا، بالإضافة إلى مشاهد فيديو جديدة. وتظهر نتائج الاختبارات مدى فعالية ودقة نهجنا حيث تمكنا من استرداد الفيديو المخزن المرغوب فيه دائما من بين أول نتيجتين، واسترجاع مقاطع فيديو مماثلة تنتمي إلى نفس مفاهيم الفيديو أو مفاهيم ذات الصلة في بقية نتائجنا العشرة الأولى. وقد تم تحقيق نتائج مماثلة عند البحث عن مقاطع فيديو جديدة، حيث تمكن نظامنا من استرداد مقاطع الفيديو من مفاهيم الفيديو ذاته أو من مفاهيم ذات الصلة في أول اثنتين من النتائج.