Characterization of Unstructured Video

by

Giridharan Ranganathan Iyengar

S. M. Media Arts and Sciences Massachusetts Institute of Technology
September 1995

SUBMITTED TO T IE PROGRAM IN
MEDIA ARTS AND SCIENCES,
SCHOOL OF ARCHITECTURE AND PLANNING
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

AT THE

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1999

Author: [illegible]
Program in Media Arts and Sciences
April 30, 1999

Certified by: [illegible]
Andrew B. Lippman
Associate Director, MIT Media Laboratory
Thesis Supervisor

Accepted by: [illegible]
Stephen A. Benton
Allen Professor of Media Arts and Sciences
Chairperson, Departmental Committee on Graduate Students
Program in Media Arts and Sciences

Title: Characterization of Unstructured Video

Author: Giridharan Iyengar

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
April 30, 1999

Abstract

In this work, we examine video retrieval from a synthesis perspective in co-operation with the more common analysis perspective. Specifically, we target our algorithms for one particular domain– unstructured video material. The goal is to make this unstructured video available for manipulation in interesting ways. I.e, take video that may have been shot with no specific intent and use it in different settings. For example, we build a set of interfaces that will enable taking a collection of home videos and making Christmas cards, Refrigerator magnets, family dramas etc out of them.

The work is divided into three parts. First, we study features and models for characterization of video. Examples are VideoBook with its extensions and Hidden Markov Models for video analysis. Secondly, we examine clustering as an approach for characterization of unstructured video. Clustering alleviates some of the common problems with "query-by-example" and presents groupings that rely on the user's abilities to make relevant connections. The clustering techniques we employ operate in the probability density space. One of our goals is to employ these techniques with sophisticated models such as Bayesian Networks and HMMs, which give similar descriptions. The clustering techniques we employ are shown to be optimal in an information theoretic and Gibbs Free Energy sense.

Finally, we present a set of interfaces that use these features and groupings to enable browsing and editing of unstructured video content.

Thesis Supervisor: Andrew B. Lippman
Associate Director, MIT Media Laboratory.

This research was supported by the Digital Life consortium.

Characterization of Unstructured Video

by

Giridharan Iyengar

Reader: ______________________________

Rosalind W. Picard
Associate Professor
Program in Media Arts and Sciences

Reader: [illegible]

Robert A. Ulichney
Senior Consulting Engineer
Cambridge Research Laboratory
Compaq Corporation

To my parents and my teachers

Contents

1 Introduction and Motivation **15**

1.1 The Problem Statement 15

1.2 Motivation 16

1.3 Questions raised and addressed in this thesis 17

1.4 Thesis contributions 20

1.4.1 Content Clustering Engine 21

1.4.2 Browser 22

1.5 Road map 23

2 Related Work in Content-based Retrieval of Images and Video **24**

2.1 Annotation based approaches 25

2.1.1 Relational database extended with image features 25

2.1.2 Text-assisted image features based systems 26

2.2 Feature similarity based approaches 28

2.2.1 Color based techniques 28

2.2.2 Approaches based on Texture, Shape and other features 29

2.2.3 Apriori classification to improve performance 32

2.3 Video event detection and representation 33

2.3.1 Video shot/scene detection and key frame extraction 34

2.3.2 Browsing and summarization of video 36

2.3.3 Content-based video retrieval 38

2.4 Bayesian Networks for Content Characterization 40

3 Video Content Characterization **43**

3.1 VideoBook: An experiment in characterization of video 44

3.1.1 Feature extraction and representation 44

3.1.2 Query and Retrieval Experiments 46

3.2 Evolving discriminators for video sequences 49

3.2.1 Evolution of VideoBook discriminators 51

3.2.2 Performance of the evolved discriminators 56

3.3 HMMs for characterizing video sequences 57

3.3.1 Television Sports/News classifier 58

3.3.2 Action/Character movie trailer classifier 64

3.4 Controlled retrieval using VideoBook 71

3.4.1 Classification of Trailers using KL divergence 73

3.4.2 Joint grammar and visual query 73

3.5 Why probabilistic descriptions? 75

4 Density-based Clustering **79**

4.1 Why clustering for unstructured video? 80

4.2 Introduction to Distribution Clustering 82

4.3 Hard partition clustering 83

4.3.1 Clustering using KL divergence 83

4.3.2 Clustering using Euclidean norm 86

4.4 Hierarchical soft clustering 88

4.4.1 Framework for probabilistic clustering 88

4.4.2 Soft distribution clustering 91

4.5 Agglomerative distribution clustering 94

4.5.1 Hard bottom-up hierarchical clustering 96

4.5.2 Soft bottom-up hierarchical clustering 97

4.6 Clustering experiments 102

4.6.1 Clustering Algorithms 103

4.6.2 Performance evaluation 106

5 Browser and Editors for Unstructured Video **117**

5.1 Representation of unstructured video 118

5.2 Architecture of the System 119

5.2.1 Cluster Browser 120

5.2.2 Centroid Browser 121

5.2.3 Still editor module 122

5.2.4 Collage editor module 123

5.2.5 Movie editor module 124

6 Conclusions and Future Work **129**

6.1 Contributions 129

6.2 Future Work 130

A Likelihood of data under a novel distribution **131**

B Convexity of KL divergence **133**

Bibliography **135**

Acknowledgments **140**

List of Figures

1.1 Outline of proposed system 22

3.1 Spatio-temporal chunk representation of video 45

3.2 Key frame from the last five seconds of the Television news sequence 48

3.3 Key frame from the closest match for the above query sequence 49

3.4 Key frame from the next closest matching sequence 50

3.5 Key frame from the sports query sequence 51

3.6 Key frame from the next closest matching sports sequence 52

3.7 Tree representation of an expression 53

3.8 Example of a crossover operation. Two subtrees are exchanged 54

3.9 Example of a mutation operation. The top operator is changed 55

3.10 Plot of average fitness versus log time (number of iterations) 56

3.11 One of the 12 Low Action training templates 61

3.12 One of the 12 High Action training templates 62

3.13 Histograms of the two projections for the 24 training template frames . . . 63

3.14 Plot of average shot length versus motion energy 68

3.15 Histogram of the ME/SL ratio for the trailer "circle" 70

3.16 The two estimated pdfs. Green curve is the "character" pdf and the red curve is the "action" pdf 71

3.17 Sample query shot for $w = 0.5$ 76

3.18 Corresponding retrievals for $w = 0.5$. Note a similarity value closer to 1 implies a better match 78

4.1 Plot of β vs. Number of clusters indicating cluster regimes. 95

4.2 Dendrogram representing the agglomerative clustering algorithm. 96

4.3 Sample images in the test database. 107

4.4 Sample Rotated and Noisy Images. The Top 2 Images are Rotated by 45 degrees. Bottom Right Image is the Noisy Version of the Bottom Left Image. 109

4.5 Sample database image and its 25% sub-image used for query. 110

4.6 ROC plot for scaled query. All but the Hard AM plot are co-located at the 100% line. 112

4.7 ROC plot for rotated query. All but Hard AM are near the top. 113

4.8 ROC plot for subimage query. 114

4.9 ROC plot for 15% pixel distortion query. 115

4.10 ROC plot for contrast change query. 116

4.11 ROC plot for additive gaussian noise query. 116

5.1 Architecture of the browser/editor application 119

5.2 Cluster browser view of movie shots database 121

5.3 Cluster browser view of home video database 121

5.4 Centroid browser showing some shots in the "field" cluster 122

5.5 Centroid browser showing some shots in the “living room” cluster 123

5.6 Original still picture of butterfly 124

5.7 KL collage of the butterfly 125

5.8 Original still picture of turtle 126

5.9 KL collage of the turtle 127

5.10 Automatically generated 2-minute summary of the home video collection . . 128

List of Tables

3.1 Performance of the evolved discriminators 57

3.2 Confusion matrix for training set 64

3.3 Confusion matrix for test set 64

3.4 Table of average shot lengths, motion energy and KL distances 67

3.5 iMDB classification: "A" is action oriented and "C" is character oriented . 69

3.6 Classification results for the 3-node HMM model 72

3.7 Final classification of trailers using the SKL distance 74

4.1 Cluster experiments for scaled query image 111

4.2 Cluster experiments for query image rotated by 45 degrees 111

4.3 Cluster experiments for sub image query 112

4.4 Cluster experiments for 15% pixel distortion 113

4.5 Cluster experiments for contrast change query 114

4.6 Cluster experiments for query image distorted with gaussian noise 115

Chapter 1

Introduction and Motivation

1.1 The Problem Statement

Understanding the structure of video (more generally, media) is a fundamental problem that is worthy of research without any underlying application in mind. There are things that we simply cannot glean from a microphone or camera, and they cross all potential uses. For example, we cannot yet place a microphone in front of an orchestra and deduce the score. Similarly, we cannot yet meaningfully quantify the relationship between two video sequences; we neither know the elements that we are comparing nor do we have strong metrics for evaluating our conclusions. These are long-term research issues marked by incremental steps peppered with occasional breakthroughs.

It is of considerable value to proceed by solving particular and useful real-world problems, especially if they can help us understand the larger issue better. For example, we learn about biology and life by trying to treat a disease. We may wish to remove the sound of a cough from a concert very specifically, or we may wish to match a face before a camera with one in a database. These problems have been posed by questions of surveillance but a good solution both helps and teaches. In other words, these issues meet specific needs and at the same time enhance our ability to build structure out of media.

In this thesis we choose one such specific domain: unscripted and casually recorded video and one solution: a representation that permits it to be catalogued, searched and assembled into a variety of motion and still presentations. This material is what often fills the reels of consumer camcorders. I argue that this is a useful situation for the following reasons: It is looking at video retrieval from a synthesis point of view as opposed to the analysis view. Also, such video has no scripted structure to facilitate its navigation and therefore makes the problem more interesting. Further, it can be solved by using the machine to help sort, manipulate and select images that support or suggest an idea.

The focus of the thesis is therefore, making video useful. The idea is to build a set of tools that analyze, characterize, and prepare footage shot with a camera for unanticipated uses. We are targeting fun as the utility of the material. The idea is to take this ensemble of raw stuff and use it in interesting ways that are not just making a sequence out of them. Some of the possible uses of such unscripted/unstructured video are: Make a time slice of a person or a series of events, select a group of shots and make a postcard out of them, make a collage of shots, make a Salient Still, storyboard etc. Some of these outputs may be used to guide other constructions of the material. For example, the storyboard may form the basis of a family drama. The construction process may suggest ways to efficiently encode and transmit the resulting sequence.

1.2 Motivation

Recently, the technology of transmitting and storing high-bandwidth media such as video has far outpaced our ability to meaningfully manipulate it. This situation is going to worsen in the near future. For example, NASA's Earth Observing System will generate 1 Terabyte of image data per day when fully operational. A content-based system is required to effectively navigate and retrieve information from such immense repositories. In addition, video is becoming the medium of choice for recording human experience, both personal and collective. This position occupied by video will only be strengthened with the availability of future high-bandwidth smart networks. Moreover, a promise of digital technology is that bits of one kind can be converted to bits of another kind. I.e, a still image can be created from a video sequence or a collage can be made from a collection of stills. This thesis derives its motivation from these facts and strives to make day-to-day communication using video simple and intuitive.

1.3 Questions raised and addressed in this thesis

Unlike scripted video (such as movies and commercials), video that has been shot for the simple purpose of recording an event or place has no structure or meta-data to facilitate its access. Frequently, the only method of access is a linear navigation through the entire contents. While this may be acceptable for passive watching of video, the paucity of structural information renders it useless for later retrieval and reuse. Moreover, the commercially available non-linear editing systems are oriented towards structured video. They are good for editing but not useful for navigating and creation of structure where none was present to begin with. Further, such systems take video sequences and create another sequence out of them and are not geared towards other types of outputs. In contrast, content-based retrieval systems hold the promise to wade through large repositories of imagery. Media Streams [8] is a video representation language with special attention to the issue of creating a global, reusable video archive. It amply illustrates the usefulness of having structure in video for retrieval and reuse. However, it relies on manual annotation to add structure to video. The work done by Bobick, Iyengar, and others [2, 20, 21, 23] characterizes video based on the action in addition to the content. Once the video has been characterized using these techniques, retrieval becomes feasible albeit possibly imprecise. However, precision is not a problem as long as the retrieval is useful. With useful retrieval, comes reusability of video and transformation to other types of representations (such as stills, collages, storyboards etc).

These uses require two things: a mechanism to characterize and classify the content and a suitable navigational aid through the content. In addition, each output format requires a dedicated composer/editor. The characterization techniques developed by us may be well suited for the first task since they analyze both content and action [20, 21, 23]. Since the underlying content is going to be used in unanticipated ways, the task of building a browser to navigate through the content is formidable. Some of the traditional approaches towards indexing and annotation are ill-suited for this purpose. Annotation is time and labor-intensive, and may not be usable for all types of applications. Moreover, annotation based systems have to deal with operator subjectivity [43]. Systems that explicitly model video and permit browsing using simple language (such as the Vasconcelos' plain English browser [62]) often have a limited vocabulary by design and therefore cannot support any unintended query. When the video is structured, we can rely on editing artifacts and other structural cues to characterize the video. For example, see Ref. [23] for examples of using hidden markov models to analyze newscasts and sports and to classify movie trailers either as character or action movies. Ref. [64] exploits the script information present in television situational comedies to characterize content. These systems, make reasonable structural assumptions about the nature of particular types of video and exploit them in their classification. Such assumptions will not be valid for unstructured video in general.

In contrast, query-by-example systems coupled with user-feedback are better suited to this task since they do not make any assumptions of the nature of use of the content. However, these systems suffer a couple of limitations. In order to provide flexibility, such systems frequently permit direct access to the features and their weighting in the query-retrieval process. This is non-intuitive and often counter-productive. Another limitation is that query-by-example systems start by presenting a random sampling of the database and then tune themselves with user interaction. Presenting a useful first page is a difficult problem since it has to be representative of the database. This is particularly important as the size of the database increases. Any small set of samples cannot meaningfully represent the database well. Sclaroff et al. [6] refer to this as the *page-zero problem.* The notion of serendipitous discovery of the "right" image/video in the query-by-example paradigm gets weak if the initial subset that is presented is not representative. Moreover, such systems typically search over the entire database before returning the results. This implies that the computational requirements grows with the database size. In addition, we also need to examine issues such as the right feature set and the minimum number of features to guarantee good retrieval. This thesis does not address this issue of feature space selection, but it does address the page-zero problem by employing clustering for browsing.

Query-by-example pays the price by relying on lower-level features. The gap between the machine representation and the human perception is wider compared with model-based approaches. Another important problem with query-by-example is that it has not characterized or abstracted the database in any particular way. It searches the entire database every time a query is presented. While there are efficient search techniques to reduce this search complexity, the problem of not having generalized and abstracted the data remains. The abstraction problem can be addressed by allowing the system to learn via user iteraction and propagate newly learned labels. This can be viewed as the system having the ability of "short term memory". This system achieves the happy middle ground between pure query-by-example with no assumptions at one end and Bayesian modeling with structural assumptions at the other extreme.

Our approach to tackling the unstructured video problem has been to use clustering techniques to create multiple groupings of the video. Clustering is a natural way to organize massive quantities of data and to take advantage of the cognitive abilities of the user. For example, let us suppose that the user presents a video collection that is made up of shots from a typical home video. It is reasonable to expect that good clustering algorithms will group all shots that have similar features together. For example, most shots taken indoors will be in one cluster and most shots taken outdoors will be in another cluster, purely from the color features. The user views these clusters and attaches labels. The clustering algorithm relied on the user's ability to make inferences and attach cognitive meaning. This is not to say that it will work all the time. Some clusters may be such that we cannot provide clean labels. The interesting point is that from the perspective of extracting alternative outputs from the video (like collages, storyboards, still pictures and video dramas), it is not important that all clusters make cognitive sense to the user.

In addition, these groupings can also be augmented by specifically designed higher-level model-based approaches such as the ones in [23, 62]. Once video is thus clustered, we manipulate and access it via a set of specialized application modules that enable us to synthesize alternate outputs from video. In order to achieve these, we choose a common framework that supports these different techniques: probabilistic descriptions derived from extracted features. These probabilistic descriptions range from simple histograms of features at one end to joint density estimates from a Bayesian network at the other extreme. Using probabilistic descriptions provides us with some advantages apart from being able to combine descriptions with differing levels of complexity.

- They are robust to missing/corrupted data. That is, they can handle dropped frames, corrupted blocks etc. Their performance degradation is gradual.
- They can be multimodal. That is, an image and a video sequence can be treated similarly. An image can be used to query for video and vice-versa.
- They enable multiple groupings. For example, a picture of a crowd taken outdoors can be categorized both as a picture of a crowd and as an outdoors picture. Probabilistic descriptions provide confidence estimates for each type of categorization which is then used for multiple groupings.
- They are well understood and provide intuitive explanations and reasoning similar to the way humans relate to data.

1.4 Thesis contributions

In this thesis, we group the contributions into three distinct sections. We first present techniques for classification of video by content. These techniques range from simple features-based approaches most suited for unstructured video to semantics-based models most suited to classification of structured video such as newscasts and sportscasts. We then examine a clustering technique in the probability density space. This technique treats video sequences as a probability density along a specific feature or groups of features and partitions all the video sequences in the database along these groups of features. The final part is the application front-end that combines the features and models with the density clustering to enable access into unstructured video and permits the kinds of manipulations that we envisaged earlier.

To summarize, the contributions of this thesis are:

- A suite of (expandable) classification techniques that analyze video by action/content etc.
- A clustering approach that takes as inputs probability density models of images and video sequences and groups them in this density space. This approach is not limited to image and video but is generally applicable to all domains that model data as probability densities. We present 3 different clustering algorithms based on this approach.
- A browser to search through the video content. This browser comprises a set of displays to enable users to create and view possible outputs of such a system. Examples include a movie display, a stills display, and a collage display.

Figure 1.1 shows the outline of the proposed system.

1.4.1 Content Clustering Engine

Under the proposed model, when a video is entered into the system, it is analyzed and a set of features are extracted from it. These features are stored as indices associated with the video. For each such extracted feature, a probability density representation is computed.

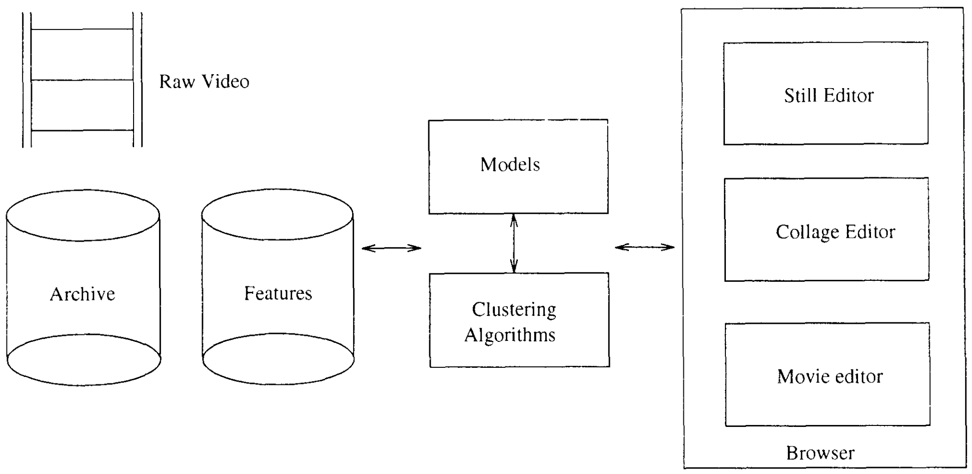


Figure 1.1: Outline of proposed system

The clustering engine groups this collection of density functions using the distributional clustering algorithms presented in Chapter 4. These groupings are made available to the browser and are the starting point for enabling structure in unstructured video.

1.4.2 Browser

The browser provides direct, low level access to the content. It is a framework for molding video into new packages. It is extensible in the different tools and metrics that it offers to search through the content. The browser supports multiple output formats, each with a specialized display module.

Stills module

The stills module provides the user with an interface to search the database and locate a particular still image. This image then can be used a regular still image - it can be printed, manipulated using tools such as Photoshop.

Collage module

The collage module is an extension of the stills module. It permits constructions such as a storyboard or a collage from the material. This storyboard can be used by the Movie module to put together a video sequence.

Movie module

The Movie module suggests movies by piecing together a storyboard from the clustered video shots. This storyboard can be seen as an alternative to summarization and a launching pad for movie constructions.

1.5 Road map

The rest of the thesis is organized as follows. Chapter 2 reviews some of the current work in video and image characterization. Chapter 3 presents my work in video characterization and makes the case for probabilistic descriptions. Chapter 4 motivates the relative entropy based clustering algorithms that constitute the primarily technical contribution of this thesis. The next chapter details the application that uses these clustering techniques for constructing interesting alternate outputs from video. Summary and future work are detailed in 6

This thesis can be read in multiple ways. If you accept the argument of clustering for unstructured video, you can skip directly to Chapter 5. For more details on the clustering algorithms used, it is recommended that you read Chapter 4. For a broad overview of content-based retrieval and reasons why we selected probabilistic representations for clustering, please read Chapters 2 and 3.

Chapter 2

Related Work in Content-based Retrieval of Images and Video

In this chapter, we categorize some of the major work in image and video retrieval. Content-based retrieval is motivated by the ever-increasing availability of images and video sequences in our information networks. Ideally, we want to be able to specify arbitrarily complex queries and yet expect meaningful answers.

Image and Video retrieval techniques can be roughly categorized as

- **Annotation based approaches**. These emphasize the usage of textual annotation for retrieval of similar images. They either use textual databases modified to handle images or guide a visual query by weighting the image features with contextual text.
- **Feature similarity based approaches**. These extract features from images and video and define similarity in this feature representation.
- **Video event detection based techniques**. These extract *significant* events in a sequence such as cuts, fades and dissolves and use this description to summarize the

video content. In addition, the extracted key frames are represented using features and the problem of video retrieval is often turned into one of image retrieval on these extracted key frames. This approach can be seen as decomposing video retrieval into a two stage process – extraction & representation of significant events in video as still images and constrained retrieval of these still images.

- **Bayesian Inference based approaches**. These approaches impose prior knowledge about the feature content for particular classes of images and video (such as outdoors, closeups, sports, action movie) and turn the task of detecting similarity into Bayesian Inference.

We will study illustrative examples from each approach. In addition, we will review some complete system perspectives.

2.1 Annotation based approaches

2.1.1 Relational database extended with image features

The Chabot/Cypress [38] system integrates relational database retrieval concepts with color analysis techniques. This system was initiated to study storage and retrieval of vast collections of digitized images at the Department of Water Resources, State of California. Each image is accompanied by extensive meta data (including the date of the photograph, photographer, location and format of the original image). The goal of this system was to augment an entirely manual operation that was in place prior to Chabot. The queries on this system ranged from highly specific, asking for images of a particular location at specified times to generically arbitrary such as "sunsets". Chabot was built on top of an object-oriented database called Postgres which permits SQL-like queries. In addition, simple functions written by the user can be registered into the database and be used on images at query time. For example, the Chabot system had functions written to compute color histograms.

In addition, Chabot permitted creation of new "concepts" using these functions and SQL constructs. The shortcoming with Chabot (and its later variation, Cypress) is that it relied primarily on the metadata for its queries. If a user had to search on "concepts", they had to specify it in terms of metadata and/or in terms of simple primitives (such as range of color values). The system did not attempt to connect the annotation with visual primitives.

2.1.2 Text-assisted image features based systems

In contrast, the Piction system [55] adopted the approach of assisted vision. It focused on a computational model for understanding pictures based on accompanying descriptive text. In their approach, "understanding" is defined as identifying relevant people and objects. They use the caption information to prime a vision system by providing a context. Similarity between query and captioned images is based on 4 distinct sources of information:

- Inexact text content (from descriptive captions).
- Exact objective text (from manually entered keywords).
- Inexact image similarity (based on extracted image features).
- Exact image-based objective term (objects/people identified by the vision system via descriptive captions).

Reliance on multiple sources of information can counter the problems with a purely textual system. E.g, "Four aircrafts performing daredevil stunts at the Air Show. President Clinton took part in the celebrations ..." implies to the human user that the picture will not contain President Clinton but a purely textual system will bring this as a match for a query on Bill Clinton. However, a vision system (if it works) will be unable to identify a face in the picture and therefore return lower similarity for this caption. The limitation with this approach is that even simple descriptive captions need a good natural language processing capabilities to properly guide a vision system. At present, our understanding of both computer vision and natural language processing is far from the level of sophistication needed to make this system work effectively. The authors report 65% performance on a small database of newspaper caption images.

LaCascia et al [6] combine visual features with textual statistics into a single index vector for content-based searches of a WWW image database. Latent Semantic Indexing (LSI) is used to capture the textual statistics in the HTML document. They assign importance to the text surrounding an image in the HTML document based on its proximity and the HTML tag type. A weighted term-image matrix is created for each such image in the document. Singular value decomposition (SVD) is then used to decompose the term-image relationships into a set of linearly independent vectors. This resultant LSI vector provides the context associated with an image and is combined with the computed visual feature vectors. The extracted visual statistics are the color histogram and the dominant orientation histogram. To reduce the dimensionality of feature vectors, they perform principal component analysis (PCA) and select a smaller subspace. The global feature vector consists of the LSI textual vector and the PCA color and directionality vectors. They then use a k-nearest neighbors algorithm to match the query with the images. In addition, relevance feedback is used to select the appropriate weighting functions for the various similarity values (text, color and directionality). The authors use this combined approach to mitigate the page-zero problem associated with query-by-example systems. Since this approach does not rely on such a close connection between the image features and textual features as the Piction system, a simpler text retrieval technique such as LSI works fairly well. However, this system relies on the extensive textual annotation that is typical in HTML documents with embedded images. For systems where there is no access to contextual text (such as unstructured video, as in our problem) this approach is not suitable. We next examine image retrieval systems that rely primarily on image features (such as color and texture) for retrieval.

2.2 Feature similarity based approaches

In these systems, the problem of retrieval is phrased as finding appropriate features that can be extracted from images and defining similarity functions in these feature spaces such that two images whose features are "close" are visually similar to humans as well.

2.2.1 Color based techniques

One of the most commonly used feature for content-based retrieval is the color histogram of the image. It is reasonable to expect that visually similar images will have similar color compositions and therefore histograms are expected to be revealing. Color histograms have several other advantages – invariance to size, scale and rotation of images that makes them attractive for content-based retrieval. Also, they are compact relative to the size of the image. They have been shown to perform very well. Once the image is described as a histogram in the colorspace, we can choose one of many different similarity measures to rank images. Swain and Ballard [57] introduced Histogram Intersection, which is equivalent to the Manhattan distance (L_1 norm)for normalized histograms. Other commonly used measures are all the L_n norms, with L_2 and L_1 being the most common[27, 50], the χ^2 estimate[3] and the KL divergence[25][1]. One of the problems using color histograms directly is that a slight shift in the feature values can result in large differences using any of these commonly used similarity measures. The QBIC system [12] attempts to counter for that by using a quadratic distance measure that quantified the similarity between bins. However, the quadratic distance similarity results in a significantly higher computation at the retrieval stage. Jain and Vailaya [27] proposed an image smoothing approach to relate the histogram bins. A similar approach has been proposed by Zabih et al [41]. In this thesis, we will present another general approach to quantify the inter-bin similarity [22, 25]. This technique, along with [27] and [41] can be viewed as a preprocessing step in the computation of the histogram.

[1]We note here that the χ^2 estimate is related to the KL divergence. It can be shown that it is the expansion to two terms of the Taylor series of the KL divergence [7].

Unlike the quadratic distance, these techniques result in an efficient computation during the retrieval stage.

While histograms are compact and very good for image retrieval, a major shortcoming with them is the lack of local information. It is easy to pick example images with similar histograms that are visually very different. For example, consider an image of a fabric that has alternate stripes of blue and green. This image will most likely have a histogram similar to a landscape image of a lawn and sky. This is because the histogram does not record the local structure of colors but only the global composition. The color coherence vector [41, 40] is a refinement of the histogram where the colors are split into two components: coherent and incoherent. Coherent component of a color are defined as those color pixels occuring in a neighbourhood of the same color pixels. Now the histogram distance is dependent on both these components. This will differentiate the images in the above example since the coherence vector encodes the composition along with the count. A further refinement on the histogram is the Color Correlogram [18] which records the probability of a color occuring in a specified neighbourhood of another color. As one would expect, each description increases in complexity and requires higher computation both for characterization and retrieval. A commonly adopted approach is to use the histogram first to select a set of candidates, and then refine further with color coherence vectors or correlograms on this smaller set to bound the computational requirements.

2.2.2 Approaches based on Texture, Shape and other features

While histograms are compact and efficient, in some particular domains they are not as useful. For example, in grey scale image retrieval, the illumination histogram is not very discriminating. In specific domains, other features have been explored for retrieval with better performance than using color histograms. The Photobook system [42] was used to query and identify frontal face images in a database of about 6000 images. It computed the eigenvector decomposition for this face space. Then, the face images in the database are represented in terms of projections onto these *eigenfaces*. Distance between two faces is the distance between the projection coefficients. In later variations, Photobook was extended to handle textures, shapes and faces, each with its separate representation. For shape representation, the objects were modeled by building a finite element representation. The stiffness matrix that results from FEM is used to create the eigenvectors in this shape space. This representation is used for query and retrieval of objects. For textured images, Photobook uses a Wold decomposition based representation which decomposes textures along directionality, periodicity and randomness. For each category of images, Photobook creates a semantics-preserving representation. This is a very powerful idea. However, one drawback is that for each class of images, a new set of features have to be extracted that best describe that class. In specific domains, such as face retrieval, this is not a problem and this technique works very well. However, for general images such as the Corel database, a single semantics-preserving representation may not exist.

The FourEyes system [37] extended the Photobook by incorporating a collection of models (they term it the society of models) and learning the pertinent weighting scheme between models for a particular query using relevance feedback. They employed this approach to counter any systematic bias the features might have introduced and at the same time to incorporate any bias that the user might want to introduce for a particular task. One mode of interaction of this system is to learn labelling from a small collection of samples by getting feedback from the user and then propagating this labelling usefully across the entire database. This learning can be viewed as "short term memory" where the learning of a concept lasts a particular session. In contrast, a Hidden Markov or a Bayesian Network approach that we review later can be seen as a mechanism of incorporating "long term memory" where the concepts are learned and used over the life of the system. Human-in-the-loop is a very useful idea, especially because it permits optimization of queries based on the particular task at hand.

In [53], Smith and Chang presented a general framework for spatial and feature based queries. The general approach is to parse the picture into a collection of regions or objects and perform queries that are constrained both spatially and in features. For example, a sunset image on the ocean can be specified as an orange colored circular region spatially located above a rectangular blue region. A similar region-based approach was presented by Carson et. al [5, 13] for querying images of animals and highly textured objects. The region-based approach was motivated in part by the need to move away from low-level descriptors to something closer to human reasoning. It is desirable to extract descriptors from imagery that can enable higher-level reasoning. However, it is unclear if such combined spatial-feature approaches provide that since they require the query to be broken down into a description in terms separable regions. A particular query might have multiple descriptions in terms of these simple regions. For example, there can be many kinds of sunsets over ocean being just one of them. Another problem is that many images from unrelated categories might have similar descriptions in terms of such spatial regions.

For texture image retrieval, Liu and Picard [32] proposed the retrieval of the Brodatz texture images using Wold decomposition for characterization of texture. Each texture image is decomposed into the three axes of periodicity, directionality and randomness. A probability model along these axes is built to characterize images. Ma and Manjunath [35] proposed using Gabor decomposition for texture image representation and retrieval. Gabor filters have compact support both in frequency and time and and can be tuned to different orientations and scales. Statistics over a range of scales and orientation forms a representation of texture and this is used for retrieval of textured images. The authors use a distance function similar to the Mahalanobis distance to compute similarity between images.

Jain and Vailaya [27] use edge histogram as a representation of shape in images. A Canny edge detector provides local orientation information which is used to build an edge histogram. This is used for content-based retrieval of trademark images, where shape is a very relevant feature. In order to account for rotations, they shift the histogram bins and define similarity between two shape histograms as the smallest distance across all shifts. However, even in these images where shape is quite relevant, the authors claim better performance with color histograms. A combined shape and color histogram based approach has the advantages (and the computational complexity) of the two techniques and it results in better performance than either color or shape alone.

2.2.3 Apriori classification to improve performance

Visual features based techniques such as histograms, coherency vectors etc do not account for semantic similarity in the content. In many cases, if the class labels are known apriori, visually better retrieval can be achieved. For example, if the query image belongs to the class of man-made objects, visually better matches can be obtained using color histograms if we restrict the search to the subset of the database containing man-made objects. Szummer and Picard [58] proposed a technique of classification of pictures as Indoor vs Outdoor images. This technique extracts a collection of features from images (color histograms, MSAR and DCT coefficients) and performs a k-nearest neighbor classification to identify images as indoor/outdoor. An incoming query image can then be classified into one of these two categories and then queried on a subset of the database. This process can be repeated to limit the database further and thereby get both computational savings and better results. This approach was employed by Vailaya et al [1] where they hand-labeled a collection of images into a hierarchy. A probability model for each class in the hierarchy is estimated using sample images and Vector Quantization. The query image is sent to the appropriate branch in the hierarchy based on these probability models and then a retrieval based on color histograms and coherency vectors is attempted on the subset. One of the problems with this approach is that the creation of hierarchy and the class labels by hand can be a laborious task. However, this can be countered by performing the labeling on a representative subset and then using an automated procedure to label the rest of the database. More importantly, the assumption of the hierarchy requires that the classes be mutually exclusive and that is not true in general. For example, an image can belong to both the *sunset* and *forest* classes. In reality, each image will more likely belong to many classes and can be associated with multiple labels. A better solution is to attach a certainity for each label for each image. For example, with 80% certainity, the image is a sunset and with 70% certainity, the image is a forest etc.

Liu and Dellaert [34] present a similar classification approach for retrieval of 3D neuroradiological CT images. They exploit the fact that CT images of normal human brains are fairly symmetrical compared to the pathological ones. As opposed to apriori classification, their approach is to use a similarity function that is useful for classification as well. The intuition is that a function that classifies well will probably retrieve similar images. They estimate a probability model over the extracted features and use this to both classify and retrieve images.

2.3 Video event detection and representation

The examples in the previous two sections are primarily still image representation and retrieval systems. These techniques have been extended to handle video sequences with limited success. The field of video retrieval is nascent. Some the pecularities of video that make it harder are – more data to handle, the concept of similarity is harder to specify since there is the added dimension of time.

One of the commonly adopted approaches to video retrieval is to extract significant events from video, such as shots. The intuition behind such a description is that a shot is a single camera stream and therefore can be represented with a still image. Once a video sequence is represented in terms of shots, still image techniques can be employed for query and retrieval. The two additional orthogonal tasks that result from such a decomposition are – video shot detection and key frame selection. The video sequence is then summarized by a collection of key frames with associated features for every key frame.

In addition to content-based retrieval, browsing is an important application for video. The rationale for browsing is that in interactive applications, it is useful to provide a quick summary before committing the extensive network resources required to send the full rate video to the consumer. If a storyboard or a summary representation is sufficient to distinguish between video sequences, then only those video sequences that are desired need to be sent at full rate. In this thesis, we make use of browsing as the interaction paradigm with video. Browsing can be seen as an alternative (even a complement) to query-by-example. Instead of specifying queries in terms of possibly non-intuitive primitives, browsing presents a view into the video database and enables identification and selection of material.

2.3.1 Video shot/scene detection and key frame extraction

Shot detection in a video sequence can be performed either in the compressed representation of video such as MPEG or in the uncompressed form. A shot is often the most easy to extract unit of video information. It represents a single camera viewpoint and therefore color, motion etc are expected to be revealing. In uncompressed domain the popular techniques use color histograms, dominant motion, motion compensated pixel differences to detect shot changes [4]. These shot change detection algorithms exploit the lack of continuity that accompanies a change in camera viewpoint and editing. For example, it is reasonable to expect that the dominant motion in a video shot, which represents the camera's action, changes from shot to shot. Similarly, the colorimetry of the scene changes with camera viewpoint change. Therefore, there is lack of continuity in the color values of a sequence at the shot boundary. Shahraray [51] proposes a shot change detection technique where the two adjacent frames that are compared are divided into rectangular blocks and a motion and intensity values from these blocks are used to determine a match. If two blocks have similar motion and intensity values, a value of 0 is returned and a 1 indicates a severe mismatch. The cumulative similarity (or dissimilarity) is the sum of similarities over these blocks. A shot change is flagged if this dissimilarity rises above a threshold. A gradual transition is similarly flagged when the dissimilarity rises above a smaller threshold and stays above this smaller threshold for a set of frames before falling below the plateau. The main advantage of this technique is that it combines both intensity and motion parameters in evaluating a shot change and therefore this technique is quite robust. Zabih et al [69] propose a technique using edge information to detect shot transitions. The intuition behind this technique is that during cuts and dissolves, new intensity edges appear far from the location of old edges and old edges disappear far from the location of new edges. They use this to classify a set of edge pixels as *incoming* or *outgoing.* Scene cuts are located by looking for local peaks in the maxima between *incoming* and *outgoing* pixels. Prior to computing the edge pixels, the authors compensate for the motion by computing the dominant motion. This is ensure that camera operations such as pan and zooms do not contribute to false shot detections. Shot change detection algorithms in the uncompressed domain can be computationally expensive. Moreover, such changes in color and motion etc can be extracted directly from the compressed representation.

Shot change detection in MPEG compressed video is based on statistics of the DCT coefficients or motion estimation parameters. Some examples are inner product of DCT coefficients of I frames as a measure of frame similarity, number of intra coded P frame blocks versus the number of inter coded blocks, number of forward predicted B frame blocks versus the number of backward predicted blocks, variance of DCT coefficients etc. Some approaches also partially reconstruct the frame from the DC coefficients and use histograms on these frames to perform scene cut detection. For a comprehensive comparison of some of the popular techniques for shot change detection please refer to [15]. Yeo and Liu [65] propose a technique of extracting a DC frame from the compressed stream. This DC frame is created by inverting the 1st DCT coefficient of each block. In a sense, this is equivalent to creating a low pass version of the frame. Since only one coefficient is used, the inverse transform operation can be done in real-time or faster. The authors show that this low pass image is adequate for performing shot change detection. This technique is more robust than relying on the transform parameters completely and yet retains the advantage of the significant computational savings offered by not decoding the entire compressed stream. A key characteristic of all shot change detection algorithms is the assumption about the effect the shot change has on the extracted features. Most of these assumptions are valid for abrupt shot changes but gradual changes such as dissolves are especially hard for all algorithms [15]. Moreover the compressed domain techniques are highly dependent on the specific encoder and the output quality [15].

Once the video events have been detected using a shot change algorithm, the next task is to represent these shots by key frames. These key frames are both used for browsing and for content-based retrieval. For key frame selection, many proposals have been suggested – first frame, last frame, middle frame, closest to average frame, the two furthest frames etc. One of the most sophisticated technique for key-frame extraction is presented in [71] where the authors use unsupervised clustering to group frames in a shot and select frames from these clusters. There is no clear evidence that any of the proposed set of techniques have a significant advantage over the others, once a video sequence has been parsed into a collection of shots.

2.3.2 Browsing and summarization of video

Zhong et al [70] propose a hierarchical clustering technique for video. Representative shots from each level in the hierarchy form the icons at that level. These icons enable browsing into the video content. The input to their system is a contiguous video presentation (such as a news broadcast). They begin by performing a shot detection to extract shots from the video sequence. A representative frame for each shot is chosen and features associated with this frame are used for the hierarchical clustering process. The features they use for clustering are the color histogram, motion orientation histogram and motion mean and variances. They employ a fuzzy K-Means algorithm for clustering. This is an interesting technique for browsing into the video content and related to our work. However, relying on features from a single representative frame may not capture the temporal import of the event. Also, the motivation for the fuzzy cost function that they employ is unclear.

Yeung et al [66, 68, 67] proposed the *Scene Transition Graph (STG)* as a model for organizing and browsing video. The STG is a representation where the component shots of the video are organized into related groups and the transitions between the shots is indicative of the video content. For example, assume that a video sequence is made up of a dialog scene, an action scene followed by another dialog scene. Each of these component scenes is made up of many shots. For example, the dialog scene will typically have two or three camera viewpoints, each representing a shot. A simple parsing of the video sequence into shots does not convey the presence of a dialog. Whereas, if the representative images from each shot is examined, the dialog scene might cluster into two/three groups of shots (one for each camera viewpoint). The authors use this intuition to cluster related shots and create a scene transition graph where each node represents a group of related shots. Examination of this graph makes it more apparent that the underlying event is a dialog/action scene etc and thereby provides a better summary of the video content. This works well for structured content. For unstructured video, the premise on which the STG rests is invalid and therefore cannot be applied. Interestingly, the authors also argue against using a single representative frame per shot [66] as it loses key temporal structure and can potentially classify related shots as different because of bad choice of the key frame. The authors, instead, use multiple key frames selected on the basis of the amount of activity within the shot. A shot with more action will have more key frames to represent it. This is a mechanism to compromise between the computational complexity of using the full rate video and the lack of information in a single key frame.

Pope et al [45] describe a scheme for summarizing video content using mosaics. The mosaic is a static snapshot built from aggregating the frames in a video shot. Moving objects are then placed into the mosaic to create an alternative video representation. In their scheme, the incoming video is segmented into shots and the static content of each shot is used to create the mosaic. The dynamic content is segmented, tracked and overlaid onto this mosaic to provide a mechanism for semantic event detection (such as collisions, chases, suspicious activity). This approach is targetted towards aerial surveillance and reconnaissance missions. The mosaicing process works well for relatively static shots with significant overlaps to ensure quality registration. In addition, since sensor inputs (such as GPS co-ordinates) are frequently available, an orthorectified mosaic using a physical model of the region surveyed is also used. Once the images comprising a shot are registered and compensated for camera motion, pixels of significant coherent residual motion are marked as dynamic objects. Each such contiguous region of moving pixels is modeled by it size, velocity, overall shape and color in order to track it over multiple frames. A typical representation of each object is extracted from these model parameters and overlaid into the mosaic. This technique of tracking objects relies on the fact that they occupy a relatively small fraction of the overall size of the frame.

2.3.3 Content-based video retrieval

Content-based retrieval of video, as we mentioned earlier, is often turned into the simpler problem of image retrieval on extracted key frames. The basic idea is to extract a collection of shots from a video sequence, represent each shot by a key frame. Once the key frames are selected, a set of features for each key frame are extracted and used for queries and retrieval.

Dimitrova and Abdel-Mottaleb [11] present a technique for extraction of signatures from MPEG and MJPEG compressed video sequences. These signatures are extracted from the DC coefficients and motion information of representative blocks in a video frame. The authors extract signatures from video clips (a clip is a small sequence, possibly made up of a few shots) using two techniques: a signature for every I frame in the clip or a signature for every key frame in the clip. The I-frame only signatures tend to be longer because there can be potentially many more I-frames than shots in a video clip. A set of window pairs are selected apriori. These window pairs are arbitrarily chosen macro blocks within the frame. For each window pair, the DC coefficients and the block motion vector form the signature. The frame signature is the concatenation of the set of window pair signatures. Similarity between two video clips is expressed in terms of similarity between these signatures. A major advantage with this representation is that the signature is fast to compute (since no decompression is required) and search. However, the signature seems highly dependent on the MPEG encoding parameters used. For example, the same video sequence which has been encoded using two different MPEG parameters will not necessarily result in signatures that are proximal. In contrast, a color histogram might be more revealing in this case. Because of its computational simplicity, it might be useful to use this technique as a filter to select candidate matches and then refine the smaller subset further with a more expensive approach.

Vailaya et al [61] propose a similar approach for query by video clip (once again a clip is defined as a small video sequence). They use both a key frame based approach and a uniform subsampling approach for content-based retrieval. In the key frame based approach, a set of key frames are extracted from the query clip using shot change detection. In the uniform subsampling approach, the query and the database are temporally subsampled identically. Both the clip and the database are represented by the set of these subsampled frames. For the purposes of further discussion, these subsampled frames can be called "key frames". For each key frame the color, texture and motion associated with it are used to compute similarity between the the query key frame and the database key frames. For color, euclidean distance between histograms is used for similarity. For texture, the euclidean distance between the edge direction histograms (edges computed using a Canny edge detector) is used. For motion, the euclidean distance between the optical flow histograms is used as the similarity measure.

Similar video clips are the ones whose consecutive key frames are similar to the key frames of the query clip. In the case where the query and the database sample have the same number of key frames, the dissimilarity between the two is the cumulative dissimilarity between the features of the key frames. Because the query clip and the database clips are likely to have different number of key frames, they define a similarity measure that depends on the closest key frames in the query and sample and also on the difference in the number of key frames between them.

The general problem with key-frame based techniques for video retrieval is that such techniques lose important information that is present in the evolution of video. For example, in the above key-frame based approach, two clips which have similar key frames but in different temporal order will be classified as being similar. This may not be desirable since in structured content sources such as movies, the information is revealed in a particular temporal order (chosen by the director/editor) and usually semantic meanings are attached to the order in which information is revealed. Also, because of the particular algorithm used to select key frames, two key frames from very similar shots can look visually very different. This can be fairly common when there is significant camera movement within a shot. The sub-sampling approach or the I-frame approach (which is a particular case of sub-sampling) do not suffer from these problems with key frames. Interesting classifications of video that rely on their temporal evolutions cannot be made in a static representation. For example, we have presented some work [23] (presented in detail in Chapter 3) where we use Hidden Markov Models to classify video sequences. These models rely purely on the temporal characteristics of video and make useful inferences. In [24], such a temporal representation is coupled with a features based representation of video for improved retrieval performance. Deng and Manjunath [10] present retrieval work on MPEG video streams. Once this stream is parsed into video shots, color and texture features are averaged over all the I frames that make up the shot. For motion, the P and B frames are used to to extract motion information. In this representation, data from all the frames in a shot is used to characterize the shot, as in [20]. Our techniques will be presented in detail in Chapter 3.

In an approach to extract interesting temporal structures from video, Liu and Picard [33] proposed using Wold models for characterizing periodic movements. Just as in the case of spatial texture, temporal movements is quantified similarly in terms of their Wold components. This approach can be used to characterize and identify related temporal evolutions.

2.4 Bayesian Networks for Content Characterization

Hidden Markov Models, which are a special class of Bayesian Networks, have been used successfully in speech analysis [46, 47] and in gesture recognition [56]. The usage of Bayesian

Networks for image and video content characterization is a relatively new topic of research. Most current work in image content filtering, browsing and retrieval rely on low-level descriptors such as color, texture and motion that can be unintuitive to most users. In this framework, the goal is to provide an alternative that exploits the structured nature of content sources to achieve a semantically meaningful characterization, and a useful user interaction.

The fundamental assumption in Bayesian content characterization is that the bulk of the content that one would care to store in, or retrieve from video databases exhibits a significant amount of structure which can be exploited. This assumption is certainly valid in biological and other evolutionary systems and been talked about in terms of *organizing principles*, *natural modes*, and *non-accidental properties.* For traditional media, such as movies and structured news/sports programs, struture is very commonly used since it provides a standard framework in which the content producers and content consumers can share information. For example, Hollywood producers typically show action movies and suspense with specialized editing techniques. For example, an easy way to embody action is to pace the video with short cuts from different shots and rely on the content consumers' cognitive ability to bridge the gap between the shots and build the narrative[2]. Similarly, character oriented movies typically use close-ups and dialogue scenes since they are effective conveyors of the actors' emotions. These fairly common principles can be effectively exploited in the design of content-characterizers. For example, it is reasonable to expect that an action movie will have a very different cut pattern compared to a character movie. An action movie will consist of more shots with a shorter time interval between shots. In addition, the activity within each shot will also be high on an average. In contrast, a character movie will tend to have longer duration shots and lesser activity within each shot. A computational model such as a Bayesian Network or a Hidden Markov Model can be trained on these features and be used for classification. We present an approach for this in Chapter 3.

[2]Alfred Hitchcock's *Psycho* shower murder scene is a paramount example of montage editing

Bayesian Inference can then be posed as a problem in sensor fusion where a set of targeted sensors gather observations on relevant patterns and an architecture combines multiple sensor inputs to provide a useful inference. For an excellent article on content characterization using Bayesian Networks see Ref. [62], where the authors present a Bayesian network that use a motion energy sensor, a skin-tone detector and a texture energy sensor to characterize a video shot. The types of labels associated with the shot are – close ups, crowd, natural vs man-made setting, action vs character oriented. The rationale behind the classification from the sensor outputs are as follows: For action oriented shots, the motion energy will be high based on our expectation of typical stuctures present in the movies. The texture sensor gives a high value whenever it finds significant horizontal or vertical structures which fits well with typical man-made structures. Similarly, if a significant portion of the shot contains skin tone, it is either a closeup or a crowd scene. However, a crowd scene would also have a high textured output. So, the network by integrating these multiple sensors achieves a content characterization that is more semantically meaningful. We note here that this characterization can also be used to perform queries as in the query-by-example paradigm. In Chapter 3, we will study two Hidden Markov Models that make structural assumptions about video and use them for useful inference.

Chapter 3

Video Content Characterization

This work is divided into three parts: We first examine a query-by-example system for video where each sequence is divided into a collection of spatio-temporal chunks. Each such chunk is represented by a feature vector derived from the colorimetry, motion and texture of the chunk. We then examine a template based approach for categorizing video. This uses Genetic programming to evolve rules for the chunk representation and can be viewed as an exercise in supervised classification. Finally, we examine two Hidden Markov Models that exploit the editing artifacts present in video to categorize them. This work is then extended to be incorporated with the chunk approach for a semantically controlled video retrieval.

As opposed to the image retrieval examples presented in the previous chapter, all the techniques presented in this chapter are used for content-based retrieval of video sequences. A key difference between these approaches and some of the video retrieval approaches presented earlier is that these feature representations use the temporal aspects of video. See Refs. [64] and [34] for other temporal representations of video. The Hidden Markov models and the probabilistic modeling of shot statistics rely primarily on the temporal dynamics of video for characterization. We believe that for video characterization, much can be accomplished by analyzing the temporal dynamics. Content producers (such as Hollywood directors and editors) structure a narration in terms of *montage* and *mise-en-scene*. Roughly speaking, montage relates to the temporal juxtaposition of scenes and mise-en-scene refers to the spatial arrangements. Features such as colorimetry capture the mise-en-scene better and features targetting the temporal dynamics capture the notions of montage better. Extracting a semantic description from colorimetry and shapes of objects in the scene is a hard problem. Comparatively, extracting temporal events is a simpler task and we can make useful *higher*-level inferences from the temporal statistics.

3.1 VideoBook: An experiment in characterization of video

This experiment was motivated by the need for a representation of video in which time was an integral part. In addition, image based techniques can be computationally expensive if directly applied to video. Using key frames to reduce the computational loads sacrifices the temporal information. By combining spatial and temporal representations, we hope to capture aspects of both mise-en-scene and montage.

3.1.1 Feature extraction and representation

VideoBook [20] is a query-by-example system which has a compact feature representation of video sequences. This enables fast searching over large collections. In VideoBook, every sequence is split into spatio-temporal chunks as shown in Figure 3.1.

The advantage that we get using a chunk representation of video as opposed to global histograms of features is that it preserves the spatial and temporal relationships between the individual features, much like the color correlogram or the color coherency vectors [41, 40, 18]. In addition, the number of chunks and number of features per chunk is very small compared to the number of pixels in a video frame. This enables fast query and

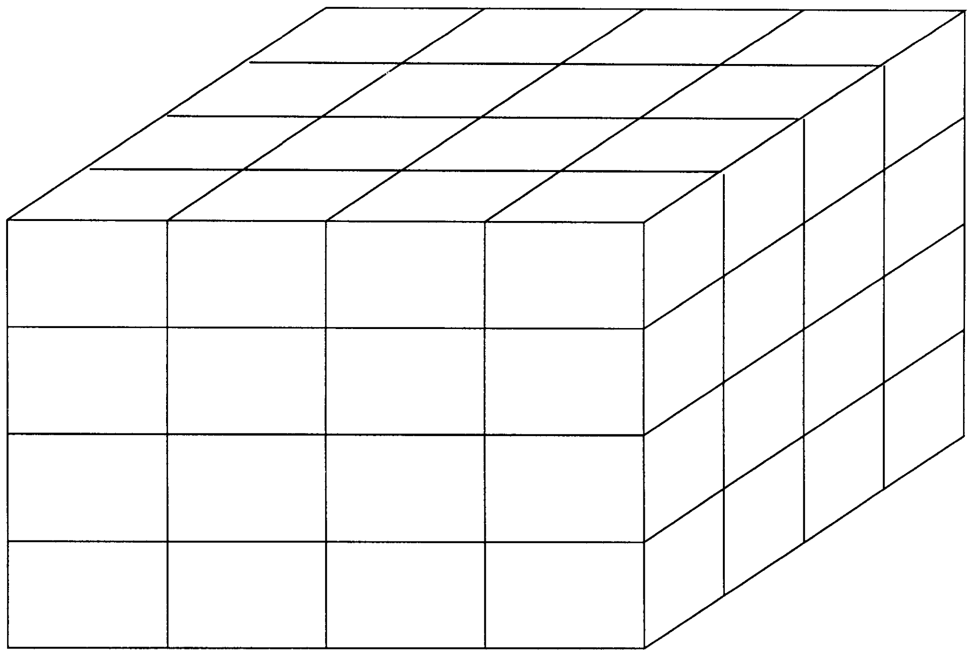


Figure 3.1: Spatio-temporal chunk representation of video

retrieval for interactive applications. For each chunk a number of features are extracted from motion, color and texture of these chunks. In the pattern recognition literature, the correlation function is a frequently used quantity to measure dependence. It is now better understood that while the correlation function captures linear dependence, mutual information is a general dependence measure [30]. In the VideoBook, we use motion, texture and color along with their respective mutual information functions as the feature vector as opposed to using the correlation function. This results in representative feature vectors that work quite well for video retrieval.

Specifically, in VideoBook the chunk dimensions are $64 \times 64 \times 16$ pixels in $x, y \& t$ respectively. 8 features are extracted from each chunk – 3 for color, 3 for motion and 2 for texture respectively. For color, we represent each chunk by the weighted averages of the luminance (Y) and the two color differences (Cb and Cr). Along motion, weighted averages of the optical flow vectors and the mutual information of flow vectors over this chunk are extracted. Texture energy is estimated using a local Laplacian operator. This Laplacian measure along with its mutual information averaged over the extent of the chunk form the next 2 features. A 320×240 pixel video sequence at 30 frames per second is therefore represented by a $5 \times 4 \times 1$ array of spatio-temporal blocks. Thus 160 features are used to represent $1,228,800$ pixels, which provides a compact representation for fast query and retrieval.

3.1.2 Query and Retrieval Experiments

We digitized 165 video clips, each 15 seconds long from television and laser disc movies. These clips form the test database for our experiments.

Once the entire database is characterized along these 8 features, searching and querying experiments can then be performed using these feature vectors. Searching and querying are done in real-time because of the extremely compact representation space. In the present version of the VideoBook, the search mechanism is euclidean and all parameters are uniformly weighted. We note here that this a basic and simplistic search mechanism. One of the reasons we implement this search mechanism is to illustrate the usefulness of the feature vectors. This search mechanism also ensures real-time performance using software only methods. The distance criterion we use is Mean Square Error (MSE) between the feature vectors. The clip with the least MSE distance is the best matching clip to the search cue. We now detail the different experiments that were performed using these feature vectors to characterize the video and using MSE as the distance metric.

In the first set of experiments, we split each digitized video sequence into two parts: The first 10 seconds and the last 5 seconds. We thus have two groups of video clips, each consisting of 165 clips. We randomly pick a clip from the second group (last 5 seconds) and use it to query amongst the first group. The distance metric used to determine the closest match is the MSE as explained earlier. If the feature vectors capture any higher level structure about video, we would expect the search cue (which was extracted from the last 5 seconds) to match with the counterpart (first 10 seconds) in the exact same sequence from which it was extracted. The similarity between two clips of different lengths (the search cue is 5 seconds long and the database contains 10 second long clips) is determined by sliding the cue feature vectors over the database feature vectors and returning the smallest distance between them. This is as shown below.

$$D(s,d) = \min_{k \in [0,n-l]} \sum_{i=1}^{l} [s(i) - d(k+i)]^2 \tag{3.1}$$

where s is the search cue, d is the database sample, n is the length of the database sample, l the length of the cue. We compute the mean square error between feature vectors at the specified indicies. In addition to returning the matching clip, this process also identifies the time index within which the cue matches the database clip the best.

We performed 50 random trials and in 46 of them the closest match was the corresponding part from the same sequence. In two trials where the search cue matched with another sequence, the other sequence belonged to the same content source. Therefore, we can conclude that the feature vectors that we used for characterizing video are useful for fast content based retrieval. In addition to giving the closest match, the VideoBook returns the next two closest matches to the search cue. Here we find more evidence that the feature vectors perform very robustly. In one of the trials performed, the system was given a news clip as the search cue. A key frame from that sequence is shown in Fig. 3.2. The VideoBook identifies correctly the closest match as belonging to the same video sequence. Also, the next two matches were also news sequences. The key frames from the closest two matches are shown in Figs. 3.3 - 3.4.

In the second set of experiments, the database is made up of the 15 second clips. We randomly pick a search cue as a 5 second subset from these clips. We performed 50 such trials. The system searches the entire database and returns the three closest matches as before. Since there will always be an exact match in this case, the system rejects the match if the distance value is 0. Therefore, the three returned matches represent are guaranteed to be different from the search cue. We find that the first match is always another segment from the same video sequence from which the search cue was derived. Moreover, the next

Figure 3.2: Key frame from the last five seconds of the Television news sequence

best matches are visually similar. Fig 3.5 shows a query from a sports broadcast and Fig 3.6 shows the next closest match.

We have noticed instances where the VideoBook goes into orbits[1], with the user ending up in a collection of the same few video sequences over and over. However, this is not a problem. It is entirely possible that some video sequences form a closely clustered group in the 8-dimensional representation space. In reality, such a grouping is desired for it implies that similar content forms a cluster in this parameter space. Formation of such clusters implies that we may be able to perform higher level searches (such as querying using natural language as opposed to using image sequences) for these clusters. In the next section we exploit existence of such orbits and derive templates for characterizing video.

[1]in the non-group theory sense

Figure 3.3: Key frame from the closest match for the above query sequence

3.2 Evolving discriminators for video sequences

In order to exploit these natural groupings in the feature space, we use genetic programming (GP) to form templates. The primary motivation for using GP is that it is a useful technique for automatically extracting rules from a collection of related observations.

Genetic Programming is an evolutionary technique belonging to the general class of Genetic Algorithms (GA). GAs are highly parallel mathematical algorithms that transform populations of individual mathematical objects (typically represented as fixed length character strings) into new populations using natural genetic operations such as sexual re-combination (crossover) and fitness proportionate recreation. GAs start with randomly selected initial population and evolve new members using these various genetic operations to iteratively produce better fit populations. They can thus be viewed as an optimization procedure like K-means. Representation is a key issue in Genetic Algorithms since the algorithms directly manipulate these coded representations. Moreover, fixed length character strings are not particularly suited for a variety of problems. Genetic Programming [28] strives to mitigate

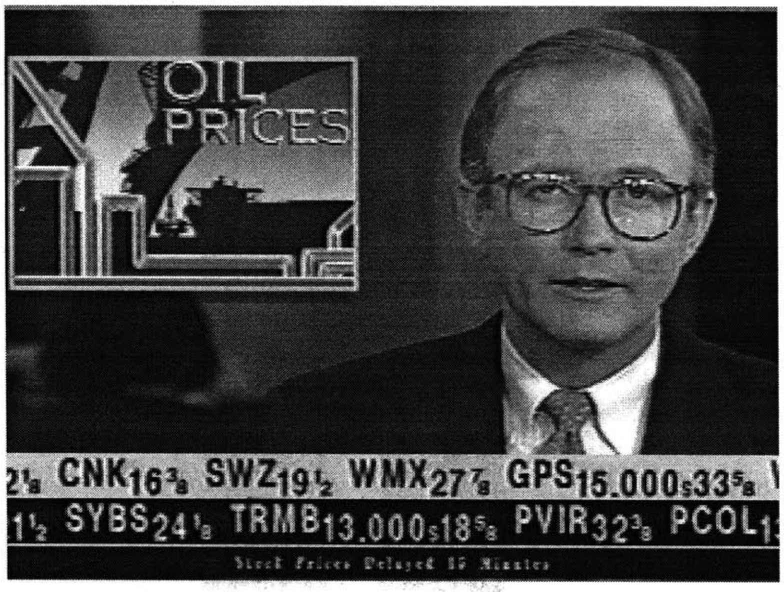

OIL
PRICES
CNK 16 3/8 SWZ 19 1/2 WMX 27 7/8 GPS 15.000s33 5/8
SYBS 24 1/8 TRMB 13.000s18 5/8 PVIR 32 3/8 PCOL
Stock Prices Delayed 15 Minutes

Figure 3.4: Key frame from the next closest matching sequence

this problem by offering greater flexibility of representation.

In GP, the main notion is that of an expression comprised of a set of *operators* and *terminals*. For example, {+,-,*} constitute operators and combined with $\{X, Y\}$ form expressions of the kind $(+(*XY)(-Y1))$ in postfix notation. Such a notation can also be viewed as a binary tree. Crossover operation can be defined as a swapping of two randomly chosen subtrees of two expressions. Similarly, mutation can be defined as an operation where either a terminal or an operator in a random subtree of an expression is replaced with another valid terminal or operator, resulting in another valid expression. Figs. 3.7, 3.8 and 3.9 illustrate the tree representation and these genetic operators. The expression represented by Fig 3.7 is

$$((A5 \times A1) - A3) + ((A4 + A1) * (A3 + A2)) \tag{3.2}$$

These reproductive operators, combined with a fitness function form the basic primitives of a GP simulation. Important parameters are crossover rates, mutation rates, population size. The form of the fitness function determines the problem that the GP is attempting

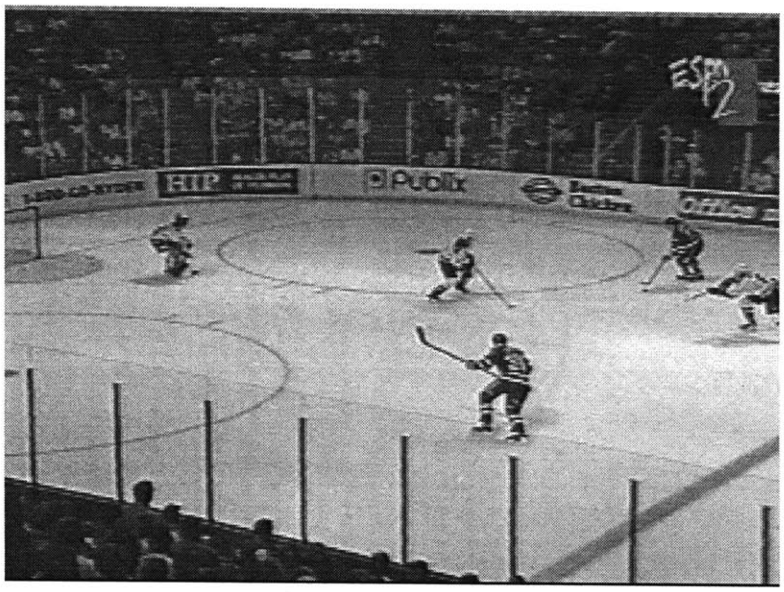


Figure 3.5: Key frame from the sports query sequence

to address. Another important issue is that of type closure which we will explain when we discuss the specifics of the evolution that we attempted. While GP is a powerful technique, its application to pattern recognition problems is not yet widespread. This is because formulating a pattern recognition problem in a GP framework is not always simple. We now examine an approach to formulate the supervised learning problem in a GP framework.

3.2.1 Evolution of VideoBook discriminators

Using GP, we attempt to build discriminators that distinguish video sequences into two broad categories: *News* and *Action*. News represents a head and shoulder sequence of a newscaster with occasional mixing of other shots. Action scenes comprise of sequences from car chases, horse chases etc.

The incoming video sequence is first parameterized as detailed in section 3.1. The *terminal* set is therefore the space of these features. If we treat each feature of the VideoBook

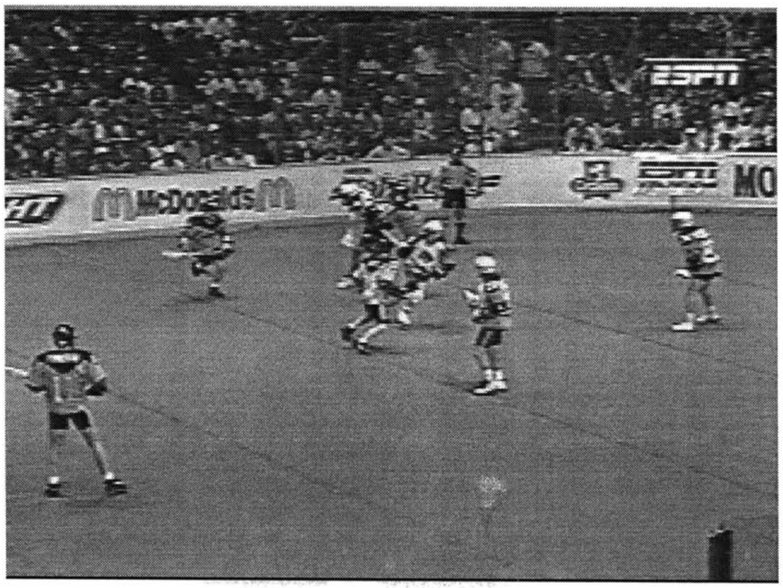


Figure 3.6: Key frame from the next closest matching sports sequence

representation as a variable, we have 160 different variables for every segment of a video sequence since there are 20 chunks per segment of video and 8 features are extracted per chunk. Hence, we have 161 different terminals, with the one special terminal randomly set to a number between 0.0 and 1.0. We normalize the feature vectors to lie in the range between 0.0 and 1.0. We choose the following operators to evolve a decision tree to help us build templates for various classes of video. They "guarantee" the space of all Boolean functions and also all real valued polynomial functions. By guarantee, we mean that the space of all these functions is achievable by combinations of these terminals and operators.

Operator set: $= >, <, AND, OR, +, *, -, NOT$

The operator set includes both Boolean and Real operators. This presents a problem known as type closure where we have to ensure that these two inherently different data types are combined in a logical and mathematically tractable manner. Type closure can be implemented in many ways [28]; we choose to implement it by requiring the boolean operators to return either a 0.0 or a 1.0. When the boolean operators are presented with

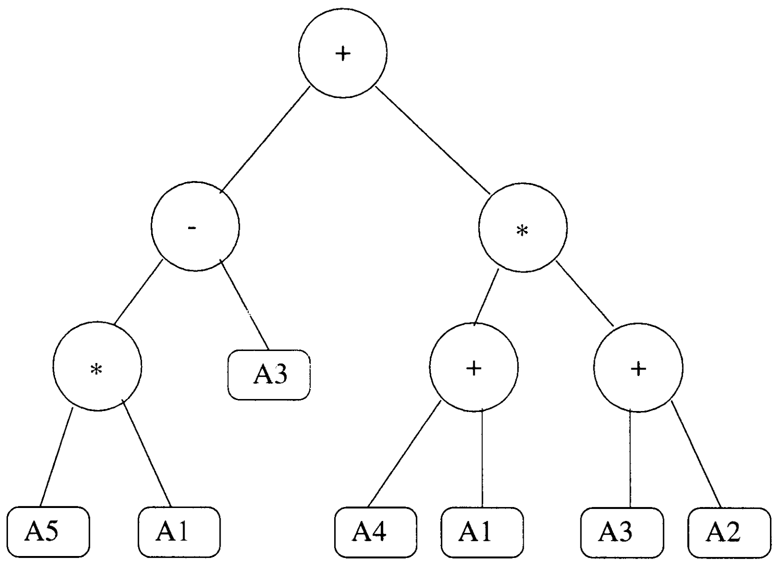


Figure 3.7: Tree representation of an expression

real valued inputs, their behavior is as follows. The AND operator becomes a modified MIN operator. It takes the minimum of the two real valued inputs and returns a 1.0 if this minimum value is greater than 1.0 else it returns a 0.0. The OR operator is a similarly modified MAX operator. This ensures that for boolean input values they work correctly and for non-boolean values, they act as gated MIN or MAX operators. Other ways of implementing type closure include changing all data types to a common third data type (e.g. integers) or replacing boolean units with neural network type sigmoidal activation functions. An interesting technique for type-closure would be to permit only "correct" expressions. That is, the simulation can be made to permit only those expressions that make algebraic sense without permitting loose interpretations of the boolean operators. It is not immediately apparent if such an approach has clear benefits given that the purpose of genetic programming is to evolve useful expressions. If a loose boolean expression is useful as demonstrated by its fitness over the test population, the argument that expressions must conform to a strict algebraic grammar becomes weak.

For the fitness function, we have a special requirement. We are evolving two class discrim-

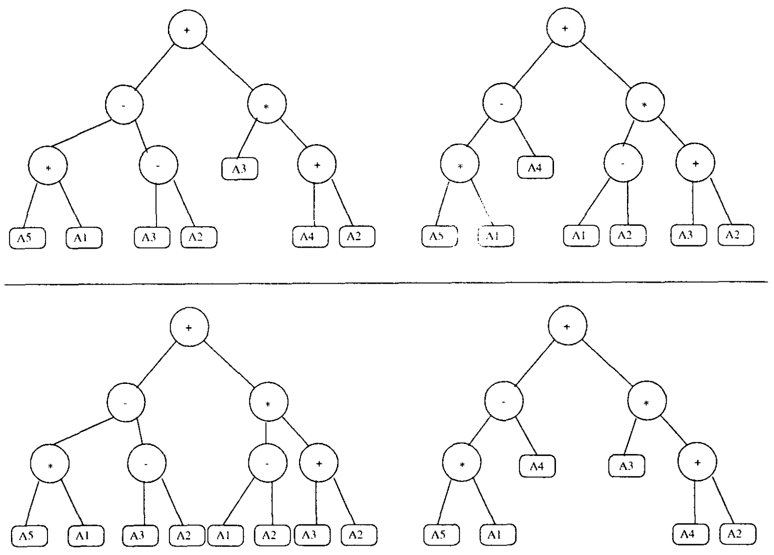


Figure 3.8: Example of a crossover operation. Two subtrees are exchanged

inators: hence, the evolved functions must be good for class A *and* bad for class B and vice versa. We thus have our fitness score as a sum of two fitness scores, *fitness for* and *fitness against*. Each instance of a correct decision by the discriminator over the training set receives a score of 1. In this paper, we use 200 training samples (100 examples of news sequences and 100 examples of action sequences). We can therefore view this evolution of discriminators as performing supervised learning.

Selection of the next generation can be performed in multiple ways. Perhaps the simplest method of selecting the next generation of the population in the GP iteration is to perform fitness weighted selection. This is akin to spinning a roulette wheel where the sector of the wheel is distributed according to the fitness of each member. Thus, a member with a higher fitness occupies a larger sector and hence is more probable in being selected in the next generation. However, this simple method has the following problem: If in one generation, one of the members dominates in fitness, this member overshadows the next generation and the simulation has the risk of being caught in a local minima. To counter this effect, we implement a tournament selection procedure where we pick two members from the population at random and select the member with higher fitness amongst the two as the

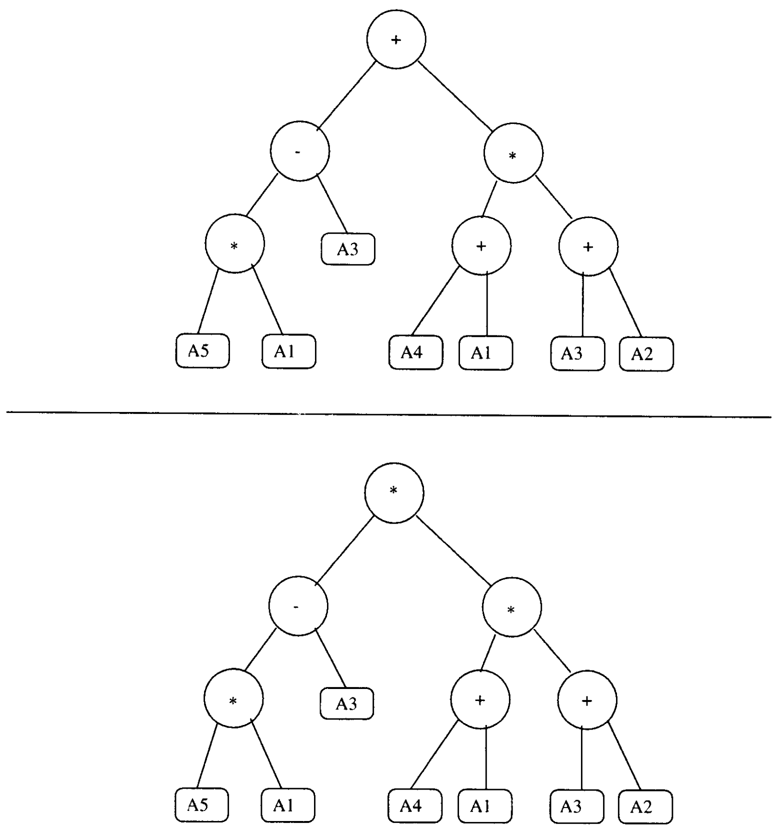


Figure 3.9: Example of a mutation operation. The top operator is changed

winner. This is repeated till the entire new population is selected. The stopping criterion for the GP simulation is when an entire generation achieves a target average fitness. The chosen discriminator is the individual with the maximum fitness in the terminating iteration. An alternative choice would be the individual with the best performance over the held out data. Such a strategy would be akin to cross-validation. In our implementation, we adopt the simpler technique of selecting the best individual at the terminating iteration. The plot of average fitness of the test population over time is shown in Figure 3.10.

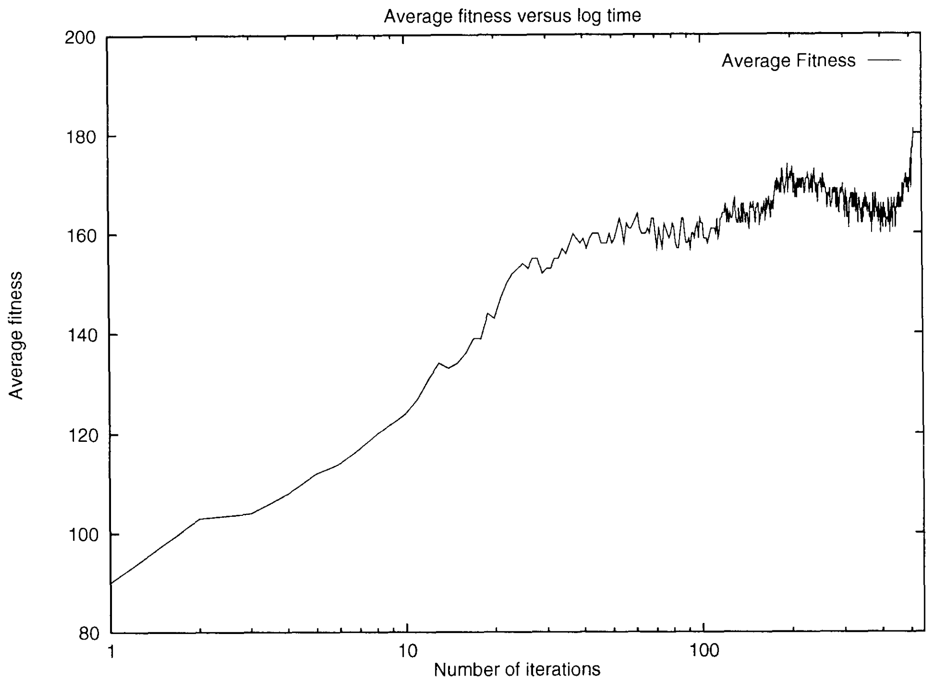


Figure 3.10: Plot of average fitness versus log time (number of iterations)

3.2.2 Performance of the evolved discriminators

With the two discriminators evolved as specified in the previous section, we tested them together on 100 video sequences (50 News sequences and 50 Action sequences). We note here that all these 100 samples were from out of the training set used to evolve the template expressions using GP. The classification was considered correct if both the discriminators gave the correct answer. If only one discriminator gave a positive response and the other discriminator gave a negative response, it was considered a false reject. The performance of evolved discriminators is shown below in Table 3.1. This is a lower bound on the performance of the discriminators, since there are cases where one of the discriminators would give the correct answer and the other discriminator's answer is unreliable. However, since the discriminators do not give out probabilities but just a classification, we are not able to exploit this. This is a limitation of our technique.

Test class	Number of sequences	Positive	False Reject	False Alarm
News (train)	100	97	3	0
Action (train)	100	98	2	0
News (test)	50	47	3	0
Action (test)	50	46	4	0

Table 3.1: Performance of the evolved discriminators

The first two rows indicate the performance on the training set. Test set results follow in the next two rows of the table. It is interesting to note that both the discriminators performed well and that the false alarm rate where the *News* sequence was classified as *Action* or vice versa is zero.

We note some problems with GP. The decision trees that it produces tend to be redundant. This problem can be mitigated by adding a penalty term for redundant and deep trees, akin to the Minimum Description Length principle. Another problem we did not address is that of novelty detection. That is, if a video sequence neither belongs to *News* nor to *Action* but to a third class, the behavior of these two discriminators is not clear. Ideally, we want the class of discriminators to identify not only the correct class to which examples belong but also instances where examples belong to none of the known classes. In the particular way we posed this problem, it is unclear if Genetic Programming based techniques have the capability to perform novelty detection. Probabilistic approaches such as Bayesian Networks and Hidden Markov Models have this additional advantage of being able to perform novelty detection. Content characterization examples using Hidden Markov Models are presented in the next section.

3.3 HMMs for characterizing video sequences

Hidden Markov Models are types of Bayesian Networks that have been used rather successfully in speech analysis applications [46, 47] and in gesture recognition [56]. These models attempt to learn the underlying probability density function that was responsible for creating a particular sample. The task of classification now becomes one of Bayesian Inference where the model that explains the occurence of a particular sample the *best* wins. The definition of *best* is usually either maximum likelihood or maximum aposteriori probability.

Hidden Markov Models are particularly useful for modelling temporal events such as gestures, speech etc. To our knowledge, using Hidden Markov Models for content-based retrieval of video has not been attempted prior to our work [23]. Sahouria and Zakhor [49] use Hidden Markov Models for classification of sports video sequences. Both these approaches exploit the temporal nature of video to make interesting inferences. In contrast with query-by-example approaches for image and video retrieval, both these approaches rely on the temporal evolution of video rather than its visual characteristics to classify. In our work, we design two Hidden Markov Models for classification of television programs and full length movie trailers.

3.3.1 Television Sports/News classifier

We begin by presenting models for classifying TV sequences as News or Sports. The next section (Section 3.3.2) presents models for movie trailer classification. Combining many such approaches, we can build a language for automatic parsing of video sequences where many such small, specialized classifiers are connected together in a Bayesian framework.

There are two interesting problems that make video characterization unique. The first one is that unlike spoken word classification for example, there is much more variation among classes in video sequences and the class boundaries are not very clear. For example, if you see a small sports clip as a part of a television newscast, is it considered news or is it a sporting event? The second problem is that of dimensionality and appropriate feature selection. Video can be extremely voluminous and it imposes severe restrictions on the sizes of the test and training sets.

In order to deal with these two problems, we made the following assumption: We assumed that TV news usually has relatively static shots of one or two people talking interspersed with bursts of activity such as a field shot, some action detail where the camera is usually mobile. Similarly, the sports sequence was assumed to be high action interspersed with relatively quiet shots of the coach, a player or the scoreboard etc. With this assumption, using optical flow or some similar motion information seems to be a likely candidate for features. We initially chose optical flow as the input to our models. Later experiments with simple frame differences also achieved similar performance as optical flow (see section 3.3.1). We digitized 50 random television broadcasts, with 25 Sports events (mostly soccer, football and basketball) and 25 Newscasts. These broadcasts were mainly derived from network news (ABS, NBC, CBS) programs and specific sports stations such as ESPN. These sequences were split into training and test sets of 24 and 26 sequences each, respectively.

Each digitized sequence is between 5 and 15 seconds long and at 15 frames/second. Based on our assumptions, we expect to see two types of flow fields: High energy and low energy fields. Thus, the sequence classification problem is turned into a time-series analysis problem with the feature being the flow energy. In a typical news sequence, we expect to see mainly low energy flow fields with occasional high energy fields whenever there is a camera cut to a field shot. Similarly, we expect the sports sequences to be the other way around with respect to these flow fields.

2-class feature extraction

Since this is a two class discrimination task, we do not need the entire optical flow field but rather can work with a projection of this optical flow information into a single dimension. From the training set of 12 News sequences and 12 Sports sequences, we isolated 24 single frames (template frames) and used the optical flow information at these frames to compute an optimal projection. The procedure for computing the projection is outlined next. For futher details on this two-class feature selection, see Ref. [14].

Consider two correlation matrices R_1 and R_2, one for each class.

$$R_i = E[(x - m_i) \times (x - m_i)'] \tag{3.3}$$

where x is a vector belonging to class ω_i and m_i is the mean of class ω_i. Let Q be a matrix such that $Q = R_1 + R_2$ with its eigenvalues λ_k and eigenvectors, v_k. Let S be a linear transformation such that $S^T Q S = I$. We know that one such S is

$$S = \begin{bmatrix} \uparrow & \uparrow & & \uparrow \\ v_1 & v_2 & \cdots & v_n \\ \downarrow & \downarrow & & \downarrow \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{\lambda_1}} & & & 0 \\ & \frac{1}{\sqrt{\lambda_2}} & & \\ & & \ddots & \\ 0 & & & \frac{1}{\sqrt{\lambda_n}} \end{bmatrix} \tag{3.4}$$

The transformation S does not diagonalize either R_1 or R_2. It however transforms the input x to $x' = S^T x$ and the new correlation matrices are given by $R_1' = S^T R_1 S$ and $R_2' = S^T R_2 S$.

We know that $R_1' + R_2' = I$ by the construction of S. Therefore $R_2' e = (I - R_1')e$ for any vector e. Now, if e were an eigenvector of R_1', then we see that e is also an eigenvector of R_2'. Moreover, if λ is an eigenvalue of R_1', then $(1 - \lambda)$ is an eigenvalue of R_2'. Since the correlation matrices R_1' and R_2' are both positive semidefinite, the eigenvalues must be non-negative. As a result, the largest eigenvalue of R_1' is the smallest eigenvalue of R_2' and vice versa. So, the eigenvectors e that are best for representation of class 1 are the worst for class 2. We can now choose a subset of these eigenvectors e and define a new transformation of the original data x such that

$$y = Tx = \begin{bmatrix} \leftarrow & e_{j1}^T & \rightarrow \\ \leftarrow & e_{j2}^T & \rightarrow \\ & \vdots & \\ \leftarrow & e_{jm}^T & \rightarrow \end{bmatrix} S^T x \tag{3.5}$$

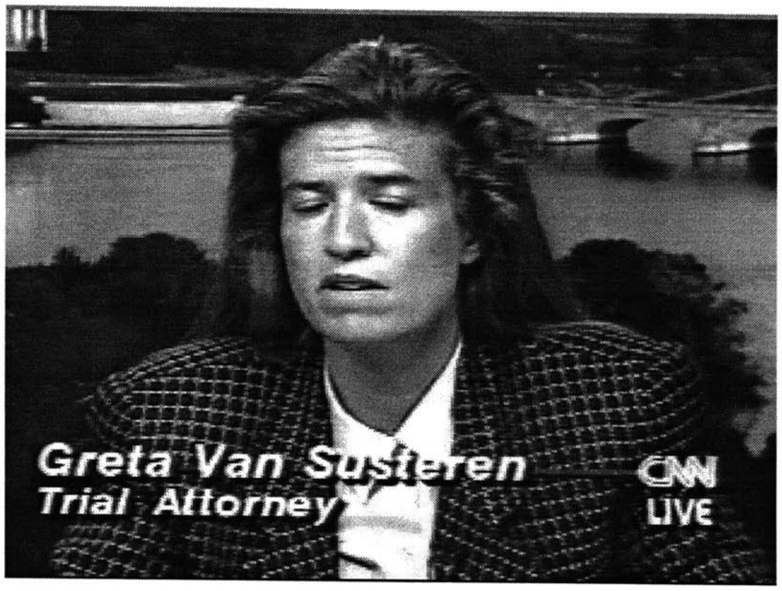


Figure 3.11: One of the 12 Low Action training templates

As part of the training, we used the 24 static frames and computed a projection vector (by selecting e_{j1}, corresponding to the largest eigenvalue of $R_1^{'}$). The input to our HMM models is this projected data. The task that the HMMs have to perform is to analyze a sequence of projection data and compute the likelihood for both the classes. These likelihoods are then used for discrimination between classes.

Figure 3.11 shows one of the "Low Action" templates and Figure 3.12 shows one of the "High Action" templates. Figure 3.13 shows the histogram for the projections for the 24 template frames. These projections are representative projections for fast and slow action frames. We note here that we cannot use the projections directly to decide the classification of the incoming video. News sequences can (and do) contain fast action as well as slow action frames. Similarly a sports sequence will contain both kinds of action. Hence, using just this projection will result in a classification that may change from frame to frame which is clearly incorrect. Moreover, since there is significant overlap between the two types of templates, a direct classification will (and does) have a very low accuracy.

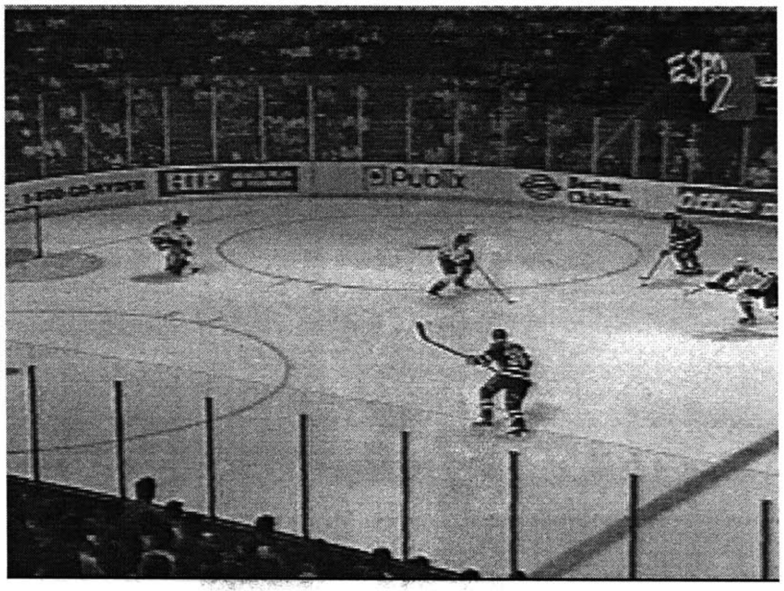


Figure 3.12: One of the 12 High Action training templates

We then trained two HMMs, one on each of the 12 training sequences to build the two HMM models. The next section discusses the experiments and the results obtained.

Sports/News classifier experiments

As mentioned earlier, we trained the 2 HMM models on the 12 video sequences each. The input sequence was the projection values of the optical flow of each frame in the training set. We implemented the HMM training algorithm as mentioned in Rabiner and Juang [47], incorporating scaling to prevent numerical underflow. In addition, we incorporated simple checks to prevent the state transition (A) and the state probability (B) matrices from having 0 values. This is partly because of the small size of the training set. This is in effect stating that there are no illegal transitions but only highly unlikely ones in our HMM model. The HMMs were fully connected with two states each, reflecting our belief that these sequences have two "modes" – low action and high action and spend varying times in these modes. In addition, we perform maximum likelihood classification by associating

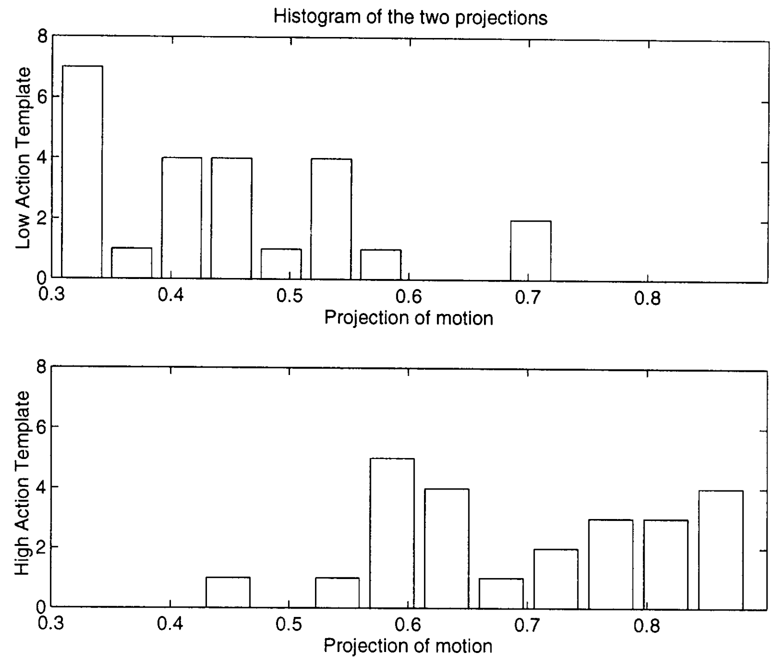


Figure 3.13: Histograms of the two projections for the 24 training template frames

equal priors to both the models.

Once the models are trained, for a given test sequence, the models compute the likelihood for that test sequence given that particular model. Once these likelihoods are computed, the sequence is classified per the model with the higher likelihood. Typically the likelihood computed by the chosen models is orders of magnitude larger than the rest. In the case where the two models in our set produce likelihoods in the same order, we conclude that the observed data is a novel category, belonging to neither of the models. Thus, such a framework gives us the capability of "none-of-the-above" classifications which was not possible in the Genetic Programming templates.

The confusion matrix for the training set is given in Table 3.2 and for the test set, it is given in Table 3.3. We see that the HMM models perform well, with correct classification rate of 90%. We then replaced the computationally expensive optical flow calculation with a much simpler frame difference operator. We similarly trained the feature extractor to

True ↓ Labeled as

	News	Sports
News	11	1
Sports	0	12

Table 3.2: Confusion matrix for training set

True ↓ Labeled as

	News	Sports
News	11	2
Sports	0	13

Table 3.3: Confusion matrix for test set

compute new optimal projections from these frame differences. Building HMMs from these projections results in similar good performance. For our final system, we therefore rely on simple frame differences rather than computationally expensive optical flow estimation. This has the added advantage of low computational expense, an important requirement for on-line systems.

3.3.2 Action/Character movie trailer classifier

This technique analyzes the shot duration and motion energy of the movie trailers to determine their placing along an Action-Character axis. Modern film theory contends that movies follow a formula $A \times C = k$ [36]. For example, emphasis on action in an adventure movie leaves little time or place for the development of complexity of character. Similarly, everyday incidents become interesting when complex characters participate in them. We can say that an action oriented movie spends less time developing characters and the opposite is true in character-oriented movies. This is a reasonable assumption for the following reasons – Both the audience and the narrator rely on socially agreed upon structures to present and imbibe movies. Without such structures, it becomes impossible to understand movies (which is the primary motivation of the audience). For the creators, lack of such common structures has both economic and narration ramifications. It is to the creators benefit if they can rely on standard structures to set up the audiences' expectations and weave the narration. Presence of such production codes makes the task of editing simpler (and therefore less expensive) and the narration more coherent (and therefore engaging the audience).

Our assumption is that action emphasis manifests in movies as high energy in shots and frequent shot changes and character emphasis is the exact opposite. These are reasonable assumptions because there are only a few ways in which character/action developments can be shown. Emphasis on action in an adventure movie leaves little time or place for the development of complexity of character. Similarly, everyday incidents become interesting when complex characters participate in them. It is extremely difficult to quantify features such as emphasis on action or complexity of character because of their qualitative nature. However, character developments are frequently shown using dialogs and closeups. Fast cuts during dialogs and closeups is distracting. So, character movies can be expected to have long scenes and similarly, a popular way to embody action is to use fast scene cuts [2] and high action in scenes to create the illusion of action. We can predict reasonably that action movies will have a very busy imagery – for example high speed chases, explosions etc and similarly character movies will have long duration shots – dialogues, closeups on emotional moments etc. This bias comes from the requirements of typical shots that comprise the narration. For example, you cannot have a meaningful dialogue in a shot lasting just a few seconds. Similarly, when an explosion occurs, it lasts just a few seconds and a common way to capture it is to show flying debris (translates to high motion in the corresponding frames). Features such as shot length and motion energy are therefore expected to be revealing about the content of the movie albiet in a limited way. These shot durations, together with the activity in these shots are used to train the Hidden Markov Models to classify movie trailers.

For shot detection, we used the Kullback-Liebler (KL) distance between histograms in *rgb*

[2]See the famous shower murder in Psycho by Alfred Hitchcock

space to determine transitions. The *rgb* space is defined as follows:

If R, G and B represents the pixel values, then

$$\begin{aligned} r &= \frac{R}{R+G+B} \\ g &= \frac{G}{R+G+B} \\ b &= \frac{B}{R+G+B} \end{aligned} \tag{3.6}$$

This transformation renders the pixel values invariant to changes in illumination intensity (but not illumination color) [16]. To account for variation between shots, an adaptive threshold was chosen for shot change detection. The threshold is a constant factor times the average of past KL distances. While this technique cannot accurately detect gradual transitions such as dissolves, it is acceptable for the purposes of this work. This work relies on an approximate shot duration and motion energy and is not sensitive to small errors shot lengths. Once the shots were extracted, the shot duration in number of frames and the average optical flow energy in each shot was computed. For each trailer, we therefore have a series of shot durations and corresponding average flow energies. Table 3.4 shows the average shot duration, motion energy and the average KL distance between the rgb histograms of two frames in the shot for each of the 24 trailers. We note here that the average KL distance and the average motion energy correlate well suggesting that the average KL distance is also a possible feature for content characterization. We also note that the average shot duration and the motion energy are inversely correlated. Figure 3.14 plots the average shot length versus motion energy. We can clearly observe the $\frac{1}{c}$ nature of the graph in accord with the $A \times C = k$ theory.

Training and testing

As mentioned earlier, the dataset consists of 24 full length movie trailers. The shortest trailer is about 1.5 minutes long and the longest trailer is 5 minutes in duration. We

Movie	avg. shot length	avg. motion energy	avg. KL btwn. rgb hist.
badboys	14.6746	29.2512	0.3583
blankman	22.6115	27.0813	0.1209
dredd	24.5882	23.5782	0.5378
fighter	16.0071	24.5057	0.3323
jungle	26.8889	11.7419	0.1228
madness	17.9000	25.6444	0.2744
riverwild	46.0000	12.4335	0.0685
terminal	20.2564	22.4440	0.3559
tide	32.2595	10.6759	0.1160
vengeance	12.8198	25.1608	0.2447
circle	60.6780	5.0598	0.0445
clouds	23.7467	8.8116	0.0829
eden	47.1000	7.6748	0.0691
edwood	27.8230	8.9310	0.0714
french	62.1429	4.5978	0.0595
junior	39.9706	7.8763	0.1055
miami	66.9302	3.3276	0.0598
payne	30.8276	11.2788	0.1794
princess	24.3909	9.1767	0.1463
puppet	22.5155	12.4883	0.1478
santa	41.6000	5.8339	0.0899
scout	30.5041	5.8996	0.0570
sleeping	34.9800	5.1866	0.0393
walking	32.6444	8.8228	0.0901

Table 3.4: Table of average shot lengths, motion energy and KL distances

consulted the internet movie database (iMDB)[19] in order to pre-classify these trailers. This classification is based on the keywords used to describe the movies in the iMDB. Table 3.5 shows the trailer name, the full movie name, the iMDB keywords and the classification that we assigned to the trailer based on these keywords. We have 10 Action and 14 Character movies. We note here that since there is a continuum between action and character movies, this classification just denotes which side of the dividing line the movie lies. As our first experiment, we considered the average shot length and the average motion energy (see Table 3.4) as the discriminating features. We randomly chose 3 trailers from each category as the training set. Computing the means and variances from this training set, We performed a likelihood ratio test assuming implicitly that the shot lengths and motion energies were

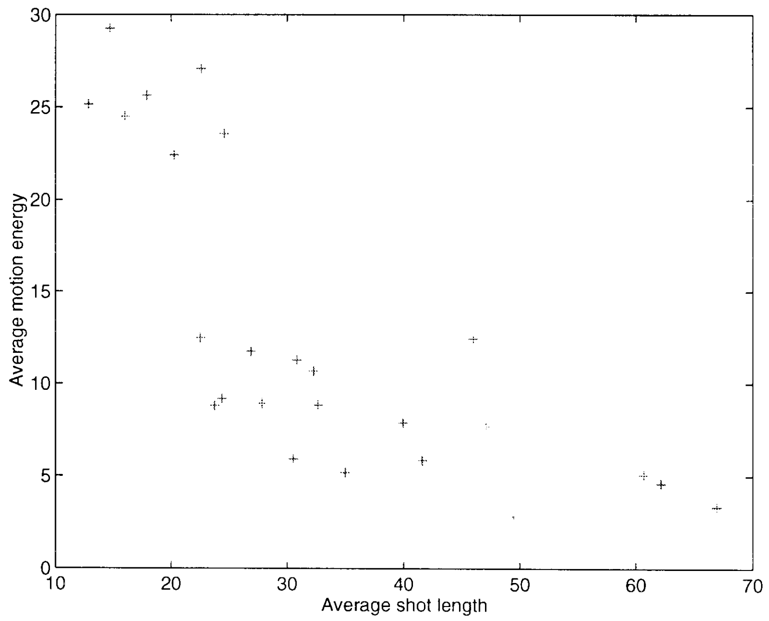


Figure 3.14: Plot of average shot length versus motion energy

distributed with a Normal density function. This resulted in 78% correct classification. This classification was also tried with the ratio of average motion energy to average shot length as the feature with a similar performance. Figure 3.15 is the histogram of the ratio of motion energy to shot length for one such trailer. We can clearly see that modeling the histogram with a single gaussian is not a very good approximation and therefore the classifier performs poorly. We decided to use the ratio of motion energy to shot length as a feature since in action movies, the motion energy is expected to be high AND shot lengths small and the converse in character movies. Therefore, a ratio is expected to be more discriminating. In the second experiment, we took the 6 randomly chosen trailers and trained two sets of Hidden Markov models on the ME/SL ratio. The models ranged from 2 fully connected nodes to 5 fully connected nodes. The training algorithm was the alpha-beta algorithm as outlined in [47]. The best classification rate that we obtained was 66.6%. This was much below the performance of the likelihood ratio test. Upon some analysis, we found the cause for the poor performance of the HMM models to be the discretization of states (specifically, the B matrix). The character movies have significantly smaller values

Trailer	Movie Name	iMDB keywords	Class
badboys	Bad Boys	violence	A
blankman	Blankman	comedy/superhero	A
dredd	Judge Dredd	violence	A
fighter	Street Fighter	action	A
jungle	Jungle Book	adventure/animals	A
madness	In the mouth of Madness	horror/thriller	A
riverwild	The River Wild	action/hostage	A
terminal	Terminal Velocity	action/skydive	A
tide	Crimson Tide	thriller/nuclear	A
vengeance	Diehard with a Vengeance	thriller	A
clouds	A Walk in the Clouds	drama/romance	C
circle	Circle of Friends	drama/romance	C
eden	Exit to Eden	comedy/sex	C
french	French Kiss	comedy/romance	C
miami	Miami Rhapsody	comedy/family	C
princess	The Little Princess	drama/school	C
santa	Santa Clause	comedy/christmas	C
scout	The Scout	drama/baseball/	C
sleeping	While you were Sleeping	comedy/romance	C
edwood	Ed Wood	biographical/historical	C
payne	Major Payne	comedy/army-life/teaching	C
junior	Junior	comedy/pregnancy/sci-fi/	C
puppet	The Puppet Masters	sci-fi/mind control	C
walking	The Walking Dead	drama/afro-american/war	C

Table 3.5: iMDB classification: "A" is action oriented and "C" is character oriented

of the ME/SL ratio compared to the action movies. Figure 3.16 shows the fitting of a mixture of Gaussians model to the two groups (specifically, to the 6 trailers in the training set). We can see that the character group of movies have significantly smaller ME/SL ratio. A simple uniform quantization, as in our first attempt with the HMM models, did not capture this well. In our second attempt with the HMM models, we transformed the data by taking the natural logarithm of the ME/SL ratio. We similarly trained two sets of HMMs ranging from 2 nodes to 5 nodes. Table 3.6 shows the results of the best performing simulation. The performance improvement from using the logarithm was very significant. The 2-node model and the 4 and 5-node models perform slightly poorly compared to the 3-node HMM implying underfitting and overfitting respectively. The best model correctly

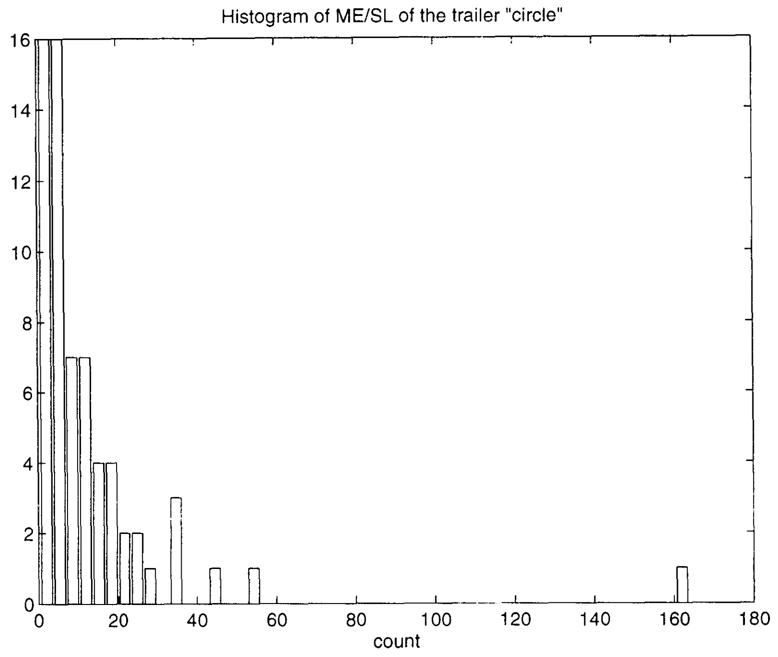


Figure 3.15: Histogram of the ME/SL ratio for the trailer "circle"

classified 22 out of 24 trailers achieving a recognition rate of 91.67%. The ratio of the two likelihoods given by the HMMs can be used to place the movies along the Action-Character axis. However, in order to verify this characterization, we can no longer rely on the simple descriptive keywords of iMDB.

Both the proposed Hidden Markov Models use the temporal evolution information of selected features to classify video sequences. Specifically, the HMMs made structural assumptions about the nature of video sequences and used them to classify these sequences. In other words, we can say that these Hidden Markov Models rely on a grammar that is present in structured video content in their analysis. In contrast, the query-by-example systems use purely visual features to retrieve video sequences. It might be interesting to combine the purely visual features with these grammar-based features. We present this approach in the next section.

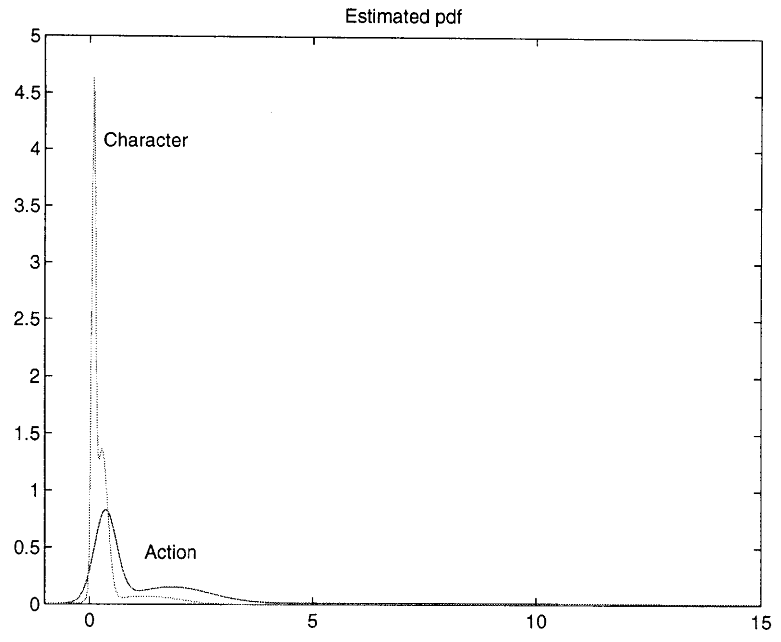


Figure 3.16: The two estimated pdfs. Green curve is the "character" pdf and the red curve is the "action" pdf

3.4 Controlled retrieval using VideoBook

Given a query sequence from a particular movie, it is reasonable to expect that visually similar sequences will be present in movies of similar genres. This expectation arises from the notion of socially accepted movie production codes (see section 3.3.2). Also, given a query sequence from a particular genre, it is possible that the user is interested in retrieving scenes from similar genres. We attempt to enable these queries by providing an additional control to the user in the VideoBook interface. This enables grammar-constrained searches on the movie trailer collection.

The Hidden Markov Models are essentially representations of the joint probability density function in the feature space of the samples that they classify. While these HMMs are useful for some applications, in many cases it is useful to represent the density functions in more amenable forms, such as semi-parametric representations. When we combine the

Movie	iMDB	log lik. (char)	log lik. (action)	3-node HMM class.
badboys	A	-584.6431	-279.2000	A
blankman	A	-372.4045	-273.7870	A
dredd	A	-116.0311	-65.7753	A
fighter	A	-412.5557	-230.2305	A
jungle	A	-235.8129	-231.4702	A
madness	A	-100.4025	-109.8857	**C** ←
riverwild	A	-285.8203	-199.0171	A
terminal	A	-266.8279	-149.0636	A
tide	A	-275.2908	-265.1946	A
vengeance	A	-147.9277	-111.3094	A
circle	C	-161.9689	-216.2792	C
clouds	C	-224.2623	-295.5342	C
eden	C	-171.2445	-192.7708	C
edwood	C	-198.2601	-218.6139	C
french	C	-158.0795	-206.4249	C
junior	C	-153.9576	-191.4976	C
miami	C	-171.0646	-210.9144	C
payne	C	-220.7809	-226.5245	C
princess	C	-233.8534	-244.2294	C
puppet	C	-304.5891	-212.0613	**A** ←
santa	C	-205.8415	-258.7029	C
scout	C	-271.4748	-296.7294	C
sleeping	C	-216.3463	-258.3200	C
walking	C	-231.4427	-255.0258	C

Table 3.6: Classification results for the 3-node HMM model

purely visual low-level features-based representation of the VideoBook with the grammar-based approach, it is useful to represent the densities in semi-parametric forms rather than as HMMs. The query-by-example retrieval task is now turned into a weighted retrieval where one component of the distance is computed from the visual features and the other component is derived from the distance between movies in the density space. At one end, this task is identical to the original VideoBook query and the other end, it is a purely grammar based retrieval. So the queries can range from "Show all shots that look like this" to "Show all movies (trailers) that are cut like this movie (trailer)". In the middle ground, we have queries of the type "Show all shots that look like this, in movies (trailers) that are cut like this movie (trailer)".

3.4.1 Classification of Trailers using KL divergence

Based on our experiments with Motion Energy and Shot Lengths as useful features for movie trailer classification, we use the EM algorithm [9] to construct a mixture of gaussians probability density representations for each of the movie in our collection. Prior to using them in VideoBook, we evaluate their performance with a classification experiment similar to that in Section 3.3.2. Since the quantities we are interested in comparing are density functions, we use the KL divergence as our classification criterion.

We use the 6 trailers used earlier to train the HMMs to create “character” and “action” density functions (see Figure 3.16). Each trailer is similarly represented using a mixture of gaussian model and the KL divergence between the trailer and the two classes is computed. We chose the symmetric form of the KL (SKL) divergence (defined in equation 3.7) as the similarity measure. The SKL divergence between the group PDFs and the sample PDF was used to classify the sample in the two categories. This classification is then compared with iMDB ground truth. Table 3.7 shows the results of the final classification.

$$SKL(p||q) = KL(p||q) + KL(q||p) \tag{3.7}$$

We note that the SKL classifier performs as well as the HMMs classifier earlier. Only one action trailer was misclassified as a character movie. Upon further analyzing the trailer, we found that the trailer was from the movie “The River Wild” and the opening shot of the trailer was a long and reasonably static shot (16 seconds) of a river. This skewed shot was probably responsible for that trailer being misclassified.

3.4.2 Joint grammar and visual query

This joint approach can be viewed as setting up the priors of the features-based retrieval interactively. That is, the bias of the features towards movies of similar semantic content

Movie	$SKL(action\|\|movie)$	$SKL(character\|\|movie)$	classification
badboys	0.933242	17.426211	A
blankman	0.404791	16.910384	A
dredd	0.968591	7.665252	A
fighter	0.165028	8.599979	A
jungle	1.281311	1.863222	A
madness	0.180119	19.061343	A
riverwild	17.284014	1.478502	C
terminal	1.285146	4.503840	A
tide	1.055045	2.193608	A
vengeance	1.506620	3.244065	A
circle	14.991360	0.118162	C
clouds	17.268471	0.036801	C
eden	8.512113	0.174151	C
edwood	6.784345	0.585779	C
french	Inf	1.509291	C
junior	Inf	1.365144	C
miami	Inf	3.116879	C
payne	1.408200	1.525291	C
princess	6.558665	0.149712	C
puppet	4.330472	1.231515	C
santa	9.895152	0.108404	C
scout	15.054414	0.180318	C
sleeping	4.935052	0.290696	C
walking	3.511839	0.381914	C

Table 3.7: Final classification of trailers using the SKL distance

can be controlled. At one extreme, individual shots do not matter and retrievals are purely based on proximity of the movies from which the shots were obtained. At the other extreme, only visual similarity is used for retrieval. A middle ground where both play a role will possibly result in more meaningful and visually similar retrievals.

To combine the two approaches, we normalize the similarity measures from both the representations into a joint rank. The normalized similarity function is as shown below:

$$S(i,j) = (1.0 - w) \times \exp(-k_1 \times d1) + w \times \exp(-k_2 \times d2) \tag{3.8}$$

where $d1$ is the euclidean distance between two shots and $d2$, the KL divergence between the corresponding trailers respectively. The weighting factor w determines the degree of semantic influence in the query. It is chosen to be between $(0, 1)$ by the user. Setting $w = 0$ indicates purely features-based query. Conversely, $w = 1$ is a purely semantic query. Factors k_1 and k_2 are scale factors chosen to appropriately scale $d1$ and $d2$. Given the above transformation, a higher similarity value implies a closer match, with 1 indicating a perfect match.

We performed 100 query experiments with $w = 0.5$, indicating equal weighting to both the representations. As in section 3.1, the trailer database was divided into two groups. A random shot would be chosen from the query group and queried against the test group. The result was considered correct if the correct trailer was identified in rank 1 position AND the retrieved video shot was visually similar to the query shot. We obtained 96 correct responses, indicating a performance gain of 4% compared to the simple features based approach. In addition, the shots returned in the next 2 positions are also both visually similar and from trailers of similar semantic content. Figure 3.17 shows key frames from a sample query shot and the corresponding retrievals are shown in Figure 3.18. The semantic weighting serves to increase the prior for similar content and this results in higher observed performance. In addition, the semantic weighting constrains the search space in favor of trailers with similar characteristics which results in higher visually proximate rank 2 and rank 3 retrievals as well.

3.5 Why probabilistic descriptions?

In this chapter and the previous one, we toured through many approaches for image and video content characterization. For image retrieval, histogram and histogram refinement techniques (such as coherency vectors and correlograms) have been demonstrated to have superior performance under a wide range of queries. From an unstructured data perspective, the advantage that histogram-based techniques offer us is that they do not make

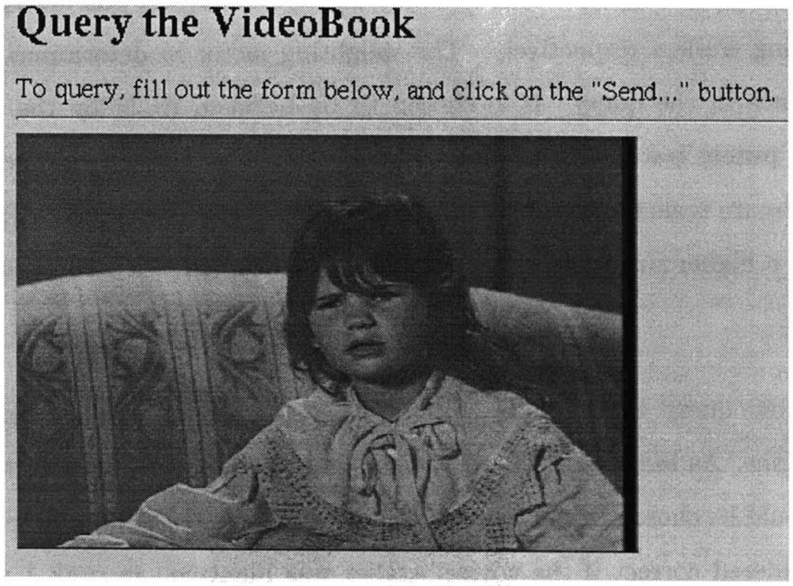

Query the VideoBook
To query, fill out the form below, and click on the "Send..." button.

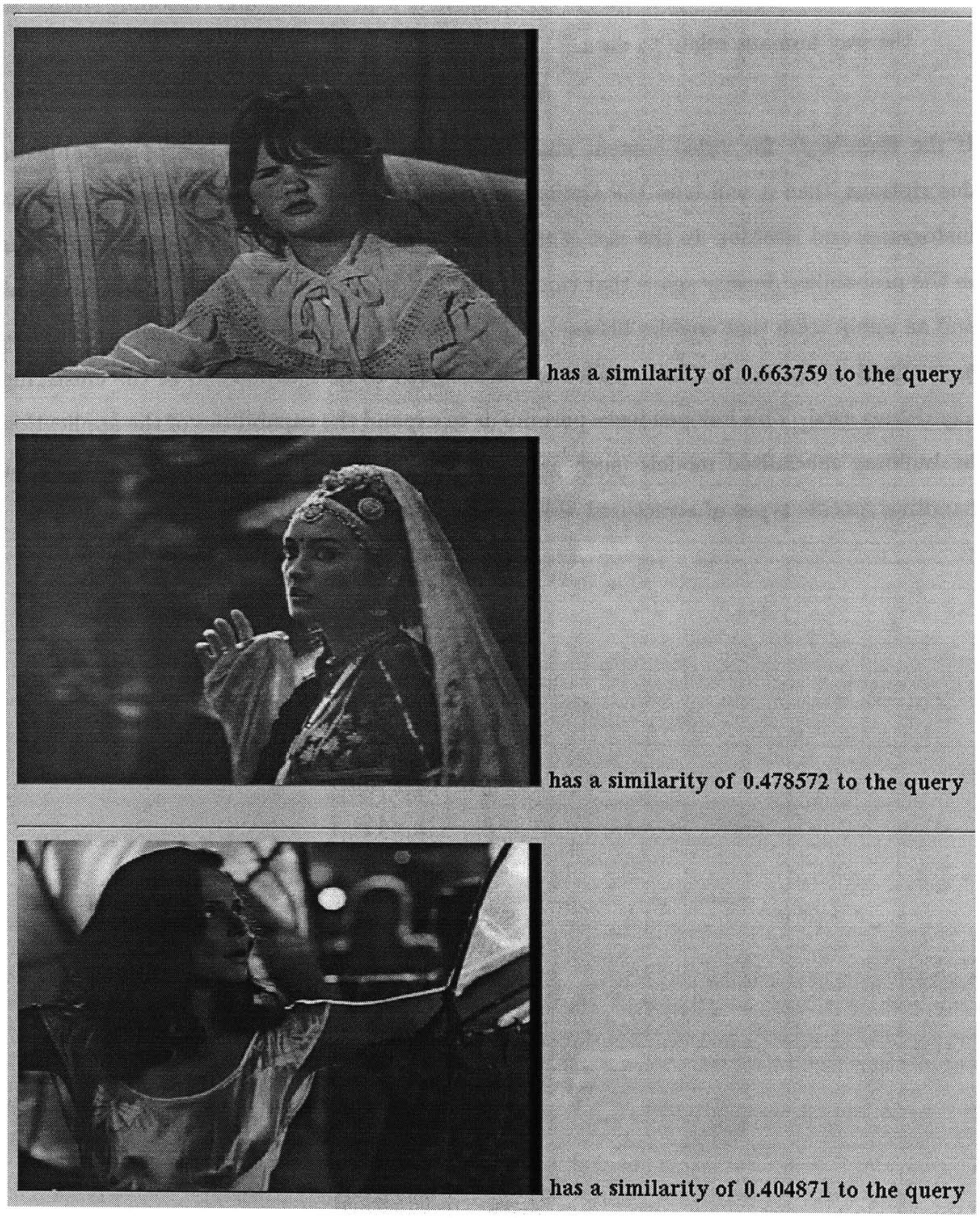
Figure 3.17: Sample query shot for $w = 0.5$

any assumptions about the content and therefore are applicable widely. Essentially, these histogram-based techniques describe images and video in terms of statistics of the extracted features. Similarly, at the other extreme of structured content, Bayesian Models such as the HMMs we presented were essentially modeling a high-level description in terms of a probability density over the extracted features. In addition, probabilistic descriptions have other advantages such as

- They are robust to missing/corrupted data. That is, they can handle dropped frames, corrupted blocks etc. Their performance degradation is gradual.
- They can be used with different data types. That is, an image and a video sequence can be treated similarly. An image can be used to query for video and vice-versa.
- They enable multiple groupings. For example, a picture of a crowd taken outdoors can be categorized both as a picture of a crowd and an outdoors picture. Probabilistic descriptions provide confidence estimates for each type of categorization which is then used for multiple groupings.

- They are well understood and provide intuitive explanations and reasoning similar to the way humans relate to data.

If the framework for video content characterization is built upon a set of probabilistic descriptions, then it will have the ability to work at multiple levels of complexity (such as histograms and HMMs). In the rest of the thesis we describe a set of clustering algorithms in the probability density space that can be used for cataloging unstructured video content and an application that enables browsing and constructing of alternate outputs from video, as outlined in chapter 1. This application is designed to be independent of the clustering algorithms used. This independence permits us to expand the capabilities of the application by building specialized models (such as the Bayesian Models and HMMs presented) for handling specific types of structured video content.


has a similarity of 0.663759 to the query

has a similarity of 0.478572 to the query

has a similarity of 0.404871 to the query

Figure 3.18: Corresponding retrievals for $w = 0.5$. Note a similarity value closer to 1 implies a better match

Chapter 4

Density-based Clustering

In this chapter, we start with the assumption that video sequences are modeled as probability density functions and present clustering algorithms in density space.

The clustering work is presented as four different algorithms:

- Hard clustering, which is similar to Vector Quantization.

- Soft clustering, similar to Expectation-Maximization. The particular technique we use is an extension of the deterministic annealing technique proposed for Vector Quantization. This is a top-down algorithm. The hard clustering algorithm can be viewed as a special case of this algorithm.

- Soft agglomerative clustering. This is a bottom-up algorithm which addresses some efficiency issues with the top-down algorithm.

- Hard agglomerative clustering. The hard variant of the above soft algorithm.

4.1 Why clustering for unstructured video?

With structured video sources, the assumptions of common editing codes and presence of a structured narration are valid. These make possible the use of high-level inference mechanisms such as the Hidden Markov Models (see Chapter 3, Section 3.3) and Bayesian Networks (see Chapter 2, Section 2.4). In [31], we argued that a Bayesian approach is a natural solution to semantic characterization when the assumptions of structured content can be made. However for unstructured video, we cannot rely on these models because some of these structural assumptions are not valid. Also, a major limitation with the structured approach is that these models are good at handling only the vocabulary that they have been trained upon. Training data requirements grow exponentially with the number of classes and can become unmanageable for any realistic sized vocabulary.

The query-by-example paradigm [42, 12, 20] (and also all examples in Chapter 2, Section 2.2) is a powerful approach for unstructured video characterization because it makes no assumptions about the nature of content. However, it pays the price by relying on lower-level features. The gap between the machine representation and the human perception is wider compared with model-based approaches. In addition, although query-by-example works well when the database is small, as the database grows, presenting the user with a small randomly chosen subset of thumbnails is no longer a viable choice. Also, the notion of serendipitous discovery of the "right" image/video in the query-by-example paradigm gets weak if the initial subset that is presented is not representative of the database. Another important problem with query-by-example is that it has not characterized or abstracted the database in any particular way. It searches the entire database every time a query is presented. While there are efficient search techniques to reduce this search complexity, the problem of not having generalized and abstracted the data remains. In [37] the abstraction problem is addressed by allowing the user to provide relevance feedback and propagating newly learned labels. This can be viewed as the system having the ability of "short term memory". This system achieves the happy middle ground between pure query-by-example with no assumptions at one end and Bayesian modeling with structural assumptions at the other extreme.

We argue that clustering offers yet another such middle ground. Clustering is a natural way to organize massive quantities of data and build interfaces that take advantage of the cognitive abilities of the user. For example, let us suppose that the user presents a video collection that is made up of shots from a typical home video. It is reasonable to expect that good clustering algorithms will groups all shots that have similar features together. For example, most shots taken indoors will be in one cluster and most shots taken outdoors will be in another cluster, purely from the color features. The user views these clusters and attaches labels. The clustering algorithm relied on the user's ability to make inferences and attach cognitive meaning. This is not to say that it will work all the time. Some clusters may be such that we cannot provide clean labels. The interesting point is that from the perspective of extracting alternative outputs from the video (like collages, storyboards, still pictures and video dramas), it is not important that all clusters make cognitive sense to the user.

Clustering also offers another advantage that alleviates some of the problems with query-by-example. As the database grows in size, clustering is a natural solution to presenting the user with a small representative subset of the database. As opposed to showing a few randomly chosen images/video clips from the database to start off on a query-by-example search, chosen images/video from each cluster can be used as the starting point. This process can be repeated ad infinitum with a hierarchical clustering scheme. This ensures that the user is presented with a representative collection at each query.

A good choice for the representation of clustering should not compromise the ability to handle higher-level characterization. In particular, if the clustering is performed in the space of probabilistic representations, we retain the advantage of using higher-level probabilistic descriptors when available. We saw earlier the other advantages with using probabilistic descriptions.

4.2 Introduction to Distribution Clustering

Study of clustering techniques is a topic to which much attention has been devoted [26]. Traditionally, clustering has emphasized discovering structure in data, compact summarization of data etc. Clustering has also been proposed for alleviating sparse data problems [29, 44].

Clustering algorithms can be grouped into two categories: partioning and hierarchical. In partitioning approaches, the sample set is divided into a set of distinct groups. That is, each sample point is placed into one of many sets. Partitioning approaches can be viewed as a special case of hierarchical clustering. In hierarchical clustering, all samples belong to one cluster at the top level and as we descend down the levels, the number of partitions increase. Hierarchies are either constructued in a top-down or bottom-up fashion. Bottom-up techniques are usually termed as agglomerative clustering.

Each clustering approach can either be soft or hard. In hard clustering, the partitionings are distinct and each sample belongs to exactly one partition. In soft clustering, each sample belongs to every partition with a "degree" of belongingness. This degree sums to unity across all partitions (and at each level, in case of hierarchical clustering). Most clustering algorithms begin by creating an initial estimate and then iteratively refine estimate by optimizing a cost function till they converge onto a local minima. Techniques such as simulated annealing are employed to select better minimas and in theory simulated annealing can converge to the global minima.

Traditionally in clustering, the sample points are n-dimensional vectors. In our case, the sample points are probability density functions as we model each image or video sequence in terms of a density over the extracted features. A simple example of a density estimate is a normalized color histogram which is one of the features that we use. The use of density functions impose special constraints on the clustering algorithms. Specifically, this requires that all centroids that we estimate must be valid densities. Clustering distributions is a topic of interest on its own accord. In our case, we have illustrated the need and advantages of modeling video as a collection feature distributions (see Chapter 3) and the merits of clustering for handling unstructured video.

4.3 Hard partition clustering

We assume that we start with a collection of discrete distributions, with each sample distribution representing a video shot. For example, these distributions could be the normalized color histograms, or the color correlograms. Given a collection of distributions, the Kullback-Liebler divergence becomes a natural choice for similarity between two samples. In order to cluster a collection of such samples, we define a distortion criterion J and minimize it. The clustering algorithm then proceeds in the following steps:

1. Initialize by randomly assigning N centroids.
2. Assign each sample to the closest centroid. Distance from each centroid is estimated using the KL divergence.
3. Move each centroid to a new location such that the cumulative distortion between the assigned samples and the centroid is minimized.
4. Repeat steps 2 and 3 till no samples change centroids. If no change in assignment, stop.

4.3.1 Clustering using KL divergence

Given a cluster centroid, the cumulative distortion contributed by that centroid is given by

$$J = \sum_j D(p||q_j) + \lambda(\sum_i p(i) - 1) \tag{4.1}$$

where q_j is the j^{th} sample density of N_c samples that belong to one cluster and p is the centroid density. In our notation, $q_j(i)$ denotes the i^{th} bin of the density and $p(i)$ is similarly the i^{th} bin of the centroid density. We add the Lagrange constraint to ensure that the resulting density is valid. We also implicitly assume that each of the q_j are normalized to be valid densities. $D(p||q_j)$ is the KL divergence between q_j and p. KL divergence can be defined in two ways, as we saw earlier.

We minimize J with respect to the cluster centroid for both the expressions of KL divergence below.

Theorem 4.1 *Given the following expression for the cumulative cluster distortion*

$$J = \sum_j \sum_i q_j(i) \log \frac{q_j(i)}{p(i)} + \lambda(\sum_i p(i) - 1) \tag{4.2}$$

the centroid distribution is the mean of the sample distributions.

Proof: Minimizing the cumulative distortion with respect to the bins of the centroid, we get

$$\begin{aligned} \frac{\partial J}{\partial p(i)} &= -\sum_j \frac{q_j(i)}{p(i)} + \lambda \\ p(i) &= \frac{1}{\lambda} \sum_j q_j(i) \end{aligned} \tag{4.3}$$

$$\begin{aligned} \frac{\partial J}{\partial \lambda} &= \sum_i p(i) - 1 \\ \sum_i p(i) &= 1 \end{aligned} \tag{4.4}$$

Together, we get

$$\begin{aligned} \sum_i \frac{1}{\lambda} \sum_j q_j(i) &= 1 \\ \lambda &= N_c \end{aligned} \tag{4.5}$$

And we get,

$$p(i) = \frac{1}{N_c} \sum_j q_j(i) \tag{4.6}$$

That is, each bin of the centroid is the arithmetic mean of the bins of the sample densities.

Theorem 4.2 *Given the following expression for the cumulative cluster distortion*

$$J = \sum_j \sum_i p(i) \log \frac{p(i)}{q_j(i)} + \lambda(\sum_i p(i) - 1) \tag{4.7}$$

The centroid distribution is the geometric mean distribution of the samples.

Proof: Minimizing this J with respect to the $p(i)$s, we get,

$$\begin{aligned} \frac{\partial J}{\partial p(i)} = \lambda + N_c(1 + \log(p(i))) - \sum_j \log(q_j(i)) = 0 \\ \sum_j \log(\frac{p(i)}{q_j(i)}) = \sum_j \log(\frac{p(k)}{q_j(k)}), \ \forall i \neq k \\ N_c \times \log(\frac{p(i)}{p(k)}) = \sum_j \frac{q_j(i)}{q_j(k)} \\ \frac{p(i)}{p(k)} = \prod_j (\frac{q_j(i)}{q_j(k)})^{\frac{1}{N_c}} \end{aligned} \tag{4.8}$$

From which we get the following expression for $p(i)$

$$p(i) = C \times \prod_j q_j(i)^{\frac{1}{N_c}} \tag{4.9}$$

where C is the constant (4.10) introduced to satisfy the Lagrange constraint to make p a valid distribution. We see that the $p(i)$ comes out to be the geometric mean of the sample densities.

$$C = \frac{1}{\sum_i \prod_j q_j(i)^{\frac{1}{N_c}}} \tag{4.10}$$

4.3.2 Clustering using Euclidean norm

Additionally, instead of using the KL divergence as the distortion measure, we could use a valid distance function such as the euclidean norm. That is, we could consider the discrete density function as an n-dimensional vector and apply any of the standard distance functions, with euclidean distance being one such example. We continue to impose the Lagrange constraint to ensure that the resulting centroids remain valid densities. This modifies the cumulative distortion to

$$J = \sum_{j}^{N_c} \frac{1}{2} \sum_{i} (q_j(i) - p(i))^2 + \lambda(\sum_{i} p(i) - 1) \tag{4.11}$$

Minimizing the cost from (4.11) with respect to each bin of the centroid, we get the following:

$$\begin{aligned} \frac{\partial J}{\partial p(i)} &= \sum_{j} -(q_j(i) - p(i)) + \lambda = 0 \\ \lambda &= \sum_{j} q_j(i) - N_c p(i) \end{aligned} \tag{4.12}$$

From which we get

$$p(i) = \frac{1}{N_c} \sum_{j} q_j(i) \tag{4.13}$$

Which results in the same expression as using (4.2) as the criterion. This is an interesting result since the euclidean norm is a very common first choice for distance function. This result implies that using euclidean distance and computationally more cumbersome KL divergence result in similar clustering.

One problem that remains is the determination of the number of clusters. The hard clustering algorithm can work for any number of clusters. A variety of different techniques can be used to determine the number of clusters from the data. For example, we could plot the final cumulative distortion versus number of clusters. If there is a true number of clusters in the data, the cumulative distortion would drop with increasing number of clusters until the desired number of clusters are reached, and after that the drop in cumulative distortion would be much smaller. Another way to estimate the number of clusters would be leave some data aside and at each iteration, plot the cumulative distortion of the left out data. After the "true" number of clusters are reached, this cumulative distortion changes direction. This approach is popularly known as cross-validation. Alternatively, we could apply a set of heuristics to determine the number of clusters. At each iteration, we split clusters that have a higher variance than the average and merge clusters that are close to each other. This is akin to the Isodata procedure outlined in [60]. A version of our algorithm that is Isodata-like will be detailed in Section 4.6. Another variation is to add a penalty term to the number of clusters (clearly, the more the number of clusters the higher is the cost of representing the data, if we are doing compression). This imposes an additional constraint on the optimization procedure and is popularly referred to as the MDL (minimum description length) principle. In Section 4.4 we outline yet another procedure for determining the number of clusters. Unfortunately, there is no right answer since we do not know how many clusters are in the data to begin with. Ultimately, the question to ask is what are we going to use the clusters for? That determines the price we are willing to pay in choosing amongst the different selection criteria. In our case, we are using the clustering to show interesting groupings to the user and to enable easy navigation through a potentially vast collection of video. In addition, we are also interested in efficient retrieval through this collection whenever the user would like to perform a specific query-by-example search. The considerations that govern the number of clusters to be shown are the granularity desired by the user, size of the database (to perform efficient query-by-example searches), possibly the network connection that the user has to the database. Briefly, if the network connection is slow it is better to show more clusters, with fewer video sequences per cluster, than it is to show a few clusters, with many sequences per cluster. Or, for slower connections, it is better to organize the clusters in a hierarchy and show with increasing detail as user requests get more specific. Nevertheless, the point here is that the number of clusters in our case are governed by considerations that seem unrelated to the basis behind the heuristics that are used to select the number of clusters.

We will detail performance of the hard clustering algorithm together with the other algorithms in Section 4.6. This section also details the specifics of each algorithm.

4.4 Hierarchical soft clustering

The clustering algorithm outlined in the previous section assigns each sample density into one cluster. While this is useful, in many cases it is desirable to have a soft assignment, i.e, have samples belong to multiple classes and have a probabilistic association with each class. For example, a particular video shot of two people in a conversation taken outdoors can be classified as both a dialog and an outdoors shot. Real world examples can and tend to have multiple labels associated with them. It is desirable to have the clustering algorithms possess the ability to assign multiple labels.

The soft assignment process is desirable for yet another reason: As we noted earlier, the hard clustering algorithm is an iterative procedure with the potential of getting locked in a local minima. The soft assignment algorithm can be viewed as a technique for selecting a better local minima. In fact, the deterministic annealing algorithm on which our soft algorithm is based was proposed as a technique for the same [48].

4.4.1 Framework for probabilistic clustering

We begin with the notion that each sample point (in our case, distribution) is associated *in probability* with each cluster. Hard clustering is now a special case where the association probabilities are either 1 or 0. We quote from [48] the following which is true for clustering (distributions or otherwise).

For a given set of centroids, the expected distortion in the probabilistic case is

$$\langle D \rangle = \sum_{j=1}^{N} \sum_{k=1}^{K} P(q_j \in C_k) d(q_j, p_k) \tag{4.14}$$

where the q_js are the sample points, with N such points and the p_ks are the centroids with K such centroids, each cluster denoted by C_k. The measure $d(q_j, p_k)$ is the distortion for representing sample q_j by centroid p_k. Under the assumption that we have no prior knowledge about the data, it is reasonable to apply the principle of maximum entropy. That is, given the different configurations a system can be in, the most likely configuration is the one with maximum entropy. It is relatively easy to show that maximizing the entropy under (4.14) results in the Gibbs distribution.

$$P(q_j \in C_k) = \frac{e^{-\beta d(q_j, p_k)}}{Z_{q_j}} \tag{4.15}$$

where Z_{q_j} is the partition function

$$Z_{q_j} = \sum_k e^{-\beta d(q_j, p_k)} \tag{4.16}$$

It is also reasonable to assume that the probabilities relating the different samples to clusters are independent. Hence the total partition function is $Z(q) = \prod_j Z_{q_j}$. The parameter β can be considered inversely proportional to temperature, given the analogy of annealing.

In [48], they further show that the process of minimizing the distortion subject to maximizing the entropy is equivalent to minimizing the Gibbs free energy which is defined as

$$F(p) = -\frac{1}{\beta} \log Z(q) \tag{4.17}$$

The set of centroids p, that minimize the free energy satisfy

$$\frac{\partial}{\partial p_k} F = 0 \tag{4.18}$$

which results in

$$\sum_j P(q_j \in C_k) \frac{\partial}{\partial p_k} d(q_j, p_k) = 0 \tag{4.19}$$

While this technique works with any distortion function, It can additionally be shown that for convex distortion functions, there is a unique solution that minimizes $\sum_j d(q_j, p)$ [48]. That is, at $\beta = 0$ there is exactly one cluster. At $\beta > 0$, a set of vectors satifying (4.18) corresponding to a local minima in free energy can be obtained. The single solution at $\beta = 0$ is a solution for all β but it changes from being stable to nonstable as β increases. This gives rise to the following generic algorithm for clustering.

1. Set $\beta = 0$ and find the global solution.

2. Replace each centroid in the solution set with many copies, each copy randomly perturbed. Increase β per annealing schedule.

3. Repeat probabilistic assignment and centroid computation till convergence.

4. Discard duplicate centroids (i.e, the ones that converge to the same point). This is the solution set at the new β.

5. Repeat from step 2 till desired number of clusters are reached or each sample is assigned to a cluster by itself.

It is interesting to note that at each β value, a *natural* number of clusters result without employing any heuristics. For a range of β values, the same set of clusters emerge. In continuing with our physical analogy, each such range is analogous to a phase. If it is possible to determine apriori at which β values a phase transition takes place, we can speed up the simulation at all other places. In [48], the authors compute these β values for νth law distortion functions. Since the phase transitions are known deterministically, the simulation is akin to *deterministic annealing.* In the distribution clustering problem, it is not easy to determine these β unstability points because of the relative unwieldiness of KL divergence. Hence, we are forced to rely on the slower simulated annealing update schedules.

4.4.2 Soft distribution clustering

Our clustering algorithm is inspired by this algorithm. We note importantly that nowhere in the Free Energy formulation or the centroid optimization was any constraint placed on the form of the distortion function. This implies that an asymmetric similarity function such as the KL divergence may also be used. See Ref. [29, pages 23-25] and Ref. [48] for derivations of the Gibbs free energy formulation. See Appendix B for a simple proof of convexity of KL divergence which implies that at $\beta = 0$ we will have one cluster. We note that this cluster is also a solution at all values of β. However, it may not be a stable solution at $\beta > 0$, signalling phase transitions.

In our case, the samples are not n-dimensional points, but probability distributions. This places the additional constraint that each resulting centroid solution be a valid distribution. We employ the above probabilistic clustering framework for soft clustering of a collection of distributions, using KL divergence as the similarity measure. The additional constraint leads to the following minimization formulation

$$\frac{\partial}{\partial p}F(p) = \frac{\partial}{\partial p}(-\frac{1}{\beta}\log Z(q) + \lambda(1 - \sum p)) = 0 \tag{4.20}$$

As in the hard clustering case, we use the two different forms of the KL divergence as the similarity measure and solve for the centroid.

Theorem 4.3 *Given the following expression for the cumulative distortion*

$$J = \sum_j \sum_k (P(q_j \in C_k) KL(q_j || p_k)) + \lambda(\sum_i p_k(i) - 1) \tag{4.21}$$

where $KL(q_j||p_k)$ is defined as

$$KL(q_j||p_k) = \sum_i q_j(i) \log \frac{q_j(i)}{p_k(i)} \tag{4.22}$$

and $P(q_j \in C_k)$ is as defined in (4.15) with the appropriate form of the KL divergence, the centroid distribution is the weighted mean of the sample distributions.

Proof: Minimizing the cumulative distortion with respect to the bins of the centroid p_k, we get

$$\begin{aligned}
\frac{\partial J}{\partial p_k(i)} &= -\sum_j P(q_j \in C_k)\frac{q_j(i)}{p_k(i)} + \lambda \\
\frac{1}{p_k(i)}\sum_j P(q_j \in C_k)q_j(i) &= \frac{1}{p_k(l)}\sum_j P(q_j \in C_k)q_j(l),\ \forall i \neq l \\
\frac{p_k(i)}{p_k(l)} &= \frac{\sum_j P(q_j \in C_k)q_j(i)}{\sum_j P(q_j \in C_k)q_j(k)} \\
p_k(i)\ \alpha &\sum_j P(q_j \in C_k)q_j(i)
\end{aligned} \tag{4.23}$$

This proportionality combined with the Lagrange constraint requires that $p_k(i)$ be:

$$p_k(i) = \sum_j P(q_j \in C_k)q_j(i) \tag{4.24}$$

Which results from the fact that $\sum_k P(q_j \in C_k) = 1$. Therefore, the centroid is the weighted mean of the sample distributions.

Theorem 4.4 *Given the following expression for the cumulative distortion*

$$J = \sum_j \sum_k (P(q_j \in C_k)KL(p_k||q_j)) + \lambda(\sum_i p_k(i) - 1) \tag{4.25}$$

where $KL(p_k||q_j)$ is defined as

$$KL(p_k||q_j) = \sum_i p_k(i)\log\frac{p_k(i)}{q_j(i)} \tag{4.26}$$

and $P(q_j \in C_k)$ is as defined in (4.15) with the appropriate form of the KL divergence, the centroid distribution is the weighted geometric mean of the sample distributions.

Proof: Minimizing J with respect to the $p_k(i)$s, we get,

$$\begin{aligned}
\frac{\partial J}{\partial p_k(i)} &= \sum_j (P(q_j \in C_k) \log \frac{p_k(i)}{q_j(i)}) + \sum_j P(q_j \in C_k) + \lambda = 0 \\
\sum_j (P(q_j \in C_k) \log \frac{p_k(i)}{q_j(i)}) &= \sum_j (P(q_j \in C_k) \log \frac{p_k(l)}{q_j(l)}) \ \forall i \neq l \\
\log \prod_j (\frac{p_k(i)}{q_j(i)})^{P(q_j \in C_k)} &= \log \prod_j (\frac{p_k(l)}{q_j(l)})^{P(q_j \in C_k)} \ \forall i \neq l \\
\prod_j (\frac{p_k(i)}{q_j(i)})^{P(q_j \in C_k)} &= \prod_j (\frac{p_k(l)}{q_j(l)})^{P(q_j \in C_k)} \ \forall i \neq l
\end{aligned} \tag{4.27}$$

From which we get the following expression for $p_k(i)$

$$p_k(i) = C \times \prod_j q_j(i)^{P(q_j \in C_k)} \tag{4.28}$$

where C is the constant introduced to satisfy the Lagrange constraint to make p_k a valid distribution. We see that the $p_k(i)$ comes out to be the weighted geometric mean of the sample densities. It is interesting to note that these two expressions are analogous to the expressions for centroids in the hard clustering case. We can similarly use euclidean distance instead of KL divergence in the cumulative distortion equation and as before, we get the weighted average as the expression for the centroid. As we mentioned earlier, partitioning into multiple clusters from a single cluster at the top relies on the centroid solution becoming unstable with change in β. In our experiments, we found that the Geometric mean expression results in one stable centroid for all values of β. We have not yet been able to find an analytical expression for the stability of centroid, but we suspect that it may be possible to show that for the Geometric mean case there is only one stable solution.

This technique can be used in two ways: As a standalone, hierarchical clustering scheme or as a technique for selecting better hard partitions. That is, the algorithm can be run for a while with increasing β values and once the desired number of clusters are reached, the simulation can be frozen by increasing β to ∞ and reverting to the hard clustering algorithm. We note here that we can employ β to find cluster regimes. That is, for every number of clusters, we can find a range of β values that gives the same number of clusters. Additionally, some ranges of β values will be longer than others. If there are ranges of β that are significantly longer than the rest, it is reasonable to expect that these correspond to natural clusters in the data. We call each such range a cluster regime, borrowing from fluid flow terminology[1]. We did an experiment with a small dataset of 45 video shots (representing 10 minutes of home video) whose color histograms were used to cluster[2]. Figure 4.1 is the plot of β versus N_c, the number of clusters. We can identify 3 regimes in the plot for $N_c = 1, 4, 7$ respectively. This plot seems to indicate that the dataset naturally splits into these groups. Even so, we have multiple choices for the number of clusters. For example, we could choose either 4 clusters or 7; But it is unclear if either choice has an advantage without the context of the application or other considerations.

A similar distribution clustering algorithm was presented in [29] as a solution to handling sparse data problems in natural language processing. A major difference is that the underlying sample points are distributions in our case and vectors in their case. Rather than clustering vectors of verb-noun pairs, which tends to be sparse, they estimate a set of conditional densities and perform clustering in the density space. Also, they do not consider both the forms of KL divergence nor the possibility of using a simple, regular (such as euclidean) distance function for clustering.

4.5 Agglomerative distribution clustering

The algorithm presented above is computationally inefficient as the simulation has to be repeated over the entire data set for each change in β value. If the dataset is large, this com-

[1]Fluid flow is characterized by a Reynolds number that has regimes. Two such regimes are the laminar flow and turbulent flow.

[2]We used a small dataset primarily because of the time it takes the soft clustering algorithm to execute each iteration.

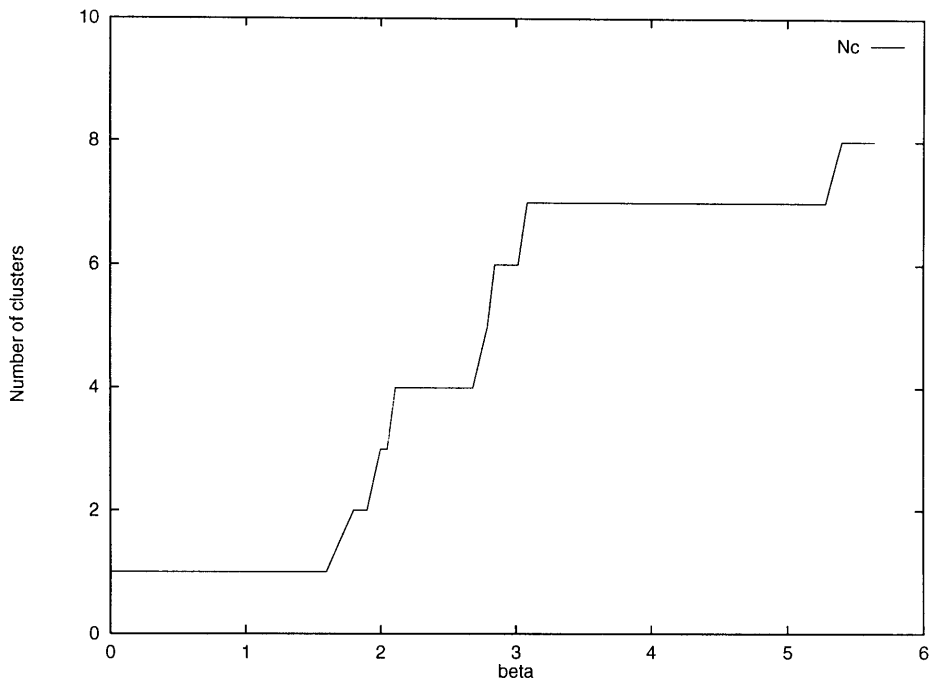


Figure 4.1: Plot of β vs. Number of clusters indicating cluster regimes.

putation can be prohibitively expensive. Ideally, if one level of the hierarchy can be related to the next level without reference to the underlying data, then the resulting algorithm will be computationally efficient since we deal with a (possibly) much smaller collection of clusters at each level as opposed to the entire dataset. Since this is a top-down hierarchical algorithm, with increasing details as we descend the levels, there does not seem to be a way to relate two levels without taking recourse of the underlying dataset. We now present a bottom-up approach in which we achieve this computational savings by relating the levels in the hierarchy. We present this algorithm in two flavors: hard and soft. The hard algorithm is akin to the agglomerative algorithms in [26] and is a special case of the soft agglomerative algorithm.

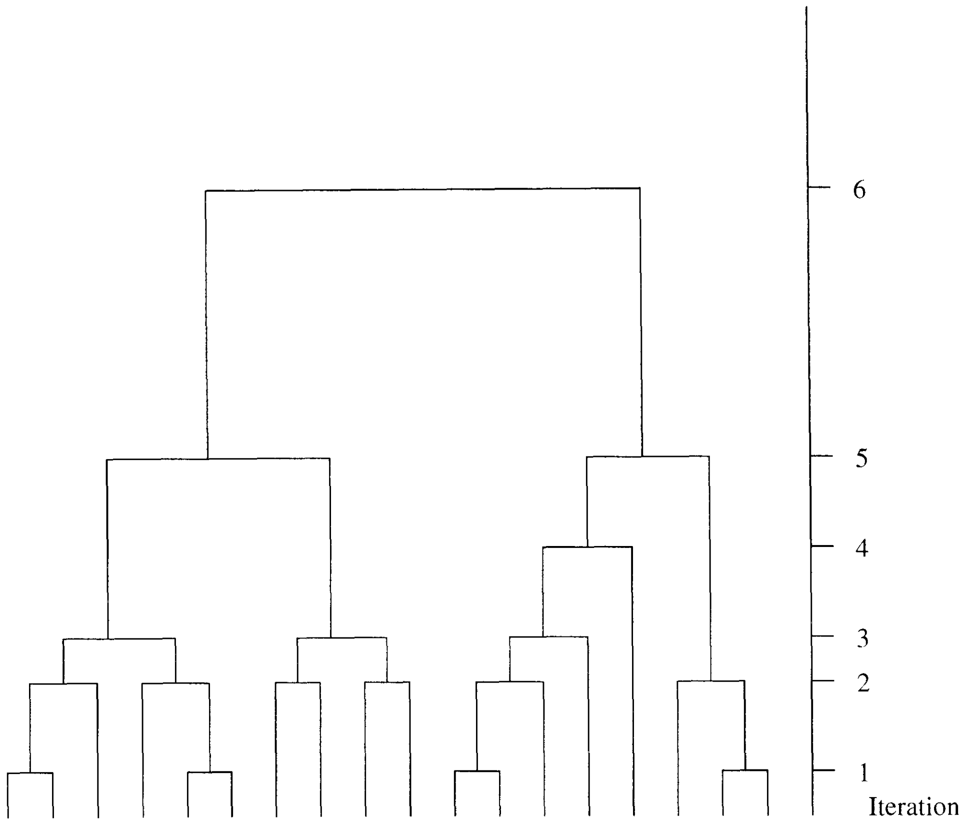


Figure 4.2: Dendrogram representing the agglomerative clustering algorithm.

4.5.1 Hard bottom-up hierarchical clustering

The hard agglomerative clustering algorithm is conceptually easy to understand and is presented to provide the intuition behind the soft clustering technique. We do not implement this algorithm in our tests. Initially, each sample is assigned to a cluster by itself. At each iteration, distance between clusters is computed and recorded. Clusters that are close to each other are merged with each other. The number of clusters that are merged at any iteration and the threshold distance are parameters input to the algorithm. The algorithm proceeds merging clusters till there is one cluster at the top level. The hierarchical cluster that results resembles a dendrogram as shown in Figure 4.2.

This algorithm will be computationally efficient if the estimation of the merged centroid at one level depends only on the centroids at the previous level and not on the data. One simple way of computing the centroid distribution of the merged clusters would be to use either the arithmetic mean or the geometric mean (based on our earlier discussions) of the component centroid distributions. This is clearly independent of the data. However, the merged centroid does not reflect the distribution of data well. For example, one of the centroids that is being merged might represent many more data points than the others. To tackle this, we employ a weighted averaging process. The relative weight that is assigned each component centroid is proportional to the number of data points it represents. For example, in Figure 4.2 at iteration 2 the left most merged cluster is made up of two component centroids. One of these centroids represents 2 data points and the other 1 data point. Therefore, we assign $\frac{2}{3}$ weighting to one centroid and $\frac{1}{3}$ to the other centroid in the estimation of the merged centroid distribution. This merged centroid now represents 3 data points. While this weighting process seems to be based on heuristics, it will be motivated in a principled manner in the soft clustering algorithm.

4.5.2 Soft bottom-up hierarchical clustering

The soft clustering algorithm is the general case of the hard clustering presented above. The clustering model is shown in (4.29). Since the quantities we are dealing with are distributions, we assume that $\mathbf{X}$ represents the underlying random variables that these distributions pertain to.

$$P(X) = \sum_{i=1}^{C^l} \pi_i^l p(\mathbf{X}|z_i^l = 1, M_l) \tag{4.29}$$

where l is the level in the hierarchy the model represents, with $l = 0$ being the coarsest level. M_l is the model at level l, C^l the number of centroids at that level and π_i^l, the corresponding priors. The variable z_i^l is an indicator that takes the value 1 iff the sample $\mathbf{X}$ is drawn from the ith component. This model is a standard likelihood model in the EM literature [9].

The basic idea is that the centroid distributions at one level l of description are completely specified in terms of the centroid distributions at the previous level, $l+1$. At each level, the collection of centroid distributions from the previous level is treated as data for estimating the new model. In addition, at each level the clustering is fuzzy. A straightforward implementation would be to draw some samples from the model at level $l+1$ and simply run the EM algorithm to estimate the new model. This would be computationally expensive and be contrary to our goal.

Let us suppose that instead of generating real samples from the cluster model at level $l+1$, we perform a Monte Carlo simulation and extract a collection of virtual samples. The idea of using virtual samples for estimating EM parameters was proposed for learning mixture parameters by Vasconcelos [63]. We use this idea for distributional clustering. As in Ref. [63], we consider a virtual sample block $\mathbf{X} = \mathbf{X_1}, \ldots, \mathbf{X_{C^{l+1}}}$ from the model at level $l+1$, M_{l+1} with each $\mathbf{X_i}$ a virtual sample block from one of the C^{l+1} clusters. Each block $\mathbf{X_i}$ is chosen to be of size $N_i = \pi_i^l N$, where N is the total number of virtual samples drawn. That is, each $\mathbf{X_i}$ is a set of vectors $\mathbf{x}_i^n$, $n \in [0, N_i]$. We further assume that these samples have been drawn independently. The likelihood of these samples under the model at level l is therefore

$$P(\mathbf{X}|M_l) = \prod_{i=1}^{C^{l+1}} P(\mathbf{X_i}|M_l) \tag{4.30}$$

The likelihood of each block is therefore

$$P(\mathbf{X_i}|M_l) = \sum_{j=1}^{C^l} \pi_j^l P(\mathbf{X_i}|z_{ij} = 1, M_l) \tag{4.31}$$

where $z_{ij} = z_i^{l+1} z_j^l$ is a binary variable with value 1 iff the samples in $\mathbf{X_i}$ are assigned to cluster j in level l. The above equation expands to

$$P(\mathbf{X_i}|M_l) = \sum_{j=1}^{C^l} \pi_j^l \prod_{n=1}^{N_i} P(\mathbf{x}_i^n|z_{ij} = 1, M_l) \tag{4.32}$$

This gives us

$$P(\mathbf{X}|M_l) = \prod_{i=1}^{C^{l+1}} \sum_{j=1}^{C^l} \pi_j^l \prod_{n=1}^{N_i} P(\mathbf{x}_i^n|z_{ij} = 1, M_l) \tag{4.33}$$

The above equation is referred to as the *incomplete* data likelihood since it does not account for the state variables z. Accounting for the indicators, we get

$$P(\mathbf{X_i}, \mathbf{Z_i}|M_l) = \prod_{j=1, z_{ij}=1}^{C^l} \pi_j^l \prod_{n=1}^{N_i} P(\mathbf{x}_i^n | z_{ij} = 1, M_l) \tag{4.34}$$

where $\mathbf{Z_i}$ is the set of indicators z_{ij} for a given model i. The product form results from the observation that within a block $\mathbf{X_i}$, each sample $\mathbf{x}_i^n$ is drawn independently. The above expression can be written compactly as

$$P(\mathbf{X_i}, \mathbf{Z_i}|M_l) = \prod_{j=1}^{C^l} [\pi_j^l \prod_{n=1}^{N_i} P(\mathbf{x}_i^n | z_{ij} = 1, M_l)]^{z_{ij}} \tag{4.35}$$

Simplifying and combining terms, we get the *complete* likelihood as

$$P(\mathbf{X}, \mathbf{Z}|M_l) = \prod_{i=1}^{C^{l+1}} \prod_{j=1}^{C^l} [\pi_j^l P(\mathbf{X_i} | z_{ij} = 1, M_l)]^{z_{ij}} \tag{4.36}$$

where $\mathbf{Z}$ is the collection of all vectors z_{ij}. Taking logarithms we get,

$$\log P(\mathbf{X}, \mathbf{Z}|M_l) = \sum_{i=1}^{C^{l+1}} \sum_{j=1}^{C^l} z_{ij} \log(\pi_j^l P(\mathbf{X_i} | z_{ij} = 1, M_l)) \tag{4.37}$$

From EM, we get the following E-step:

$$h_{ij} = E[z_{ij}|\mathbf{X_i}, M_l] = P(z_{ij} = 1|\mathbf{X_i}, M_l) = \frac{\pi_j^l P(\mathbf{X_i}|z_{ij} = 1, M_l)}{\sum_k \pi_k^l P(\mathbf{X_i}|z_{ik} = 1, M_l)} \tag{4.38}$$

The steps from generating the virtual sample to the above are standard EM equations and can be found in any reference to EM [9]. In the next few steps, we relate the two levels using Theorem A.1 stated in the appendix. $P(\mathbf{X_i}|z_{ij} = 1, M_l)$ can be expressed as

$$P(\mathbf{X_i}|z_{ij} = 1, M_l) = \prod_{n=1}^{N_i} \log P(\mathbf{x_i^n}|z_{ij} = 1, M_l) \tag{4.39}$$

Now, the sample $\mathbf{x_i^n}$ is derived from the ith centroid distribution at level $l+1$. The likelihood of observing N_i such samples under the jth centroid distribution at level l is given by Theorem A.1 as

$$2^{-N_i(H(p_i^{l+1})+D(p_i^{l+1}||p_j^l))} \tag{4.40}$$

where p_j^l represents the jth centroid distribution at level l. And, $D(p||q)$ is the KL divergence between distributions p and q. Assuming the units of entropy to be *nats*, we get

$$e^{-N_i(H(p_i^{l+1})+D(p_i^{l+1}||p_j^l))} \tag{4.41}$$

Using (4.41) in (4.38) and cancelling common terms, we get

$$h_{ij} = \frac{\pi_j^l e^{-N_i D(p_i^{l+1}||p_j^l)}}{\sum_k \pi_k^l e^{-N_i D(p_i^{l+1}||p_k^l)}} \tag{4.42}$$

We see that this equation is independent of the virtual samples and specifes the relationship between two levels purely in terms of the model parameters. Therefore, we have a computationally efficient algorithm for clustering.

Carrying out the M-step of this thought experiment, we get the following relationships. As is usual in EM, the M-step consists of maximizing

$$Q = \sum_{i=1}^{C^{l+1}} \sum_{j=1}^{C^l} h_{ij} \log(\pi_j^l P(\mathbf{X_i}|z_{ij} = 1, M_l)) \tag{4.43}$$

with the constraint that $\sum_j \pi_j^l = 1$ and the distributions p_j^l be valid. We have used the log-likelihood instead of the likelihood for the sake of ease of manipulation. Both log-likelihood and the likelihood have the same extremum points and therefore this substitution is valid. We separate the terms of Q as

$$Q = \sum_{i=1}^{C^{l+1}} \sum_{j=1}^{C^l} h_{ij} \log \pi_j^l + \sum_{i=1}^{C^{l+1}} \sum_{j=1}^{C^l} h_{ij} \log P(\mathbf{X_i}|z_{ij} = 1, M_l)) \tag{4.44}$$

The first term of (4.44) depends only on the priors π_j^l and the second term depends only on the centroid distributions p_j^l. Differentiating Q with respect to π_j^l, we get

$$\frac{\partial}{\partial \pi_j^l} Q = \sum_{i=1}^{C^{l+1}} \frac{h_{ij}}{\pi_j^l} = 0 \tag{4.45}$$

which gives

$$\pi_j^l = \frac{\sum_i h_{ij}}{C^{l+1}} \tag{4.46}$$

Simplifying the second term Q' of (4.44) before differentiating with respect to the bins of the distribution p_j^l, we get

$$Q' = \sum_{i=1}^{C^{l+1}} \sum_{j=1}^{C^l} (h_{ij} [\sum_{n=1}^{N_i} \log P(\mathbf{x_i^n} | z_{ij} = 1, M_l)]) \tag{4.47}$$

From Theorem A.1, we get

$$Q' = \sum_{i=1}^{C^{l+1}} \sum_{j=1}^{C^l} (h_{ij} [\sum_{n=1}^{N_i} (-H(p_i^{l+1}) - D(p_i^{l+1} || p_j^l))]) \tag{4.48}$$

$$Q' = \sum_{i=1}^{C^{l+1}} \sum_{j=1}^{C^l} (h_{ij} N_i (-H(p_i^{l+1}) - D(p_i^{l+1} || p_j^l))) \tag{4.49}$$

$$Q' = \sum_{i=1}^{C^{l+1}} \sum_{j=1}^{C^l} (h_{ij} N_i [\sum_k (p_i^{l+1}(k) \log p_i^{l+1}(k) + p_i^{l+1}(k) \log \frac{p_j^l(k)}{p_i^{l+1}(k)})] \tag{4.50}$$

$$Q' = \sum_{i=1}^{C^{l+1}} \sum_{j=1}^{C^l} (h_{ij} N_i [\sum_k (p_i^{l+1}(k) \log p_j^l(k))] \tag{4.51}$$

Differentiating (4.51) with respect to the bins of the centroid distribution p_j^l, we get

$$\frac{\partial}{\partial p_j^l(k)} Q' = \sum_i^{C^{l+1}} h_{ij} N_i \frac{p_i^{l+1}(k)}{p_j^l(k)} \tag{4.52}$$

Which gives us the following update equation, considering the Lagrange constraint.

$$p_j^l(k) = \frac{\sum_i h_{ij} N_i p_i^{l+1}(k)}{\sum_i h_{ij} N_i} \tag{4.53}$$

Which is very closely related to the update equation (4.24) in the top-down soft clustering algorithm. Instead of a simple weighted average, we now have the additional weighting factor that is dependent on the number of virtual samples that we create. Notice that none of the update equations that we derived relate to the samples $\mathbf{X}$ directly. The relationships are expressed in terms of the different level models. This implies a very efficient implementation. The quantity N_i plays the same role as β in the top-down algorithm. As we increase N (and therefore N_i), we increase the weighting on a particular configuration of clusters. The cluster boundaries get harder with increasing N. A natural choice from the algorithmic perspective is to start with a very high N and decrease it as the simulation progresses to allow clusters to merge.

One special point to note is that the KL divergence came out as a natural quantity in this algorithm. As opposed to the earlier two cases where we had a choice of similarity measures, the setting up of the problem resulted in the KL divergence (and that too in one particular form) as the similarity measure between two centroid distributions. In (4.53), the weighting for each component centroid distribution depends on the prior of each distribution (i.e, how many samples it represents) and the distance from the centroid being estimated (distance in the KL sense). The hard clustering algorithm is a special case of this algorithm. In the hard clustering, the second quantity is either 1 or 0 as we have boolean membership as opposed to soft memberships. We see that the weighting scheme we chose for the hard hierarchical clustering becomes justified since in that algorithm, each centroid was weighted proportional to the number of samples it represented. We note here that the hard clustering as we presented is not exactly a special case of this algorithm since we relax the constraint on KL divergence and permit both forms.

4.6 Clustering experiments

We begin by detailing the three different algorithms that result from the analysis above. Since two of the algorithms proposed are soft assignment algorithms, they cannot be directly compared with the hard clustering algorithm. In order to make these comparisons, we will derive hard clusters from these soft algorithms as well. That is, in the soft clustering case, the algorithms will be run in soft mode and be frozen into the hard mode once the chosen criterion is met. The criterion we choose is that all algorithms must give us the same number of clusters. By this, we are evaluating the ability of each of these algorithms to choose a good local minima at a given configuration (or *order*, as is customary in the clustering literature). Since each algorithm is designed to seek a local minima, this criterion seems a reasonable one. We use the hard partioning algorithm as our control algorithm since the rest of the algorithms can be stopped whenever the desired number of clusters are reached. As mentioned earlier, this hard partioning algorithm is modeled after the Isodata procedure to select the number of clusters in the data. In addition, to evaluate if the Isodata procedure results in better clustering, we also run the hard algorithm without the split and merge heuristics. That is, we pre-specify a number of clusters and perform the K-means iteration alone till it converges. For the Arithmetic mean case, we compare all four algorithms. For the Geometric mean case, the two soft algorithms are not used.

4.6.1 Clustering Algorithms

Algorithm 4.1 Hard partitioning algorithm

Define (and specify) the following parameters:

T: Minimum number of samples in a cluster.

N_d: Desired number of clusters.

s_m: Maximum spread parameter for splitting a cluster.

D_m: Maximum distance for merging clusters.

N_m: Maximum number of clusters that can be merged.

n_r: Maximum number of discarded samples to be reintroduced into a simulation.

1. *Initialize N_d centroids by assigning one sample to each centroid randomly. Set $N_c = N_d$ where N_c is the current number of clusters.*

2. *Compute the mean centroid distribution from all the samples in the dataset. Compute the* mean spread *as the average of the squared KL divergence between each sample and the mean centroid. See (4.54).*

3. *Assign each one of the valid samples to its closest centroid. The closeness is computed using KL divergence. If assignment does not change, exit.*

4. *Eliminate clusters with less than T members. Decrease N_c accordingly. Move the samples from these clusters into scratch buffer. Samples in the scratch buffer will not be used in the simulation.*

5. *Update the centroids using the update equations (for the chosen form of KL divergence) in Section 4.3.*

6. *Compute the* spread *for each centroid. If the iteration is either odd and $N_c < 2N_d$ or if $N_c \leq \frac{N_d}{2}$ then*

   (a) *split clusters that have* spreads *greater than s_m and increase N_c.*

   (b) *If any clusters have been split, goto step 3.*

7. *Merge any pair of clusters (until a max of N_m are merged) that have average centroid distance less than D_m.*

8. *If scratch buffer is non-empty, reintroduce maximum of n_r samples into the simulation. The closest n_r samples to the current centroids are chosen from the scratch buffer. Go to step 3.*

The heuristic behind splitting and merging clusters requires a "measure" to estimate the *largeness* and *closeness* of clusters. For *closeness* of clusters, we can use the symmetric KL divergence (average of the two KL divergences) between the cluster centroids. To estimate the *largeness*, we define a *mean spread* term as

$$S(a_i||a) = \frac{1}{N}\sum_{i=1}^{N} KL(a_i||a)^2 \tag{4.54}$$

where a is the mean centroid distribution and a_i are the sample distributions. This quantity plays the same role as the cluster variance in the conventional ISODATA algorithm.

The purpose of discarding and reintroducing data samples is to control the effect of outliers. The procedure of discarding small clusters will hopefully eliminate samples that are outliers. However, it will also discard good data samples. The purpose of selectively reintroducing samples based on proximity to centroids will possibly retrieve these good data samples.

Algorithm 4.2 Soft top-down hierarchical clustering algorithm

1. *Initialize β, the inverse temperature, to zero. Compute the all sample mean centroid distribution. Set parameter n, the number of copies of a centroid made in every iteration. We found that $2 < n < 4$ works well. Larger n implies additional computation at each iteration, for the extra centroid copies.*

2. *Copy each centroid n times, randomly perturb each of these copies by a small amount. Change β according to the update schedule.*

3. *For each sample, compute the responsibilities of each centroid. See Section 4.4 for equations.*

4. *Update each centroid accordingly. If any centroid is a copy of another, discard the copies and retain only one such centroid. Repeat steps 3 and4 till no change in centroids.*

5. *Repeat step 2 until β goes to ∞ or desired number of clusters reached or each sample assigned to a cluster by itself. Hard modification for experiments – When desired number of clusters is reached, set β to ∞ and freeze the simulation.*

Algorithm 4.3 Soft agglomerative clustering algorithm

1. *Initialize by assigning each sample to a cluster by itself. Set N, the number of virtual samples, to a large number. Set l, the number of levels, to a large number.*

2. *Decrement N per update schedule. Make copies of centroids at level l and perturb them by a small amount. Treat these copies as level $l-1$ centroids.*

3. *Compute the M-step and E-step as in Section 4.5.2. Repeat step 3 till convergence.*

4. *Some centroids would converge to the same points. Eliminate all but one such centroids. Decrease the number of centroids accordingly. If desired number of centroids reached, stop. Or if all centroids merged into one, stop. Hard extension - when desired number of centroids reached, freeze the simulation by setting N to a very large number.*

4.6.2 Performance evaluation

We used a database of 972 images from the Corel database for our experiments. These images are RGB color images and a few of them are shown in Figure 4.3. Prior to computing the histogram, we converted the images from RGB space to Ohta [39] space shown below. The color axes of this space are the 3 largest eigenvectors of the RGB space, found via principal components analysis of a large selection of natural images.

$$\begin{aligned} I1 &= R + G + B \\ I2 &= R - B \\ I3 &= R - 2G + B \end{aligned} \tag{4.55}$$

The Ohta space approximately decorrelates the color channels which makes it a good choice

Figure 4.3: Sample images in the test database.

for computing individual channel histograms. We compute 3 color histograms, one for each channel, for each image in our database. These histograms were then smoothed with a gaussian kernel to ensure that images of similar but slightly different colors will have similar histograms. Additionally, this process also ensures that none of the bins in the histogram have zero values. This is important for the clustering using the Geometric mean criterion.

The experiments that we perform are two-stage query experiments. That is, the test image is first queried with respect to the cluster centroids using KL divergence as the ranking criterion. The closest cluster that is returned in this first step is now considered as the database and the images within cluster are ranked using KL divergence. In the tables, Rank 1 implies that the correct match came up as the first match in this 2-stage retrieval. Rank 5 implies that the correct match appeared within the top 5 matches and similarly Rank 20 implies that the correct match appeared in the top 20 matches. In addition, to provide a baseline, the same query experiment is performed with the entire database.

It is easy to show that using KL divergence for retrieval is equivalent to performing maximum-likelihood. Assume that q is the query image density and p is the density of an image in the database. We define the ranking criterion as

$$D(q||p) = \sum q \log(\frac{q}{p})$$

$$= \sum q \log q - \sum q \log p \tag{4.56}$$

The first term is a constant with respect to the query. The second term is estimating the log-likelihood of the database model p under the distribution q. Minimum $D(q||p)$ therefore implies maximum $\sum q \log p$. This possibly explains the good performance of KL divergence as a similarity measure.

In each set of experiments, the query image is subject to a different distortion prior to query. The distortions that we used are: scaling, sub image, rotation, contrast shift, gaussian noise addition and random pixel distortion. In the scaling experiment, the query image is randomly scaled either 0.5 or 2.0 times the original image dimensions. For subimage experiments, the top $\frac{1}{4}$ of the image is used to perform the query. In rotation experiments, an affine warp of 45 degrees is applied to the query images. For contrast shift, the dynamic range of the pixels is reduced to 75% of the original. In gaussian noise addition, the query image is degraded with gaussian noise such that the Signal-to-noise ratio (SNR) between the query and the original is between 17 and 20 dB. For random pixel distortion, approximately 15% of the pixels in the query image are randomly changed. This results in the SNR between the original and query image to be between 14 and 16 dB. Example distortions are presented in Figures 4.4 and 4.5.

To select the number of clusters, as we mentioned earlier, the Isodata variation of the hard clustering algorithm is used as the controlling algorithm. The chosen number of clusters is then input to the other algorithms and they are iterated till the desired number of clusters are obtained. In addition, the two soft clustering algorithms are frozen to give hard decision boundaries. For the arithmetic mean case we obtained 9 clusters, and for the geometric mean case we obtained 10 clusters using the Isodata variation.

Table 4.1 tabulates the 2-stage query experiments outlined above for the case of scaled images. The first row in each table indicates the performance of the retrieval on the entire

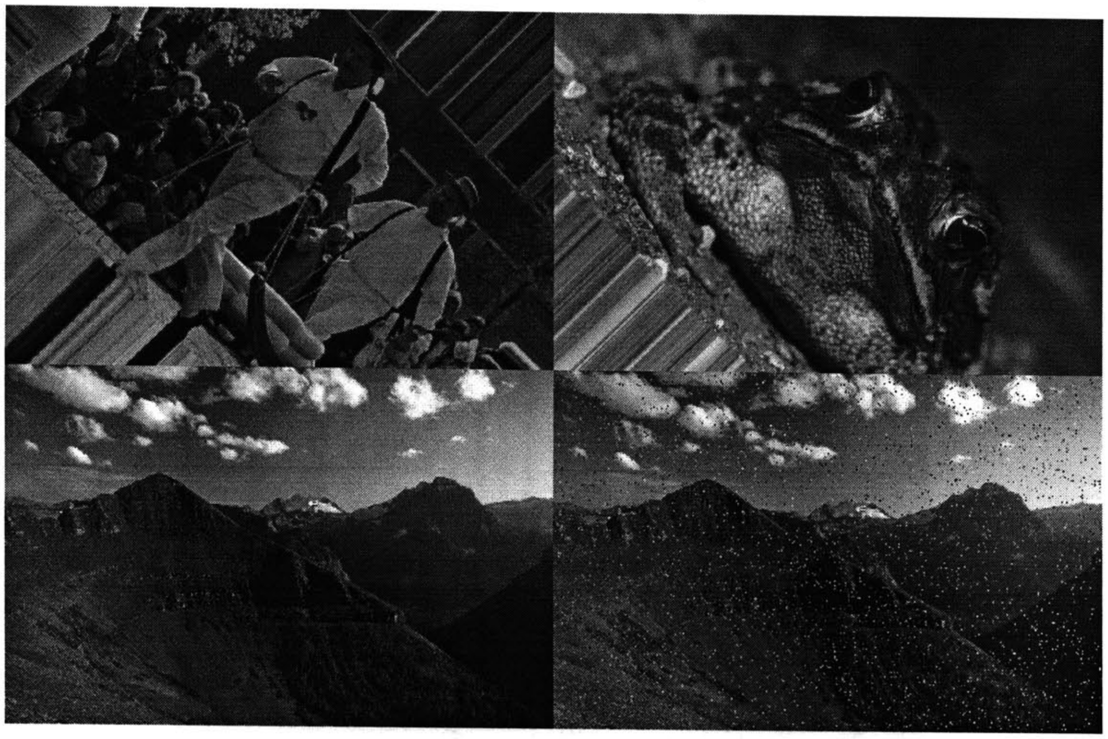


Figure 4.4: Sample Rotated and Noisy Images. The Top 2 Images are Rotated by 45 degrees. Bottom Right Image is the Noisy Version of the Bottom Left Image.

database without any clustering. In the tables, AM refers to arithmetic mean and GM to geometric mean. Similarly, TD refers to top-down and BU refers to the agglomerative algorithm. We perform 100 queries per experiment. Table 4.2 represents the query experiments where the image was rotated by 45 degrees. The corresponding ROC plots are shown in Figs. 4.6 and 4.7.

Since color histograms are scale-invariant and rotation-invariant, these results are expected. The changes in the histogram that result by scaling and rotation are mostly numerical precision errors and by forcing a rectangular aspect on the images (in the case of rotated queries). It is interesting to note that there is considerable difference between the performance of the Hard AM and Isodata AM algorithms. This seems to imply that the heuristics of preventing both very large and very small clusters in the Isodata algorithm helps by finding a better

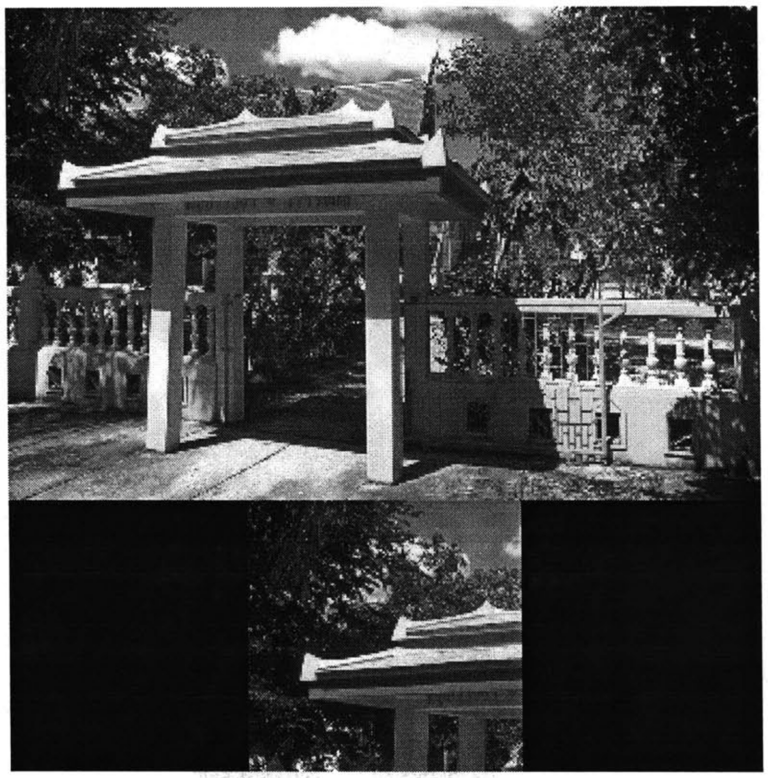


Figure 4.5: Sample database image and its 25% sub-image used for query.

representative approximation to the data.

Table 4.3 presents the results for subimage query. In this experiment, the top left 25% of every image is presented as the query and a correct match implies retrieval of the full image to which the subimage belonged. Table 4.4 tabulates the results of the random pixel distortion experiments. Compared to the first two experiments, in these two cases there is significant change to the histogram of the query image. The corresponding ROC plots are shown in Figs. 4.8 and 4.9.

In both the experiments, there is remarkable difference in performance between the hard versions and their Isodata variants. Both the soft algorithms perform the best. In the pixel distortion experiment, the geometric mean algorithms perform better whereas in the sub image query, the arithmetic mean algorithms perform better.

Algorithm	Rank 1	Rank 5	Rank 20
Full	100	100	100
Hard AM	96	96	96
Hard GM	100	100	100
Isodata AM	100	100	100
Isodata GM	100	100	100
Soft TD	100	100	100
Soft BU	100	100	100

Table 4.1: Cluster experiments for scaled query image

Algorithm	Rank 1	Rank 5	Rank 20
Full	100	100	100
Hard AM	84	84	84
Hard GM	98	98	98
Isodata AM	99	99	99
Isodata GM	99	99	99
Soft TD	99	99	99
Soft BU	99	99	99

Table 4.2: Cluster experiments for query image rotated by 45 degrees

In the next two experiments, we distort the query images by a large extent. For the contrast and illumination change experiment, we reduce the dynamic range of the pixels to 75% of the original and for the gaussian noise experiment, we add a gaussian distributed random number to every pixel in the image. Table 4.5 shows the results of the contrast change experiment. We note here that we used the Ohta color space and did not normalize the features to make them illumination invariant as in [16]. We therefore expect a considerable degradation in retrieval accuracy. Table 4.6 shows the results for the gaussian noise experiment. The corresponding ROC plots are shown in Figs. 4.10 and 4.11.

We find that compared to using the entire database, there is a small drop in performance when limiting the search to only the nearest cluster. However, the time savings of a cluster based query are significant. Each cluster based query requires approximately $\frac{1}{N_c}\%$ of the computation time, where N_c is the number of clusters.

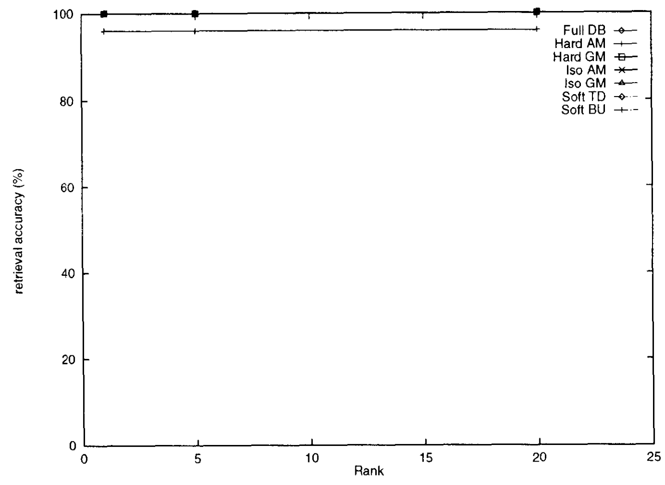


Figure 4.6: ROC plot for scaled query. All but the Hard AM plot are co-located at the 100% line.

Algorithm	Rank 1	Rank 5	Rank 20
Full	88	100	100
Hard AM	77	81	88
Hard GM	75	84	92
Isodata AM	81	94	97
Isodata GM	82	91	95
Soft TD	85	94	97
Soft BU	84	94	97

Table 4.3: Cluster experiments for sub image query

We also find that the arithmetic mean versions are more robust than the geometric mean versions under a wider variety of distortions. Geometric mean performs better under smaller distortions to the query whereas the arithmetic mean clustering is robust over a wider range of distortions. Interestingly, the percentage of correct retrievals in Rank 1 is higher for the clustered retrieval compared with normal retrieval in Tables 4.5 and 4.6. This is possibly because of fewer distractors in the clustered cases and therefore less confusion. However, for Rank 20 performance, the full database query always performs better. This implies that if the correct cluster is identified at the first stage, then having fewer distractors results in a better rank. However, there are a number of instances where the correct clusters are not

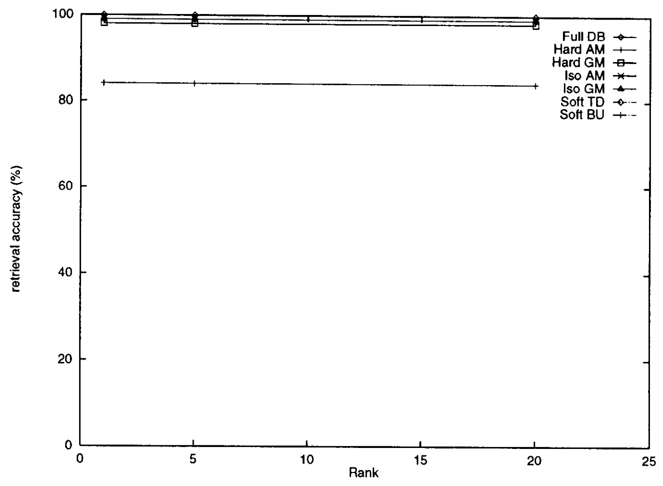


Figure 4.7: ROC plot for rotated query. All but Hard AM are near the top.

Algorithm	Rank 1	Rank 5	Rank 20
Full	87	98	100
Hard AM	71	81	81
Hard GM	86	97	98
Isodata AM	79	89	89
Isodata GM	87	97	98
Soft TD	82	90	93
Soft BU	80	89	93

Table 4.4: Cluster experiments for 15% pixel distortion

identified in the first stage and therefore we do not get similar improvements in Rank 20. To a smaller degree, this effect can be observed in Rank 5 performance.

Both the soft algorithms performed better than the Isodata variant. However, both these algorithms are significantly more computationally complex than the Isodata variant. Using soft algorithms to initialize a hard clustering as we did for these experiments is perhaps not the best way to use these algorithms. While such an approach improves the hard clustering, the significant additional computation may not justify the improvement in performance. If we use the soft clusterings without freezing, there might be additional improvement in performance. However, it does not result in computational savings during retrieval since in

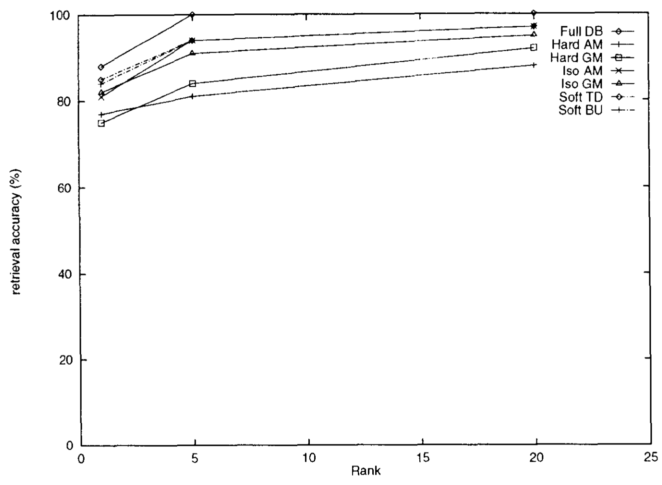


Figure 4.8: ROC plot for subimage query.

Algorithm	Rank 1	Rank 5	Rank 20
Full	51	78	90
Hard AM	49	72	80
Hard GM	50	67	79
Isodata AM	54	77	87
Isodata GM	55	76	83
Soft TD	60	78	88
Soft BU	58	78	87

Table 4.5: Cluster experiments for contrast change query

soft clustering every sample belongs to every cluster (with only the degree of belongingness changing). Soft clusterings as such only influence the order of search and do not restrict the search set. This is a good property to have – e.g., if the user is willing to wait longer (but not as long as it takes to search the entire database), these soft clusterings can be used to search the database in the order of decreasing likelihood, thereby reducing the expected time retrieval. A possible way to achieve the computational savings with soft clustering is to limit the search set to a fixed percentage of the database images. Since each cluster in the soft algorithms implies a different search order, this thresholding strategy might offer the computational savings with possibly higher retrieval accuracy.

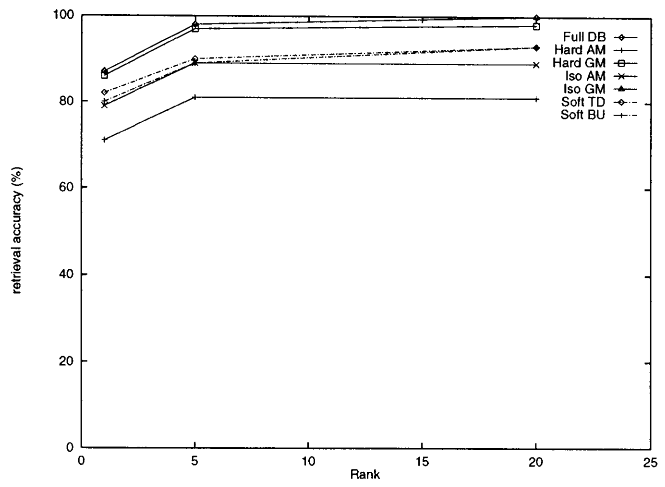


Figure 4.9: ROC plot for 15% pixel distortion query.

Algorithm	**Rank 1**	**Rank 5**	**Rank 20**
Full	66	72	88
Hard AM	52	62	70
Hard GM	49	56	68
Isodata AM	66	73	83
Isodata GM	55	63	76
Soft TD	68	77	85
Soft BU	67	75	84

Table 4.6: Cluster experiments for query image distorted with gaussian noise

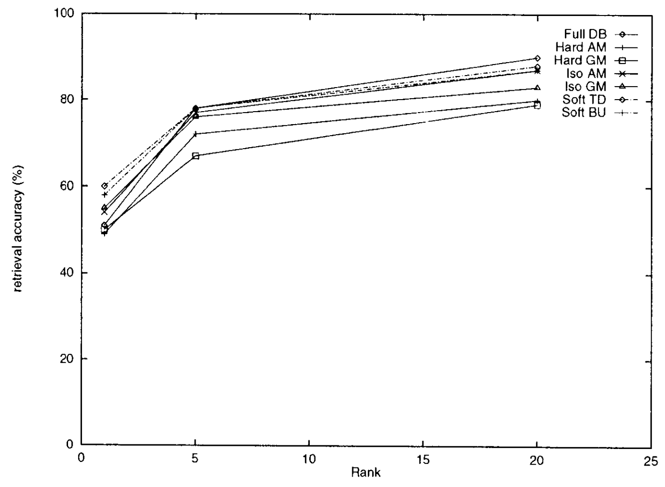


Figure 4.10: ROC plot for contrast change query.

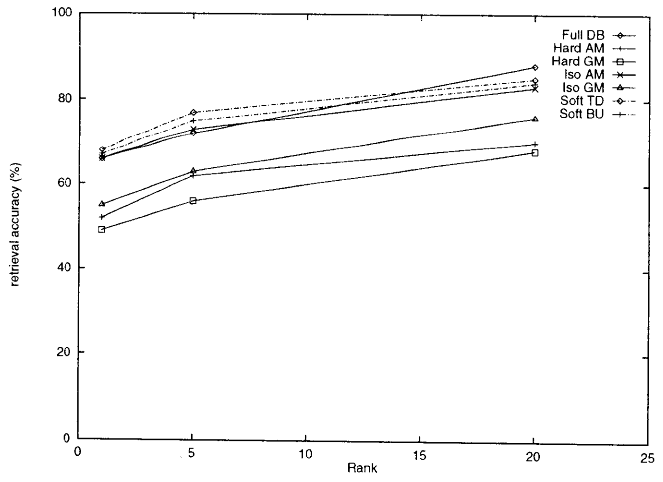


Figure 4.11: ROC plot for additive gaussian noise query.

Chapter 5

Browser and Editors for Unstructured Video

We now present the application that takes a collection of unstructured video shots and shows clusters based on the algorithms presented above. These groupings are the launching pad for the construction process by the user. As we discussed earlier, these groupings are also an alternative to the query-by-example paradigm of navigating a video database.

Within this application, the user is presented with a choice of groupings and a set of editors that enable interesting constructions from video. Specifically, we have built three different editors that work with the cluster browser. The still editor enables extraction of a still image from a video sequence for embedding into postcards, for example. The collage editor can be used to construct storyboards, photomosaics, etc. The movie editor can be used in two modes – in the default mode where the user selects all the shots and arranges the movie; or given a time budget, the movie editor picks up shots from the clusters and suggests a movie. In both modes, the user has complete control over the final selection and ordering of shots within the movie.

5.1 Representation of unstructured video

We assume that the unstructured video collection (from a home camcorder, for example) is parsed into a collection of shots before the clustering process. This parsing into shots can be accomplished using a combination of shot detection algorithms outlined in chapter 2, for example. Once the shots are suitably extracted, a set of relevant features are extracted from these shots. In this thesis, we have experimented with colorimetry, motion and texture and when structural assumptions can be made, we have used sophisticated models and features as seen in chapter 3. We note that the clustering algorithms and the application are both independent of the features extracted. The clustering algorithms require a discrete density representation of the features in order to cluster the shots.

For density representations of video, we use color and motion in our experiments. For example, density representations can be either histograms, correlograms, coherency vectors etc. We also use Vector Quantization to create a universal codebook of the collected shots and create a codebook-use histogram for each shot. This is yet another density representation of the underlying features. With video sequences, we find the Vector Quantization based density representation particularly useful. Since the underlying points are blocks of pixels, this approach offers some respite from the pure lack of local structure in the histogram and offers advantages similar to that of a color coherency vector. The local structure information gets embodied in the blocks that are used to construct the codebook. In addition, a codebook based approach also provides a model for the particular region of space that these video shots occupy and concentrates the histogram bins in this region. Thus, it also permits greater representational power than with a simple color histogram which quantizes the color space uniformly. In addition, Vector Quantization is both useful for archiving and for feature representation. Another possible feature representation that we are currently exploring is to use the DCT coefficients and the motion vectors that are available from the MPEG and JPEG streams. The advantage of this approach is that it makes use of the information already available from the encoding process of the current video and image archiving and transmission standards.

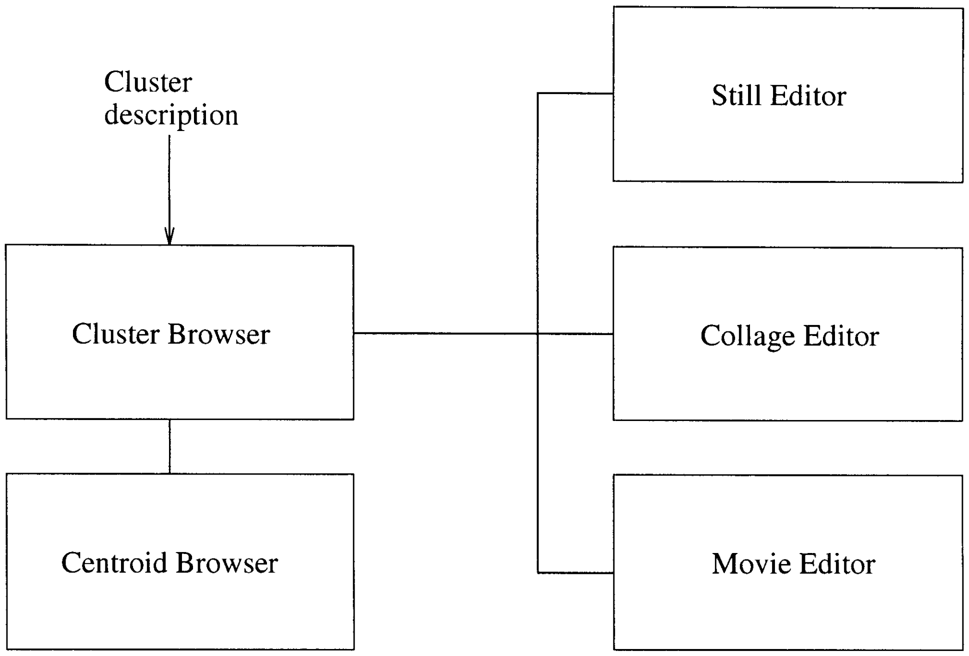


Figure 5.1: Architecture of the browser/editor application

We note here that the browsers and the editors are independent of both the features and the representations used to characterize the video shots. In addition, they are also independent of the clustering techniques used to group the shots. This independent visualization provides the flexibility to explore a variety of representations and features.

5.2 Architecture of the System

Figure 5.1 shows the architecture of the browser/editors for unstructured video. The application is visualized as a standalone module that takes as input a cluster description and a database of shots. This enables the application to be useful with a wide variety of clustering techniques. In addition, it also can be used as a tool to visualize and evaluate different clustering techniques. It comprises of 5 modules as shown in Figure 5.1.

5.2.1 Cluster Browser

The cluster browser module is the central module of the system. It is the primary interface of the system to the external world. That is, it mediates all user interactions and also exchanges information with the environment regarding the clusters, shots, files etc. It serves as the launching pad for the centroid browser and the different editors. It also serves as the conduit for messages passed between the different modules. The cluster browser reads in the cluster descriptions (shot membership of each cluster) and shot descriptions (physical location of shots, timing information etc) and uses this information to orchestrate the tasks of the different editors and browsers.

Figure 5.2 shows the cluster browser interface. In the top image display area, one image per centroid is shown. The image that is shown is the first frame of the top ranking shot for each centroid. The selected feature for which the centroids are shown is highlighted in the information panel below. Via this panel, the feature set for which the clusters are shown can be changed. For the selected feature set, the image display area shows one image per cluster. For example, in Figure 5.2, there are 6 images indicating 6 centroids for the chosen feature. Figure 5.3 shows a set of clusters for a typical home video collection.

In addition, the cluster browser permits a hierarchical organization of clusters, in which case, clusters at a particular level are shown in the image display area. It is hoped that this interface provides the user with an idea of the database in contrast with the views offered by a query-by-example system which presents a collection of randomly chosen shots. For example, assume that the image database consists of pictures of postage stamps. If the user is browsing for a stamp of a particular color, in the query-by-example system it is not immediately apparent whether stamps of that color exist or not. However, if there are sufficient stamps of that color in the database, a clustering based on color will reveal that by positioning a centroid around that color. The cluster browser thus would be more revealing to the user. The cluster browser thus offers a solution to the page-zero representation problem (see chapter 1 and [6]).

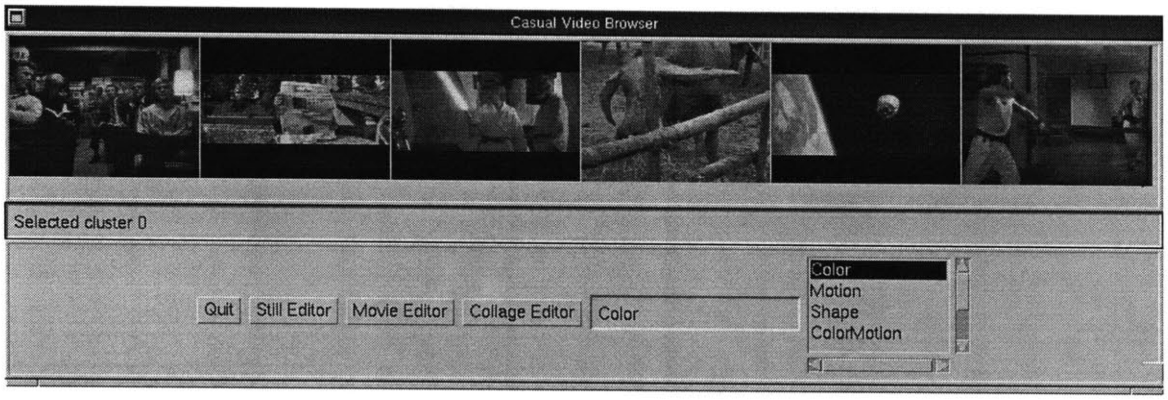


Figure 5.2: Cluster browser view of movie shots database

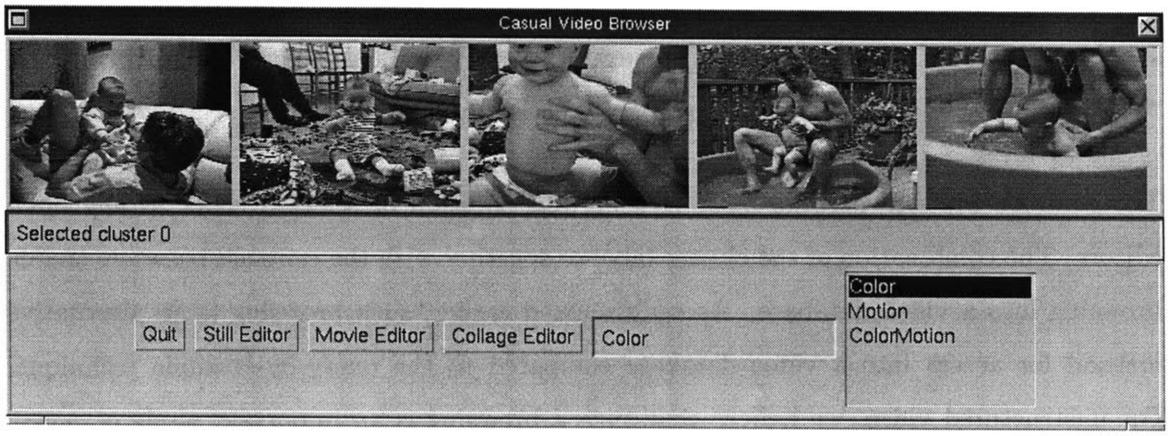


Figure 5.3: Cluster browser view of home video database

5.2.2 Centroid Browser

In order to view the contents of a particular centroid, a separate browser for each centroid is launched upon user request. One such browser is shown in Figure 5.4. This viewer shows all the shots that belong to a centroid. These shots are arranged in decreasing order of proximity to the centroid. The shot in the top-left corner is the closest to the centroid and the bottom-right shot is the farthest away from the centroid. One of the advantages of clustering is that it relies on the cognitive ability of the user to bridge between the shots of

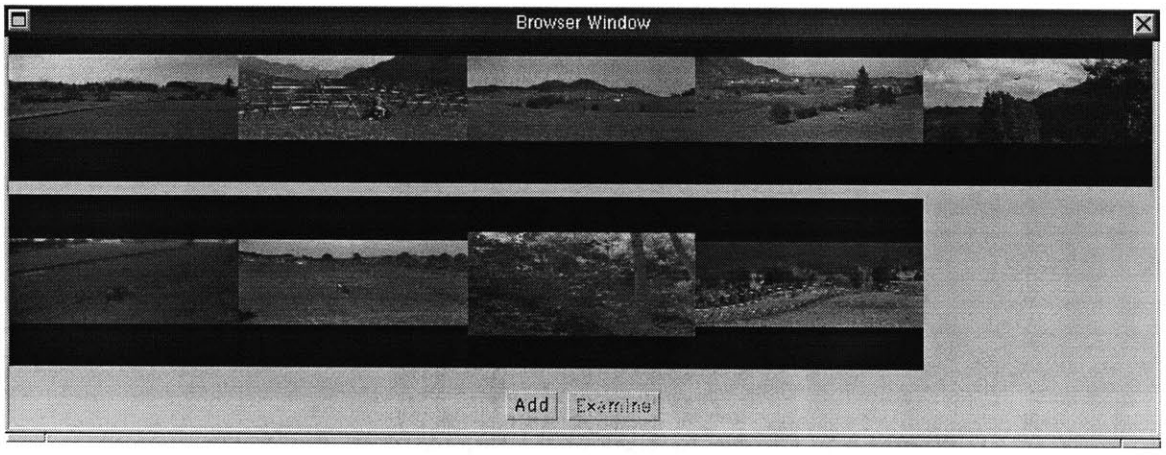


Figure 5.4: Centroid browser showing some shots in the "field" cluster

a grouping and attach a label to it. For example, in Figure 5.4, a possible label could be "fields". Similarly, a possible label for Figure 5.5 is "living room".

The centroid browser permits both viewing and addition of selected shots to any open editors. The combination of the cluster browser together with the centroid browsers enable browsing into a video database. As we discussed earlier, such browsing is an alternative method for access into a video database compared to the query-by-example technique. For unstructured video, we believe that such a browsing is an important mode of access as the goal of the system is to provide views into the database and enable serendipitous constructions.

5.2.3 Still editor module

The still editor permits selection of single snapshots from the clustered shots. As described above, shots are selected in the centroid browser and added to the editor. To aid the selection of individual frames within a shot, the still editor provides tools to manipulate the shot (play forward, reverse, freeze frame etc). In addition to extracting single frames from shots, the still editor provides an interface to the Salient Stills algorithm by Teodosio and Bender[59].

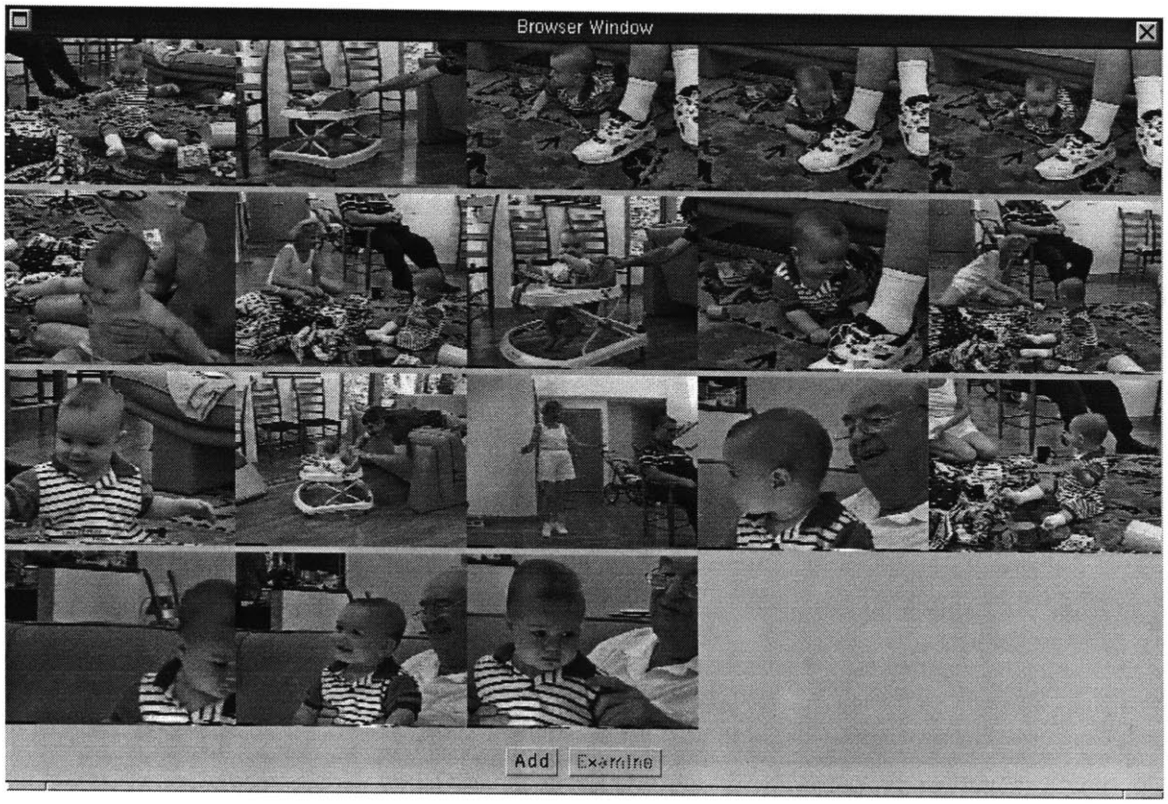


Figure 5.5: Centroid browser showing some shots in the "living room" cluster

Briefly, Salient Stills is a static representation of a video shot. It aligns individual frames with respect to a common reference by estimating their motion parameters and registering the individual images in a shot with respect to a common reference coordinate system. Once these individual frames are registered with respect to the common reference, a static composite can be created. For another example of such mosaicing, see Pope et al[45].

5.2.4 Collage editor module

The collage editor extends the still editor by permitting extraction of a collection of single frames from multiple shots. These can be laid out as a storyboard or can be used to make collages similar to Photomosaics[52, 17]. In our application, we use the Kullback-

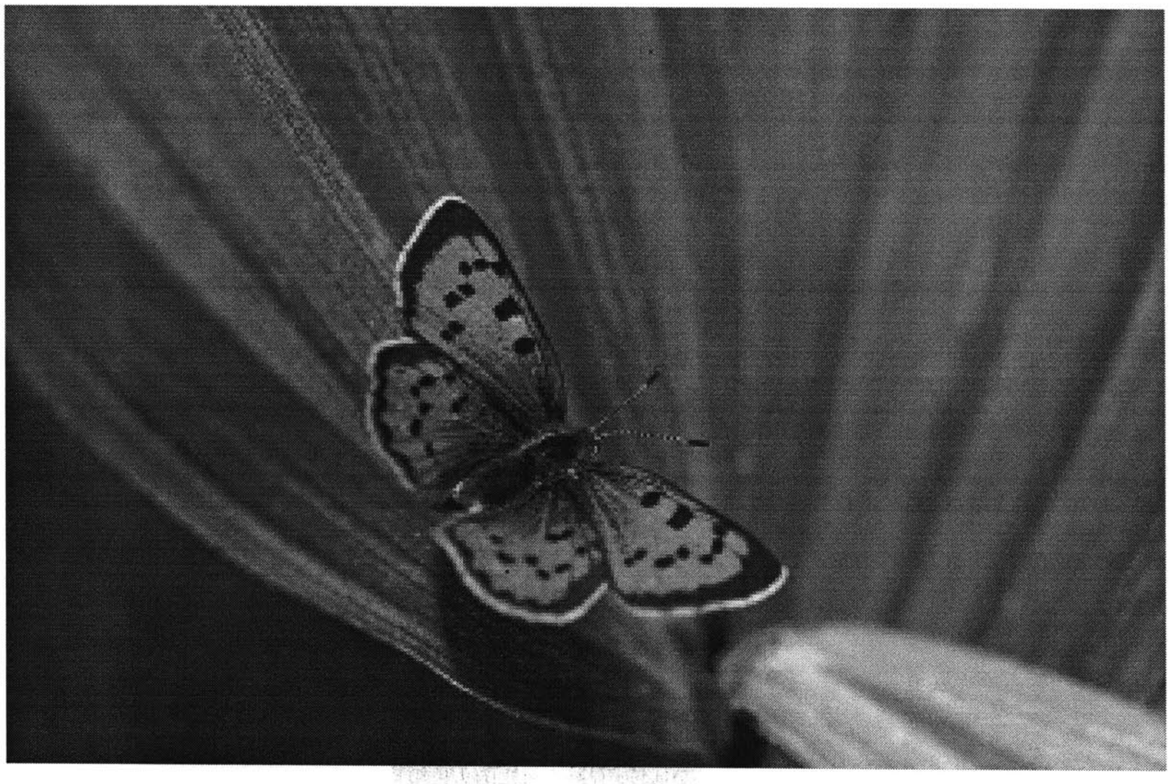


Figure 5.6: Original still picture of butterfly

Liebler divergence between the histograms of the target frame region and a database of still images to select the mosaic images. While Photomosaic is also an artistic process with an artist-in-the-loop to refine the mosaic selection and target image creation, we view this process of KL collages as a first step in selection of images. These images can be refined further with a human-in-the-loop performing a query-by-example and replacing images with more appropriate ones. Figures 5.6 and 5.8 show two original images. Their respective photomosaics generated by the automated KL process are shown in Figures 5.7 and 5.9.

5.2.5 Movie editor module

The movie editor module has two modes of operation. In the default mode, the selection process is completely under user control as in the cases of still and collage editors. Once

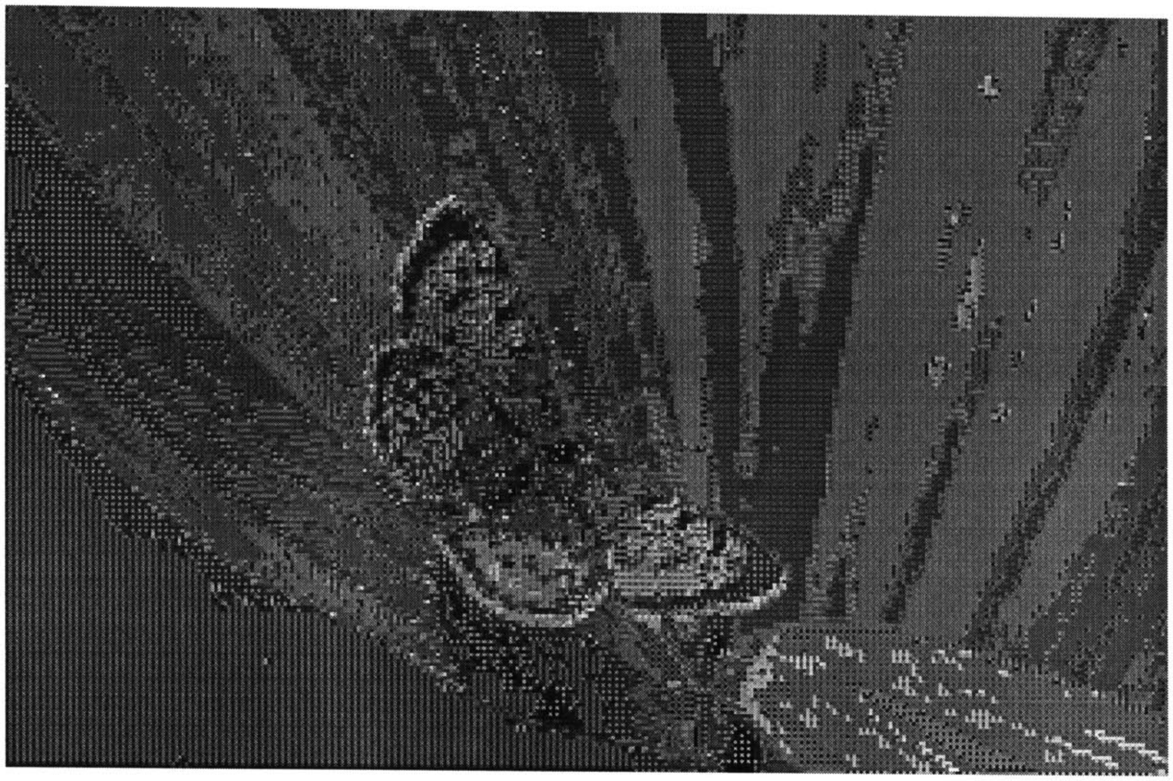


Figure 5.7: KL collage of the butterfly

the storyboard is assembled by selection shots via the centroid browsers, it is converted to a movie by collating all the relevant shots and converting them into a single MPEG stream.

In the suggestion mode, the editor aids the creation of a movie by requesting a time budget and selecting shots from the clusters to fit within this time budget. To select shots, the editor picks one shot from each cluster in a round robin selection procedure with the ordering of each shot within a cluster depending on its proximity to the centroid. This selection is done till the specified time budget is filled. At this point, the shots are rearranged into their correct temporal order (assuming that all shots came from one video tape, for example) and presented as a storyboard to the user. The user can then modify this storyboard by changing the relative placement of shots or by adding/deleting shots. This process of selecting the storyboard can be likened to a "maximum likelihood" movie selection since the shots are

Figure 5.8: Original still picture of turtle

selected from each cluster in accordance to their likelihood within each cluster. This can be viewed as a video summarization technique akin to Smith et al[54] and Yeung et al [68]. Our technique is very flexible since it provides with a summarization for every time budget and across every feature set. In addition, since the shots are picked from different clusters, the summarization is expected to be visually less redundant. Figure 5.10 shows a selected and rearranged 2-minute summary of the home video collection.

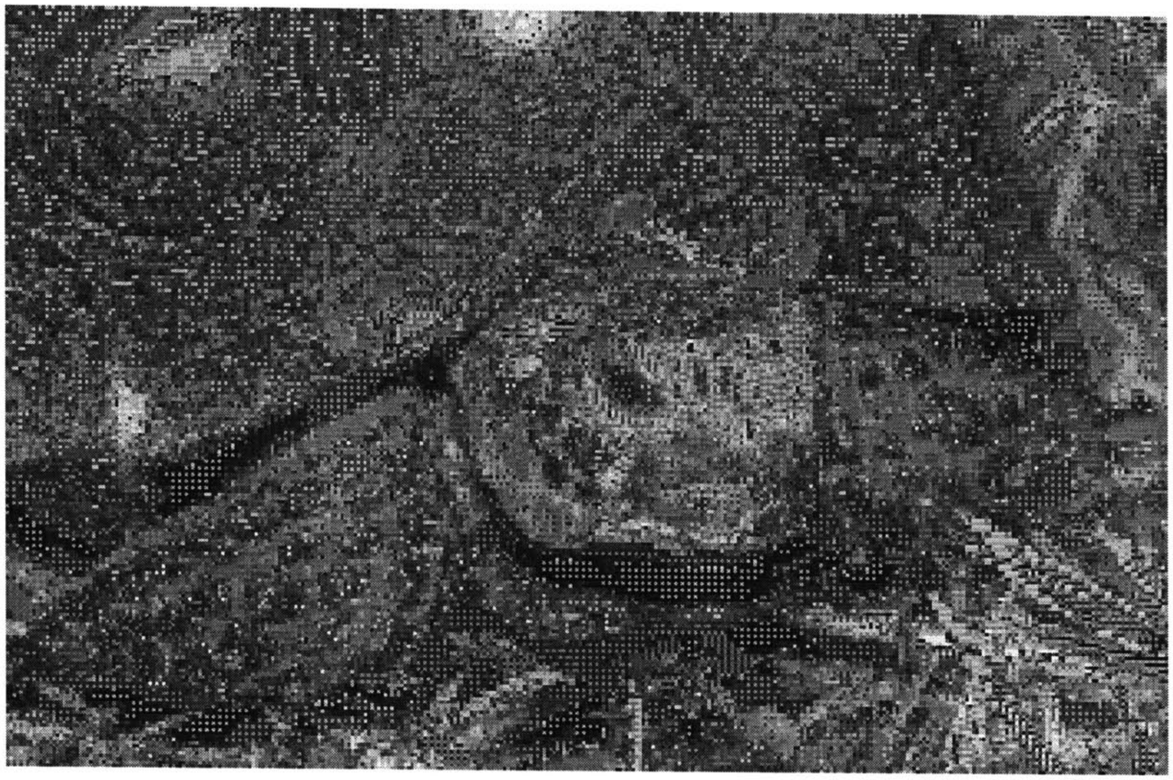


Figure 5.9: KL collage of the turtle

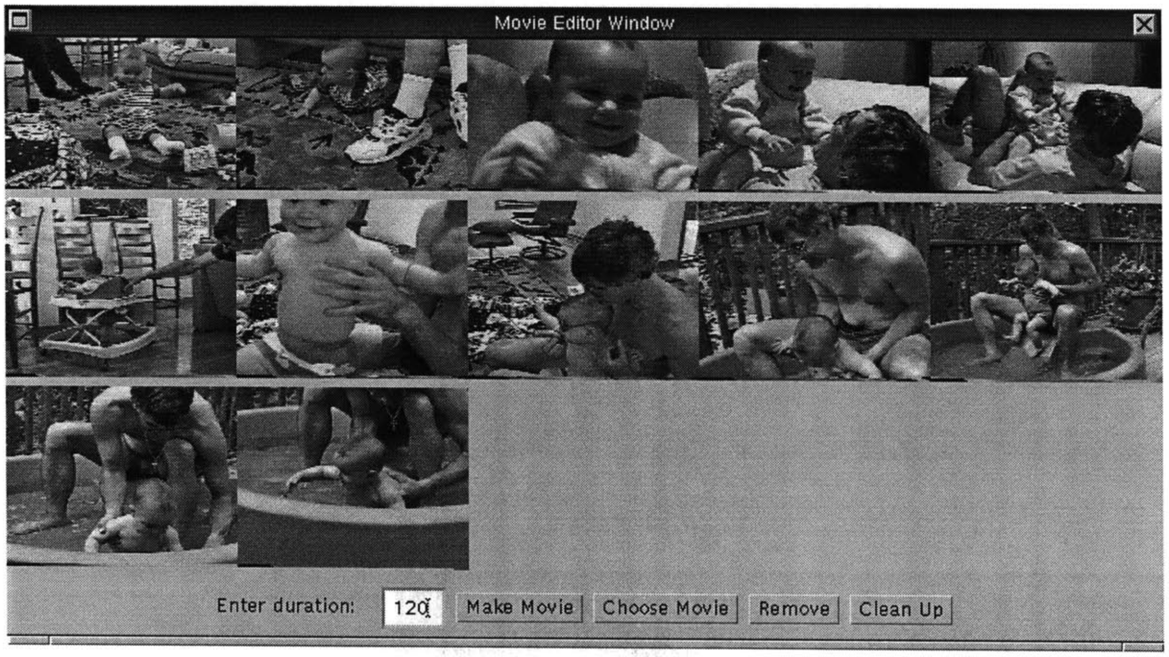


Figure 5.10: Automatically generated 2-minute summary of the home video collection

Chapter 6

Conclusions and Future Work

We conclude the thesis by summarizing the contributions made and presenting some areas for future work.

6.1 Contributions

We highlight the contributions in three distinct areas.

- **Models for video characterization.** We presented a suite of classification techniques that analyze video by action/content etc. In addition to characterizing along color and texture, these techniques used motion as an integral component of video. Some of these techniques exploited the temporal evolution of video to make interesting inferences.

- **Distribution clustering.** We contributed to a clustering approach that takes as inputs probability density models of images and video sequences and groups them in this density space. This approach is not limited to image and video but is generally applicable to all domains that model data as probability densities. We presented

3 different clustering algorithms based on this approach. These groupings form the underlying representation that is used by the video browser that we developed for browsing through unstructured video and creating interesting outputs.

- **Application for unstructured video browsing and editing.** We presented a browser to search through video content. This browser consists of a set of displays to enable interesting outputs from video. Examples include a movie display, a stills display, and a collage display.

6.2 Future Work

The human population watches video - in television and in movies- regularly and in large quantities; this seems to imply that we possess models for video cognition. Movies and television are relatively new constructions, which seems to imply that these models are learned, both by the creators of such narrations and by the consumers. Models and techniques that attempt to bridge the gap between machine representation and these learned cognitive models offer the potential for rich success both in our understanding of this narrative process and that of cognitive modelling. In this thesis we primarily explored models that were close to machine representation and on occasion attempted to extract higher-level meaning (with the Hidden Markov Models) albeit in a limited way. This work needs to be explored further. For example, models that do not just characterize movies as Action/Character or Sports/News but also permit further finer distinctions.

In the distribution clustering work, we primarily explored discrete distributions. Often times, it is useful to represent a discrete distribution in semi-parametric form (such as a mixture of gaussians) which are compact. It would be interesting to phrase distribution clustering in this semi-parametric framework.

Appendix A

Likelihood of data under a novel distribution

We present Cover and Thomas' proof [7] for the data likelihood theorem used in Chapter 4.

Let us define a *type* P_x of a sequence $\mathbf{x} = x_1, x_2, \ldots, x_n$ as the relative proportion of occurences of each symbol in a symbol set A. i.e, $P_{\mathbf{x}}(a) = N(a|\mathbf{x})/n$ for all $a \in A$, where $N(a|\mathbf{x})$ is the number of times the symbol a occurs in the sequence $\mathbf{x}$.

Let P_n denote the set of types of sequences in A^n. A *type class* is the set of sequences of length n and type P and is denoted by $T(P)$, i.e.,

$$T(P) = x \in X^n : P_x = P \tag{A.1}$$

We state a theorem from [7]. The proof is paraphrased here from [7, page 281] for convenience.

Theorem A.1 *If $X_1, X_2, \ldots, X_n$ are samples drawn i.i.d. according to a distribution $Q(x)$,* *the probability of the sample* $\mathbf{X} = X_1, X_2, \ldots, X_n$ *is denoted by* $Q^n(\mathbf{X})$*; and this depends only on its type:*

$$Q^n(\mathbf{X}) = 2^{-n(H(P_\mathbf{X})+D(P_\mathbf{X}||Q))} \tag{A.2}$$

where the entropy of $\mathbf{X}$ *is assumed to be specified in bits.*

Proof:

$$Q^n(\mathbf{x}) = \prod_{i=1}^{n} Q(x_i) \tag{A.3}$$

$$= \prod_{a \in X} Q(a)^{N(a|\mathbf{x})} \tag{A.4}$$

$$= \prod_{a \in X} Q(a)^{nP_\mathbf{x}(a)} \tag{A.5}$$

$$= \prod_{a \in X} 2^{nP_\mathbf{x}(a)\log Q(a)} \tag{A.6}$$

$$= \prod_{a \in X} 2^{n(P_\mathbf{x}(a)\log Q(a) - P_\mathbf{x}(a)\log P_\mathbf{x}(a) + P_\mathbf{x}(a)\log P_\mathbf{x}(a))} \tag{A.7}$$

$$= 2^{n\sum_{a\in X}(-D(P_\mathbf{x}||Q)-H(P_\mathbf{x}))} \tag{A.8}$$

Where we implicitly used the Law of large numbers to go from step 2 to step 3. The powerful statement that the theorem makes is that the individual sample does not govern its likelihood but rather the statistics of the distribution to which the sample belongs.

Appendix B

Convexity of KL divergence

From Theorem 2.6.1 in Cover and Thomas [7, page 24], if a function has a second derivative which is non-negative everywhere, then the function is convex.

Consider, for example, a distortion function $D(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{N}(x(i) - y(i))^4$. In our notation $\mathbf{x}$ is an N-dimensional vector, $\mathbf{x} = x(1), \ldots, x(N)$. From the above-stated theorem, we just need to show that this function has non-negative second derivatives with respect to $x(i)$s and $y(i)$s. Taking partial derivatives with respect to $x(i)$, we get

$$\frac{\partial^2}{\partial x(i)^2} D(\mathbf{x}, \mathbf{y}) = 12(x(i) - y(i))^2 \tag{B.1}$$

which is clearly $\geq 0 \ \forall x(i)$. By symmetry, it is true for $y(i)$s as well.

Now consider the KL divergence function,

$$D(p||q) = \sum_{i=1}^{N} p(i) \log \frac{p(i)}{q(i)} \tag{B.2}$$

where both p and q are PDFs. That is $\sum_{i=1}^{N} p(i) = 1$ and $0 \leq p(i) \leq 1 \ \forall i \in [1, N]$. We note these additional constraints when we take the partial derivatives.

$$\frac{\partial^2}{\partial p(i)^2} D(p||q) = \frac{1}{p(i)} \tag{B.3}$$

Clearly, the second derivatives with respect to $p(i)$s are non-negative. The same is true with respect to $q(i)$:

$$\frac{\partial^2}{\partial q(i)^2} D(p||q) = \frac{p(i)}{q(i)^2} \tag{B.4}$$

which is also clearly non-negative. Therefore, KL divergence is convex.

Bibliography

[1] Vailaya A, M. Figueiredo, A. K. Jain, and H. Zhang. Bayesian framework for semantic classification of outdoor vacation images. In *Storage and Retrieval from Image and Video Databases*, volume VII, pages 415–426. SPIE, Jan 1999.

[2] Aaron Bobick and Claudio Pinhanez. Divide and conquer: Using approximate world models to control view-based algorithms. Media Lab. Perceptual Computing TR 357, MIT, Cambridge, MA, Oct. 1995.

[3] J. S. De Bonet, P. Viola, and J. W. Fisher III. Flexible histograms: A multiresolution target discrimination model. *Proceedings of SPIE*, 3370(12), 1998.

[4] J. S. Boreczky and L. A. Rowe. Comparison of video shot boundary detection techniques. In *Storage and Retrieval from Image and Video Databases*, volume IV, pages 170–179. SPIE, 1996.

[5] C. Carson, S. Belongie, H. Greenspan, and J. Malik. Region-based image querying. In *Workshop on Content-Based Access of Image and Video Databases.* IEEE, Jun. 1997.

[6] M. La Cascia, Saratendu Sethi, and S. Sclaroff. Combining textual and visual cues for content-based image retrieval on the world wide web. In *Workshop on Content-Based Access of Image and Video Databases.* IEEE, June 1998.

[7] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory.* John Wiley and Sons, New York, NY, USA, first edition, 1991.

[8] M. Davis. Media streams: an iconic visual language for video annotation. *Telektronik*, 89(4):59–71, 1993.

[9] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, B 39(1):1–38, 1977.

[10] Y. Deng and B. S. Manjunath. Content-based search of video using color, texture and motion. In *Intl. Conf. Image Processing*, pages 534–537. IEEE, Oct. 1997.

[11] N. Dimitrova and M. Abdel-Mottaleb. Content-based video retrieval by example video clip. In *Storage and Retrieval from Image and Video Databases*, volume V, pages 59–70. SPIE, Feb 1997.

[12] M. Flickner, H. Sawhney, W. Niblack, and et al. Query by image and video content: The QBIC system. *IEEE Computer Magazine*, 28(9):23–32, Sept. 1995.

[13] D. Forsyth, J. Malik, and R. Wilensky. Searching for digital pictures. In *Scientific American*, pages 88–93, June 1997.

[14] K. Fukunaga and W. L. G. Koontz. Applications of Karhunen-Loeve expansion to feature selection and ordering. *IEEE T. Comp.*, C-19:917–923, 1970.

[15] U. Gargi, R. Kasturi, and S. Antani. Performance characterization and comparison of video indexing algorithms. In *Intl. Conf. on Computer Vision and Pattern Recognition.* IEEE, Jun. 1998.

[16] Theo Gevers. *Color Image Invariant Segmentation and Retrieval.* PhD thesis, Wiskunde Informatica Natuurkunde and Sterrenkunde, Amsterdam, The Netherlands, 1995.

[17] The Photomosaic homepage. http://www.photomosaic.com/.

[18] J. Huang, S. Ravi Kumar, M. Mitra, W-J Zhu, and R. Zabih. Image indexing using color correlograms. In *Intl. Conf. on Computer Vision and Pattern Recognition*, pages 762–768. IEEE, Jun. 1997.

[19] The Internet Movie Database. http://us.imdb.com/.

[20] G. Iyengar and A. B. Lippman. Videobook: An experiment in characterization of video. In *Intl. Conf. Image Processing.* IEEE, September 1996.

[21] G. Iyengar and A. B. Lippman. Evolving discriminators for querying video sequences. In *Storage and Retrieval from Image and Video Databases*, volume V, pages 154–165. SPIE, Feb 1997.

[22] G. Iyengar and A. B. Lippman. Clustering images using relative entropy for efficient retrieval. In *Workshop on Very Low bitrate Video Coding.* University of Illinois-Urbana Champaign, Oct 1998.

[23] G. Iyengar and A. B. Lippman. Models for automatic classification of video sequences. In *Storage and Retrieval from Image and Video Databases*, volume VI. SPIE, Jan 1998.

[24] G. Iyengar and A. B. Lippman. Semantically controlled content-based retrieval of video sequences. In *Multimedia Storage and Archiving Systems*, volume III. SPIE, Nov 1998.

[25] G. Iyengar and A. B. Lippman. Video and image clustering using relative entropy. In *Storage and Retrieval from Image and Video Databases*, volume VII, pages 436–446. SPIE, Jan 1999.

[26] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data.* Prentice Hall, Englewood Cliffs, NJ, USA, first edition, 1988.

[27] A. K. Jain and A. Vailaya. Image retrieval using color and shape. *Pattern Recognition Journal*, 29:1233–1244, August 1996.

[28] John R. Koza. Evolution and co-evolution of computer programs. In *From Animals to Animats: Proceedings of The First International Conference on Simulation of Adaptive Behavior (1990)*, pages 366–375. J-A. Meyer and S. W. Wilson eds, The MIT Press/Bradford Books, 1991.

[29] Lillian J. Lee. *Similarity-based Approaches to Natural Language Processing.* PhD thesis, Harvard University, Department of Computer Science, Cambridge, Massachussetts, 1997.

[30] Wentian Li. Mutual information functions versus correlation functions. *Journal of Statistical Physics*, 60:823–837, 1990.

[31] A. B. Lippman, N. Vasconcelos, and G. Iyengar. Humane interfaces to video. In *Asilomar conference on Signals, Systems and Computers*, volume 32. IEEE, Nov 1998.

[32] F. Liu and R. W. Picard. Periodicity, directionality, and randomness: Wold features for image modeling and retrieval. *IEEE T. Patt. Analy. and Mach. Intell.*, 18(7):722–733, July 1996.

[33] F. Liu and R. W. Picard. Finding periodicity in space and time. *International Conference on Computer Vision*, Jan. 1998.

[34] Yanxi Liu and Frank Dellaert. A classification based similarity metric for 3d image retrieval. *Intl. Conf. Image Processing*, Oct. 1998.

[35] W. Y. Ma and B. S. Manjunath. Texture features and learning similarity. In *Intl. Conf. on Computer Vision and Pattern Recognition*, pages 425–430. IEEE, Jun. 1996.

[36] Wallace Martin. *Recent Theories of Narrative*, chapter 5. Cornell University Press, Ithaca, NY, USA, first edition, 1986.

[37] Thomas P. Minka. An image database browser that learns from user interaction. Master's thesis, MIT EECS, Cambridge, MA, 1995.

[38] Virginia E. Ogle and Michael Stonebraker. Chabot: Retrieval from a relational database of images. *IEEE Computer Magazine*, 28(9):40–48, Sept. 1995.

[39] Y-I Ohta, T. Kanade, and T. Sakai. Color information for region segmentation. *Comp. Graph. and Img. Proc.*, 13:222–241, 1980.

[40] Greg Pass and Ramin Zabih. Histogram refinement for content-based image retrieval. In *Workshop on Application of Computer Vision*, pages 96–102. IEEE, Dec. 1996.

[41] Greg Pass, Ramin Zabih, and Justin Miller. Comparing images using color coherence vectors. In *ACM Intl. Multimedia Conference*, pages 65–73. ACM, Nov. 1996.

[42] A. Pentland, R. W. Picard, and S. Sclaroff. Photobook: Content-based manipulation of image databases. *Int. J. of Comp. Vis.*, 18(3):233–254, June 1996.

[43] Rosalind W. Picard, Thomas P. Minka, and Martin Szummer. Modeling user subjectivity in image libraries. *Intl. Conf. Image Processing*, Oct. 1996.

[44] K. Popat and R. Picard. Cluster-based probability model and its application to image and texture processing. *IEEE T. Image Proc.*, 6(2):268–284, 1997.

[45] A. Pope, R. Kumar, H. Sawhney, and C. Wan. Video abstraction: Summarizing video content for retrieval and visualization. In *Asilomar Conference on Signals, Systems and Computers*, volume 32nd. IEEE, Nov. 1998.

[46] L. R. Rabiner and B. H. Juang. An introduction to hidden markov models. In *ASSP Magazine*, pages 4–16. IEEE, 1986.

[47] Lawrence R. Rabiner and Biing-Hwang Juang. *Fundamentals of Speech Recognition.* Prentice Hall, Englewood Cliffs, NJ, USA, first edition, 1993.

[48] Kenneth Rose, Eitan Gurewitz, and Geoffrey C. Fox. Vector quantization by deterministic annealing. In *IEEE T. Info. Theory*, pages 1249–1257. IEEE, Jul. 1992.

[49] Emile Sahouria and Avideh Zakhor. Content analysis of video using principal components. In *Intl. Conf. Image Processing.* IEEE, Oct. 1998.

[50] S. Sclaroff, L. Taycher, and M. La Cascia. Imagerover: A content-based image browser for the world wide web. In *Workshop on Content-Based Access of Image and Video Databases.* IEEE, June 1997.

[51] Behzad Shahraray. Scene change detection and content-based sampling of video sequences. In *Digital Video Compression: Algorithms and Technologies.* SPIE, 1995.

[52] Robert Silvers. Photomosaics: putting pictures in their place. Master's thesis, MIT Media Arts and Sciences, Cambridge, MA, 1996.

[53] John R. Smith and S-F Chang. Safe: A general framework for integrated spatial and feature image search. In *Workshop on Multimedia Signal Processing.* IEEE, 1997.

[54] Michael A. Smith and Takeo Kanade. Video skimming for quick browsing based on audio and image characterization. School of Computer Science Report CMU-CS-95-186, CMU, Pittsburgh, PA, 1995.

[55] Rohini K. Srihari. Automatic indexing and content-based retrieval of captioned images. *IEEE Computer Magazine*, 28(9):49–56, Sept. 1995.

[56] Thad Starner, Joshua Weaver, and Alex Pentland. Real-time american sign language recognition using desk and wearable computer based video. *IEEE T. Patt. Analy. and Mach. Intell.*, 20(12), 1998.

[57] Michael J. Swain and Dana H. Ballard. Color indexing. *Int. J. of Comp. Vis.*, 7(1):11–32, 1991.

[58] Martin Szummer and Rosalind W. Picard. Indoor-outdoor image classification. In *Workshop on Content-Based Access of Image and Video Databases.* IEEE, Jan. 1998.

[59] L. Teodosio and W. Bender. Salient video stills: Content and context preserved. *ACM International Conference on Multimedia*, pages 39–46, Aug. 1993.

[60] Charles W. Therrien. *Decision Estimation and Classification*, chapter 5–9. John Wiley and Sons, New York, NY, USA, first edition, 1989.

[61] A. Vailaya, W. Xiong, and A. K. Jain. Query by video clip. In *Intl. Conference on Pattern Recognition.* IEEE, January 1998.

[62] Nuno Vasconcelos and Andrew B. Lippman. A Bayesian framework for semantic content characterization. In *Intl. Conf. on Computer Vision and Pattern Recognition.* IEEE, Jun. 1998.

[63] Nuno Vasconcelos and Andrew B. Lippman. Learning mixture hierarchies. In *Neural Information Processing Systems.* IEEE, Dec. 1998.

[64] Joshua S. Wachman and Rosalind W. Picard. Tools for browsing a tv situation comedy based on content specific attributes. Media Laboratory Perceptual Computing TR 477 (to appear in Journal of Multimedia Tools and Applications), MIT, Cambridge, MA, Dec. 1998.

[65] B.-L. Yeo and Bede Liu. Rapid scene analysis on compressed video. *IEEE Transcations on Circuit and Systems for Video Technology*, 5, December 1995.

[66] Minerva M. Yeung and Bede Liu. Efficient matching and clustering of video shots. In *International Conf. on Image Processing.* IEEE, Oct. 1995.

[67] Minerva M. Yeung, Boon-Lock Yeo, and Bede Liu. Extracting story units from long programs for video browsing and navigation. In *International Conf. on Multimedia Computing and Systems.* IEEE, Jun. 1996.

[68] Minerva M. Yeung, Boon-Lock Yeo, Wayne Wolf, and Bede Liu. Video browsing using clustering and scene transitions on compressed sequences. In *Multimedia Computing and Networking.* IS & T/SPIE, 1995.

[69] R. Zabih, J. Miller, and K. Mai. Video browsing using edges and motion. In *Intl. Conf. on Computer Vision and Pattern Recognition*, pages 439–446. IEEE, Jun. 1996.

[70] D. Zhong, H. J. Zhang, and S. F. Chang. Clustering methods for video browsing and annotation. In *Storage and Retrieval from Image and Video Databases*, volume IV, pages 239–246. SPIE, 1996.

[71] Y. Zhuang, Y. Rui, T. S. Huang, and S. Mehrotra. Adaptive key frame extraction using unsupervised clustering. In *Intl. Conf. Image Processing.* IEEE, Oct. 1998.

Acknowledgments

First, thanks to Andy Lippman, my advisor, for the good ideas and direction, and an unique perspective of research and life; to the thesis readers, Roz Picard and Bob Ulichney, for their patience to read it and the feedback they provided. A special thanks to Roz for her detailed proofreading of my thesis and suggesting the Maximum Likelihood movie. Andy and Roz were kind enough to loan me their home videos and have fun with them. Thanks to Debbie, Polly, Linda, and Kim for all their help and support.

Special thanks to all my family and friends for the love and support that made my stay away from home much easier: For all the long distance morale boosting and support through difficult times. For all their delicately concocted Indian food to give that unique taste of home.

No acknowledgements will be complete without a salute to the garden denizens who made life memorable – Bill Butera, John Watlington, Nuno Vasconcelos, Stefan Agamanolis, Vadim Gerasimov and Henry Holtzman.

To *puzzles* for keeping me focused and providing countless hours of good times. I heartily welcome the three new additions to the *puzzles* family – Sidharth, Kavya and Amrita. To friends – Anu, Ravikanth, Sofia, Ravi, Srikrishna, Shobana, Neeraj, Swati, Sneha, Parag and Sonali.

Finally a very special thanks to Sri, the real strength behind the success of my stay here. Thank you, Sri, for being there every single moment.

[illegible]