

Improving Image to Video Matching to Support Entity Resolution with Motion Detection and Feature Extraction

H. S. Senevirathna

149232V

Faculty of Information Technology

University of Moratuwa

December 2017

Improving Image to Video Matching to Support Entity Resolution with Motion Detection and Feature Extraction

H. S. Senevirathna

149232V

Dissertation submitted to the Faculty of Information Technology, University of Moratuwa,
Sri Lanka for the partial fulfilment of the requirements of the MSc in Information
Technology

December 2017

Declaration

We declare that this thesis is our own work and has not been submitted in any form for another degree or diploma at any university or other institution of tertiary education. Information derived from the published or unpublished work of others has been acknowledged in the text and a list of references is given.

Name of Student (s)

Signature of Student (s)

Date:

Supervised by

Name of Supervisor(s)

Signature of Supervisor(s)

Date:

Dedication

To my parents.

Acknowledgements

From the numerous people who helped me in this project, I must first thank my supervisor Mr. Saminda Premarathne who suggested this research area to me and guided me throughout with his advice, encouragement, expertise and wisdom. Then I must thank my parents, husband and parents-in-law for encouraging me to extend effort and for supporting me through the numerous difficulties I faced during the course of the project, with their wisdom. Last, but not the least I must thank my fellow students, the course coordinator Mr. Sudantha B. H. and coordinating assistant Achala Subhashini for their cooperation to make this project a reality.

Abstract

The need for image based video search is increasing rapidly as today with the expansion of big data and the increasing power of hardware. But there are only a few highly successful implementations in existence. In this project I have developed a search method combining motion detection and Different Feature Detection algorithms, then evaluated the method's effectiveness and compared the two approaches of Real-time Video Search and searching against a Database of feature data taken from videos. Key frames of videos are extracted using motion detection, by the difference of consecutive key frames and the Otsu's threshold. Speeded Up Robust Features (SURF), Harris-Stephens corners with Fast Retina Keypoint (FREAK) descriptor and color features are the feature detection/description methods used for extracting features. The features extracted from key frames are matched with those of the given image and M-estimator SAmple and Consensus (MSAC) algorithm is used to find 'Affine transformations' from the matching points. Different thresholds are taken by combining the feature extraction methods for filtering the results. Two prototypes were produced for comparing searching normally and searching against a database of features. Images of cats are being used to search videos where, some of which have cats throughout, some which have intermediate intervals of cats and while others have no cats. After evaluating against sets of images of incrementing size, the search method produced an intermediate level accuracy (48.89%) of search results. Furthermore, comparing the two prototypes for 5 images and 3 sets of videos, the stored summary prototype is seen slower than the real-time video search, and a trivial difference in result statistics is found.

Table of Contents

Dedication	iii
Acknowledgements	iv
Abstract	v
Table of Contents	vi
List of Figures	vii
List of Tables	viii
Introduction	1
Developments and Challenges in Image Matching, Image Based Video Search and Video Segmentation	4
Technologies Used For Improving Image to Video Matching	15
Method of Employing Motion Detection and Feature Matching in the Prototypes	21
Designs of the Prototypes	25
Implementation	30
Evaluation	39
Conclusion and Further Enhancements	48
References	52
Appendix A - Approach	54
Appendix B - Design	55
Appendix C - Implementation	57
Appendix D - Evaluation	71
Glossary of Terms	76

List of Figures

Figure 2.1 - Architecture of an early CBVR	13
Figure 3.1 - Video components	15
Figure 3.2 - Video segmentation hierarchy	16
Figure 4.1 - Level 1 DFD of Real-time searching prototype	22
Figure 4.2 - Level 1 DFD of Stored summary searching prototype	23
Figure 5.1 - Architecture of Real-time searching prototype	25
Figure 5.2 – Activity diagram of Real-time searching prototype	26
Figure 5.3 - Architecture of Stored summary searching prototype	27
Figure 5.4 - ER diagram	28
Figure 6.1 - Screenshot of the database design (relationships)	31
Figure 6.2 - List of code components	32
Figure 6.3 - Screenshot of execution of the run-time prototype	37
Figure 7.1 - Result of searching more than 90% appearance videos	43
Figure 7.2 - Result of searching Medium maching videos	44
Figure 7.3 - Result of searching 0% maching videos	44
Figure 7.4 - Total result of searching all maching videos	45
Figure 7.5 - Total result for video sets	45
Figure 7.6 - Comparison between prototypes' runing times	46
Figure 7.7 - Comparison between prototypes' percentages of results	46
Figure B.1 – Activity diagram of Stored summary searching prototype search senario	55
Figure B.2 - Stored summary searching prototype's Add Video scenario	56

List of Tables

Table 2.1 - Comparison of researches	12
Table 6.1 - Detailed data dictionary	32
Table 7.1 - Matching results for sample images with all video sets using real-time prototype	41
Table 7.2 - Matching results for different sets using real-time prototype	42
Table 7.3 - Comparison of search times for different sets using both prototypes	42
Table 7.4 - Comparison of matching video percentages for different video sets using both prototypes	43
Table A.1 - DFD data dictionary	54
Table D.1 - Evaluation details of prototype comparison	72
Table D.2 – Evaluation details of video results for different image sets	74
Table D.3 – Evaluation details of video results for different video sets	75

Chapter 1

Introduction

1.1 Preamble

There is growing need for the development of accurate techniques in image based video searching. This project proposes a new method for this purpose by employing motion detection and exiting techniques, which is tested for real-time feature matching and matching with features found in videos which were stored in a database. This chapter includes the background and motivation information for this project, along with problem definition, hypothesis, objectives, and a brief overview of the solution.

1.2 Background of Image based Video Retrieval

In today's internet, there are vast amounts of data available and people are able to search this content using search engines within a lesser duration of time. The production and distribution of videos has achieved exponential expansion, with the advent of big data, increasing power of hardware, high speed internet, digital video production, mobile video recording and social networks. There exist large collections of videos owned by industries such as news and entertainment as well as private collections. Therefore, the need for image based video retrieval is most relevant today than ever before and the need arises for more user-friendly methods of searching videos.

Content based video indexing and retrieval is a growing field of research in the world of machine vision and data mining. Although image matching techniques are widely used today (especially in search engines), image to video matching and retrieval is not being used in the same level up to current time. But since the early days of the century, there have been researches on image based video search (Araujo et al., 2014, pp. 723–724; Sav et al., 2006, pp. 1–10; Sivic et al., 2004; Sivic and Zisserman, 2004, 2003; Yang et al., 2011; Yildirim et al., 2013). YouTube is well known for having implemented Convolutional Neural Networks (CNN) for searching videos using feature information for user given textual input. Google can search videos after having retrieved a textual description for a given image. Googling the same textual query, results in the same set of videos. One of the greatest challenges here is how to retrieve videos efficiently with a sufficient accuracy of results.

There are some existing applications and researches which address similar matters in different ways which are very complex and try to find some specific type of visual information out of a collection of videos (Hu et al., 2011; Jing et al., 2015; Patel and Meshram, 2012; Sivic and Zisserman, 2003). Zisserman and others (2003) have attempted to implement such a solution for searching videos against an image, based on bags of visual words. Hu and others (2011) and Patel and Meshram (2012) have surveyed on strategies of context based video retrieval.

It is possible to search videos either on the run-time or by storing the features in a database aiming for greater speeds. But one of the problems this presents is the high dimensionality of feature vectors. Different compression techniques address this problem using quantization (McGuinness et al., 2012) and Bags of Features (BOF) (Sivic and Zisserman, 2004) for storing feature details.

There have been different clustering attempts using Hidden Markov Models and k-means. Visual Bag of Words (VBOW) and Bag of Features use such theories (Eickeler et al., 2001).

Bayesian classification (Fergus et al., 2003) and multiclass SVM classification has been used for classification (Elnemr, 2016) mostly for classifying images than videos.

Video segmentation to acquire key frames has been done in various ways in the literature: for example, fixed time intervals between key frames and histogram based selection methods (Rathod and Nikam, 2013).

1.3 Problem and Hypothesis

As stated above, there is currently only few implementations that successfully searches videos by matching features against a user specified image. And the literature suggests that searching a database of videos is relatively less efficient.

A new method of searching videos by an image which uses motion detection based video segmentation by Otsu's thresholding and SURF, Harris-Stephens and colour features has to be evaluated. The efficiency in searching a stored database of video feature summaries can be found by building two prototypes for each of real-time searching and database searching and comparing them for speed and accuracy.

1.4 Objectives for the Project

1. To survey the literature related with image based video retrieval to select appropriate algorithms.
2. To develop two prototypes and evaluate the two techniques of real-time searching and stored video summaries.
3. To evaluate the technique of motion detection based image based video retrieval

1.5 Methodology of Image Based Video Searching

In this project, the proposed method of implementation has a number of stages: video segmentation, feature extraction, feature matching, summarization of results and displaying the results. This project proposes a motion based segmentation for video segmentation using Otsu's thresholding. SURF, Harris-Stephens and colour features are being matched between the image and video key frames. Percentage matches are used to summarize and rate the results. The video list is then displayed according to the rating. In the stored features method, image features are compared with the key frame features for each video, which are stored in a database.

Query image path and the video path are the inputs needed. MATLAB is used in the development. The yfcc100m dataset and Google were used to find the images and videos. The domain of search is limited to images/videos of cats.

1.6 Structure of the Thesis

The rest of the thesis is organized as follows. Chapter 2 critically reviews the literature on image based video search and image search and identify the research problems and algorithms. Chapter 3 is about the algorithms and technology being used. Chapter 4 present our new approach to image based video search and the comparison. Chapter 5 and Chapter 6 describe the design and implementation respectively. Chapter 7 describe the evaluations of system and the comparison. Chapter 8 notes on problems faced and further work.

1.7 Summary

This chapter gave an overall picture of the entire project presented in this thesis. As such we described the background/motivation, problem definition, hypothesis, objectives, and a brief overview of the solution. Next presents a critical review of the literature related.

Chapter 2

Developments and Challenges in Image Matching, Image Based Video Search and Video Segmentation

2.1 Introduction

The chapter one gave an overall description of the project described in this thesis. In this chapter we discuss different related researches that are related to image based video retrieval. The review of past researches has been presented here in separate sections by surveys and other researches which are separated into image matching, image based video search and video segmentation. Many researchers have conducted research on content based video retrieval methods and semantic content retrieval from videos (Sivic and Zisserman, 2003; Yildirim et al., 2013) and some have conducted surveys on existing researches (Patel and Meshram, 2012; Weiming Hu et al., 2011).

At the end, this chapter defines the research problem and identifies the technologies that can be used to address the problem.

2.2 Extracts of literature

2.2.1 Surveys

A survey on Visual Content-Based Video Indexing and Retrieval is been done by Hu and others (2011). The general strategies in visual content based video indexing and retrieval are explored by them, including methods for video structure analysis, extraction of features, video data mining, video annotation, video retrieval and video browsing (Weiming Hu et al., 2011). Weiming Hu and others present a general overview of process of video indexing and retrieval. The recent developments are discussed as components of video indexing and retrieval, and also each task, sub process and the approaches involved are discussed by them. They further go on to discuss in detail the possible future directions. This survey by Weiming Hu and others comprehensively discuss contemporary Video Indexing and Retrieval research in detail. These details are from the year 2011.

Existing content based retrieval systems are surveyed by Patel and Meshram (2012). They survey the types of features that can be extracted for indexing and retrieval. They also survey

similarity measurement methods (Patel and Meshram, 2012). To search by the contents of colours, shapes and textures, the steps used are video segmentation and feature extraction. Features may be low level or high level. The high dimensionality of a feature-vector set causes the curse of dimension problem. This is solved by dimension reduction techniques such as PCA (Principle Component Analysis). Meta learning can be used to select or combine appropriate features. The selection process can be assisted by an interactive user interface. Results of feature extraction can be compared with actual human interest as well (Patel and Meshram, 2012). This survey only covers research until 2012.

2.2.2 Video Retrieval

2.2.2.1 Ontology-Based Fuzzy Video Semantic Content Extraction

The research by Yildirim and others (2013) has explored how objects, events and concepts can be retrieved automatically. They use an ontology-based fuzzy video semantic content extraction model that employs spatial/temporal relations in event and concept definitions (Yildirim et al., 2013). This research has produced a framework which can be used in many areas. Yildirim and others (2013) have proposed a generic model: VISCOM. This provides a rule construction standard that can be used in a wide domain area and allows a user to create ontology for a given domain. Additional rule definitions can be used to lower spatial relation computation cost and define complex situations more effectively. They also show that semantic representation and extraction can be improved by adding fuzziness to class, relation and rule definitions. The developed system shows success, but further enhancements can be made by considering the viewing angle of the camera, and motions in the depth dimension (Yildirim et al., 2013).

2.2.2.2 Visual Vocabulary of Viewpoint Invariant Region Descriptor Vectors (Bag of Visual Words) For Videos

Sivic and Zisserman (2003) adopted the analogy of text retrieval used by Google in searching web documents to search for objects within videos. Viewpoint invariant region descriptor vectors and nearest neighbor matching are used in this method. A set of viewpoint invariant region descriptors is used in object representation and these vectors are quantized to perform nearest neighbor matching. Temporal continuity within a shot is used to track the regions. An inverted file has an entry for each visual word, which stores all the matches (Sivic and Zisserman, 2003). While this method enables run-time object retrieval and opening the

possibilities of latent semantic indexing and automatic clustering, currently only two films are used for illustrating the method. The visual vocabulary will have to be upgraded to include different scene types (Sivic and Zisserman, 2003).

In 2004, a research was conducted by Sivic and Zisserman using data mining techniques on configurations of viewpoint invariant regions. Here principal objects, characters and scenes are extracted by measuring the frequency of occurrence of spatial configurations (Sivic and Zisserman, 2004). The video is partitioned into shots using color histograms and motion compensated cross-correlation. The images are segmented by using a sliding region. By clustering spatial configurations, frequent co-occurring parts are identified earlier by Sivic and Zisserman (2004), rather than first detecting objects and clustering them as in earlier research (Schneiderman and Kanade, 2000; Eickeler et al., 2001; Agarwal and Roth, 2002; Fitzgibbon and Zisserman, 2002; Fergus et al., 2003). The visual descriptors of object/face/scene are invariant of (do not change with) size (scale) and affinity. The significance of a cluster is measured by the number of shots and key-frames that it covers (Sivic and Zisserman, 2004). By using this method, clustered configurations show sufficient quality. But this search is biased towards textured regions and can be extended to more extensive spatial and temporal co-occurrence searching.

Sivic and others (2004) have done a research on efficient object retrieval from videos, as fast, accurate and in the same manner as similar to the Google text search (Sivic et al., 2004). By using the visual analogy of words, videos are fetched incorporating the three component methods of viewpoint invariant region descriptors, contiguous frames within a shot and vector quantization. Irrespective of viewpoint, lighting and partial occlusion, are the objectives of the type of weak segmentation used here. Rather than semantically segmenting the image it is represented by a set of overlapping regions. Then the object is matched to the descriptor vectors of regions. Later using local spatial coherence, disambiguation is done. The descriptor vectors are quantized into visual words, so that matches can be pre-computed at run-time. Based on independent 3D rigid motion constraints, regions are tracked and tracks are grouped. The resulting matches contain no false negatives and the ranking of frames has good precision.

2.2.2.3 Video Indexing Using Neural Network Based Face Detector

A system for video indexing using face detection and face recognition methods has been developed by Eickeler and others (Eickeler et al., 2001). Here, steps followed are detecting faces, recognising them and analyzing their actions. Faces are scanned by a Neural Network

based face detector and extracted. Then using pseudo two-dimensional Hidden Markov Models and the k-means clustering algorithm, faces are recognised and clustered. Then labeling and evaluation of occurrence can be done. Face tracking simplifies the detection of the same person in consecutive frames. And this approach indexes a video sequence without any prior knowledge of the sequence. This only works for JPEG data currently and can be enhanced to work directly on MPEG data. And also can be extended to detect main characters in movies. Furthermore the functionality of face detection in compressed domain has to be done.

2.2.3 Image Retrieval

2.2.3.1 Recognition of Object Classes from Unlabeled and Un-Segmented Cluttered Scenes by Scale Invariant Learning and Probabilistic Representation

Fergus and others have a method for recognition of object classes from unlabeled and un-segmented cluttered scenes by scale-invariant learning (Fergus et al., 2003). Representation is done for object categories (as constellations of parts) by a probabilistic way for the shape, appearance, occlusion and relative scale; then detection and learning are done (Fergus et al., 2003). An object category has appearance and position. This allows considering shape variations, presence/absence of features and image clutter. Both appearance variability and shape variability are taken into account. Appearance variability is learnt simultaneously with shape. An interest operator detects regions and their scale. Also, efficiently new object categories can be learnt automatically, and training sets have a large variation of the scale of objects and clutter. Several object categories are also available in the datasets. Currently the framework is very dependent on the feature detector. In the future more classes of feature can be incorporated. The model structure can be generalized to have a multi-modal appearance density with a single shape distribution. Full affine-invariance to cater for larger viewpoint variation is also a possible extension (Fergus et al., 2003).

2.2.3.2 Bag of Visual Words Based Techniques

Elnemr (2016) has offered a new CBIR technique combining different algorithms. He has combined SURF and MSER along with Color Features (color correlograms and ICCV (Improved Color Coherence Vector)) and then Bag of Visual Words technique for quantizing the features extracted. The K-means algorithm is used for clustering the words. Later a multiclass SVM (support vector machine) is used to classify the query images. Corel-1000 and COIL-100 datasets are used for both training and testing. Then Elnemr (2016) has compared it

with three alternative techniques: first using SURF descriptors, then a combination of SURF, color correlograms and ICCV, and third a combination of MSER, color correlograms and ICCV. This framework outperforms significantly other existing systems (Velmurugan and Baboo, 2011), (Bahri and Zouaki, n.d.), (Kavitha and Sudhamani, 2013), (Kavya and Shashirekha, 2014). The proposed method shows a great increase in the retrieval precision, but takes slightly longer time than the other three alternative methods being discussed (Elnemr, 2016).

2.2.3.3 Comparison of Image SURF Feature Point Storage and Compression Methods, and a New 8-Bit Quantization Method

An efficient method for image SURF feature point storage and compression has been presented by (McGuinness et al., 2012). An entropy based 8-bit encoding of quantized feature vectors is proposed by McGuinness and others, after comparing it with other existing methods. For measuring storage consumption and disk-read efficiency, various methods for compression and storage of SURF feature points have been compared. They have compared each scheme with a baseline plain-text encoding scheme as used by many existing SURF implementations. The final proposed scheme significantly reduces both the time required to load and decode feature points, and the space required to store them on disk (McGuinness et al., 2012). Product quantization which can give a further more compact representation can be combined with the proposed approach to reduce the memory footprint (McGuinness et al., 2012), is not compared with the other methods in their research.

2.2.3.4 Assessments of Feature Detectors and Descriptors

Madbouly and others (Madbouly et al., 2015) have assessed the performance of many feature detectors and descriptors, by combining them, which were famous at that time period. Those are SURF, FAST, BRISK, Harris and MinEigen as detectors, and FREAK, SURF and BRISK as descriptors. They have been tested on images under rotate, scale constraints and distortion such as illumination on different scenes (bedroom, industrial and CALsuburb datasets). The best selected as for the detector in number of detected key-points when handling rotation, scale and illumination and not affected with scene is MinEigen, for handling rotation and scale constraint in different levels and scenes is SURF with SURF detector, and for illumination distortion in different levels is FAST/SURF and Harris/FREAK.

Low level feature extraction algorithms have been compared by (El-gayar et al., 2013). The widely used algorithms of FAST, SIFT, PCA-SIFT, F-SIFT and SURF have been compared for matching against scale changes, rotation, blur, illumination changes and affine transformations. Repeatability measurement and the number of correct matches were used as evaluation measurements. SIFT is stable but slow. F-SIFT has the best overall performance. El-gayar and others (2013) do not address single object detection.

(Khan et al., 2011) have investigated the performance of SIFT, Shorter SIFT descriptors of 64D and 96D SIFT and SURF against the datasets of David Nister, Indoor, Hongwen and Caltech. 64D and 96D SIFT perform as well as traditional 128D SIFT at the much less computational cost. SURF also gives good results (slower and less effective on scaling, large blur and viewpoint invariance). 64D SIFT is superior for viewpoint invariance and has almost three times faster image matching and half the memory requirements. 32D SIFT has underperformed.

2.2.4 Video Segmentation

S.G. Anuradha and others (2013) have worked on change detection in videos. For this purpose they provide an entropy based real-time adaptive non-parametric window thresholding algorithm. The value of scatter of sections of change in a difference image is approximated, and using entropy structure a threshold for every image block is calculated discriminatively, and then all thresholds for image blocks of the frame are averaged to obtain the global threshold. Calculation of the block threshold is done contrarily for regions of change and background (Anuradha et al., 2013). Adaptive window selection is required to automatically select window size in order to obtain an optimal result and Otsu's thresholding scheme gives threshold value for a particular window (Anuradha et al., 2013). This method outperforms the other traditional methods. It is steadier and more efficient than the spatial properties based methods (Anuradha et al., 2013).

2.3 Discussion of the Literature and Problem Definition

We can see from the above descriptions that, a number of research has been done on image based video search using different technologies.

Research	Purpose	Technology/purpose	Considerations on selecting for this project
S.G. Anuradha and others (2013)	Change detection in videos	An entropy based real-time adaptive non-parametric window thresholding algorithm using Otsu's thresholding scheme.	Change detection of this method could be used for video segmentation.
Madbouly et al. (2015)	Assessed the performance of many feature detectors and descriptors, by combining them	SURF, FAST, BRISK, Harris and MinEigen as detectors, and FREAK, SURF and BRISK as descriptors are tested on images under rotate, scale constraints and distortion such as illumination on different scene.	A combination of SURF/SURF and FAST/SURF are suitable for our application due to the speed and coverage of all aspects.
Elnemr (2016)	A new CBIR technique combining different algorithms	Combined SURF and MSER along with Color Features (color correlograms and ICCV) and then Bag of Visual Words technique	Great increase in the retrieval precision, but takes slightly longer time
Sivic and Zisserman (2003)	Analogy of text retrieval used by Google in searching web documents to search for objects within videos	A set of viewpoint invariant region descriptors for object representation and then quantization to perform nearest neighbor matching	The visual vocabulary will have to be trained to include cat figures. But it might be slightly slower to retrieve results in the implementation.

Sivic and Zisserman (2004)	Using data mining techniques on configurations of viewpoint invariant regions	Frequency of occurrence of spatial configurations, data mining techniques of clustering spatial configurations (viewpoint invariant regions)	Clustered configurations show sufficient quality, and is biased towards textured regions and can be extended. But it also might be slightly slower to retrieve results.
Sivic and others (2004)	Efficient object retrieval from videos	By using the visual analogy of words, videos are fetched incorporating the three component methods of viewpoint invariant region descriptors, contiguous frames within a shot with vector quantization.	Matches can be pre-computed at run-time and has good precision. Only object retrieval is considered. The temporal aspect searching within shots is not required here.
Yildirim and others (2013)	How objects, events and concepts can be retrieved automatically	An ontology-based fuzzy video semantic content extraction model that employs spatial/temporal relations in event and concept definitions	Yildirim and others propose semantic content extraction based on ontology (VISCUM), which is beyond the scope of this research. We only handle visual/spatial aspects relating to user given images.
Eickeler and others (2001)	Video indexing using face detection and face recognition methods	Neural network based face detector, pseudo two-dimensional hidden Markov models and the k-means clustering algorithm	Face detection is not required for this project, but two-dimensional Hidden Markov Models and the k-means clustering may be used.
Fergus and others (2003)	Object class recognition from	A Bayesian based probabilistic representation of object	Efficiently new object categories can be learnt automatically. Clustering is

	unlabeled and un-segmented cluttered scenes	classes, which are collections of parts with appearance and a position, detection and learning.	itself a time consuming task. Kadir-Brady detector used here can be used to detect salient features. There is no indication of comparison with other methods.
McGuinness et al. (2012)	Efficient method for image SURF feature point storage and compression	An entropy based 8-bit encoding of quantized feature vectors and comparison with other existing methods	This process significantly reduces both the time required to load and decode feature points. But it is hard to replicate in code.
El-gayar et al. (2013)	Comparison of low level feature extraction algorithms	FAST, SIFT, PCA-SIFT, F-SIFT and SURF have been compared for matching against scale changes, rotation, blur, illumination changes and affine transformations. Repeatability measurement and the number of correct matches are used.	F-SIFT is a good choice for feature matching since it has the best overall performance.
Khan et al. (2011)	Investigating the performance of SIFT and SURF descriptors	Shorter SIFT descriptors of reduced dimensionality, 64D and 96D SIFT were generated to compare with normal 128D SIFT and others.	64D SIFT implementation is not readily available in Matlab, but could be done with additional libraries which need to be thoroughly tested.

Table 2.1 - Comparison of researches

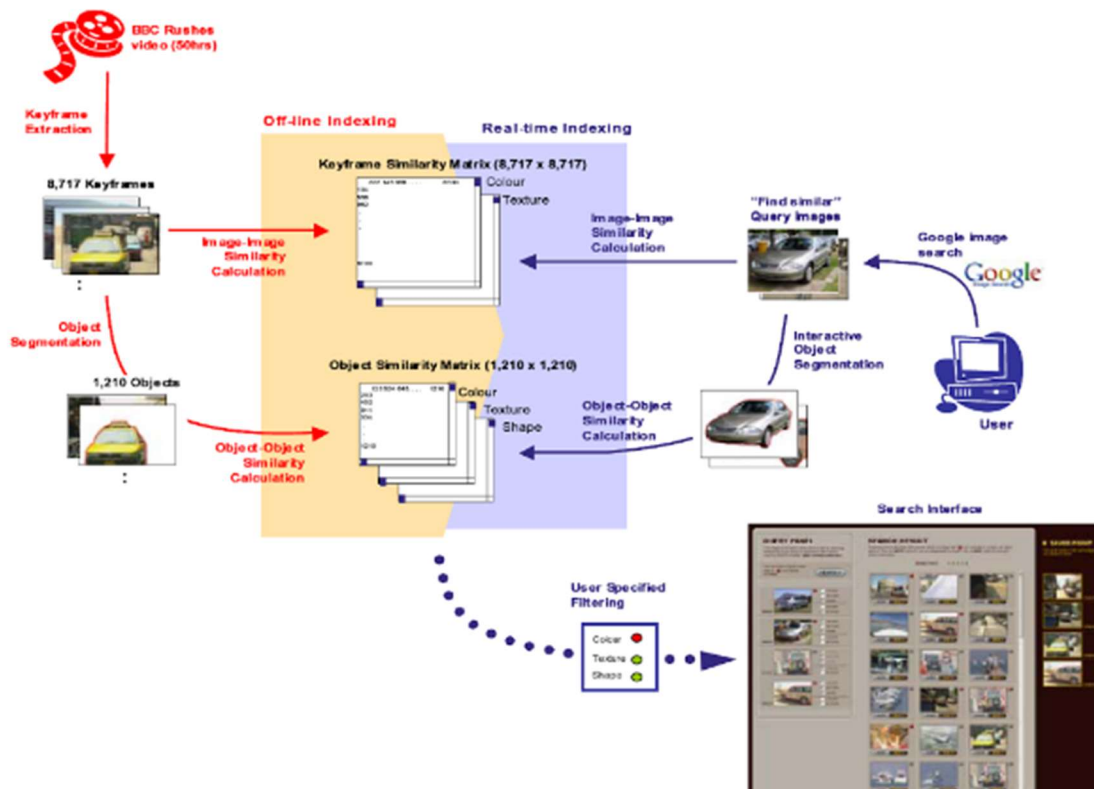


Figure 2.1 - Architecture of an early CBVR (Sav et al., 2006)

It is evident from the above summary that there is still a lot of space for new ideas. Video segmentation as discussed in S.G. Anuradha and others (2013) is complex, and only a simpler segmentation method will be used since searching a large number of videos which requires a greater speed. Otsu's thresholding scheme as used in S.G. Anuradha and others (2013) is good for video segmenting but adaptive window thresholding use there takes more time.

From the technologies used we can see that, viewpoint invariant region descriptors for building a dictionary of bags of visual features is a useful technique (Sivic and Zisserman, 2003), but is also a complex process. Ontology-based fuzzy video semantic content extraction model relates to ontology (Yildirim et al., 2013), and thus not suitable for this project. Automatic clustering and data mining is also appropriate (Sivic and Zisserman, 2004), if more than just textured region matching can be performed. But that technique is time consuming. We can see that indexing has been done by Sivic and Zisserman in 2004 and by Eickeler and others in 2001. These attempts have been successful. Eickeler and others (2001) only used indexing for faces. Temporal co-occurrence searching is not required here as done by Sivic and Zisserman in 2004.

From feature detectors/descriptors SIFT is found to be powerful (El-gayar et al., 2013), yet too slow and thus not suitable for video retrieval. Some methods such as MinEigenValue

(Madbouly et al., 2015), SIFT, 64D SIFT (Khan et al., 2011), F-SIFT (El-gayar et al., 2013) were not readily available in Matlab. Kadir and Brady is useful for salient object matching (Fergus et al., 2003). SURF is popular, available and suitable for rotation invariant matching. FAST/SURF is suitable for illumination changes matching (Madbouly et al., 2015). MSER is as good as a region detector/descriptor (Elnemr, 2016).

We can find from these researches that the stored method is less efficient than real-time retrieval, which is counter intuitive. Normally, database storage is used to increase the searching speed. This theory needed to be further tested.

Therefore it was useful to compare both stored summary method and on-the-run retrieval. The SIFT, Harris, color features, MSER feature matching and Otsu's thresholding are suitable for this project. Details of the technologies behind the solution will be discussed in Chapter 3.

2.4 Summary

This chapter presented a comprehensive literature review on the on image based video search research and identified the research problem as comparing both indexing and on-the-run retrieval. We also identified the Android technology to address the above problem. Next chapter will discuss the technology used for our solution.

Chapter 3

Technologies Used for Key Frame Extraction by Motion Detection and Feature Extraction/Matching

3.1 Introduction

Earlier we discussed the different kinds of research and in most researches, key frames are extracted which are compared to the given image using feature comparison algorithms. Thus, here we discuss different algorithms selected for use in the prototypes. First we discuss the video segmentation method. Then SURF (Speeded Up Robust Features), Harris corners and Color features extractor are discussed. The use of MSAC algorithm is also explained. The reasons for using these technologies and their advantages/disadvantages will be described. Afterwards the platform used for development, Matlab is presented in terms of pros and cons and its use for this project.

3.2 Video Segmentation with Otsu's Threshold Based on Motion Detection

It is important to select key frames since it is difficult to process all frames within a short time. Videos can be considered to be consisting of components as suggested by Patel and Meshram is shown below.

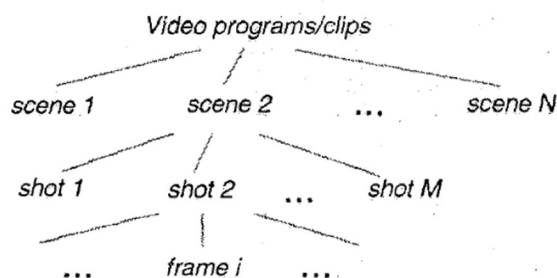


Figure 3.1 - Video components (Patel and Meshram, 2012)

Similar to the above classification, we are segmenting a video according to a hierarchy that makes it faster to compute as the same time as being able to find representative key frames.

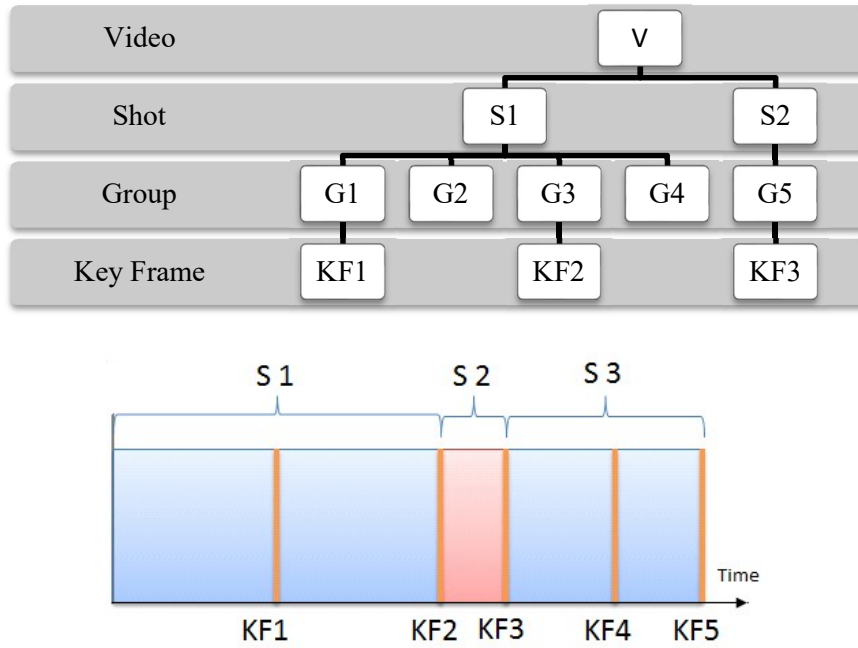


Figure 3.2 - Video segmentation hierarchy

We can consider a video to consist of shots. And a shot consists of groups of frames. A shot is a similar set of frames. So we use difference between frames to find shots. The frames with greatest difference from earlier frames become shot boundary frames. This is similar to techniques used in motion detection.

Frames are separated into groups using a formula depending on No of frames. (NOF). Frames are divided into groups of:

$$NOF / \{ \lfloor NOF / 1000 \rfloor^2 + 2 \}$$

One frame is selected for each group, in this case, the final frame of each group.

A **difference image** or the difference between selected frames is used to calculate the **Otsu's threshold**. This is based on the method used by S.G. Anuradha et al. (2013) with simplification for achieving a greater speed. Rather than calculating thresholds for foreground and background locally, we use one global threshold for each key frame.

The frames with a difference above a threshold (super global) define a shot boundary. For taking the super global threshold, out of a series of thresholds, a value is taken using the below formula suggested by Rathod and Nikam (2013):

$$T = mean + a * standard_deviation \text{ ('a' is configurable where default value = 1)}$$

Advantages:

The simplicity of the scheme makes it possible to segment the video within a sufficiently less time and find the best representative frames of all the frames of a video.

Disadvantages:

When first selecting frames by grouping them, it is possible that some frames with significant changes can be lost and some images can also be lost. The grouping equation is designed to keep the number of selected frames within a manageable range. Thus lesser and lesser number of frames are selected as the total number of frames increases. This means that only objects that are shown for a significant portion of time can be matched. Since the key frame selection algorithm selects key frames from the middle of shots, some blurred images can be selected.

3.3 SURF Feature Extraction Algorithm

SURF uses a blob detector based on the Hessian matrix for texture detection. In theory, SURF is also known as approximate SIFT and employs integral images and efficient scale space construction to generate key points and descriptors very efficiently. SURF uses two stages namely key point detection and key point description. In the first stage, integral images allow the fast computation of approximate Laplacian of Gaussian images using a box filter. Integral image is an algorithm for quick and efficient generation of the sum of values in a rectangular subset of a grid, where the value at any point (x, y) in the summed area table is just the sum of all the pixels above and to the left of (x, y) . Determinants of the Hessian matrix are then used to detect the key points. So SURF builds its scale space by keeping the image size the same and varies the filter size only.

Advantages:

This is a very efficient algorithm compared to SIFT, BRIEF and ORB. SURF is also accurate. It is good for handling rotation changes and has scale independence (in matching two images). SURF is Suitable for classification tasks. SURF is used in this project due to its speed, accuracy, rotation invariance and texture detection capabilities.

Disadvantages:

But SURF is not preferred for handling large blur, illumination and viewpoint differences (invariance) when matching two images. And it produces a less number of match points.

3.4 Harris-Corners (Harris-Stephens) Detection Algorithm with FREAK Descriptor Algorithm

Harris-corners is one of the most used corner detectors and in Matlab Harris-corners produces a **BinaryFeatures** object. In this project, **FREAK** descriptors are used after extracting Harris corner points as FREAK is the default descriptor for Harris corners in MatLab. In theory, this method directly considers the differential of the corner score with respect to direction, as an improvement to improved Moravec's Corner Detector. The surrounding area is averaged. If we use a circular window then the response will be isotropic solving one of the problems of Moravec's approach. By analysing the magnitude of the eigenvalues of the Harris matrix, we can find out if a concrete pixel has or not features of interest.

Advantages:

Harris-corners detects corners. And it is suitable for matching images with man-made structures. It can match images with rotation changes and illumination distortion in different levels.

Disadvantages:

Harris-corners is not scale independent. It is not suitable for classification tasks.

3.5 Color Features Extraction

Color feature matching can be done in various ways, and here we use a very simple method.

Step 1: Convert RGB images to the $L^*a^*b^*$ color space.

Step 2: Compute the "average" $L^*a^*b^*$ color within 16-by-16 pixel blocks.

Step 3: L2 normalize color features.

Step 4: Append the [x y] location.

Step 5: Normalize pixel coordinates.

Step 6: Concatenate the spatial locations and color features.

Step 7: Use color variance as feature metric.

* Return empty features for non-color images.

Simple **square neighborhood** method is used as the descriptor by the standard feature description method of MatLab.

Advantages:

This method is very simple to implement, and can be used to match the similarity of images.

Disadvantages:

This method is slow to execute and it is not a standard method.

3.6 MSAC to Detect Inliers

MSAC is a modification of RANSAC which is popular in the field. It is a mathematical technique and an iterative method to estimate parameters of a mathematical model from a set of observed data that contains outliers, when outliers are to be accorded no influence on the values of the estimates. It is used in computer vision to detect inliers (best matching points) and remove outliers (least matching points) between two images.

Advantages:

MSAC gives a modest to hefty benefit to all robust estimations and with lesser computational cost than MLESAC. MSAC outperforms RANSAC. For this project, this algorithm is used since it is available in Matlab and is a simple and fast way to remove outliers. MSAC can detect Affine Transformations (e.g.: translation, scale, shear and rotation).

In this research MSAC is used for finding the percentage of inliers against the total no of features in the image. This is done for all 3 feature extraction algorithms. To rank the videos these percentages and the number of key frames which give a sufficient match are used.

Disadvantages:

It can produce different results in different runs. This is due to the fact that MSAC uses random samples of feature points for calculations. Thus there is a possibility of selecting a set of features which are not inliers, while there may be some inliers among the ignored set of features.

3.7 Matlab

Matlab is a good platform for experimenting new ideas of image processing rapidly.

Advantages:

- Flexibility: script writing or using the command window. As an interpreter language, it acts as a very advanced calculator
- Mathematics: essential in computer vision
- Debugging: very convenient, calling stack, conditional breakpoints and the instant use of the command line makes it really easy to find a bug.
- Plots: several built in plotting features which helps to visualize the steps/results of an algorithm
- Ready to use CV/ML functions: large function library for many research areas. It is particularly well equipped with computer vision (edge detection, color filtering, resize, rotation, feature matching, histograms, etc.) and machine learning (support vector machines, decision trees etc.)

Disadvantages:

- Parts of the syntax used in this language are different than the other languages like C, Java etc. For example we use '{ }' braces to indicate the start and the ending of the 'for', 'if', 'while' statements usually, but in Matlab the word 'end' is used to indicate the ending.
- There are lot of in-built commands in Matlab which can make coding easier but these have to be learned.
- Matlab is probably not appropriate for production. Matlab is best suited for research rather than implementations. Since we are building prototypes as a proof of the concept, Matlab is adequate.

3.8 Summary

In this chapter we discussed the technologies used in the development of the prototypes, which were video segmentation with Otsu's thresholding, SURF, Harris-corners, MSAC and Matlab. In the next chapter we shall elaborate on the approach taken to build the prototypes for comparing the two image based video retrieval methods.

Chapter 4

Approach to Image To Video Matching in the Prototypes

4.1 Introduction

Here we describe our approach to image based video retrieval which is based on the algorithms and technologies described in the earlier chapter. Two prototypes had to be built in order to compare the two approaches of database and real-time based searching which use the same algorithms. In this chapter the details of the approach will be described, such as the hypothesis, inputs and outputs, the processes of the prototypes and the required features of the prototypes.

4.2 Hypothesis

The problem of building an image based video retrieval system can be solved by using motion detection based feature vector matching. And comparing real-time searching and stored video summaries searching can be done by building two prototypes and comparing their performance and quality.

4.3 Inputs

As inputs, the query **image path** and the **video folder path** are given for the prototypes. The prototypes will then match the image with all the videos in the given path. In the stored summary prototype, the video path is given separately early on, so that the feature data can be saved into the database.

The **YFCC100M** dataset and **Google Image Search** were used to find the images and videos which are used as sample data. The search sample data is limited to **images and videos of cats** for simplifying the evaluation. Cats have different types of color combinations, different types, different sizes and slightly complex shapes. And furthermore, images and videos of cats are very common and easily found in the YFCC100M dataset as well as in the Web.

4.4 Outputs

Each of the two prototypes produce a **list of sorted videos** as output. Also the **execution times**, relative **video rankings** and **matching percentages** are obtained from the output.

4.5 Process

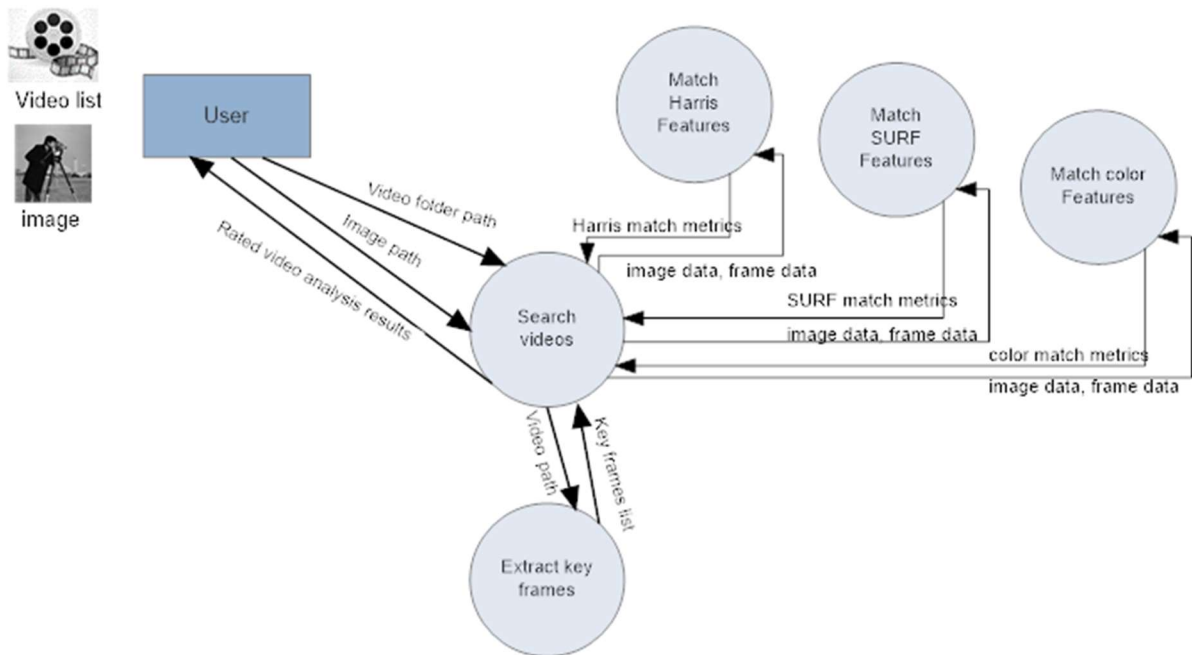


Figure 4.1 - Level 1 DFD of Real-time searching prototype

The above figure 4.1 explains the process of the Real-Time Searching prototype in brief. The user inserts the paths of a list of videos and an image into the prototype. The prototype then preprocesses the image and finds the Harris, SURF and Color features of the image. Then the prototype extracts key frames from each video, and finds the Harris, SURF and Color features of each key frame. The prototype matches these key frame features with the features of the image to find which frames match and how well they match. Afterwards, the prototype calculates how well each video matches with the image and displays the results in the descending order of matching rank.

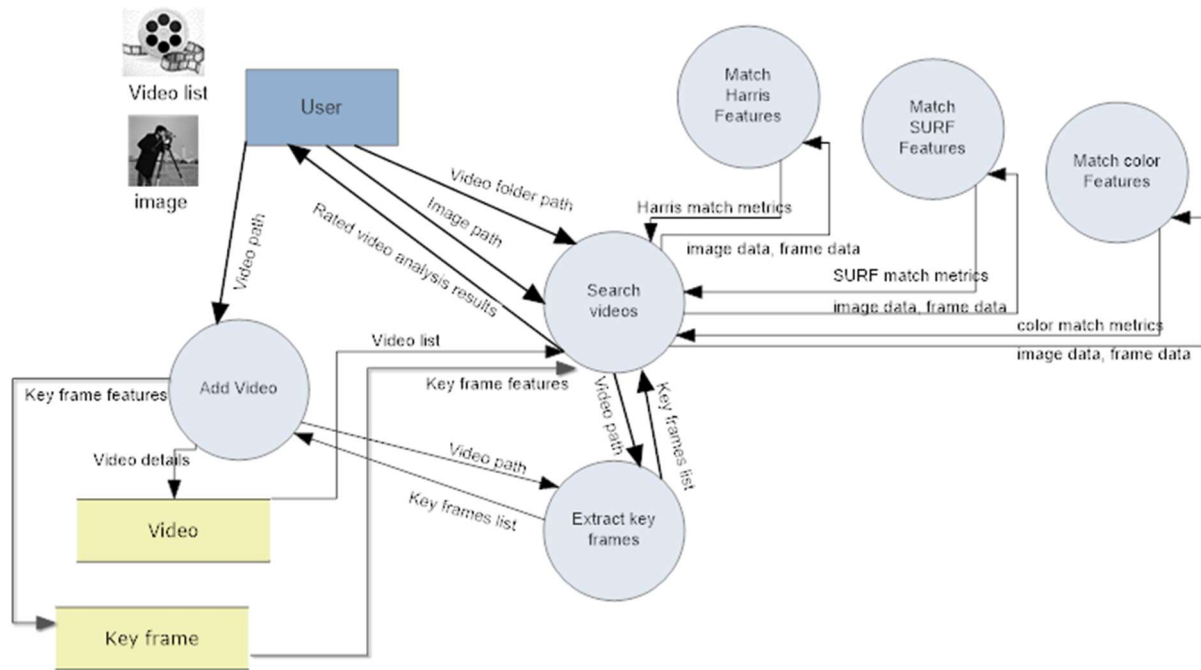


Figure 4.2 - Level 1 DFD of Stored summary searching prototype

Similarly, the stored summary prototype, is given a video path by the user. The prototype then extracts the key frames, finds the Harris, SURF and Color features of the key frames and stores the resultant information in the database. Next the user inserts the path of the image into the prototype. Similar to the earlier prototype, the image is preprocessed and its Harris, SURF and Color features are extracted. Afterwards, the prototype retrieves the video information from the database to match each key frame's features with the features of the image. After matching the features, the results are calculated and displayed as was explained before.

4.6 Features

Required features of the prototypes:

- Efficiency of execution
- Accuracy of results (and of the measurements taken)

4.7 Users

Since the results of this research are prototypes of algorithm implementation, users can be identified by the possible application areas. This algorithm can be applied in search engines which are used by **Web surfers**. Multimedia applications which require searching videos can cater **multimedia application users** with this functionality.

4.8 Summary

The approach chapter described how this project is going to be approached; the hypothesis, input, output and processes. Also features have been described. The next chapter will explain about the design of the prototypes in greater detail.

Chapter 5

Designs of the Prototypes

5.1 Introduction

Earlier we discussed the approach being used to improve the entity resolution of image based video search. In this chapter we shall discuss the design for the prototypes we use to compare the two methods of searching, and how to conduct the comparison. There are two prototypes, each for a method of searching. In the real-time search, we extract the features and compare, while in the stored summary method, we store the extracted features and later use them in our search. We shall discuss these designs with architecture diagrams, activity diagram and scenario descriptions further in this chapter. The database design is explained afterwards.

5.2 Real-Time Search Method Prototype

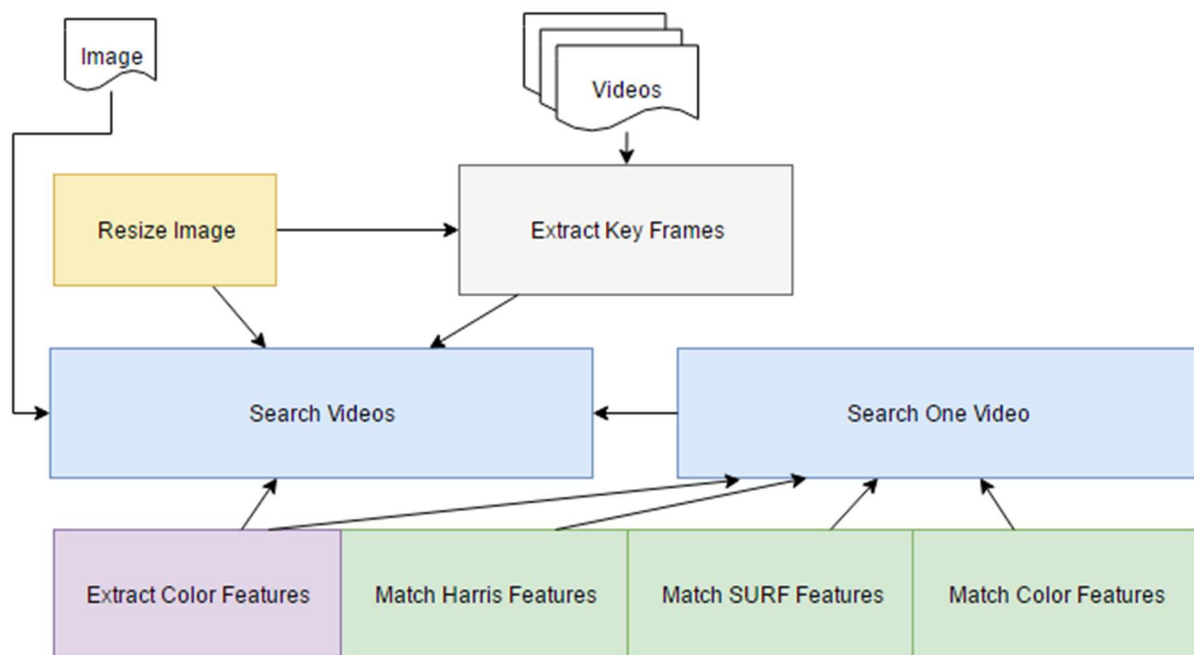


Figure 5.1 - Architecture of Real-time searching prototype

In this prototype, first a user inputs an image. Then the software takes the list of all videos and then calculates the difference between each two frames for each video and the Otsu's threshold. This value is used to calculate the overall threshold. Then it determines which key frames are best representative of the different shots. Afterwards the key frames can be searched, matching against the given image (using feature vectors).

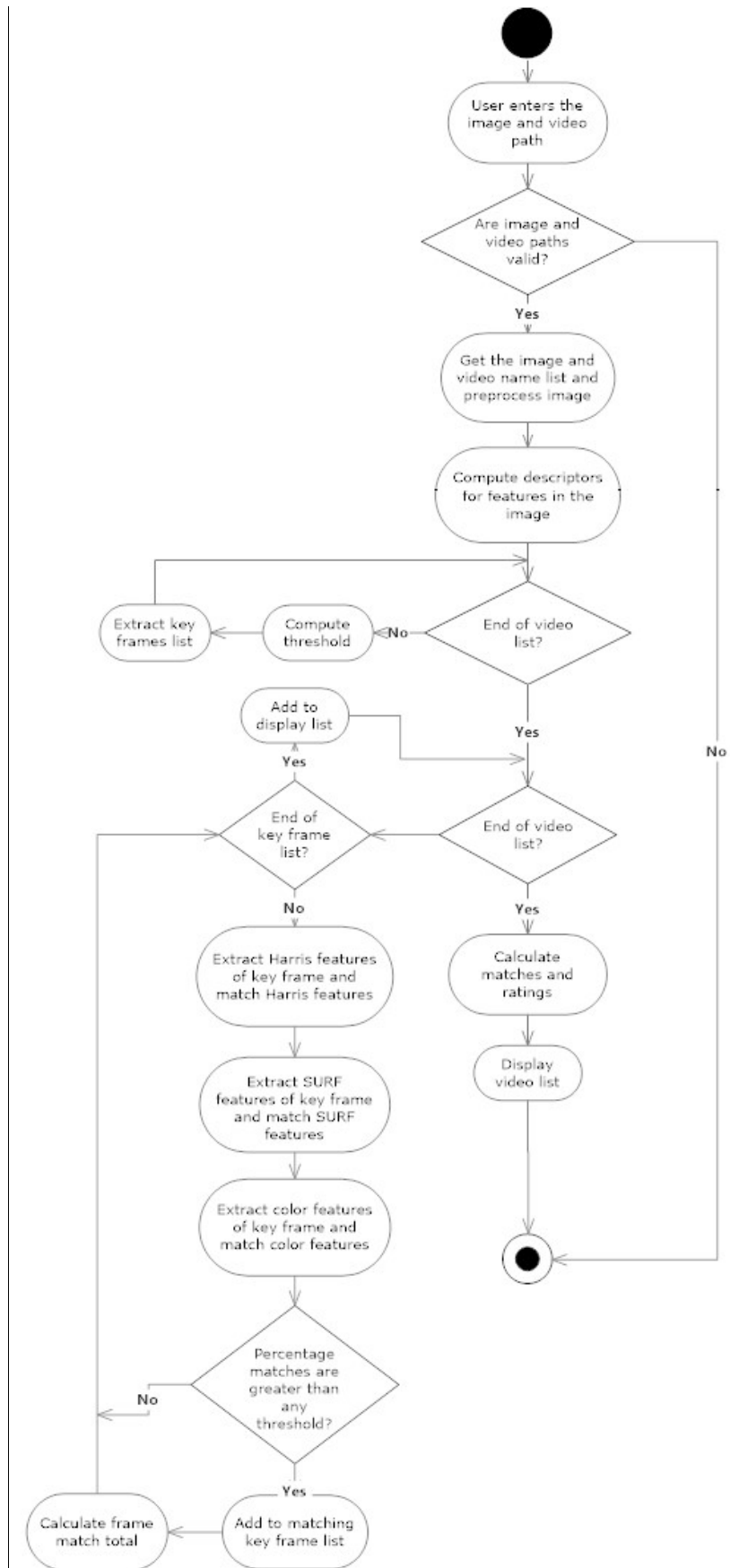


Figure 5.2 – Activity diagram of Real-time searching prototype

5.3 Stored Summary Method Prototype

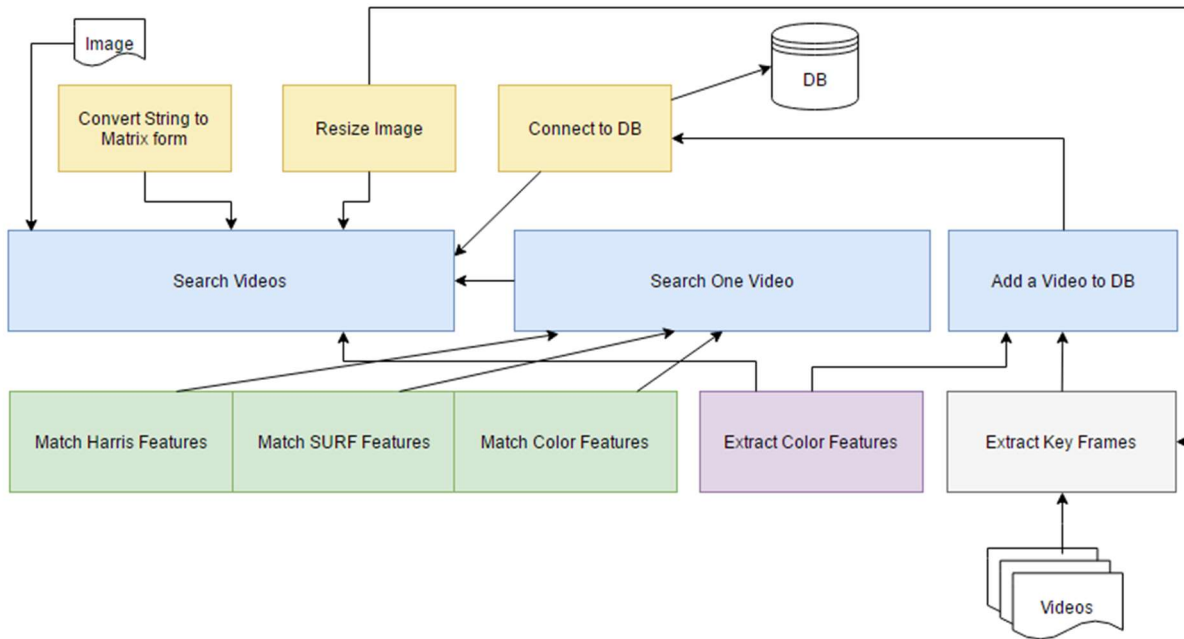


Figure 5.3 - Architecture of Stored summary searching prototype

The stored method employs functions that are similar to the real-time method. When adding a video, the prototype extracts the key frames. These key frames are used to extract the features by executing the respective algorithms on the key frames. Afterwards the features/signatures/summaries are saved into the database. When an image is given, these features are retrieved and compared against the image.

For the Activity diagram of this prototype, please refer to the **Appendix B**.

Scenario descriptions of stored summary

Scenario 1

1. System takes each video in the collection.
2. Segment the video and extract key frames.
3. Descriptors are computed for features in each key-frame.
4. Save the video details in database.
5. Store feature descriptions and feature points of the key frames in the database.

Scenario 2

1. User enters the image.

2. Descriptors are computed for features in the image.
3. System extracts the stored features.
4. The descriptors are compared to descriptors of the image and the match percentage is calculated for each key frame.
5. Metrics of comparison (average frame level match percentage, matching key frame percentage and total match) are calculated.
6. Compare the video matches and build the final video search results list.

5.4 Database

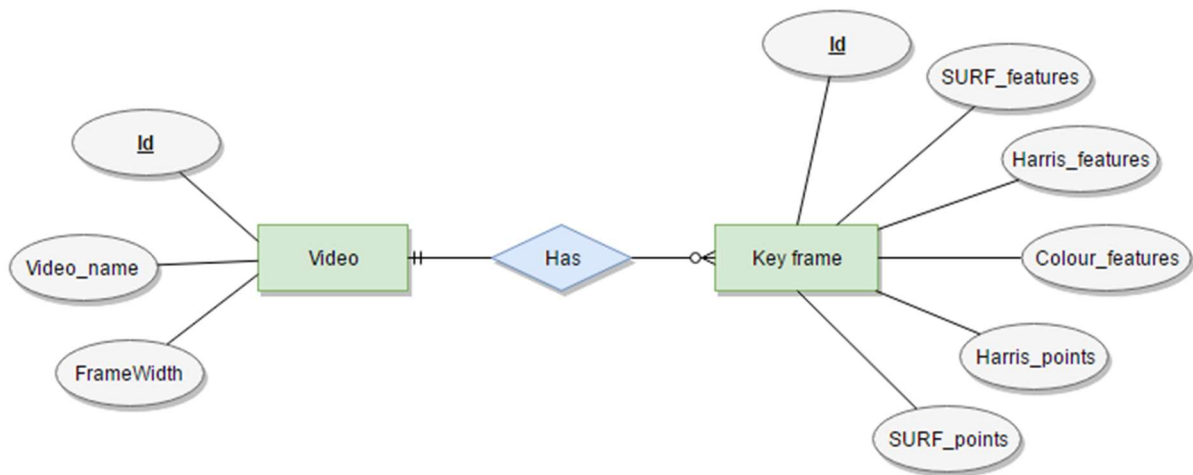


Figure 5.4 - ER diagram

The database contains two tables: the video table and the key frames table. Key frames have the properties of algorithm based features, extracted from the key frames.

5.5 User Interfaces

Simple command prompt can be used for inserting the video folder path and the image path which is simple and straightforward.

5.6 Comparison

Run the two applications separately, using the same 3 sets of videos (size 22 each), input the same image and run the search. The time to produce results will be calculated automatically. This will be repeated for 3 sets of 5 images (90% cat, 89-1%cat, 0%cat). The comparison will be further explained in the evaluation chapter.

5.7 Summary

In this chapter we discussed how the two prototypes for the two methods of searching are designed. The designs were described with the aid of diagrams and scenario descriptions. These include the architecture, activity flow and the design of the database. Afterwards, the method of comparison for the prototypes was discussed. The next chapter shall discuss how to implement the prototypes and related issues.

Chapter 6

Implementation

6.1 Introduction

This chapter provides the details of the prototype implementation. After describing the decisions regarding the system in the previous chapter this is the chapter where the actual implementation of those planned features is documented. Each component and module will be explained in this chapter with the relevant implementation designs, decisions, tools, screenshots and code samples.

6.2 Implementation Overview

The two implementations are similar except for the fact of using a database of key frame signatures by one implementation. Both search prototypes are written using MatLab. MS Access acts as the DBMS.

The codes of the prototypes are given in the **Appendix C**.

6.3 Database Implementation

6.3.1 MS Access 2013

As explained earlier, the storage of feature information needed to be saved in the stored summary prototype is implemented using MS Access. MS Access was selected on the basis that it is very easy to implement and the transaction of the system are fairly simple. But for a business level application of this methodology of Image to Video Matching requires a more powerful database implementation.

6.3.1 Physical Database Design

Accordingly a relational database design was needed for the data. An ER diagram was drawn to design the relational database as shown in the previous chapter. The same is implemented here as follows:

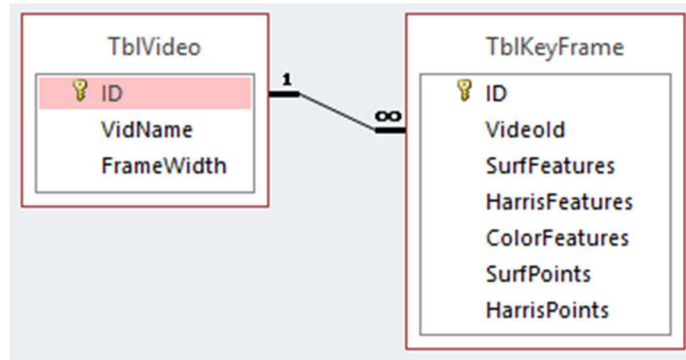


Figure 6.1 - Screenshot of the database design (relationships)

There are two tables as TblVideo for storing video names and TblKeyFrame for storing the features of the Key Frames for each video. SurfFeatures column for SURF feature detector/descriptor results, HarrisFeatures column for Harris detector with FREAK descriptor and ColorFeatures column for color extractor algorithm with simple Square Neighborhood descriptor results are used. And the locations of the feature points are saved in HarrisPoints and SurfPoints columns.

The feature data had to be saved in the string format because it requires less space to store and easy to convert from matrix format. But converting strings into matrices is a slow process.

6.3.3 Data Dictionary

Entity Name	Video		
Table Name	TblVideo		
Description	Hold basic information on videos		
Primary Key	ID		
Foreign Keys	-		
Reference tables	-		
Attributes	Field	Type	Description
	<i>ID</i>	Number	Primary key
	<i>VidName</i>	ShortText	Name of the video file
	<i>FrameWidth</i>	Number	Width of a frame
Entity Name	Key Frame		
Table Name	TblKeyFrame		
Description	Hold information on key frames of videos		

Primary Key	ID		
Foreign Keys	VideoId		
Reference tables	TblVideo		
Attributes	Field	Type	Description
	ID	Number	Primary key
	VideoId	Number	Foreign key to TblVideo. Cascade rule is applied for UPDATE and DELETE.
	VidName	ShortText	Name of the video file
	SurfFeatures	LongText	Features list derived by SURF algorithm
	HarrisFeatures	LongText	Features list derived by Harris algorithm
	ColorFeatures	LongText	Features list derived by color features extraction algorithm
	SurfPoints	Long Text	Locations of SURF points
	HarrisPoints	Long Text	Locations of Harris corner points

Table 6.1 Detailed data dictionary

6.4 Component/Module Implementation

The components are implemented in MatLab. So rather than the detailed pixel level implementation, the emphasis is given to the use of existing algorithms to support the prototype designs.

- ▲ AddVidToDB
- ▲ bulkRun
- ▲ Color
- ▲ ConnectToDB
- ▲ featuresColorExtractor
- ▲ keyfOtsu
- ▲ matchColorFeatures
- ▲ matchHarrisFeatures
- ▲ matchSurfFeatures
- ▲ resizeImage
- ▲ searchOneImageFromDB
- ▲ searchOneVid
- ▲ searchVids
- ▲ searchVidsFromDB
- ▲ str2matrix

Figure 6.2 - List of code components

6.4.1 Extract Key Frames

This component extracts key frames from a given video. For these prototypes, we consider only videos of length up to 45 seconds. This will make the evaluation simpler.

What it does is, obtaining a threshold for each frame and then using it to extract the key frames. First, selected frames are applied the Otsu's Threshold to find the change in consecutive selected frames. Thresholds are calculated for the grey image. This threshold is used to extract key frames (see also Technology chapter).

Afterwards, the frame that correspond to the values of threshold greater than the average threshold are extracted. To compromise for the states between each high threshold frames, the median frame between two consecutive high threshold frames are also extracted.

Video > Shot > Group of frames > Key frame

The simplicity of the scheme makes it possible to segment the video within a sufficiently less time. The following algorithm is used in this procedure. The code is given in the Appendix C.

```
READ VIDEO
CALCULATE NUMBER OF FRAMES
DECIDE THE FRAME EXTRACTION INTERVAL
GET THE FIRST FRAME
RESIZE FIRST FRAME
FOR EACH INTERVAL OF FRAMES
    GET THE NEXT FRAME AFTER INTERVAL
    RESIZE NEXT FRAME
    GET FRAME DIFFERENCE
    GET OTSU'S THRESHOLD
    KEEP OTSU'S THRESHOLD
END
GET MEAN AND STANDARD DEVIATION OF OTSU'S THRESHOLDS
GET THE GLOBAL THRESHOLD
FOR EACH SELECTED FRAME
    IF DIFFERENCE > THRESHOLD OR IS (LAST -1) TH SELECTED FRAME
        EXTRACT KEY-FRAME
```

```
        EXTRACT KEY-FRAME OF MEDIAN POSITION BETWEEN THIS KEY-FRAME AND THE LAST
ONE
        END
END
```

But there are a lot of possible improvement to this codes, such as supporting video durations greater than 45 seconds and even one hour. A generalized frame interval selection formula could be used for his purpose. And a greater number of more accurate number of frames could be selected if there is more processing power, using two global thresholds for the upper and lower limits.

6.4.2 Get Colour Features

For getting the Harris features and SURF features, the methods given by MatLab Computer Vision Toolkit are used. For the colour features, we use a simple extractor as described in the Technology chapter.

6.4.3 Match Harris/ SURF/ Color Features

Here, the features of the image are matched with the features of the given frame. Then the program calculates the percentage of matching features as well as percentage of inlier features against the total number of features of the image. This value is thresholded separately for each type of features (Harris/SURF/color). The average distance between Key Pairs is compared with the image width to guess whether the Key Points are inliers or not. A partial match is calculated for combining it with the results of other types of features later.

E.g. (from Harris feature matching):

```
if(size(I1,2) < size(I2,2))
    Tdist = size(I2,2);
else
    Tdist = size(I1,2);
end
averageDist = sum(indexPairs(:,2)-indexPairs(:,1))/numPairs;
if ((status ~= 0 && percentage >= 2 && (averageDist <= Tdist/2 *
0.9)) || ((sizeOfinliers1*100/numPairs) >= 2 && status == 0) ||
percentage >= 6)
    match = true;
else
    match = false;
end
if (percentage >= 4)
    partial_match = true;
end
```

6.4.3 Run-time prototype

6.4.3.1 Search Videos

This is the main component of the Run-time prototype. The full code is provided in the Appendix C. Following is brief of this procedure:

```
GET VIDEO FILE LIST AND IMAGE  
  
RESIZE IMAGE  
  
GET FEATURES OF THE IMAGE  
  
FOR EACH VIDEO FILE  
    EXTRACT KEY FRAMES  
    FOR EACH KEY FRAME  
        SEARCH ONE VID  
    END  
  
CALCULATE MATCH METRICS  
  
IF METRICS ARE SIGNIFICANT  
    KEEP VIDEO RESULT  
  
END  
  
END  
  
DISPLAY VIDEO LIST
```

The match metrics are designed to exclude trivial matches by thresholding. It uses the percentage matching frames and average percentage match of matching frames to make this selection. The thresholds and metrics are largely based on observational matching.

```
((percMatch+avgFeatureMatch) >= 64.412 && avgFeatureMatch >= 5.92)
```

There are more statistically improved metrics, such as Confidence that can be used.

6.4.3.2 Search One Video

This component takes one video, takes each key frame and calls the matching components for Harris, SURF and color features. Then it checks to see whether any of them return positive results or if all match partially. Then it calculates the matching percentage and number of matches and returns them.

6.4.4 Stored summary prototype

6.4.4.1 Add a Video to DB

This module helps to insert one video into the database. First, the video path is taken. And then the video name information is saved into the database (TblVideo). Afterwards the key frames are extracted by calling the relevant component (Extract Key Frames). Lastly, for each key frame, features are obtained by calling the relevant components for Harris, SURF and colour features, and these are converted to string by calling a MATLAB function and then saved to the TblKeyFrame. The code is given in the Appendix C.

6.4.4.2 Search Videos from DB

Similar to the Search Videos component, this function searches each video for matching key frames. The difference is that the set of video names is obtained from the database, and then each key frame's feature information is retrieved from the database. For each of these key frames, the features are matched with the features of the given image by calling the relevant component (Search One Image from DB).

6.4.4.3 Search One Image from DB

For each key frame from the database, the feature information string will be converted into matrix form. Then those features will be matched using the matching modules for Harris, SURF and color algorithms. Then the best match selection process goes in the same way as in the Real-Time prototype.

6.4.4.4 Match Harris/ SURF/ Color Features from DB

This is similar to the function in the Real-time prototype but since the features of key frames have been given, only the matching part is done.

6.4.4.5 String to Matrix

Converting string to a matrix. Please refer to the Appendix C for the actual code.

```
GET NUMBER OF ROWS AND NUMBER OF COLUMNS

CREATE MATRIX

GET NUMBER OF ROWS

FOR EACH ROW

    GET NUMBER OF VALUES

    FOR EACH VALUE

        GET VALUE START POSITION

        GET VALUE END POSITION

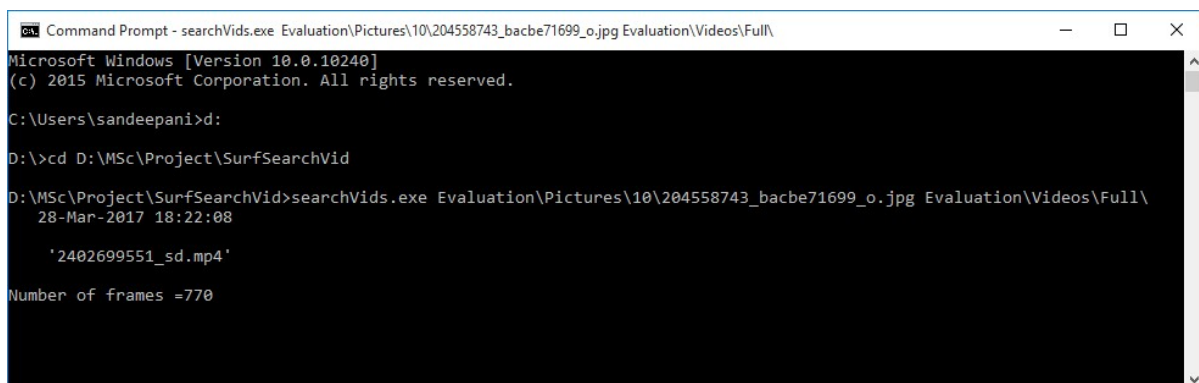
        GET VALUE AND ASSIGN TO MATRIX

    END

END
```

6.5 Command Line Execution

MatLab allows to deploy programs as executable files. The executable files are actually slower than executing in the MatLab environment. But this makes the programs easier to demonstrate. It requires the MatLab Runtime to execute.



```
Command Prompt - searchVids.exe Evaluation\Pictures\10\204558743_bacbe71699_o.jpg Evaluation\Videos\Full\
Microsoft Windows [Version 10.0.10240]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\sandeepani>d:
D:\>cd D:\MSc\Project\SurfSearchVid
D:\MSc\Project\SurfSearchVid>searchVids.exe Evaluation\Pictures\10\204558743_bacbe71699_o.jpg Evaluation\Videos\Full\
28-Mar-2017 18:22:08

    '2402699551_sd.mp4'

Number of frames =770
```

Figure 6.3 - Screenshot of execution of the run-time prototype

6.6 Summary

The implementation chapter describes the implementation, with its code in the Appendix C. First we briefed the technologies used and then the database implementation. In the DB implementation, the DBMS used, the physical design and the data dictionary were presented. The components/modules were then explained in detail. In the next chapter we shall discuss the evaluation of the prototypes.

Chapter 7

Evaluation

7.1 Introduction

After all the implementation details of the last chapter, this is the point at which the project and the product will be evaluated. The project will be evaluated against the requirements and objectives of the project. The prototypes have to be evaluated for comparing the stored summary method vs. real-time method as well. The environment of evaluation, the plan, data collection and data analysis will be expounded from here on.

7.2 Evaluation Environment

The fast execution and accurate results are expected from the algorithm being proposed. And the comparison of speed of execution and accuracy of results between the two implementations must be carried out as well. The evaluation plan needs to support these goals.

Thus a large number of videos need to be searched for a large number of images in order to produce better results. There are also different types of images of different types of cats, appearing in different postures, angles and appearing partially. But as clear as possible images are needed that have distinct cat images since otherwise the background can affect the search result. The videos have the same variation. Additionally, there may be videos with no cats and different durations of cat appearance. All these factors need to be taken into account in order to measure the accuracy and speed.

7.3 Evaluation Plan

Three sets of videos are used in the search. The first set of videos has the image of a cat/cats in 90% of duration, the second one has 89 to 1% and the third set has 0% cats.

The duration of videos is limited to maximum 45 seconds for faster searching. And the videos saved in the database are limited in their file size to reduce the size of the Access database (maximum 2GB total).

1. For measuring the accuracy of the video search technique, the 3 sets of 30 videos each are used with groups of incrementally sized (incrementing by 5) image sets, i.e. 5, 10, 15 etc.

2. To compare the two applications are run separately, using the same 3 sets of videos (size 22 each), input the same image and run the search. The time to produce results is calculated manually using start time and end time given from the program. This is repeated for the 3 sets of videos with 5 images as input.

Additionally, it is required to run the two search prototypes and Add Video module in batches to conduct this test. A large number of image searches as well as databases insert operations needs to be run. For this reason, Bulk Run programs were created for running the tests. The evaluator inserts a folder path for all the images or videos which is used by the program to loop over each and run a given Module consecutively until the end of the image/video list.

7.4 Automated Testing Implementation

Since the evaluation process is long and redundant, it had to be automated. Two small programs were created to automate the evaluation of the implementations. And three databases were created to carry out the comparison task, each for one type of videos based on the similarity category. This was required especially since MS Access only allows to store 2GB in a database at one time. Details are given in the Appendix D.

1. 'bulkRunReal' to run the real-time prototype on a folder of images
2. 'bulkRun' to run the stored summary prototype to save a folder of videos into the database or search a folder of images against a database
3. 'VidSearchFull' to store information of videos with $\geq 90\%$ of cat images
4. 'VidSearchMid' to store information of videos with $< 90\%$ and $> 0\%$ of cat images
5. 'VidSearchNone' to store information of videos with 0% of cat images

7.5 Data Collection

Evaluation Number 1

The following results could be obtained by running the evaluation of real-time prototype. Table 7.1 gives a sample of the results obtained. Detailed test results are given in the Appendix D.

Image	Results	Top three video results
	27 out of 90 videos	
	22 videos out of 90	
	7 out of 90 videos	
	23 out of 90 videos	
	23 out of 90 videos	
	23 out of 90 videos	

Table 7.1 - Matching results for sample images with all video sets using real-time prototype

Video set	Image set	Total average of matching videos
More than 90% appearance	5	11.33%
	10	15.67%
	15	17.78%
	20	19.50%
Medium % appearance	5	17.33%
	10	33.67%
	15	29.78%
	20	25.67%
0 % appearance	5	9.33%
	10	18.67%
	15	20.67%
	20	15.17%

Table 7.2 - Matching results for different sets using real-time prototype

Evaluation Number 2

For the comparison of the two prototypes, the following results were obtained. Further details are given in the Appendix.

		Average time (h:m:s)
Real-time	More than 90%	0:21:09
	Medium % match	0:27:48
	0% match	0:14:31
Stored summary	More than 90%	0:28:18
	Medium % match	0:28:17
	0% match	0:46:25

Table 7.3 - Comparison of search times for different video sets using both prototypes

Following numbers of matches could be observed from the two prototypes.

		Average percentage of matching videos
Real-time	More than 90%	8.67%
	Medium % match	11.33%
	0% match	6.67%
Stored summary	More than 90%	2.00%
	Medium % match	9.33%
	0% match	6.67%

Table 7.4 - Comparison of matching video percentages for different video sets using both prototypes

7.6 Data Analysis

Evaluation Number 1

The below graphs were produced using the results of evaluation number 1: the comparison of percentages of matching videos.

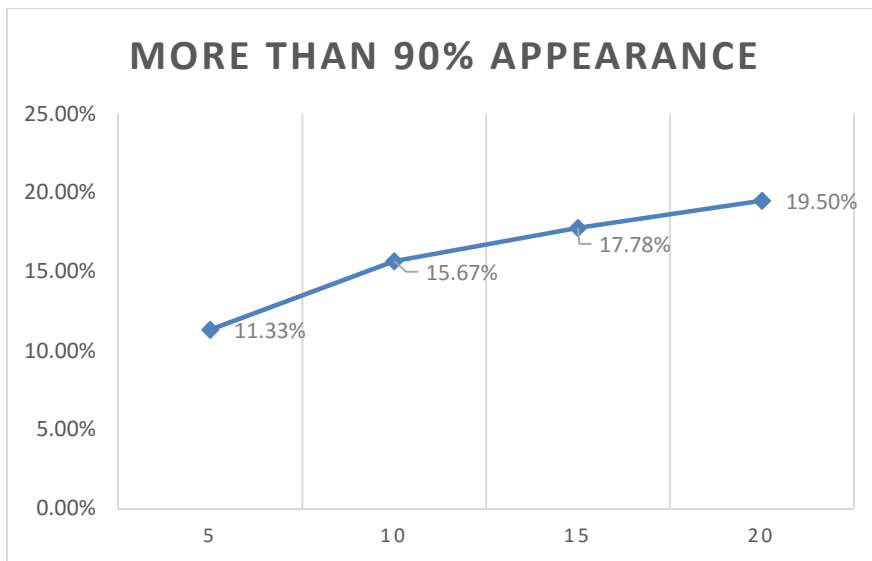


Figure 7.1 - Result of searching more than 90% appearance videos

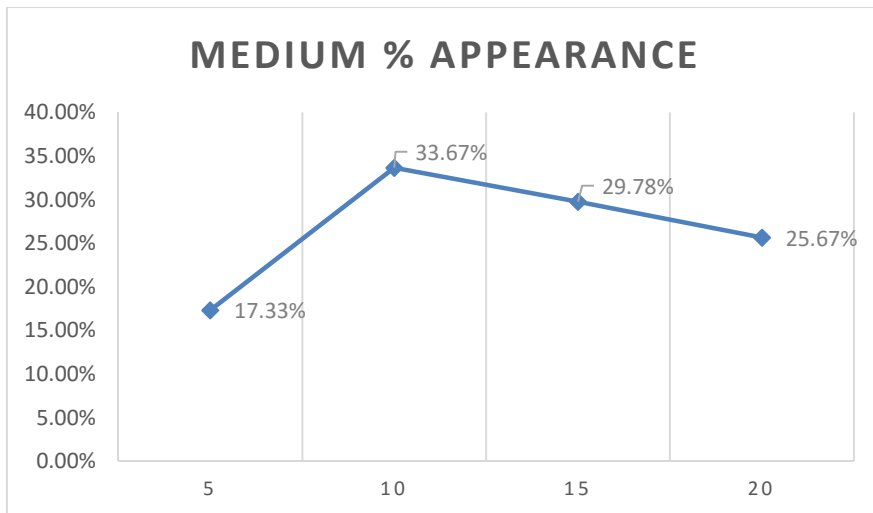


Figure 7.2 - Result of searching Medium maching videos

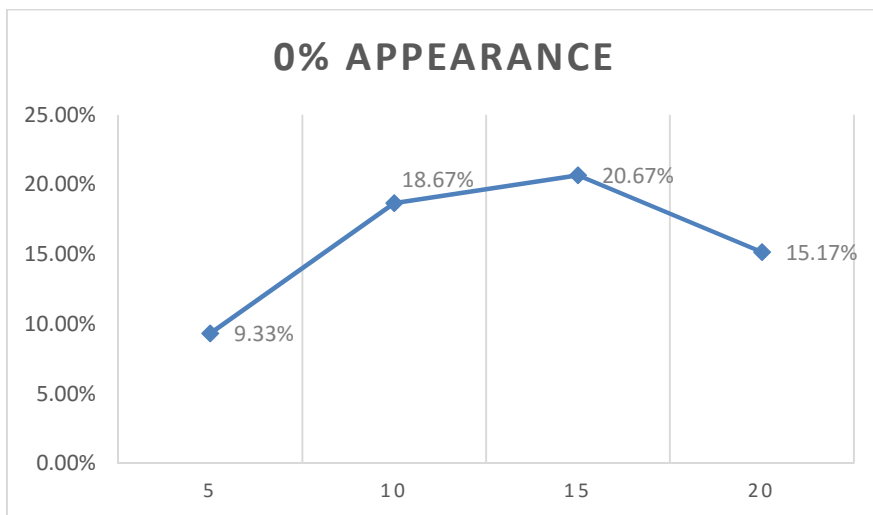


Figure 7.3 - Result of searching 0% maching videos

The summary graph of all the video sets:

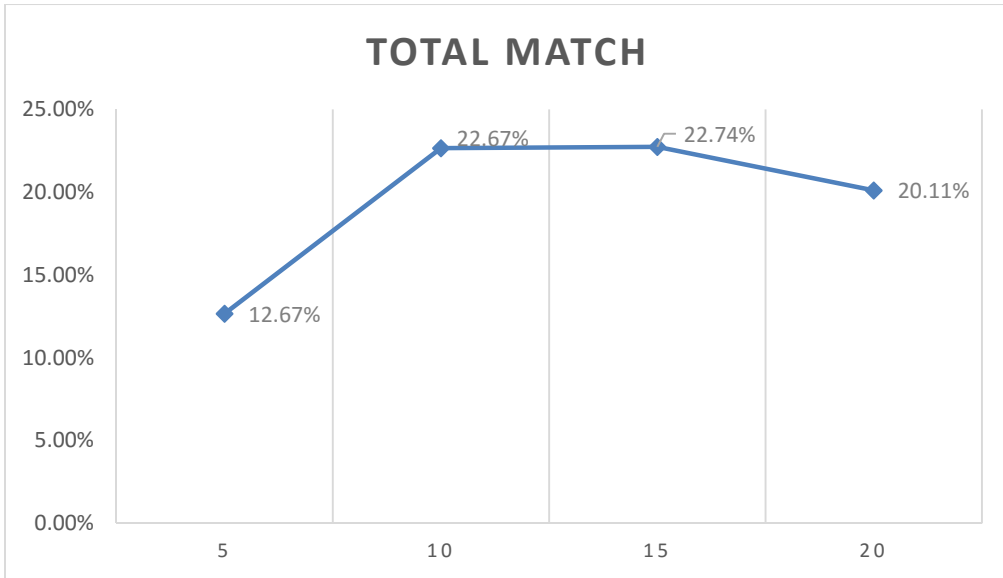


Figure 7.4 - Total result of searching all matching videos

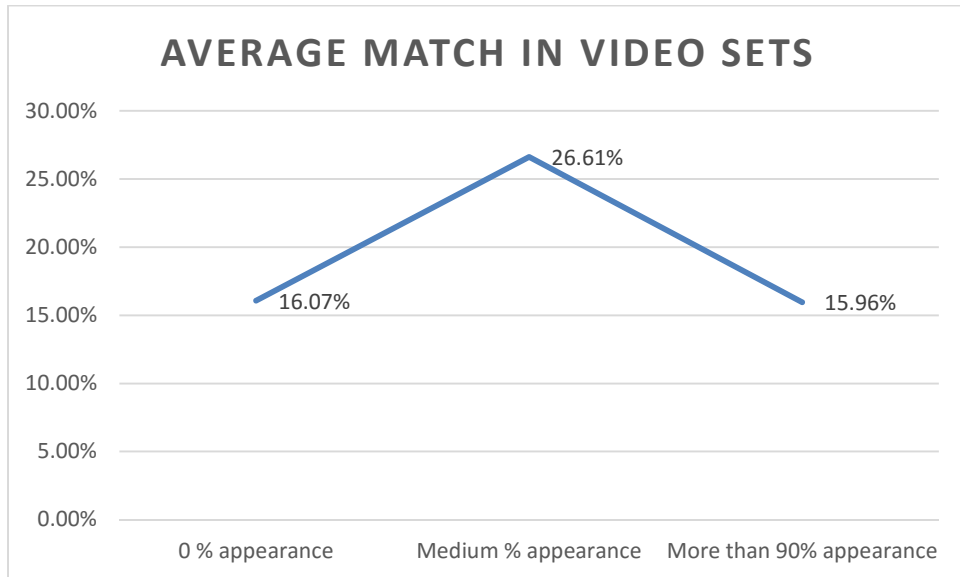


Figure 7.5 - Total result for video sets

Evaluation Number 2

Comparison of the running times of the two prototypes can be shown as follows:

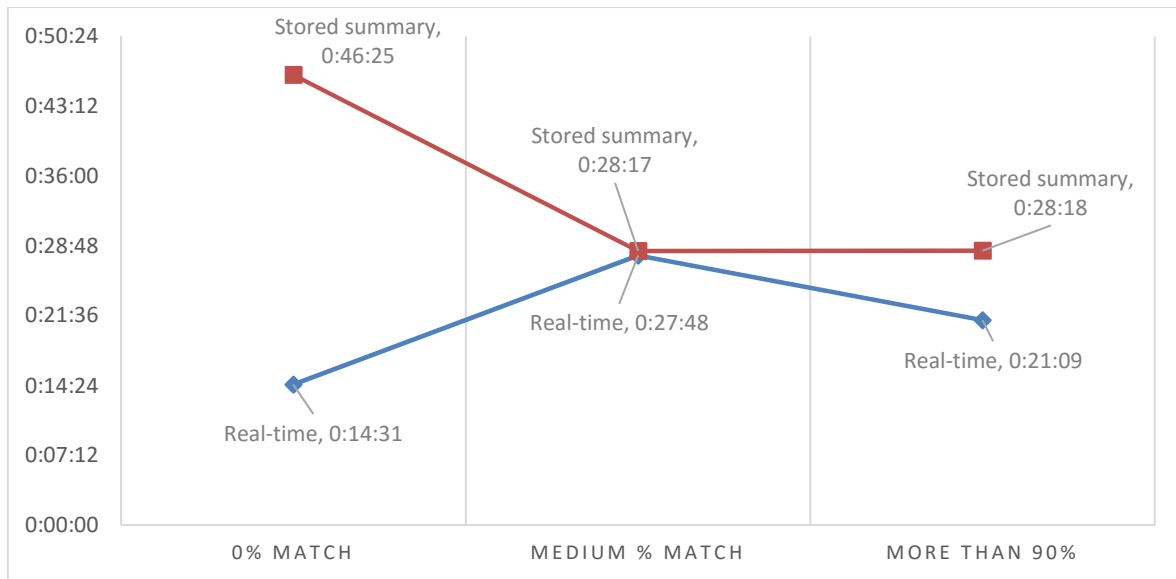


Figure 7.6 - Comparison between prototypes' running times

The change of value between the two prototypes from the comparison run can be shown as below.

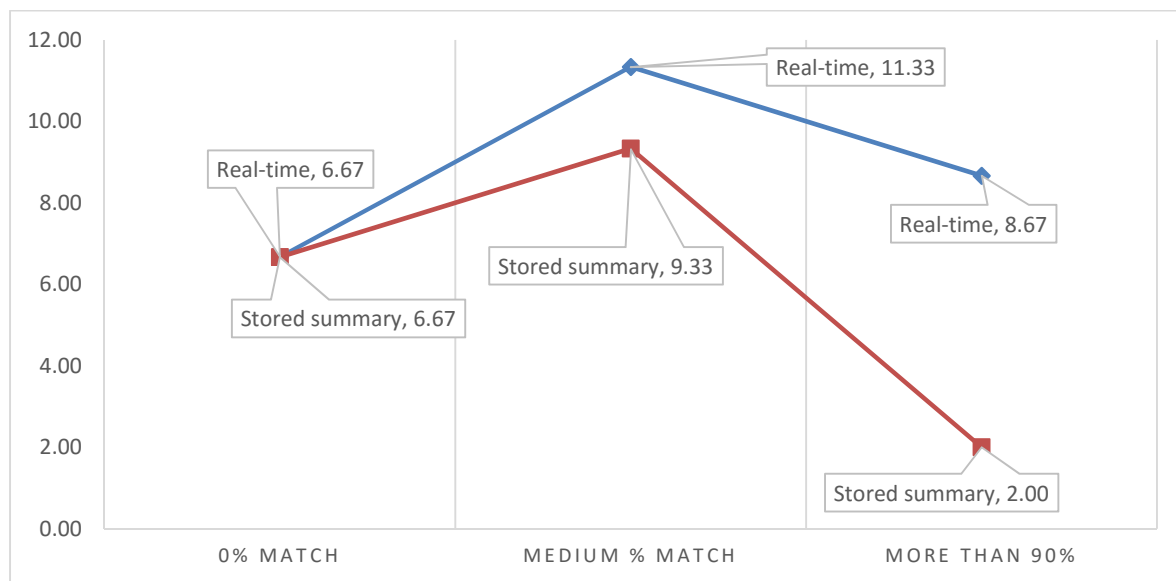


Figure 7.7 - Comparison between prototypes' percentages of results

From the above results it can be seen that:

1. There can be observed generally a growth of number of matching videos as the number of images are increased, except for the set of 20 images (0% and medium percentage matching) and medium percent matching image set of 15.
2. From the video sets, medium matching videos show a greater match difference between image sets and a greater match overall.

3. The real-time based prototype is clearly the faster approach for closely-matching and scarcely-matching videos. For medium level matching videos, it is only slightly faster than the other method.
4. There is a difference in the number of results in the two prototypes. The Stored summary prototype produces lesser number of results.

The summary based prototype has become slower than the runtime based prototype. This has occurred because of the time spent on converting feature summary strings into matrix format, despite the actual speed of searching having been increased.

7.7 Summary

This chapter portrayed the evaluation of the proposed method of image to video matching and the comparison of the two prototypes proposed for this purpose. First of all the background of the evaluation and the evaluation plan was described. Then the collected data were presented and the analysis of the data was done. In the next chapter shall detail the conclusions finally derived from implementation and evaluation.

Chapter 8

Conclusion and Further Enhancements

8.1 Introduction

Previous chapters have elaborated all the aspects of the project, ending with evaluation of the proposed method of enhancing image based video searching. Following the evaluation, this is the concluding chapter of the thesis, with the conclusions gained from evaluation. Here it summarizes the entire project and discusses the quantitative conclusions gained through evaluation and the achievements of the project against the initial objectives. There are accomplishments, failures and the limitations faced in the project. We shall also see how the final product can be improved through further extensions.

8.2 Project Summary

This MSc project was started with the aim of exploring the field of image to video matching and proposing a possible solution. After surveying the related literature, existing implementations and implementing sections of code, it was decided to implement a simple solution using feature matching and video segmentation. The possibility of storing summary data of the video was decided to be carried out as a comparison. Thus, different technologies were selected for the final prototype implementation: SURF feature extraction, Harris corner feature extraction, colour features extraction, Otsu's thresholding for change detection and string format to store feature information summary. The prototypes were later evaluated using 3 sets of 22 videos and 5 images. Then the real-time prototype was used to test the effectiveness of the proposed method in iterations using 3 sets of 30 videos and 3 sets of images.

8.3 Quantitative Conclusions

As has been elaborated in the last chapter, the evaluation of the run-time based prototype shows that the proposed method of image based video searching shows accuracy for expected results or confidence for the 0% matching videos is 83.93%. The confidence for 90% matching videos is 15.96%. For medium matching videos, a 46.78% confidence is shown (for an expected 50%). Thus, the average accuracy is 48.89%. This shows little success. The number of matches varies with the number of images and the occurrence of similar features. The videos with more than 90% cat appearance show less matches than videos with 0% cats in them. The reason is that when searching for cat images, the system also finds image of animals and matches the

background colors and background textures. The videos with intermediate appearance of cats show the highest match overall.

For the comparison between the prototypes, real-time searching is 23.74% faster than searching feature summary database. This was the opposite of what was expected. And the result was effected greatly by implementation decisions. There was a difference in the results in the two prototypes. It has been observed as an average of 2.89% difference.

There has been observed a chance of around 8.75% of possible change in the resultant videos when repeating the same search. This is due to the randomness of results of the MSAC algorithm. But this has not been observed to have affected the final evaluation averages obtained.

8.4 Objective-Wise Conclusions

In an academic project, skills and experience gained during its progress are as important as its result. Nevertheless, the final outcome of the project needs to be discussed in terms of the initial objectives of the project, one by one. Then we can explore whether these have been achieved or not.

1. To survey the literature related with image based video retrieval to select appropriate algorithms.

The survey of literature has been done up to reviewing 14 researches (as documented in the literature survey) and referencing 28 total of sources. The 14 reviews have been further discussed and compared to find the suitable technologies and methods. And likewise some suitable technologies and methods have been found from these sources. Thus, we may say that a sufficient amount of survey has been done, although it would have been better if it was possible to produce reviews for the other 10 sources as well. Nevertheless it was possible to learn a great lot about the field of image based video retrieval by doing this literature survey.

2. To develop two prototypes and evaluate the two techniques of real-time searching and stored video summaries.

These two prototypes were developed and were compared for speed. The accuracy of both prototypes was the same. The comparison was done using 3 sets of 22 videos which were search for the same 5 images for each set (total 15 test runs). This comparison is described in detail in the Evaluation chapter. Result was that the Real-time based prototype was faster than the

summary based prototype. Factor which effect the speed of the database is the conversion of large feature data in a format suitable for the database into a format that can be processed by the programing language.

3. To evaluate the technique of motion detection based image based video retrieval

The technique used in the implementation was tested using the real-time search prototype, 15 images and 90 videos as described in the Evaluation chapter. This evaluation showed that the number of matches varies with the number of images and the occurrence of similar features and an intermediate level of accuracy was observed.

8.5 Problems Encountered, Limitations and Decisions Made

There had to be made a compromise between the accuracy of results and the speed of execution in all aspects of the prototypes. For searching a large video set, the speed is the most important factor. This effected the video segmentation and the selection of feature extractors. It has also affected the final accuracy.

The 'i5 processor' for this algorithm takes a great length of execution time (around 30 minutes for 30 videos with less than 45 second duration for each for the real-time based prototype), after numerous optimizations of efficiency. But in an ideal implementation, the searching will be done using a GPU and many high speed servers in data centers.

Many algorithms such as PCA-SIFT feature matching were dropped as good open source implementations could not be found for such algorithms or long execution times etc.

A decision of compromise had to be made with regard to the DBMS (MS Access): its simplicity of implementation against the lack of speed, power and tools. This was explained in earlier chapters.

Also key frame feature data had to be saved in the string format because strings take less space than XML. But strings are much slower to be converted to the matrix format.

Inlier detection method MSAC used in the prototype gives random results for inlier points, which can cause the resultant number of video results to be randomized.

8.6 Further Enhancements

If faster processors, GPUs or dedicated servers could be used for the searching, it would provide much greater speed than the current prototype implementations.

A faster method can be developed in future enhancements by techniques such as, quantization of the feature matrices so that they take less space and are easier to store/retrieve or Bag of Features methods which store only the relevance of features to cluster centers (clustering), obtained through training.

A superior DBMS such as Microsoft SQL Server can provide better tools, integrity and power, than Microsoft Access for the implementation.

A region detector such as MSER can be used in the feature detection and matching process. Currently only the corners, textures and colours are being matched.

Salient object detection and object matching is an option that will improve the matching accuracy greatly. Rather than matching a whole frame with the whole image, we may abstract the salient objects in the given image and match those with the moving objects in the video's key frames.

Deep convolutional neural networks are currently being used in the user image based images searching field. We may use his technique in image based video retrieval as well, after simplifying it for increasing the speed.

8.7 Summary

This chapter being the last chapter of the thesis, provided an overview or summary of the project. Next, a quantitative conclusion was given. Then it assessed the project against its initial objectives and discussed whether these were met or not. It also contained the problems and limitations, and the decisions taken regarding those in the project. Later, the enhancements or improvements that can further extend the project were proposed. This concludes the thesis. Additional information related to thesis chapters can be found in the Appendix.

References

- Agarwal, S., Roth, D., 2002. Learning a sparse representation for object detection, in: European Conference on Computer Vision. Springer, pp. 113–127.
- Araujo, A., Chen, D., Vajda, P., Girod, B., 2014. Real-time query-by-image video search system. ACM Press, pp. 723–724. doi:10.1145/2647868.2654867
- Bahri, A., Zouaki, H., n.d. A Surf-Color Moments for Images Retrieval Based on Bag-of-Features [WWW Document]. EA J. URL <http://www.eajournals.org/journals/european-journal-of-computer-science-and-information-technology-ejsit/vol-1-issue-1-june-2013/a-surf-color-moments-for-images-retrieval-based-on-bag-of-features/> (accessed 4.23.17).
- Eickeler, S., Wallhoff, F., Lurgel, U., Rigoll, G., 2001. Content based indexing of images and video using face detection and recognition methods. IEEE, pp. 1505–1508. doi:10.1109/ICASSP.2001.941217
- El-gayar, M.M., Soliman, H., meky, N., 2013. A comparative study of image low level feature extraction algorithms. Egypt. Inform. J. 14, 175–181. doi:10.1016/j.eij.2013.06.003
- Elnemr, H.A., 2016. Combining SURF and MSER along with Color Features for Image Retrieval System Based on Bag of Visual Words. J. Comput. Sci. 12, 213–222. doi:10.3844/jcssp.2016.213.222
- Fergus, R., Perona, P., Zisserman, A., 2003. Object class recognition by unsupervised scale-invariant learning, in: Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on. IEEE, pp. II–264.
- Fitzgibbon, A., Zisserman, A., 2002. On Affine Invariant Clustering and Automatic Cast Listing in Movies, in: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (Eds.), Computer Vision — ECCV 2002, Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 304–320.
- Hu, W., Xie, N., Li, L., Zeng, X., Maybank, S., 2011. A Survey on Visual Content-Based Video Indexing and Retrieval. IEEE Trans. Syst. Man Cybern. Part C Appl. Rev. 41, 797–819. doi:10.1109/TSMCC.2011.2109710
- Jing, Y., Liu, D., Kislyuk, D., Zhai, A., Xu, J., Donahue, J., Tavel, S., 2015. Visual Search at Pinterest, in: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '15. ACM, New York, NY, USA, pp. 1889–1898. doi:10.1145/2783258.2788621
- Kavitha, H., Sudhamani, M.V., 2013. Object based Image Retrieval from Database using Combined Features. Int. J. Comput. Appl. 76, 38–42.
- Kavya, J., Shashirekha, H., 2014. A Novel Approach for Image Retrieval using Combination of Features. Int. J. Comput. Technol. Appl. 6, 323–327.
- Khan, N.Y., McCane, B., Wyvill, G., 2011. SIFT and SURF Performance Evaluation against Various Image Deformations on Benchmark Dataset. IEEE, pp. 501–506. doi:10.1109/DICTA.2011.90
- Madbouly, A.M.M., Wafy, M., Mostafa, M.-S.M., 2015. Performance Assessment of Feature Detector-Descriptor Combination. Int. J. Comput. Sci. Issues IJCSI 12, 87.
- McGuinness, K., McCusker, K., O'Hare, N., O'Connor, N.E., 2012. Efficient storage and decoding of SURF feature points, in: International Conference on Multimedia Modeling. Springer, pp. 440–451.
- Patel, B.V., Meshram, B.B., 2012. Content Based Video Retrieval Systems. Int. J. UbiComp 3, 13–30. doi:10.5121/iju.2012.3202

- Rathod, G.I., Nikam, D.A., 2013. An algorithm for shot boundary detection and key frame extraction using histogram difference. *Int. J. Emerg. Technol. Adv. Eng.* 3, 155–163.
- Reverse image search - Search Help [WWW Document], n.d. URL <https://support.google.com/websearch/answer/1325808?hl=en> (accessed 6.14.16).
- Sav, S., Johns, G.G.F., Lee, H., 2006. , in: *Image and Video Retrieval: 5th Internatinoal Conference, CIVR 2006, Tempe, AZ, USA, July 13-15, 2006, Proceedings*. Springer Science & Business Media, pp. 1–10.
- Schneiderman, H., Kanade, T., 2000. A statistical method for 3D object detection applied to faces and cars, in: *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*. IEEE, pp. 746–751.
- S.G., A., Karibasappa, K., Eswar Reddy, B., 2013. Video Segmentation for Moving Object Detection Using Local Change & Entropy Based Adaptive Window Thresholding. *Academy & Industry Research Collaboration Center (AIRCC)*, pp. 155–166. doi:10.5121/csit.2013.3916
- Sivic, J., Schaffalitzky, F., Zisserman, A., 2004. Efficient object retrieval from videos, in: *Signal Processing Conference, 2004 12th European*. IEEE, pp. 1737–1740.
- Sivic, J., Zisserman, A., 2004. Video data mining using configurations of viewpoint invariant regions, in: *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*. IEEE, pp. I–488.
- Sivic, J., Zisserman, A., 2003. Video Google: A Text Retrieval Approach to Object Matching in Videos, in: *Ninth IEEE International Conference on Computer Vision (ICCV 2003)*. Presented at the IEEE International Conference on Computer Vision (ICCV 2003), IEEE.
- Velmurugan, K., Baboo, L.D.S.S., 2011. Content-based image retrieval using SURF and colour moments. *Glob. J. Comput. Sci. Technol.* 11.
- Weiming Hu, Nianhua Xie, Li Li, Xianglin Zeng, Maybank, S., 2011. A Survey on Visual Content-Based Video Indexing and Retrieval. *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.* 41, 797–819. doi:10.1109/TSMCC.2011.2109710
- Yang, L., Cai, Y., Hanjalic, A., Hua, X.-S., Li, S., 2011. Video-based image retrieval, in: *Proceedings of the 19th ACM International Conference on Multimedia*. ACM, pp. 1001–1004.
- Yildirim, Y., Yazici, A., Yilmaz, T., 2013. Automatic Semantic Content Extraction in Videos Using a Fuzzy Ontology and Rule-Based Model. *IEEE Trans. Knowl. Data Eng.* 25, 47–61. doi:10.1109/TKDE.2011.189

Appendix A - Approach

A.1 DFD Data Dictionary

Data Member Name	Type	Default Value	Mandatory ?
Video folder path	Character	"" (empty)	Yes
Image path	Character	"" (empty)	Yes
Rated video analysis results	Complex	null	No
Harris_features	Complex	null	No
SURF_features	Complex	null	No
color_features	Complex	null	No
image	Complex	null	Yes
Key frames list	Complex	null	No
Video details	Complex	null	Yes
Video list	Complex	null	No

Table A.1 - DFD data dictionary

Appendix B - Design

B.1 Activity Diagrams

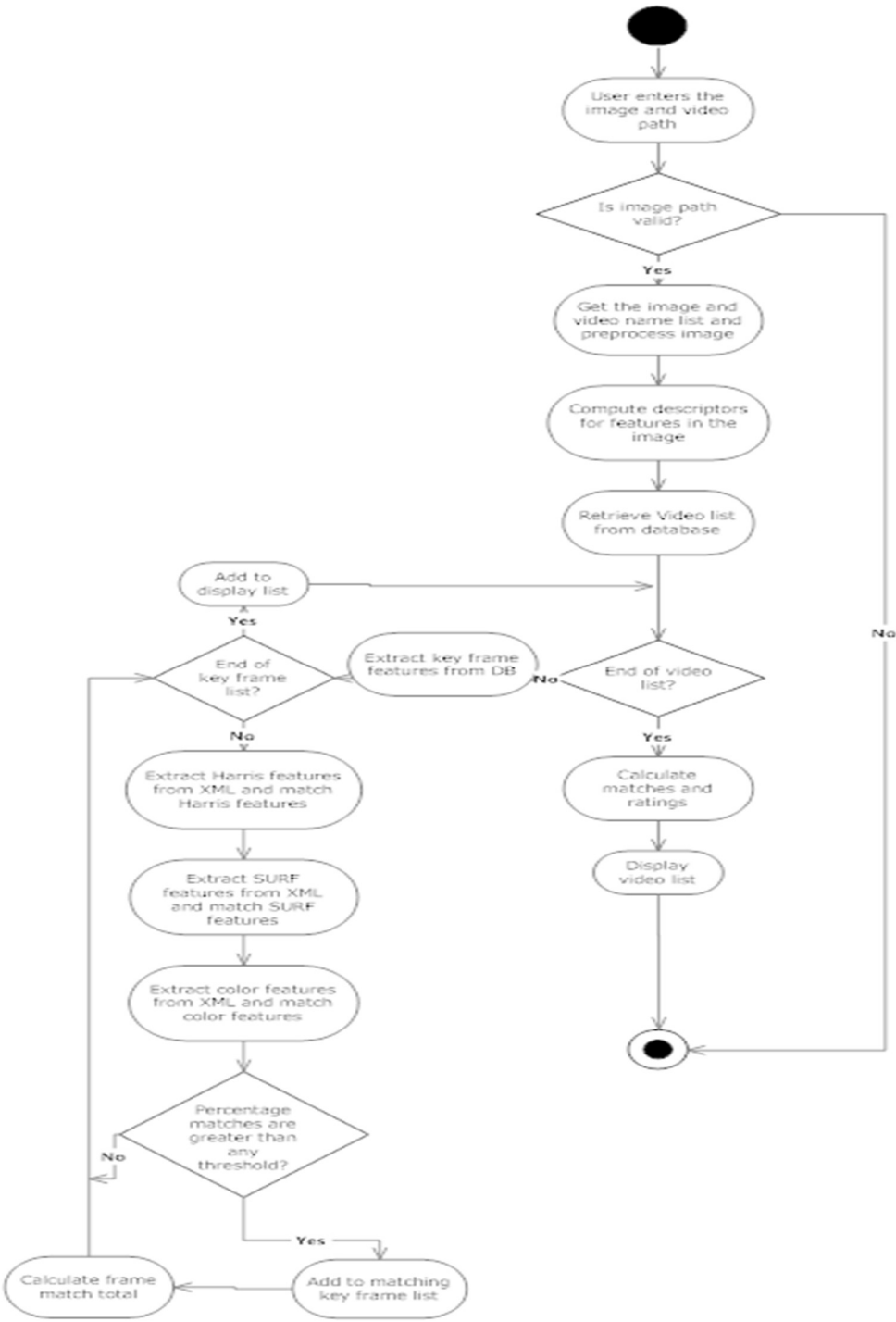


Figure B.1 – Activity diagram of Stored summary searching prototype search senario

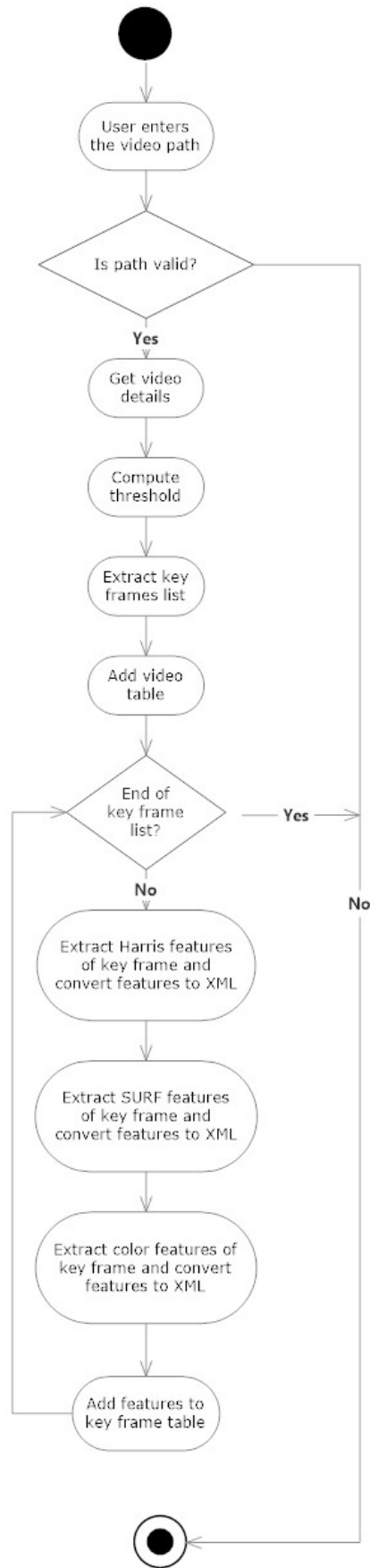


Figure B.2 - Stored summary searching prototype's Add Video scenario

Appendix C - Implementation

C.1 Code Listing

C.1.1 Get Key Frames

```
function [Keyframes, NOKF] = keyfOtsu(V, keyFramesFolder)
% Copyright (c) 2015, krishnapriya Subramanian
% All rights reserved.
% Modified by H. S. Senevirathna Copyright (c) 2017.

xyloObj = VideoReader(V);    %Using video reader reading video
%Extracting frames
NOF = xyloObj.NumberOfFrames;    % Calculating number of frames
disp(strcat('Number of frames = ', num2str(NOF)));

% special lengths of video
x = 1;
if (xyloObj.Duration >= 15 && 45 > xyloObj.Duration )
    x = 2+((floor(NO/1000)).^2);
elseif(xyloObj.Duration > 45) % not used
    x = 3+((floor(NO/1000)).^2);
end
x = floor(x);
NOFhalf = floor(NO/x);

Diffs = zeros(NOHalf,1); Frames = cell(NOHalf,1); %initialize
Frames{1} = read( xyloObj,x);    % Retrieve data from video and Add to cell
array
Frames{1} = resizeImage(Frames{1});

for g=1:NOHalf
    if(g~= NOHalf)
        y = (g + 1)* x;
        Frames{(g+1)} = read( xyloObj,(y));
        Frames{(g+1)} = resizeImage(Frames{(g+1)});
        diff = imabsdiff(Frames{g},Frames{g+1});
        I=rgb2gray(diff);    % Convert into gray scale
        level = graythresh(I); % Using Otsu's thresholding
        Diffs(g) = level;

%To calculate histogram difference between two frames
        %
        end
    end

%calculating mean and standard deviation and extracting key frames
mean=mean2(Diffs);
std = std2(Diffs);
threshold= (mean + std);
NOKF = 0;
Keyframes{1} = cell(1);
lastKFno = 0;
for g=1: NOHalf
    if(g~=NOHalf)
        th=Diffs(g);
        if(g == (NOHalf - 1) || th > threshold)    % Greater than
threshold select as a key frame
```

```

        NOKF = NOKF + 1;
        midKFno = ceil((lastKFno + g + 1)/2);
        if(midKFno ~= lastKFno && midKFno ~= g+1)
            Keyframes{NOKF} = Frames{midKFno};
            NOKF = NOKF + 1;
        end
        lastKFno = g + 1;
        Keyframes{NOKF} = Frames{g+1}; % Add to cell array
    end
end
end
end

```

C.1.2 Extract Color Features

```

function [features, metrics] = featuresColorExtractor(I)
% Example color layout feature extractor.
% Local color layout features are extracted from truecolor image, I and
% returned in features. The strength of the features are returned in
% metrics.

[~,~,P] = size(I);

isColorImage = P == 3;

if isColorImage

    % Convert RGB images to the L*a*b* colorspace. The L*a*b* colorspace
    % enables you to easily quantify the visual differences between colors.
    % Visually similar colors in the L*a*b* colorspace will have small
    % differences in their L*a*b* values.
    Ilab = rgb2lab(I);

    % Compute the "average" L*a*b* color within 16-by-16 pixel blocks. The
    % average value is used as the color portion of the image feature. An
    % efficient method to approximate this averaging procedure over
    % 16-by-16 pixel blocks is to reduce the size of the image by a factor
    % of 16 using IMRESIZE.
    Ilab = imresize(Ilab, 1/16);

    % Note, the average pixel value in a block can also be computed using
    % standard block processing or integral images.

    % Reshape L*a*b* image into "number of features"-by-3 matrix.
    [Mr,Nr,~] = size(Ilab);
    colorFeatures = reshape(Ilab, Mr*Nr, []);

    % L2 normalize color features
    rowNorm = sqrt(sum(colorFeatures.^2,2));
    colorFeatures = bsxfun(@rdivide, colorFeatures, rowNorm + eps);

    % Augment the color feature by appending the [x y] location within the
    % image from which the color feature was extracted. This technique is
    % known as spatial augmentation. Spatial augmentation incorporates the
    % spatial layout of the features within an image as part of the
    % extracted feature vectors. Therefore, for two images to have similar
    % color features, the color and spatial distribution of color must be
    % similar.

```

```

% Normalize pixel coordinates to handle different image sizes.
xnorm = linspace(-0.5, 0.5, Nr);
ynorm = linspace(-0.5, 0.5, Mr);
[x, y] = meshgrid(xnorm, ynorm);

% Concatenate the spatial locations and color features.
features = [colorFeatures y(:) x(:)];

% Use color variance as feature metric.
metrics = var(colorFeatures(:,1:3),0,2);
else

% Return empty features for non-color images. These features are
% ignored by the color feature matching.
features = zeros(0,5);
metrics = zeros(0,1);
end

```

C.1.3 Match SURF Features

```

function [match, percentage, partial_match] =
matchSurfFeatures(I1,I2,features1, features2, valid_points1, valid_points2,
colour_percentage)
partial_match = false;
indexPairs = matchFeatures(features1,features2);

totalFeatures = length(features1); %baseline number of features
numPairs = size(indexPairs,1); %the number of pairs
if(totalFeatures ~= 0 && numPairs ~= 0)
    status = 1; sizeOfinliers1 = 0;
    matchedPoints1 = valid_points1(indexPairs(:,1),:);
    matchedPoints2 = valid_points2(indexPairs(:,2),:);
    if(numPairs > 2)
        % disp(size(indexPairs,1));
        [~, inliers1, ~, status] =
estimateGeometricTransform(matchedPoints1, matchedPoints2, 'affine');
        if(status == 0)
            sizeOfinliers1 = size(inliers1,1);
        end
    end
    percentage = (numPairs * 100)/totalFeatures;
else
    percentage =0; sizeOfinliers1=0;
end

% disp('Surf : ');
if (numPairs == 0)
    % disp('We do not have this');
    % disp(percentage);
    match = false;
else
    if(size(I1,2) < size(I2,2))
        Tdist = size(I2,2);
    else
        Tdist = size(I1,2);
    end
end

```

```

averageDist = sum(indexPairs(:,2)-indexPairs(:,1))/numPairs;
percAffine = (sizeOfinliers1*100/numPairs);
if ((status ~= 0 && percentage >= 8 && (averageDist <= (Tdist/2) * 2.5)
)...
    || (percAffine >= 8 && status == 0) || percentage >= 10)

    match = true;
else
    % disp('We do not have this');
    % disp(percentage);
    match = false;
end
if(percentage >= 8)
    partial_match = true;
end
end

end

end

```

C.1.4 Match Harris Features

```

function [match, percentage, partial_match] =
matchHarrisFeatures(I1size,I2size,features1, features2, valid_points1,
valid_points2, colour_percentage)

partial_match = false;
indexPairs = matchFeatures(features1,features2);

totalFeatures = features1.NumFeatures; %baseline number of features
numPairs = size(indexPairs,1); %the number of pairs
if(totalFeatures ~= 0 && numPairs ~= 0)
    status = 1; sizeOfinliers1 = 0;
    matchedPoints1 = valid_points1(indexPairs(:,1),:);
    matchedPoints2 = valid_points2(indexPairs(:,2),:);
    if(numPairs > 2)
        [tform, inliers1, inliers2, status] =
estimateGeometricTransform(matchedPoints1, matchedPoints2, 'affine');
        if(status == 0)
            sizeOfinliers1 = size(inliers1,1);
        end
    end
    percentage = (numPairs * 100)/totalFeatures;
else
    percentage =0; sizeOfinliers1=0;
end

% disp('Harris : ');
if (numPairs == 0)
    % disp('We do not have this');
    % disp(percentage);
    match = false;
else
    if(I1size < I2size)
        Tdist = I2size;
    else
        Tdist = I1size;
    end
    averageDist = sum(indexPairs(:,2)-indexPairs(:,1))/numPairs;

```

```

percAffine = (sizeOfInliers1*100/numPairs);
% disp(strcat('max=',num2str(max(indexPairs(:,end)))))
if ((status ~= 0 && percentage >= 2 && averageDist <= (Tdist/2) * 0.9)
...
    || (percAffine >= 2 && status == 0) || percentage >= 6)
%     disp('We have this');
%
disp(percentage);disp(averageDist);disp(size(features1.Features, 2));
    match = true;
else
%     disp('We do not have this');
%     disp(percentage);
    match = false;
end
if(percentage >= 4)
    partial_match = true;
end
end
end
end

```

C.1.5 Match Color Features

```

function [percentage, match, partial_match] =
matchColorFeatures(I1,I2,features1, features2)

partial_match = false;
indexPairs = matchFeatures(features1,features2);

totalFeatures = length(features1); %baseline number of features
numPairs = length(indexPairs); %the number of pairs
if(totalFeatures ~= 0)
    percentage = (numPairs * 100)/totalFeatures;
else
    percentage =0;
end

% disp('Color : ');
if (numPairs == 0)
%     disp('We do not have this');
%     disp(percentage);
    match = false;
else
    if(size(I1,2) < size(I2,2))
        Tdist = size(I2,2);
    else
        Tdist = size(I1,2);
    end
    averageDist = sum(indexPairs(:,2)-indexPairs(:,1))/numPairs;
    if (percentage >= 3.9 && (averageDist <= Tdist/2 * 2.5)) || percentage
>= 50
%     disp('We have this');
%     disp(percentage);disp(averageDist);disp(size(features1, 2));
        match = true;
    else
%     disp('We do not have this');
%     disp(percentage);
        match = false;
    end
end

```

```

        if(percentage >= 15)
            partial_match = true;
        end
    end
end
end

```

C.1.6 Resize Image

```

function image = resizeImage( image )
if(size(image,2) > 640)
    p1 = imresize(image(:,:,1), [NaN 640]);
    [m,n,~]=size(p1);
    image=zeros(m,n,3);
    image(:,:,1) = p1;
    image(:,:,2) = imresize(image(:,:,2), [m 640]);
    image(:,:,3) = imresize(image(:,:,3), [m 640]);
elseif(size(image,1) > 480)
    p1 = imresize(image(:,:,1), [480 NaN]);
    [m,n,~]=size(p1);
    image=zeros(m,n,3);
    image(:,:,1) = p1;
    image(:,:,2) = imresize(image(:,:,2), [480 n]);
    image(:,:,3) = imresize(image(:,:,3), [480 n]);
end
end

```

C.1.7 Connect to Database

```

function conn = ConnectToDB()
dbpath = ['D:\MSc\Project\SurfSearchVid\VidSearchNone.accdb'];
url = [['jdbc:odbc:Driver={Microsoft Access Driver (*.mdb,
*.accdb)};DSN=''';DBQ='] dbpath];
conn = database('','','','sun.jdbc.odbc.JdbcOdbcDriver',url);
end

```

C.1.8 Search Videos

```

function searchVids(imgPath,vidFolder)
disp(datetime('now'));
keyFramesFolder = 'Evaluation\Keyframes';
image = imread(imgPath);
k = dir(strcat(vidFolder,'*.mp4'));
filenames = {k.name}'; % get video files
noOfVids = length(filenames);
if (noOfVids == 0)
    disp('The folder location of videos is not found or empty.');
```

```

end
image = resizeImage(image);

% Get features of the image
image_features_Surf = GetSurfFeatures(image);
image_features_Harris = GetHarrisFeatures(image);
[image_features_Color, metrics] = featuresColorExtractor(image);

numOfResults=0;

```

```

% searching each vid
for g=1: size(filenamees)
    video = filenamees(g);
    path = strcat(vidFolder,video);
    disp(video);
    pathString = char(path);

    [keyfArr, NOKF] = keyfOtsu(pathString, keyFramesFolder);
    disp(strcat('Number of keyframes = ',num2str(NOKF)));

    if (NOKF > 0)
        [sumOfPerc, numMatches] = searchOneVid(image, keyfArr,
keyFramesFolder, image_features_Surf, image_features_Harris,
image_features_Color, video);
    end
    disp(strcat(video, ' Number of results = ',num2str(numMatches)));
    if(NOKF ~= 0 && numMatches ~= 0)
        averageMatch = numMatches/NOKF;
        percMatch = averageMatch * 100;
        avgFeatureMatch = sumOfPerc/(3*numMatches);
    else
        avgFeatureMatch = 0;
        percMatch = 0;
    end
    disp(strcat(video, ' Matching ratio = ',num2str(avgFeatureMatch)));
    if(percMatch >= 10 && avgFeatureMatch >= 12.028) % thresholded
percentage of matching key frames - empirically derived
        numOfResults = numOfResults+1;
        VideoList{numOfResults,1} = char(video);
        VideoList{numOfResults,2} = avgFeatureMatch;
        VideoList{numOfResults,3} = percMatch;
        VideoList{numOfResults,4} = percMatch+avgFeatureMatch;

    end
end

% Display video list
if(numOfResults>0)
    VideoList = sortrows(VideoList, -4);
    disp(VideoList);
end

disp(strcat('No of matching videos : ',num2str(numOfResults)));
disp(datetime('now'));
end

```

C.1.9 Search One Video

```

function [sumOfPerc, numMatches] = searchOneVid(image,keyfArr,
keyFramesFolder, image_features_Surf, image_features_Harris,
valid_points_Surf, valid_points_Harris, image_features_Color, video)
numMatches = 0;
sumOfPerc = 0;
n = length(keyfArr);
for g=1: n
    %     disp(strcat(num2str(g),' of ',num2str(n)));
    [KF_features_color,metrics] = featuresColorExtractor(keyfArr{g});

```



```

    [colour_percentage, match3, partial1] =
matchColorFeatures(image, keyfArr{ (g) }, image_features_Color,
KF_features_color);

    I1 = rgb2gray(keyfArr{ (g) });
    points1 = detectSURFFeatures(I1);
    [KF_features_Surf, KF_valid_points_Surf] = extractFeatures(I1, points1);
    points2 = detectHarrisFeatures(I1);
    [KF_features_Harris, KF_valid_points_Harris] =
extractFeatures(I1, points2);

    [match1, percentage1, partial2] =
matchSurfFeatures(size(image,2), size(keyfArr{ (g) },2), image_features_Surf,
KF_features_Surf, valid_points_Surf, KF_valid_points_Surf,
colour_percentage);
    [match2, percentage2, partial3] =
matchHarrisFeatures(size(image,2), size(keyfArr{ (g) },2), image_features_Harris,
KF_features_Harris, valid_points_Harris, KF_valid_points_Harris,
colour_percentage);

    if ((match1 == true || match2 == true || match3 == true) || (partial1
&& partial2 && partial3))
        numMatches = numMatches + 1;
        sumOfPerc = sumOfPerc + percentage1 + percentage2 +
colour_percentage;
    end
end

end

```

C.1.10 Add video to database

```

function AddVidToDB( path )
%insert keyframe features into DB
disp(datetime('now'));

conn = ConnectToDB();
keyFramesFolder = 'Evaluation\Keyframes';
slash_place = strfind(path, '\');
video = path(slash_place(length(slash_place))+1:end);
if exist(path, 'file') ~= 2
    disp(strcat('The file ', video, ' does not exist in the path
', path(1:hash_place(1)-1)));
end

[keyfArr, NOKF] = keyfOtsu(path, keyFramesFolder); % extract key frames
% disp(size(keyfArr,2));
disp(strcat('Number of keyframes = ', num2str(NOKF)));

if(NOKF >0)
    KFsize = size(keyfArr{1},2);
else
    KFsize = 0;
end
colnames = {'VidName', 'FrameWidth'};
data = {video, KFsize};
tablename = 'TblVideo';

```

```

datainsert(conn,tablename,colnames,data);
result = exec(conn,'SELECT Distinct @@Identity FROM TblVideo');
result = fetch(result);
ID = result.Data{1};

colnames2 = {'VideoId', 'SurfFeatures', 'HarrisFeatures', 'ColorFeatures',
'SurfPoints', 'HarrisPoints'};
tablename2 = 'TblKeyFrame';

featuresSet = cell(NOKF,1);
for g=1: NOKF
    featuresC = featuresColorExtractor(keyfArr{g});
    I1 = rgb2gray(keyfArr{g});
    points1 = detectSURFFeatures(I1);
    [featuresS,KF_valid_points_Surf] = extractFeatures(I1,points1);
    points2 = detectHarrisFeatures(I1);
    [featuresH,KF_valid_points_Harris] = extractFeatures(I1,points2);

    strfS = strcat(int2str(size(featuresS,2)),'#',mat2str(featuresS));
    strfH =
strcat(int2str(size(featuresH.Features,2)),'#',mat2str(featuresH.Features))
;
    strfC = strcat(int2str(size(featuresC,2)),'#',mat2str(featuresC));
    strVPS =
strcat(int2str(size(KF_valid_points_Surf.Location,1)),'#',mat2str(KF_valid_
points_Surf.Location));
    strVPH =
strcat(int2str(size(KF_valid_points_Harris.Location,1)),'#',mat2str(KF_vali
d_points_Harris.Location));

    featuresSet{g} = {ID strfS strfH strfC strVPS strVPH};
%     disp(g);
end
disp('features extracted');
for g=1: NOKF
    data3 = featuresSet{g};
    datainsert(conn,tablename2,colnames2,data3);
%     disp('KF inserted. ');
    disp(g);
end
disp('End. ');

close(result);
close(conn);
disp(datetime('now'));
end

```

C.1.11 Search Videos from Database

```

function searchVidsFromDB(imgPath)
disp(datetime('now'));

keyFramesFolder = 'Evaluation\Keyframes';
image = imread(imgPath);
image = resizeImage(image);

% Get features of the image

```

```

I1 = rgb2gray(image);
points1 = detectSURFFeatures(I1);
[image_features_Surf,valid_points_Surf] = extractFeatures(I1,points1);
points2 = detectHarrisFeatures(I1);
[image_features_Harris,valid_points_Harris] = extractFeatures(I1,points2);
[image_features_Color, metrics] = featuresColorExtractor(image);

conn = ConnectToDB();
result = exec(conn,'SELECT VidName, ID, FrameWidth FROM TblVideo');
setdbprefs('DataReturnFormat','cellarray');
result = fetch(result,30);
videos = result.Data;
keyfArr = cell(5);
dims = size(videos);

% searching each vid
numOfResults=0;
if (dims(1) > 0 && dims(2) > 0)

    for g=1: dims(1)
        numMatches = 0;
        video = videos(g,:);
        vidId = uint32(video{2});
        frameWidth = uint32(video{3});
        disp(video{1});
        query = strcat('SELECT SurfFeatures, HarrisFeatures, ColorFeatures,
SurfPoints, HarrisPoints FROM TblKeyFrame WHERE VideoId =
',num2str(vidId));
        resultVid = exec(conn,query);
        setdbprefs('DataReturnFormat','cellarray');
        resultVid = fetch(resultVid);
        images = resultVid.Data;
        dims2 = size(images);
        sumOfPerc = 0;
        noOfKFs = dims2(1);
        disp(strcat('Number of keyframes = ',num2str(noOfKFs)));

        %Get key frames
        featuresSet = cell(noOfKFs,1);
        for i=1: noOfKFs
            vid_image = images(i,:);
            featuresS = single(str2matrix(vid_image{1},1));
            featureVectH = uint8(str2matrix(vid_image{2},1));
            featuresH = binaryFeatures(featureVectH);
            featuresC = str2matrix(vid_image{3},1);
            pointsMatS = single(str2matrix(vid_image{4},3));
            pointsS = SURFPoints(pointsMatS);
            pointsMatH = single(str2matrix(vid_image{5},3));
            pointsH = cornerPoints(pointsMatH);
            featuresSet{i} = {featuresS featuresH featuresC pointsS
pointsH};
        end

        %searching each image
        for i=1: noOfKFs
            %
                disp(strcat(num2str(i),' of ',num2str(noOfKFs)));
                [sumOfPerc, numMatches] = searchOneImageFromDB(numMatches,
sumOfPerc, image, keyfArr, keyFramesFolder,...

```

```

        image_features_Surf, image_features_Harris,
image_features_Color, featuresSet{i}, frameWidth, valid_points_Surf,
valid_points_Harris);
    end
    disp(strcat(video{1}, ' Number of results =
',num2str(numMatches)));
    if(noOfKFs ~= 0 && numMatches ~= 0)
        averageMatch = numMatches/noOfKFs;
        percMatch = averageMatch * 100;
        avgFeatureMatch = sumOfPerc/(3*numMatches);
    else
        avgFeatureMatch = 0;
        percMatch = 0;
    end
    disp(strcat(video{1}, ' Matching ratio =
',num2str(avgFeatureMatch)));
    if(percMatch >= 10 && avgFeatureMatch >= 12) % thresholded
percentage of matching key frames
        numOfResults = numOfResults+1;
        VideoList{numOfResults,1} = video{1};
        VideoList{numOfResults,2} = avgFeatureMatch;
        VideoList{numOfResults,3} = percMatch;
        VideoList{numOfResults,4} = percMatch+avgFeatureMatch;
    end
%     end
end

% Display video list
if(numOfResults > 0)
    VideoList = sortrows(VideoList, -4);
    disp(VideoList);
end
end
disp(strcat('No of matching videos : ',num2str(numOfResults)));
disp(datetime('now'));
end

```

C.1.12 Convert String to Matrix

```

function M = str2matrix( fulltext, mode )%,x,y
% creates a matrix document for the string

if (mode == 1)
    %     if(fulltext ~= '')
    hash_place = strfind(fulltext,'#');
    if(numel(hash_place)~=0)
        y = str2double(fulltext(1:hash_place(1)-1)); %get cols
        if(fulltext(hash_place(1)+1:hash_place(1)+5) ~= char('zeros'))
            text = fulltext(hash_place(1)+2:end-1); % remove #s and []
            row_end = strfind(text,',' );
            x = length(row_end)+1;
            M = zeros(x,y);

            % extract row contents
            if x>0
                for i=1:x
                    if i ~= 1
                        a = row_end(i-1)+1;

```

```

        else
            a = 1;
        end
        if i ~= x
            b = row_end(i)-1;
        else
            b = length(text);
        end
        row = text(a:b);
        ele_end = strfind(row, ' ');
        if y>0
            for j=1:y
                if j ~= 1
                    c = ele_end(j-1)+1;
                else
                    c = 1;
                end
                if(numel(ele_end)==0)
                    disp('end');
                end
                if j ~= y
                    d = ele_end(j)-1;
                else
                    d = length(row);
                end
                M(i,j) = str2double(row(c:d)); %get element
            end
        end
    end
end
else
    M = zeros(0,y);
end
else
    M = zeros(0,1);
end
elseif (mode == 2)
    hash_place = strfind(fulltext,'#');
    if(numel(hash_place)~=0)
        if(fulltext(hash_place(1)+1:hash_place(1)+5) ~= char('zeros'))
            text = fulltext(hash_place(1)+2:end-1); % remove #s and []
            rows = strsplit(text, ';');
            n = length(rows);
            M = zeros(n,1);
            % extract element contents
            if n>0
                for i=1:n
                    M(i,1) = str2double(rows{i});
                end
            end
        else
            M = zeros(0,y);
        end
    else
        M = zeros(0,1);
    end
elseif (mode == 3) % 2 column array
    hash_place = strfind(fulltext,'#');
    if(numel(hash_place)~=0)
        if(fulltext(hash_place(1)+1:hash_place(1)+5) ~= char('zeros'))

```

```

text = fulltext(hash_place(1)+2:end-1); % remove #s and []
row_end = strfind(text, ';');
x = length(row_end)+1;
M = zeros(x,2);

% extract row contents
if x>0
    for i=1:x
        if i ~= 1
            a = row_end(i-1)+1;
        else
            a = 1;
        end
        if i ~= x
            b = row_end(i)-1;
        else
            b = length(text);
        end
        row = text(a:b);
        ele_end = strfind(row, ' ');
        c = 1;
        d = ele_end(1)-1;
        M(i,1) = str2double(row(c:d));
        c = ele_end(1)+1;
        d = length(row);
        M(i,2) = str2double(row(c:d));
    end
end
else
    M = zeros(0,2);
end
else
    M = zeros(0,2);
end
end
end

```

C.1.13 Search an Image from Database

```

function [sumOfPerc, numMatches] = searchOneImageFromDB(numMatches,
sumOfPerc, image, keyfArr, keyFramesFolder, image_features_Surf,
image_features_Harris,...
    image_features_Color, key_frame, frameWidth, valid_points_Surf,
valid_points_Harris)
% perform matching

imageWidth = size(image,2);
[colour_percentage, match3, partial1] = matchColorFeatures(imageWidth,
frameWidth, image_features_Color, key_frame{3});
[match1, percentage1, partial2] =
matchSurfFeatures(imageWidth,frameWidth,image_features_Surf, key_frame{1},
valid_points_Surf, key_frame{4}, colour_percentage);
[match2, percentage2, partial3] =
matchHarrisFeatures(imageWidth,frameWidth,image_features_Harris,
key_frame{2}, valid_points_Harris, key_frame{5}, colour_percentage);

if ((match1 == true || match2 == true || match3 == true) || (partial1 &&
partial2 && partial3))
    sumOfPerc = sumOfPerc + percentage1 + percentage2 + colour_percentage;

```

end
end

Appendix D - Evaluation

D.1 bulkRunReal

```
function bulkRunReal(imgFolderPath,vidFolder1,vidFolder2,vidFolder3)
diary('diaryReal.txt');
k = dir(strcat(imgFolderPath,'*.jpg'));
filenames = {k.name}'; % get img files
% matching each img
disp(vidFolder1);
for g=1: size(filenames)
    disp(filenames{g});
    imgPath = strcat(imgFolderPath, filenames{g});
    searchVids(imgPath,vidFolder1);
end
disp(vidFolder2);
for g=1: size(filenames)
    disp(filenames{g});
    imgPath = strcat(imgFolderPath, filenames{g});
    searchVids(imgPath,vidFolder2);
end
disp(vidFolder3);
for g=1: size(filenames)
    disp(filenames{g});
    imgPath = strcat(imgFolderPath, filenames{g});
    searchVids(imgPath,vidFolder3);
end
diary off;
end
```

D.2 bulkRun

```
function bulkRun(FolderPath,mode)
diary('DB_prot_log2.txt');
if(mode==2)
    k = dir(strcat(FolderPath,'*.mp4'));
    filenames = {k.name}'; % get files
    % each video
    for g=1: size(filenames)
        disp(filenames{g});
        vidPath = strcat(FolderPath, filenames{g});
        AddVidToDB(vidPath);
    end
elseif(mode==3)
    k = dir(strcat(FolderPath,'*.jpg'));
    filenames = {k.name}'; % get img files
    % matching each img
    for g=1: size(filenames)
        disp(filenames{g});
        imgPath = strcat(FolderPath, filenames{g});
        searchVidsFromDB(imgPath);
    end
end
diary off;
end
```


D.3 Detailed Evaluation Data

D.3.1 Comparison of Speeds of the Two Prototypes

		Image	Start time	End time	Time difference
Stored summary	Good match	1	19:22:22	19:49:58	0:27:36
		2	19:49:58	20:18:01	0:28:03
		3	20:18:01	20:47:00	0:28:59
		4	20:47:00	21:15:20	0:28:20
		5	21:15:20	21:43:51	0:28:31
	Moderate match	1	21:45:13	22:13:34	0:28:21
		2	22:13:34	22:42:08	0:28:34
		3	22:42:08	23:10:28	0:28:20
		4	23:10:29	23:38:33	0:28:04
		5	11:38:33	12:06:41	0:28:08
	No match	1	0:35:23	1:21:38	0:46:15
		2	1:21:38	2:07:57	0:46:19
		3	2:07:57	2:54:06	0:46:09
		4	2:54:07	3:40:13	0:46:06
		5	3:40:13	4:27:28	0:47:15
Real-time	Good match	1	5:31:08	5:52:08	0:21:00
		2	5:52:08	6:13:16	0:21:08
		3	6:13:16	6:34:40	0:21:24
		4	6:34:40	6:55:48	0:21:08
		5	18:55:48	19:16:55	0:21:07
	Moderate match	1	7:16:55	7:44:58	0:28:03
		2	7:44:59	8:12:28	0:27:29
		3	8:12:28	8:40:31	0:28:03
		4	8:40:32	9:08:21	0:27:49
		5	9:08:22	9:36:00	0:27:38
	No match	1	9:36:01	9:50:32	0:14:31
		2	9:50:32	10:05:02	0:14:30
		3	10:05:02	10:19:42	0:14:40
		4	10:19:42	10:34:09	0:14:27
		5	10:34:09	10:48:34	0:14:25

Table D.1 - Evaluation details of prototype comparison

D.3.2 Search Results For Different Image Sets

Image No	0% set		mid% set		>90% set	
	Matches	%	Matches	%	Matches	%
5 images						
1	1	3.333333333	2	6.666667	0	0
2	5	16.66666667	11	36.66667	6	20
3	5	16.66666667	11	36.66667	6	20
4	1	3.333333333	1	3.333333	1	3.333333
5	5	16.66666667	1	3.333333	1	3.333333
		11.33333333	17.33333		9.333333	
10 images						
1	5	16.66666667	11	36.66667	6	20
2	5	16.66666667	11	36.66667	6	20
3	5	16.66666667	11	36.66667	6	20
4	5	16.66666667	11	36.66667	6	20
5	0	0	2	6.666667	2	6.666667
6	5	16.66666667	11	36.66667	6	20
7	6	20	11	36.66667	6	20
8	5	16.66666667	11	36.66667	6	20
9	5	16.66666667	11	36.66667	6	20
10	6	20	11	36.66667	6	20
		15.66666667	33.66667		18.66667	
15 images						
1	5	16.66666667	11	36.66667	6	20
2	5	16.66666667	11	36.66667	6	20
3	5	16.66666667	11	36.66667	6	20
4	5	16.66666667	11	36.66667	6	20
5	1	3.333333333	1	3.333333	5	16.66667
6	5	16.66666667	11	36.66667	6	20
7	5	16.66666667	11	36.66667	6	20
8	2	6.666666667	2	6.666667	3	10
9	5	16.66666667	11	36.66667	6	20
10	5	16.66666667	11	36.66667	6	20
11	5	16.66666667	11	36.66667	6	20
12	6	20	11	36.66667	6	20
13	9	30	5	16.66667	9	30
14	5	16.66666667	11	36.66667	6	20
15	12	40	5	16.66667	10	33.33333
		17.7777778	29.77778		20.66667	
20 images						
1	0	0	1	3.333333	0	0
2	4	13.33333333	1	3.333333	1	3.333333
3	5	16.66666667	2	6.666667	0	0
4	4	13.33333333	3	10	2	6.666667

6	6	20	11	36.66667	5	16.66667
6	6	20	11	36.66667	5	16.66667
7	6	20	11	36.66667	5	16.66667
8	6	20	11	36.66667	5	16.66667
9	6	20	11	36.66667	5	16.66667
10	6	20	11	36.66667	5	16.66667
11	6	20	11	36.66667	5	16.66667
12	6	20	11	36.66667	5	16.66667
13	6	20	11	36.66667	5	16.66667
14	6	20	11	36.66667	5	16.66667
15	6	20	11	36.66667	5	16.66667
16	12	40	6	20	11	36.66667
17	6	20	11	36.66667	5	16.66667
18	3	10	1	3.333333	4	13.33333
19	7	23.33333333	2	6.666667	2	6.666667
20	10	33.33333333	6	20	11	36.66667
19.5			25.66667		15.16667	

Table D.2 – Evaluation details of video results for different image sets

D.3.2 Comparison of Results of the Two Prototypes

		Image	Matches
Stored summary	Good match	1	0
		2	1
		3	1
		4	0
		5	1
	Moderate match	1	0
		2	7
		3	7
		4	0
		5	0
	No match	1	0
		2	5
		3	5
		4	0
		5	0
Real-time	Good match	1	1
		2	3
		3	3
		4	1
		5	5
	Moderate match	1	1
		2	8
		3	8
		4	0
		5	0
	No match	1	0
		2	5
		3	5
		4	0
		5	0

Table D.3 – Evaluation details of video results for different video sets

Glossary of Terms

- BRISK: Binary Robust Invariant Scalable Key-points
- CBVR: Content based video retrieval
- FAST: Features from Accelerated Segment Test
- FREAK: Fast Retina Keypoint
- F-SIFT: Fast-SIFT
- GPU: Graphics Processing Unit
- LSI: Latent semantic analysis
- MSAC: M-estimator SAmple Consensus
- MSER: Maximally Stable Extremal Regions
- PCA: Principal Component Analysis
- SIFT: Scale-invariant feature transform
- SURF: Speeded Up Robust Features
- SVM: Support Vector Machine