Video Summaries and Cross-Referencing

Aya Aner

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2002
Video Summaries and Cross-Referencing

Aya Aner

A system is presented to construct a highly compact hierarchical representation of video. It is based on a tree-like representation where the bottom level is composed of frames, and the highest level represents a non-temporal segmentation of the video, and is suitable for well-structured video genres. We first propose a method to segment the video into shots, which is based on a leaky memory model, previously proposed for scene transition detection. It is unique since it presents a unified approach for shot and scene detection, and for key-frame selection. We next demonstrate the benefits of using mosaics for representing shots. A novel method for mosaic alignment and comparison is proposed, which is shown to be both efficient and effective. A scene distance measure based on mosaic comparison is then defined and used to cluster the scenes into a higher abstraction of video content, the physical settings.

We demonstrate this hierarchical representation using situation comedies, in which this abstraction has a strong semantic meaning in summarizing videos. By comparing physical settings across different episodes of the same situation comedy, we determine the main plots of each episode. We conclude by demonstrating a video summary tool specialized for fast browsing of situation comedies. In another example, we apply our mosaic comparison method to detect significant events in basketball games, enabling fast forwarding from one event to the next.
Contents

List of Figures v

List of Tables vii

Acknowledgments viii

Chapter 1 Introduction 1

1.1 Our Proposed Video Summarization ............................... 1

1.2 Organization of the Thesis ....................................... 5

Chapter 2 Related Work 7

2.1 Shot Transition Detection ........................................ 8

2.1.1 Metrics for Frame Similarity Measure ......................... 9

2.1.2 Other Approaches ............................................. 14

2.1.3 Gradual Transitions Detection ............................... 15

2.1.4 Shot Transition Detection in MPEG Compressed Domain ... 17

2.1.5 Color Spaces .................................................. 18

2.1.6 Summary of Shot Transition Detection ....................... 22

2.2 Scene Transition Detection ...................................... 23
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.2.1 Modeling Structural Organization of Shots</td>
<td>24</td>
</tr>
<tr>
<td>2.2.2 Clustering of Shots</td>
<td>24</td>
</tr>
<tr>
<td>2.2.3 Incorporating Audio Information</td>
<td>27</td>
</tr>
<tr>
<td>2.2.4 Summary of Scene Transition Detection</td>
<td>28</td>
</tr>
<tr>
<td>2.3 Video Summarization and Comparison</td>
<td>28</td>
</tr>
<tr>
<td>2.3.1 Still Image Summaries</td>
<td>29</td>
</tr>
<tr>
<td>2.3.2 Short Video Summaries</td>
<td>31</td>
</tr>
<tr>
<td>2.3.3 Video Query and Video Comparison</td>
<td>32</td>
</tr>
<tr>
<td>2.3.4 Summary of Video Summarization and Comparison</td>
<td>36</td>
</tr>
<tr>
<td>Chapter 3 A Unified Memory-Based Approach to Cut, Dissolve, Key Frame</td>
<td></td>
</tr>
<tr>
<td>3.1 Motivation</td>
<td>37</td>
</tr>
<tr>
<td>3.2 Theoretic Notations</td>
<td>39</td>
</tr>
<tr>
<td>3.3 A Unified Approach</td>
<td>41</td>
</tr>
<tr>
<td>3.3.1 Modeling a dissolve</td>
<td>42</td>
</tr>
<tr>
<td>3.3.2 Dissolves, fades and cuts</td>
<td>43</td>
</tr>
<tr>
<td>3.4 Coarse-to-Fine Method</td>
<td>45</td>
</tr>
<tr>
<td>3.5 Experimental Results</td>
<td>46</td>
</tr>
<tr>
<td>3.6 Conclusion</td>
<td>48</td>
</tr>
<tr>
<td>Chapter 4 Mosaic-Based Shot Representation and Comparison</td>
<td></td>
</tr>
<tr>
<td>4.1 Construction of Color Mosaics</td>
<td>52</td>
</tr>
<tr>
<td>4.2 Mosaic Comparison</td>
<td>54</td>
</tr>
<tr>
<td>4.2.1 Rubber Sheet Matching Approach</td>
<td>56</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
</tr>
<tr>
<td>---------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Difference Measure and Color Space</td>
</tr>
<tr>
<td>4.2.3</td>
<td>Finding Best Diagonal</td>
</tr>
<tr>
<td>4.2.4</td>
<td>Coarse Matching</td>
</tr>
<tr>
<td>4.2.5</td>
<td>Determining Threshold</td>
</tr>
<tr>
<td>4.2.6</td>
<td>Fine Matching</td>
</tr>
<tr>
<td>4.3</td>
<td>Conclusion</td>
</tr>
</tbody>
</table>

**Chapter 5  Hierarchical Video Abstraction and Video Cross-Referencing  71**

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Clustering Shots in Sports Videos for Event Classification</td>
<td>72</td>
</tr>
<tr>
<td>5.2</td>
<td>Scene Transition Detection</td>
<td>76</td>
</tr>
<tr>
<td>5.3</td>
<td>Hierarchical Summaries and Cross-Video Indexing</td>
<td>78</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Mosaic-Based Scene Representation in Sitcoms</td>
<td>79</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Video Representation by Physical Settings</td>
<td>83</td>
</tr>
<tr>
<td>5.3.3</td>
<td>Comparing Physical Settings Across Videos</td>
<td>87</td>
</tr>
<tr>
<td>5.4</td>
<td>Analysis and Evaluation</td>
<td>91</td>
</tr>
<tr>
<td>5.5</td>
<td>Conclusion</td>
<td>92</td>
</tr>
</tbody>
</table>

**Chapter 6  A Video Browsing Tool  95**

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1</td>
<td>Hierarchical Vs. Temporal Browsing</td>
<td>96</td>
</tr>
<tr>
<td>6.2</td>
<td>Browser Evaluation</td>
<td>97</td>
</tr>
</tbody>
</table>

**Chapter 7  Conclusion  102**

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.1</td>
<td>Future Work</td>
<td>105</td>
</tr>
</tbody>
</table>

**Appendix A  107**

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.1</td>
<td>Comparison of Scene Clustering Results for Different Color Spaces</td>
<td>107</td>
</tr>
</tbody>
</table>
List of Figures

1.1 Hierarchical structure of video ........................................... 2
2.1 Dissolve and fade-in examples ......................................... 9
2.2 RGB Color Space ............................................................. 19
2.3 HSV Color Space ............................................................. 20
2.4 Lab Color Space ............................................................. 22
3.1 Response function with examples for dissolve and fade .......... 41
3.2 Comparison of $L_1$ norm and EMD norm under different quantizations of RGB color space .................................................. 47
4.1 Mosaic vs. key frame representation .................................. 51
4.2 Representative mosaics of a scene ..................................... 55
4.3 HSI color space ............................................................... 58
4.4 Mosaics after re-converting back from $HSI$ to $RGB$ ............. 59
4.5 Mosaic alignment visualization .......................................... 63
4.6 Threshold selection ......................................................... 66
4.7 Aligning mosaics of varying scale ..................................... 69
List of Tables

3.1 Comparison of our shot segmentation method to previous work . . . 49

5.1 Table summary for volume of data used for one episode . . . . . 86

A.1 Summary of mosaic matching results for different color spaces . . . 108

C.1 Table summary for volume of data used for episode 3 . . . . . . 113

C.2 Table summary for volume of data used for episode 4 . . . . . . 114

C.3 Table summary for volume of data used for episode 1 . . . . . . 115
Acknowledgments

First and foremost, I give special thanks to my advisor, John Kender. I was fortunate to have him as my advisor, not just for his professional advise and generous support over the past 5 years, but also for his personal support and understanding. I learned much from him, and benefited from his wide knowledge in many areas. His guidance and advice will accompany me for many years.

I would like to thank the rest of my dissertation committee—Shree Nayar, Rick Kjeldsen, Dan Ellis and Michal Irani—for their participation in my defense and for their helpful comments and suggestions.

Thanks to Shree Nayar, for monitoring my progress in the PhD program and for meeting with me many times, supplying both professional and personal advise.

Thanks to Rick Kjeldsen from IBM Watson Research Labs, to whom I owe my first publication, who kept in touch with me and expressed interest in my work over the years.

Special thanks to Prof. Michal Irani from the Weizmann Institute of Science in Israel, for hosting me in her lab for the past two years and for supplying valuable advice and support. I am grateful both to her and to her students Lihi Zelnik, Yaron Caspi, and Tal Hassner for helping me learn about image registration and
mosaic generation, and for answering my many technical and theoretical questions.

Thanks to Prof. Shih-fu Chang from the Department of Electrical Engineering, for participating in my thesis proposal committee as well as my candidacy exam, for following my work closely over the years and for supplying valuable advice.

Thanks to Prof. Dan Ellis from the Department of Electrical Engineering, who agreed to be on my dissertation committee at a last minute’s request.

I would also like to thank the research labs where I worked during three summers: Siemens Research Labs at Princeton, New Jersey, with special thanks to Marie-Pierre Jolly; IBM T. J. Watson Research Center at Hawthorne, New York, with special thanks to Ruud Bolle and Rick Kjeldesn; IBM Matam Research Labs in Haifa, Israel, with special thanks to Zohar Sivan. I learned and experienced in a variety of areas during these summer internships.

This work was supported in part by NSF grant IIS-98-1202, and gifts from Intel and Microsoft.

Thanks to fellow students who supplied various help over the years, and mostly to my friends from the department: Rahul Swaminathan, Noemie El-Hadad and Andrew Miller.

Thanks to all of my friends and family for baring with me during the busy stressful times. I couldn’t have done this without you.

Finally, my gratitude to my husband-to-be Lior, for his endless love and support.
Chapter 1

Introduction

Video summarization has become more important recently as more advanced media services are offered with digital video being the most popular media form. Example services include digital and interactive video, large distributed digital libraries and multimedia publishing. There is an increasing challenge in managing the large capacity of data required for such video services.

In the near future it is likely that databases containing enormous amounts of digital video serving different applications will be available. The implementation of such digital libraries will depend on the ability to quickly search and browse video content. For this purpose, an efficient and effective video summary technique seems a necessity.

1.1 Our Proposed Video Summarization

We present a complete video summarization scheme which, given a video sequence, generates its semantic hierarchical representation. An illustration of such a tree-like
representation is shown in Figure 1.1. This representation becomes more compact at each level of the tree. For example, the bottom frame level is composed from approximately 50K image frames, whereas the highest level of physical settings has 5-6 representative images. The next two levels of the tree represent a temporal segmentation of the video into shots and scenes, whereas the highest level of physical settings is a newly proposed higher abstraction of video and is based on a non-temporal representation.

![Hierarchical display of video](image)

Figure 1.1: Hierarchical display of video, illustrating both temporal (frames, shots and scenes) and non-temporal (physical settings) representation.

The first temporal segmentation of the video is its segmentation into shots, presented at the second level of the tree, and in our examples there were approximately 300 shots for approximately 50K frames. A shot is a sequence of consecutive frames taken from the same camera, and the first part of this work concentrates on determining two basic types of shot transitions: cuts and gradual transitions. Cuts are well defined and are easier to detect, whereas gradual transitions are generated by editing effects and are much harder to detect. This is mainly due to the fact that they are often confused with camera motion and substantial object motion. There have been many formats suggested for modeling gradual transitions but only with partial success. Other types of shots transitions, all generated by editing effects,
are less frequent or problematic and will not be discussed here.

Segmenting video into its shot structure is the fundamental procedure in analyzing video. Many video indexing or video browsing applications are implemented using only this structure. This is appropriate for some video genres which lack any higher structure. However, many video genres (e.g. TV programs) are well structured and for them further compacting of their summarized representation would be beneficial.

The second temporal segmentation, and the third level in the tree, is the segmentation of shots into scenes. We use the following definition for a scene: a collection of consecutive shots, which are related to each other by the same semantic content. In our examples, consecutive shots which were taken at the same location and that describe an event or a story which is related in context to that physical location are grouped together into one scene. We rely on previous work for generating this level in the tree, and in our examples there were 13-15 scenes.

The last level is the non-temporal representation of the video, of segments which we name “physical settings”. These are groups of scenes, each group taking place in the same physical setting. In our examples there were 5-6 physical settings per episode (half-hour long video). These segments are well defined in many TV series. We chose to demonstrate our work using situation comedies, or in short, sitcoms. One example (although not used in this work) is the known sitcom “Seinfeld”, in which two known physical settings are Jerry’s apartment and the diner. These settings re-occur in many episodes of this sitcom and are therefore directly related to the main theme of the sitcom. In contrast, each episode has
2-3 physical settings which are unique to that episode, and we can determine them by comparing physical settings across different episodes of the same sitcom. These special settings infer the main plots of each episode, and are therefore excellent candidates for representing the content of the episode.

Many existing video indexing and browsing applications rely on frames for visual representation (mainly of shots). We show that mosaics are more reliable for representing shots than key frames, since they represent the physical settings more clearly. We also use mosaics to represent scenes and physical settings, and show their advantage in clustering shots and scenes. In our work a mosaic is an image which is generated from a sequence of frames of a single shot. We compute the camera-induced motion for these frame which enables the alignment of the image frames and their projection into a single image (either a plane or another surface). In the mosaic creation process moving objects are detected and masked out such that the generated mosaic image contains only the static background.

We represent each shot by a mosaic image generated from its frames. A scene is represented by a subset of all the mosaics associated with the shots of that scene. This group of mosaics is carefully chosen according to cinematography rules and camera characteristics to best describe the scene. In order to represent physical settings, further analysis based on clustering is used.

In our work we use only the background information of mosaics, since this information is the most relevant for the comparison process of shots; it is further used for shot and scene clustering, and for gathering general information about the whole video sequence.

This representation is not complete - a more complete representation of a
single shot also involves foreground information like characters. Examples of complementary tools suitable for this task are the synopsis mosaic presented in [Irani and Anandan, 1998] or the dynamic mosaic presented in [Irani et al., 1996], as well as motion trajectories shown in [Gelgon and P.Bouthemy, 1998].

1.2 Organization of the Thesis

In chapter 2, we review the most relevant and important work in the large domain of video indexing. It is divided into three main categories, which approximately correspond to the levels in our suggested tree-like representation of video - shot segmentation, scene segmentation, and video summaries.

Chapter 3 describes our method for temporal segmentation of the video into shots, which uses a similar model previously used for scene segmentation. Together with a proposed scheme for choosing key-frames, it proposes a uniform approach for temporal segmentation of video.

Chapter 4 presents our method for mosaic comparison. Mosaic comparison is not a straightforward task. However, by recognizing the special properties of different video genres, we can apply some constraints when comparing mosaics of the same video genre, and reach very accurate results.

In chapter 5 we describe several applications for which we apply our mosaic comparison method. The first is clustering of shots in basketball sequences for event classification, for which mosaics result in more accurate clustering results. The second is the hierarchical organization of sitcoms, using mosaics to compare shots, scenes, and physical settings. We also demonstrate how this very compact representation of video via mosaics allows fast and efficient comparison across videos.
We demonstrate it by comparing different episodes of the same sitcom and use it to identify the main plots of each episode.

Chapter 6 shows an example of a video browser designed for fast and simple browsing of sitcoms. It is based on segmentation and clustering results and techniques from previous chapters, and has been tested by several users.

Finally, chapter 7 provides a summary of this work and details our plans for future research.
Chapter 2

Related Work

From a broad perspective, the current work could be divided into the following categories: shot transition detection, scene transition detection, video summaries generation and video comparison. In this review we will focus on research done in those areas only in the context of our research. For this reason, we only briefly overview shot segmentation in the compressed domain, and omit many general works in video indexing and browsing that does not utilize the video structure. We also omit work that describes key-frames detection techniques and work that describes complete video indexing and browsing systems. In this chapter we first review segmentation of the video into shots, then we review the clustering of shots into scenes and the shot descriptors used for this comparison. We then review methods for generating summaries for videos, and conclude with methods for comparing across videos.
2.1 Shot Transition Detection

Shots are often considered to be the most fundamental structural element of video. Therefore, the first necessary step in analyzing and summarizing video is determining its shot structure. A shot is a sequence of consecutive frames taken from the same camera. In order to determine the shot structure of a video sequence, the transitions between consecutive shots are determined. There are two basic types of shot transitions: abrupt and gradual. **Abrupt** transitions (cuts) occur within a single frame: one frame belongs to the end of a first shot and the following frame is the starting frame of the second shot. **Gradual** transitions are more complex and occur along several frames. They are also more difficult to detect than cuts. Since the change is gradual, it is often confused with camera motion and significant foreground object motion which result in the same temporal change and cause false positives in the transition detection process.

There are many cinematic effects which could be applied to a video sequence in order to artificially combine two shots, yet only two of them are most frequently used: **dissolves** and **fades**. **Dissolves** are generated by super-imposing one frame on the other, as the frames of the first shot get dimmer and these of the second shot get brighter (Figure 2.1(a)). **Fades** are a special case of dissolves (Figure 2.1(b)): a **fade in** occurs when the first shot is composed of black frames (gradual increase in intensity for the second shot), and a **fade out** occurs when the second shot is composed of black frames (gradual decrease in intensity for the first shot).

Most algorithms for detecting shot transitions use information from uncompressed video, and we will first review those techniques. In order to detect a shot transition, a frame similarity measure is usually defined and measured between
successive frames. Cuts are detected when two successive frames are sufficiently dissimilar, whereas for gradual transitions cumulative differences and other more sophisticated methods are used. We will first describe methods for detecting cuts, and then review enhancements of these techniques as well as other novel techniques for detecting gradual transitions.

The most basic and common frame information used is based on pixel color values. There are different color space representations on which metrics for frame similarity could be defined. Different types of shot transition algorithms could be divided according to the color space they use and the metric defined in those color spaces.

2.1.1 Metrics for Frame Similarity Measure

There are two main categories for defining frame similarity when using color information: pixel-wise comparison and histogram comparison. Pixel-wise comparison, also called template matching, evaluates the difference of color values of corresponding pixels in two frames. For frames of size $M \times N$, a straightforward definition for frame difference measure is the absolute sum of pixel differences. For gray level
images:
\[ \text{Diff}(i, i + 1) = \frac{\sum_{x=1}^{M} \sum_{y=1}^{N} |P_i(x, y) - P_{i+1}(x, y)|}{MN} \]  

(2.1)

and for color images:
\[ \text{Diff}(i, i + 1) = \frac{\sum_{x=1}^{M} \sum_{y=1}^{N} \sum_{c \in \{R,G,B\}} |P_i(x, y, c) - P_{i+1}(x, y, c)|}{MN} \]  

(2.2)

Where \( F_i \) and \( F_{i+1} \) are two successive frames, \( P_i(x, y) \) is the gray level value for pixel \((x, y)\) in frame \( F_i \), and \( P_i(x, y, c) \) is the color value for a pixel, where \( c \) denotes the index for the color component, for example \( c \in \{R,G,B\} \). Other color spaces might be used, as described in section 2.1.5. This value is compared against a threshold in order to determine cuts. A variant of this method is to count the number of pixels for which their value changes by more than a predetermined threshold \( T \) [Nagasaka and Tanaka, 1992] [Zhang et al., 1993]:

\[ \text{Diff}(i, i + 1) = \frac{\sum_{x=1}^{M} \sum_{y=1}^{N} \sum_{c \in \{R,G,B\}} |P_i(x, y, c) - P_{i+1}(x, y, c)|}{MN} \]  

(2.3)

\[ DP(i, i + 1, x, y) = \begin{cases} 1 & \text{if } |P_i(x, y) - P_{i+1}(x, y)| > T, \\ 0 & \text{otherwise.} \end{cases} \]  

(2.4)

A cut is determined if this total value is greater than a second predetermined threshold. The main disadvantage of these methods is their sensitivity to object and camera motion.

Another metric which is based on pixel-differences is block-based matching. It is more robust to camera and object motion and is based on dividing each frame into a fixed number \( B \) of blocks. Corresponding blocks from two frames are then
compared and the frame difference is defined by:

\[
\sum_{k=1}^{B} w_k \cdot Diff^k(i, i + 1) \quad (2.5)
\]

Where \( w_k \) is a predetermined weight for block \( k \) and \( Diff^k(i, i + 1) \) is the difference between the corresponding \( k \)th blocks of \( F_i \) and \( F_{i+1} \).

Kasturi and Jain [Kasturi and Jain, 1991] define \( Diff^k(i, i + 1) \) using a likelihood ratio, by computing the mean and variance intensity values for two corresponding blocks. Shaharay [Shaharay, 1995] uses pixel-wise differences of intensity values, but applies a non-linear order statistics filter to combine the match values of different blocks.

The second category is histogram comparison. A gray level histogram of a frame \( i \) is defined to be a vector of length \( n \cdot H_i(j), j = 1, \ldots, n \), where \( n \) is the number of gray levels and \( H(j) \) is the number of pixels from the frame \( i \) with grey level \( j \). Color histograms are defined in the same manner, or could be defined as \( k \)-dimensional arrays where \( k \) is the dimension of the color space used, usually 3. This representation is equivalent to a vector of length \( 3n \) and the metrics for comparing histograms which are mentioned below could be applied in the same manner as grey level values. Another possible representation is to use \( k \) separate vectors, one vector for each color channel. The metric for comparing these histograms is usually the sum of metrics defined for each channel.

Histograms are robust to small camera motion (including rotation, changes in viewing angle and zoom) and to object motion within the frame. Their obvious flaw is that two images might have very similar histograms, even though they are visually very different. However, this phenomena rarely occurs. There are multiple ways to compute the difference between two histograms, the simplest one is to
compute the sum of absolute differences:

\[ Diff(i, i + 1) = \sum_{j=1}^{n} |H_i(j) - H_{i+1}(j)| \]  \quad (2.6)

where \( H_i(j) \) is the histogram value for the pixel value \( j \) in frame \( i \). If that difference value \( Diff(i, i + 1) \) is greater than a threshold, a cut is declared. [Nagasaka and Tanaka, 1992, Tonomura, 1991, Zhang et al., 1993] applied this metric on grey level histograms generated from image intensities. [Zhang et al., 1993] also applied this metric to quantized color histograms - instead of using all \( 2^8 \) possible values of each of the 3 color channels, only the upper two bits of each color value where used, such that the histogram had only \( 2^6 = 64 \) bins instead of \( 2^{24} \).

In order to enhance frame dissimilarity, several researchers have used the \( \chi^2 \) test [Nagasaka and Tanaka, 1992]:

\[ Diff(i, i + 1) = \sum_{j=1}^{n} \frac{|H_i(j) - H_{i+1}(j)|^2}{H_{i+1}(j)} \]  \quad (2.7)

or histogram intersection [Swain and Ballard, 1991]:

\[ Diff(i, i + 1) = 1 - \text{Intersection}(H_i, H_{i+1}) = 1 - \frac{\sum_{j=1}^{n} \min(H_i(j), H_{i+1}(j))}{\frac{1}{2} \sum_{j=1}^{n} (H_i(j) + H_{i+1}(j))} \]  \quad (2.8)

These measures are more tuned to determine the dissimilarity between images, as opposed to other measures that are better at measuring the similarity between images. However, they are more computationally expensive and in some cases are sensitive to camera and object motion.

Another approach is the weighted bin differences:
\[
Diff(i, i + 1) = \sum_j \sum_{k \in N(k)} W(k)(H_i(j) - H_i(k))
\] (2.9)

where \(N(k)\) is the neighborhood of bin \(j\) and \(W(k)\) is the weight value assigned to that neighbor. For 1-dimensional histograms a neighborhood of 3 is used whereas for 2-dimensional histograms a \(3 \times 3\) neighborhood is used. A special case of this approach is the Quadratic Form Distance [Hafner et al., 1995]: the weight is the difference between two color values according to a pre-computed color difference matrix. This matrix could be generated using the CIE color difference measure [CIE, 2000].

Another method for comparing histograms was suggested by Rubner et. al. [Rubner et al., 1998]. They define the Earth Mover’s Distance (in short, EMD) which measures the distance between two distributions by computing the minimum amount of ‘work’ needed to transform one distribution into the other. This measure could be applied to color signatures of images (histograms, for example). They used this metric for image indexing and retrieval.

In the same manner that block-based comparison was used to improve the pixel-wise comparison by making it more robust to camera and object motion, it is used with histogram-based comparisons in order to make them more sensitive to spatial information and therefore more accurate. Here, the distance between two frames is defined by:

\[
Diff(i, i + 1) = \sum_k DP(i, i + 1, k)
\]

where \(DP(i, i + 1, k) = \sum_j |H_i(j, k) - H_{i+1}(j, k)|\) (2.10)
where $H_i(j, k)$ is the histogram value for pixel value $j$ within block $k$ and $B$ is the total number of blocks. [Nagasaka and Tanaka, 1992] used this technique by dividing the frame into 16 blocks, computing the $\chi^2$ test for each block and discarding the block-pairs with the largest differences to reduce effects of noise, camera and object motion. In [Swanberg et al., 1993] RGB color histograms of blocks were compared using a modified normalized $\chi^2$ test.

### 2.1.2 Other Approaches

The previous methods all rely on having a predefined threshold for determining cuts from peaks in the frame distance measure. A method which avoids this problem was proposed by [Gunsel et al., 1998] and is based on unsupervised clustering using the k-means algorithm to cluster consecutive frame dissimilarity values. Frames classified into the "shot change" cluster which are also temporally adjacent are labeled as belonging to a gradual transition, whereas the rest of the frames from this cluster are labeled as cuts. This technique is not dependent on the features for computing the frame difference.

An interesting approach, not dependent on color information, was suggested by Zabih et. al. [Zabih and J. Miler, 1999]. They analyze intensity edges between consecutive frames, basing their transition detection on the fact that edges disappear and reappear or change location significantly, and declaring transitions according to the number of entering and exiting edges.

Hua and Oh [Oh et al., 1999] apply a non-linear approach in which the video sequence is scanned in non-linear order. They use an adaptive skip in which they compare every $d$th frame, and only compare successive frames once a shot
transition has been detected. The value of \( d \) is changed according to the current frame’s similarity measure with the previous one. The comparison measure for each pair of frames is computed using color histogram generated only from the frame’s wide boundary. Two vertical strips on the right and left and an upper horizontal strip are combined into one long strip, which is sub-sampled to produce a signature. These signatures are also used to estimate possible camera motions and to make the detection more robust.

Few researchers have incorporated audio information along with visual features for determining shot transitions [Boreczky and Wilcox, 1998]. Sometimes “audio cuts” are detected separately from visual shot transitions [Srinivasan et al., 1999, Pfeiffer et al., 1996], and mostly audio features were used to classify shots for further segmentation of the video (see section 2.2.3).

### 2.1.3 Gradual Transitions Detection

Most comparison techniques rely on the fact that the frame dissimilarity function has a high peak at cuts due to the large difference between the two frames. In such cases a single threshold is sufficient to determine a shot transition. However, this approach is not suitable for gradual transitions, for which the frame difference is not as high and pronounced as in cuts. Moreover, camera and object motion may result in similar relatively large differences, therefore it is not sufficient to use a single threshold with reduced value for detection.

To solve this [Zhang et al., 1993] presented the “twin-comparison” method. They use two thresholds, the first threshold \( T_h \) is relatively high and is used to detect cuts in a first pass over the frame difference. In a second pass, a second lower
threshold $T_s$ is used to detect a possible starting frame $F_s$ of a gradual transition. $F_s$ is then compared only with subsequent frames which also exceed $T_s$, by computing the accumulated frame difference. If the accumulated frame difference exceeds $T_b$, a gradual transition is declared.

Other researchers have proposed different mathematical models to describe fades and dissolves [Alattar, 1993, Hampapur et al., 1995, Fernando et al., 2000, Lienhart, 1999b]. A simple model used by [Lienhart, 2001] defined a dissolve sequence $D(x, y, t)$ of duration $T$ as the mixture of two video sequences $S_1(x, y, t)$ and $S_2(x, y, t)$, where the first sequence is fading out while the second sequence is fading in:

$$D(x, y, t) = f_1(t) \cdot S_1(x, y, t) + f_2(x, y, t) \cdot S_2(x, y, t), \quad t \in [0, T] \quad (2.11)$$

where $f_1(t)$ and $f_2(t)$ vary between fade-out, fade-in and dissolve types. The most common dissolve types are cross-dissolves with:

$$f_1(t) = \frac{T - t}{T} = 1 - f_2(t), \quad t \in [0, T]$$
$$f_2(t) = \frac{t}{T} \quad (2.12)$$

Boreczky and Wilcox [Boreczky and Wilcox, 1998] use Hidden Markov Models (HMM) to model different transitions. They use separate states to represent a shot, cut, fade, dissolve, pan and zoom, and set the arcs between the states to model the allowable progress of states. The video is first segmented into regions defined by shots, shot boundaries, and camera movement within shots. The features used for segmentation include an image-based distance between adjacent video frames, an audio distance based on the acoustic difference in intervals just before and after
the frames, and an estimate of motion between the two frames. These features are then combined within the HMM framework. Their system avoids the use of thresholds by using the automatically trained HMMs.

### 2.1.4 Shot Transition Detection in MPEG Compressed Domain

Nowadays video is stored in compressed format, often MPEG (Moving Picture Expert Group [Gall, 1991]). An MPEG video stream consists of three types of frames - I, P and B - that are combined in repetitive pattern called group of picture (GOP). *Intra* (I) frames are coded using only information present in the frame image itself by Discrete Cosine Transform (DCT), Quantization (Q), Run Length Encoding (RLE) and Huffman entropy coding. *Predicted* (P) frames are coded with forward motion compensation using the nearest previous reference (I or P) frames. *Bi-Directional* (B) frames are also motion compensated, but with respect both to past and future reference frames. Each frame is divided into $16 \times 16$ Macro Blocks (MBs), and for each MB the encoder finds the best matching block in the respective reference frames, calculates and DCT-encodes the residual error and also transmits one or two motion vectors (MVs).

It is obviously more computationally efficient to avoid the expensive operation of uncompressing the video and to work directly with the lower data rate of the compressed video data. Temporal video segmentation techniques often use features already present in the encoded video, such as MVs and block averages.

There are several algorithm suggested for shot transition detection in the MPEG compressed domain. They could be divided into several categories according

However, when further analysis of the video beyond the segmentation into shots is required, uncompressing it is often unavoidable, therefore the saving in computation time here is insignificant.

### 2.1.5 Color Spaces

In order to compute frame difference measures, the video data is converted to one of a number of different color space representation such as $RGB$, $HSV$, $YIQ$, $L^*a^*b^*$, $L^*u^*v^*$ or $Munsell$. Then either pixel-wise comparison is used to compute frame to frame difference or histograms in one or more dimensions or computed and compared using different metrics. We will next present a brief overview of the different color spaces:

1) **RGB color space**

The RGB color space consists of the three additive primaries: red, green, and blue. Spectral components of these colors combine additively to produce a resultant color. A natural model for the $RGB$ space would be a uniform cube with three primaries
forming the axis. An example is shown in Figure 2.2. Black is at the origin. White is at the opposite end of the cube. The gray scale follows the line from black to white. In a 24-bit color graphics system with 8 bits per color channel, red is (255,0,0). On the color cube, it is (1,0,0).

The RGB model simplifies the design of computer graphics systems but is not ideal for all applications. The red, green, and blue color components are highly correlated. This makes it difficult to use in some image processing applications.

![RGB color cube](image)

**Figure 2.2:** (a) Illustration of RGB color space.

### 2) HSV color space

This is the Hue, Saturation, Value space often used in computer graphics. It is convenient in that it corresponds to the intuitive notion of color as being composed of a hue component (yellow vs. turquoise) and a saturation component (light pink vs. deep red) as well as value component representing luminance. The color model can be represented by a single inverted hexagonal cone with the tip at the bottom representing black and the center of the hexagon at the top representing white. The axis of the cone then represents Value while the angle around the hexagon represents hue and distance from the axis represents saturation. An example is
shown in Figure 2.3.

Figure 2.3: (a) Illustration of HSV color space.

3) Munsell color space

The Munsell Renotation System [Kuehn and Luxenberg, ] [Miyahara and Yoshida, 1988] also uses a computational breakdown of color into Hue, Value and Chroma. Munsell established numerical scales with visually uniform steps for each of these attributes. The Munsell Book of Color displays a collection of colors laid out in rows and columns for different hue values. Each color is identified numerically using different scales. For a given constant hue, the chips are intended to be perceived with constant brightness in one dimension and constant chroma in another dimension.

4) YIQ color space

This is the NTSC transmission standard for encoding television pictures. Y represents the luminance while IQ represent the in-phase and quadrature-phase components of the chrominance signal. The I channel is optimized for flesh colors.
5) XYZ, \( L^*u^*v^* \) and \( L^*a^*b^* \) color spaces

These color spaces were presented by the CIE \( \text{CIE, 2000} \). A mathematical formula converts \( RGB \) values into normalized \( XYZ \) values. These values approximately correspond to red, green, and blue. The \( Y \) value curve indicates the human eye’s response to the total power of a light source, therefore is called the luminance factor, and is the same as the \( Y \) in the \( YIQ \) model. The \( XYZ \) color space is device-independent and therefore not restrained by gamut, but it has some distortion in the proportions of color differences. The \( L^*u^*v^* \) color space presented in 1976 attempts to correct this and offers a more uniform chromaticity scale diagram with better visual uniformity. The \( L^*a^*b^* \) color space is an opponent color system based on the earlier system from 1942. Color opposition correlates with discoveries in the mid-1960s that somewhere between the optical nerve and the brain, retinal color stimuli are translated into distinctions between light and dark, red and green, and blue and yellow. These values are indicated with three axes: \( L^* \), \( a^* \) and \( b^* \). An illustration of this space is shown in Figure 2.4. The central vertical axis, \( L^* \), represents lightness whose values run from 0 (black) to 100 (white). This is the same lightness valuation used in CIE \( L^*u^*v^* \). The color axes are based on the fact that a color cannot be both red and green, or both blue and yellow, because these colors oppose each other. On each axis the values run from positive to negative. On the \( a-a' \) axis, positive values indicate amounts of red while negative values indicate amounts of green. On the \( b-b' \) axis, yellow is positive and blue is negative. For both axes, zero is neutral gray.
Evaluation of color spaces

Each color space has its flaws and benefits. It is often more convenient to use the RGB values directly (assuming it is the source color space) and avoid conversion. In comparing the computational cost of conversion from RGB, [Gargi et al., 1995] found that the HSV and YIQ were the least expensive, followed by \( L^*a^*b^* \) and \( L^*u^*v^* \) and the Munsell space. HSV and YIQ are linear conversions for which uniform quantization could be applied in generating histograms. Moreover, these quantized values could be easily converted back to RGB for evaluation purposes. Out of these two HSV is more perceptually uniform when using polar coordinates. Munsell was the preferred color space used in the QBIC project [Niblack et al., 1995]. The \( L^*a^*b^* \) and \( L^*u^*v^* \) spaces are considered to be the most perceptually uniform color spaces. The evaluation of color spaces is also dependent on the metrics used to compare them.

2.1.6 Summary of Shot Transition Detection

There have been numerous algorithms suggested for shot transition detection. Recent algorithms are more accurate than the older and simpler algorithms, however,
most of them involve a complicated mathematical model or a reliable threshold
determination. Some are more efficient than others, and are applied in the MPEG
compressed domain. However, the video should often be uncompressed when fur-
ther information extraction of the video is required, eliminating their efficiency.

We have developed a shot transition detection algorithm that performed very
accurate for the video genres that we analyzed (situation comedies and basketball
games), and therefore was most appropriate for our purposes. In contrast to other
algorithms, it combines a unified approach for detecting both shot transitions and
key frames, and works with accordance to a model for scene transition detection.
It is also efficient since it doesn’t require comparison of all successive frames, and
it uses the $RGB$ color space directly.

\section{2.2 Scene Transition Detection}

There are several video genres which are well-structured, for which shots could be
further grouped into scenes. The search for visual structures in edited videos, at a
semantic level above that of the shot level, is relatively new and the literature is
relatively sparse. We define a scene to be a collection of consecutive shots which are
related by a semantic content. This content is often the physical location at which
all shots were all taken, describing some event or a story. The grouping of shots is
done based on a similarity measure defined on shot descriptors. These descriptors
could be built according to key frame information only, or they can incorporate some
motion features and other temporal and spatial information. Good descriptors will
enable easy and straightforward clustering of shots into higher level structures.


2.2.1 Modeling Structural Organization of Shots

Some initial work presented by Wolf [Wolf, 1997] involved the application of HMM models to a single short (2 minute long) video sequence by hand-segmenting and hand-labeling symbol streams. It proposes the idea of classifying shots according to their key frames and to model different scene structures (e.g. dialog) using HMM.

Aigrain et al. [Aigrain and Jolly, 1997] derived medium-level shot structuring rules from observations of a hand-segmented 15 minutes long documentary trying to recover known cinematic structuring and editing rules. The rules include: transition effects, shot repetition, contiguous shot setting similarity, editing rhythm, soundtrack, and camera work. After detecting these local rules an analysis of their temporal organization is done in order to generate macro-segmentation of the video into larger segments and to choose representative shots for each segment.

2.2.2 Clustering of Shots

Few researchers suggested clustering for generating high level segments. Zhong et al. [Zhong et al., 1996] used K-means clustering algorithm to cluster shots according to color histograms of their key frames and applied it on a documentary and on a news broadcast. Clustering shots of news broadcasts was very popular [Zhang et al., 1995a, Mani et al., 1997, Merlino et al., 1997] since the structure of news is well defined (anchor person, news story and so forth).

Gong e. al. [Gong et al., 1995] also use a model to segment soccer TV programs. Their model consists of four major parts: soccer court, ball, players and motion vectors. They identify different regions of the court which correspond to different positions of the soccer game, and analyze different features such as camera
motion, ball motion and players uniform colors in order to categorize and cluster different shot types.

Corridoni and Del Bimbo [Corridoni and Bimbo, 1996] deal with a special kind of film technique scenes, the shot/reverse-shot (SRS) scenes [Monaco, 1977]. An example of a shot/reverse-shot is a sequence of three shots: The first shot is a person’s face; the second shot shows what that person is looking at (whether it be another person or an object); and the final shot is shows how the first person feels about what he or she is looking at (SRS is meant to help the audience build identification with the character, as well as emphasize object location in the scene). In those scenes there’s a clear periodicity in the composing shots. They check similarity across a window of $N$ successive shots at a time, using key frames for comparison. The similarity measure between key frames is chosen as the normalized cross-correlation of cumulative color histograms.

Yeung and Yeo [Yeung and Yeo, 1996] segment a video into "story units" using a graph of shots, in which the key frames’ histograms serve as the similarity criteria. Shots were clustered together if they were similar and appeared within a constraining time window, so that shots further apart in time could not belong to the same story unit. Cut edges in the graph defined the different story units.

Kender and Yeo [Kender and Yeo, 1998] proposed scene segmentation using a continuous measure of coherence and a time constrained buffer model. By measuring the coherence value of each shot in regard to neighboring shots within the time buffer (where shots further away in the buffer had less influence), they determined whether two shots belong to the same scene.

Another approach for scene segmentation was proposed by Hanjalic and
Lagendijk [Hanjalic et al., 1999]. They used a combination of histogram differences between pairs of blocks in key frames from two different shots to determine the dissimilarity measure between shots and used that for clustering consecutive shots into “episodes”. They automatically identify “establishing shots” and distinguish them from the rest of the episodes to generate clear “logical story units”.

Rui et. al. [Rui et al., 1999] construct an intermediate level described as “time-adaptive grouping”, before generating the final scene structure. The descriptors used to measure shot similarity are the shot activity measure (histogram intersection summed over all shot frames) and key frame spatial information (histogram based). The shot similarity is the weighted sum of the similarity of these features, and it groups similar shots together. In order to group semantically related shots together into the final scene structure, they cluster consecutive shots together and merge similar groups.

Oh and Hua [Oh et al., 2000a] compute shot difference using the same background signatures used for comparing frames in their shot transition detection algorithm, discussed in 2.1.2. The distance between two shots is set as the normalized maximum difference across their frame signatures, and is used to construct a relationship tree between the shots, resulting in a temporal segmentation of the video into scenes.

Pan and Faloutsos [Pan and Faloutsos, 2001] propose their VideoGraph to group together similar shots which are apart in time, and display this structure with a graph where each shot group is represented by a node and edges indicate temporal succession. They group shots together by collecting statistics such as the number of I frames, the number of shots, the number of shot-groups in a video clip and
so forth. The distance between every pair of shots is computed using information from DCT coefficients of MBs of I frames. This is done after reducing dimensions using FastMap [Faloutsos and Lin, 1995]. Each shot is a ‘cloud’ of m-dimensional points, and two such ‘clouds’ are grouped together if the density of the union is comparable to the original, individual densities.

Most of the methods for general (all genres) scene segmentation described above are confined in that they only use color information from single key frames to compare and cluster shots.

\section{2.2.3 Incorporating Audio Information}

The Informedia project [Hauptmann, 1995] automatically segments audio into paragraphs and uses it as one of the cues that feeds into detecting story boundaries in broadcast news video. They cue off the silence or low energy areas in the acoustic signal to detect transitions between speakers and topics. Their algorithm computes the power of digitized speech samples and uses a low power level to indicate less active speech. [Huang et al., 1999] also use audio cues in segmentation of news stories and semantic structure reconstruction for broadcast news.

Few researchers combine audio and visual features together in determining scene transitions [Lienhart et al., 1997, Huang et al., 1998, Saraceno and Leonardi, 1997]. [Saraceno and Leonardi, 1997] also classify the audio into four main classes: silence, speech, music and noise.

Sundaram and Chang [Sundaram and Chang, 2000] define the audio scene: a semantically consistent sound segment that is characterized by a few dominant sources of sound. They detect a scene change when a majority of the sources
present in the data change. They detect visual scenes and audio scenes separately, and justify the detection of more audio scenes than visual scenes claiming that visual scenes are more complex and involve many dialogs.

2.2.4 Summary of Scene Transition Detection

The main problem with all the general methods mentioned above is that they mainly use static shot descriptors to determine shots similarity measures. The information about a shot is usually only extracted from its key frames, and the key frames that were chosen were not necessarily the best descriptors of their shots. In addition, no temporal information is used to describe the shots and to calculate shot similarity measures. The use of audio could contribute and improve accuracy but has not been fully utilized for correct scene transition detection.

We propose the use of mosaic images to represent shots (see chapter 4), since a mosaic captures the whole physical setting which is viewed within a single shot, without the occlusion of moving foreground objects. This representation mainly used for higher-level representation and comparison, but it was also tested using the algorithm proposed by [Kender and Yeo, 1998]. We used mosaics instead of frames to represent and compare shots, and received satisfying results.

2.3 Video Summarization and Comparison

There has been much work in generating shot and scene descriptors, however, we will concentrate in reviewing work which directly led and was used to generate summaries for long video sequences and for comparison across videos. This area
is often related to as **Video Abstraction**, a short summary of the content of a longer video, and could be classified into two main categories: still images and short videos.

Video summaries are easily generated using well-structured videos, where the video is not only segmented into shots but also further segmented into scenes. However, some work exists which summarizes videos where only the shot structure is used, as in [Davis et al., 1999, Lienhart, 1999a, Kender and Yeo, 2000, Girgensohn et al., 2001].

The most common descriptors used for shot comparison are key frames. A key frame is a single frame chosen from the sequence of shot frames to describe the shot. For static shots a single key frame is sufficient whereas for shots with significant motion multiple key frames are needed. There exist many suggestions in literature for efficient automatic key-frame extraction algorithms [Yeung and Liu, 1995, Zhang et al., 1995d, Zhuang et al., 1998]. Most of these algorithms try to pick frames within a shot which are most visually different. Others detect local minima of motion.

### 2.3.1 Still Image Summaries

Tonomura et al.[Tonomura et al., 1993] present the “VideoMAP”, an interface which expresses different features for indexing the video. It is in the shape of a video time-line, with single key frames for each shot displayed at the top, followed by the different feature graphs: intensity histogram, intensity average, inter-frame difference of intensity histogram, video x-ray image (summation of frame edges), and hue histogram. They also present the “VideoSpaceIcon” which is a mosaic
image generated from a single shot, with the first frame of the shot pasted on top of it. This way both spatial and temporal information is present in the image.

Bouthemy et. al. [Bouthemy et al., 1999] propose a similar extended approach and call it “Video Hyper-links”. For each shot they generate mosaics and also extract the moving objects, such that comparison between shots can consider both spatial information and object information.

Yeung and Yeo [Yeung and Yeo, 1997] present a pictorial summary of a video using images of different sizes. Each image corresponds to a video shot and its size is set by the shot’s dominance, which is measured by the frequency in which this shot appears in the video. This frequency has been computed using their shot clustering method to detect story units (described in 2.2). They detect dialog scenes using a dialog model (repeating shots of three groups, person A, person B, and both) and confirm it by investigating audio information.

Nam and Tewfik [Nam and Tewfik, 1999] generate summaries for videos relying on two semantic events: emotional dialog and violent featured action. They classify sub-shots (which, they claim, have a more meaningful visual content for their purposes) and use each group of classified shots to generate a temporal summary (of specific semantic content) for the video. They base their emotional dialog detection on the method proposed by [Yeung and Yeo, 1997], and also detect two acoustic features: average pitch frequency and temporal variations of speech signal intensity levels. Violent featured action is identified by detecting rapid and significant movements of persons and objects, fast camera motion, shorter shots, gunfire and explosions, flames, and bleeding. Soundtrack is also characterized as matching violent and non-violent scenes.
Uchihashi et. al. [Uchihashi et al., 1999] generate pictorial video summaries that resemble comic books. Key frames are chosen according to semantic importance analysis, computed from image and audio information. The relative importance of different shots is computed from their length and novelty of their key frames. The two-dimensional layout summary is generated using a frame packing algorithm which sizes key frames according to their computed importance score. They further enhance these summaries using summarized text captions.

All of these summaries and methods for video browsing rely on temporal segmentation of the video into shots and both the comparison and summary are based on shot descriptors, mostly color information from key frames, but also general estimation of motion across the shot. For long video sequences containing hundreds of shots, using this shot structure is not efficient.

2.3.2 Short Video Summaries

Pfeiffer et. al. [Pfeiffer et al., 1996] generate a video abstract of predetermined length by first detecting characteristic scenes such as dialog scenes, high-motion scenes, and high-contrast scenes. They choose one short scene from each type and complete the video abstract (if there’s space available) with more scenes which have an average color close to the average color composition of the whole video.

Smith and Kanade[Smith and Kanade, 1998] proposed summarizing medium-length videos (10 minute long) through the selective omission of frames, but without any structural decomposition.

Hanjalic and Zhang[Hanjalic and Zhang, 1999] presented video highlights, containing only the most interesting parts of the original video (like a movie trailer).
They first cluster all frames of a video sequence into multiple groups and then select the most suitable clustering options using an unsupervised procedure for cluster-validity analysis. Key-frames are then chosen for each valid cluster. The shots from which those key-frames were chosen are concatenated to form the video highlight. This method was applied only to sequences of specific events with a well-defined and reasonably structured content.

2.3.3 Video Query and Video Comparison

Zhang et. al.[Zhang et al., 1995c] propose two levels for querying videos. The first uses key frames and compares their visual features (color histograms, dominant color, color moments, mean brightness, tamura textures [Tamura et al., 1978], local statistical textures, object shape and edge maps). The second is shot-based and uses either visual information of key frames or temporal features (camera operation, temporal variation of brightness and colors, shot mosaics).

In [Naphade et al., 2000] previously proposed shot descriptors, called “Multijects”, are used to characterize shots and perform video indexing using a factor graph framework. Examples for visual multijects include: sky, water, forest, rocks and snow, and examples for audio multijects include: human-speech, aircraft-flying, and music. They construct a global graphical model which describes the probabilistic relations between various multijects. This “multinet” is used to compare between several video clips.

In QBIC [Flickner et al., 1995] the video is first segmented into shots, then key frames are selected for each shot. Video queries are based on key frame characteristics such as average color, color histograms, texture, and object shape as well
as sketching an outline of an object.

VideoQ [Chang et al., 1997] supports queries of short video sequences (few seconds long) according to different combinations of shot classifiers. They present “Video Objects” as a region within a frame which could be tracked across several frames and has some semantic meaning. The object’s color, texture, and shape are recorded and its motion and scaling over time are determined. Motion trails are then matched using also comparison of color, texture, shape and size features.

Jain et al. [Jain et al., 1998] propose two schemes for comparing video clips and for video query using an example video clip. The first is based on comparing key frame descriptors (color, texture, and motion around frame). Key frames of consecutive shots that are highly similar to the query key frames are then used to generate a set of retrieved video clips. The second approach is based on uniformly sub-sampled frames from both the query clip and the video database. Retrieval is based on matching color and texture features of the sub-sampled frames. These methods were tested on two long video sequences (basketball game and broadcast news).

Liu et. al. [Liu et al., 1999] also propose a method for video query by example video clip. They extract color and texture features from key frames and use them to compute similarity between shots. A correspondence graph between the two videos is generated by connecting most similar shots, and the final video similarity is computed using a weighted sum of which incorporates shots’ similarities, shot ordering, shot disturbance (occurrence of shots without corresponding shots), and shot congregate (shots with many corresponding shots).

Oh and Hua [Oh et al., 2000a] use their signature based comparison mea-
sure (described in 2.1.2) to define a video similarity measure. They compute the statistical variances of the video shots and use it for comparison between shots.

Wu et al. [Wu et al., 2000] compute shot similarity using Euclidean distance between multi-dimensional feature vectors extracted from key frames. They use color features ($HSV$ histograms) and texture features (Tamura textures [Tamura et al., 1978]: coarseness, contrast and direction). They weight these features and incorporate information about the shot duration and generate a similarity matrix between all shots in the video. They use a clustering algorithm and dynamically find a threshold for grouping similar shots. Next a similarity measure between a shot $S_i$ and a sequence of shots $V$ is defined. It considers the duration of candidate matching shots to compute the distance and chooses the most similar shot. For comparison of two video sequences, both forward and backward sequence matching are considered, to remove some restrictions on the temporal ordering of matching shots. The similarity is weighted by the ratio of matching shots across the two sequences. They allow the use of relevance feedback to weight the importance of each shot and incorporate that into the video similarity measure.

Adjeroh et. al. [Adjeroh et al., 1998] propose several measures for video comparison, at different levels. One interesting suggestion they make is a string-matching based comparison. They suggest a modified “edit distance” between strings as the difference measure. However, none of their proposed methods is tested.

Lienhart et. al. [Lienhart et al., 1998] propose comparison of videos over different levels of temporal resolution: frame, shot, scene and video level. Frame level comparison is the straightforward comparison of different image features across
frames. The features they propose are color coherence vectors (which computes both histogram difference and distinguishes between coherent and incoherent pixels within each color class, depending on the size of the color region they belong to). *Shot level comparison* divides into three approaches. The first uses features extracted from all frames of the shot, and is based on approximate sub-string matching of features, considering the length of the shot as an important similarity measure. The second approach uses features extracted only from key frames of each shot. It involves automatic selection of key frames based on visual dissimilarity, and comparison of these key frames across shots. The third approach describes a shot as a whole unit and constructs a single feature vector for the whole shot. The shots are compared directly by the feature’s normalized distance function. *Scene level comparison* the shot level comparison described above, resulting in a recursive computation scheme. *Video level comparison* uses two normalized measures: the first is the correspondence measure which specifies the percentage of entities of the 1st video which are similar to the entities in the 2nd video. The second is the re-sequecing measure which analyzes whether the entities that the two videos have in common appear in the same sequence or in a reordered sequence. It counts the number of permutation necessary to reorder the matched entity sequence of the 1st video into the matched entity sequence of the 2nd video.

Several methods for classifying whole videos were proposed. [Iyengar and Lippman, 1998] classified movie trailers (15 seconds long) and distinguished between two classes: action and drama movie trailers. 8 features of color, texture and motion were used for content-based retrieval tests. [Vasconcelos, 1998] classified a movie using four semantic shot attributes: presence/absence of close-up, presence/absence
of crowd, nature/urban setting, and action/non-action. However, these researchers do not show any inter-video comparison examples.

2.3.4 Summary of Video Summarization and Comparison

The query methods described above mostly use shot-level information. Only the content of shots is tested in comparing video sequences, without reference to the semantic content of the video as a whole. For long video sequences containing hundreds of shots, these comparison methods are not efficient. A video summary method suitable for long videos should attempt to combine semantic information about the whole video, for example, its hierarchical structure and the physical settings at which it was filmed.
Chapter 3

A Unified Memory-Based Approach to Cut, Dissolve, Key Frame and Scene Analysis

In this chapter we describe the initial step in our video summarization scheme, which is detecting shot transition and construction of the first level of our tree-like representation of video. After the shot structure is generated, the shots are further segmented into scenes. Motion analysis for each shot determines whether it is an appropriate shot for representing a scene, and if so, a mosaic is generated from its frames.

3.1 Motivation

Automating the process of temporal video segmentation is an important task for many applications that handle large volumes of media. For some digital video data,
particularly future data and MPEG-7 formats, their temporal segmentation might be available along with the video itself, but in practice this is still not common since there is not enough public demand for this information. Therefore, both a fast and effective automatic segmentation technique is required.

In previous work [Kender and Yeo, 1998] have presented a method for detecting breaks between physical scene changes, without the need for a discrete clustering of shots into related classes [Zhong et al., 1996], [Yeung and Yeo, 1996]. The method was motivated by a model of short-term visual memory, in accordance both with intuition and with what appears known with auditory perception [Bregman, 1990].

In keeping with common models in psychology of sensory buffers as "leaky integrators" [Garner, 1974], they modeled the short term visual memory buffer of frame perception as having a limited capacity, preserving the order of visual stimulus, and losing older frames evenly throughout the buffer at the same aggregate rate as new frames are perceived. Thus, the buffer is always filled to capacity, but more recent frames are more likely to be present and therefore to be recalled by the present frame; conversely, older frames are more likely to have been lost. This is suggested by the following diagram, where $V$, $X$, $Y$, and $Z$ represent frames of the first four shots when the first frame of shot $Z$ enters the buffer:

$$V \ldots XXX \ldots YYYY \leftarrow Z \ ZZZZZZZ \ldots$$

Under these assumptions, the probability of a frame being present in the buffer decreases geometrically with discrete time. If the buffer size, $B$, is large, the likelihood of a frame remaining at time $t$ is given by $e^{-t/B}$ [Kender and Yeo, 1998].

In this work we rely on the same model for the detection of scene transitions. Our approach is different in the sense that the short term visual memory buffer is
used to model both simple abrupt shot transitions as well as more complicated gradual transitions as dissolves and fades. It further extends the definition of “key frame”. For this purpose some modifications and enhancements were made.

3.2 Theoretic Notations

We first briefly review the basic notations and definitions used here, which are based on definitions made in [Kender and Yeo, 1998]. We assume a symmetric measure of frame dissimilarity, $D(F_i, F_j)$, where $F_i$ is the frame at time $t = i$, and $i < j$. This dissimilarity can be any one of several measures, and after experimenting with several measures we chose the $L_1$ norm on color histograms. It proved to be both accurate and cheap.

In order to determine frame-to-frame similarity, we define the term “recall”, which is based on the assumption that neighboring frames are expected to have the same similarity behavior. For example, if frame $j$ is similar to some previous frame $i$, then probably the frames in the close neighborhood of frame $i$ are also similar to frame $j$. When we measure the similarity between frames $i$ and $j$, we can enhance it by considering the neighborhood of frame $i$, and it directly related to the temporal distance between the frames. Using our notations, we say that a given single frame, $F_j$, recalls a prior single frame, $F_i$, with dissimilarity $D(F_i, F_j)$ and with probability $e^{-(j-i)/B}$. Its expected frame-to-frame dissimilarity is therefore their product $D(F_i, F_j)e^{-(j-i)/B}$. However, if frame $F_j$ is dissimilar to frame $F_i$, it is also very likely dissimilar to those frames immediately preceding $F_i$. The aggregated expected dissimilarity experienced at frame $F_j$ due to all the frames occurring before frame $F_i$ is therefore given by integrating over the entire past up
to frame $F_i$:

$$Rec(i, j) = \int_{-\infty}^{i} D(F_t, F_j) e^{-(j-t)/B} dt \quad (3.1)$$

with $D$ made suitably continuous.

Further, frames nearby frame $F_j$ also experience analogous dissimilarity with the frames near and prior to $F_i$. Thus, the total amount of dissimilarity induced at the interframe gap just after frame $F_i$ is given by integrating the impact of the past at $F_i$ on those frames more recent than $F_i$:

$$Dis(i) = \int_{i}^{\infty} Rec(i, j) dj = \int_{-\infty}^{\infty} \int_{i}^{i} D(F_t, F_j) e^{-(j-t)/B} dt dj \quad (3.2)$$

In terms of image processing, these pair wise comparisons of frames before and after the interframe gap detect more than simple “intensity” edges there. It will also detect “texture” edges if the shots are internally varied and if these variations are different between shots. The measure is therefore more robust than simple adjacent frame comparison.

The function $Dis(i)$ must therefore have a peak at shot boundaries, have a local but more rounded maximum at dissolve boundaries, and be at a local minimum at those (key) frames which are very representative of their local time intervals. Under some idealizing assumptions, it is possible to derive a closed form for $Dis(i)$, which as given above is simply a double convolution of pair-wise frame dissimilarities with an exponential kernel that models the memory buffer loss.
Figure 3.1: (a) Theoretic response of \( \text{Dis}(i) \) to ideal dissolves of size \( w = 0 \) to \( w = 5 \), with frame distances measured in units of buffer size. An ideal dissolve with \( w = 0 \) is a cut, and shows the impulse response. (b) A segment of the response of the model to a section of the video, showing a cut at \( F_{4869} \) and a dissolve at \( F_{4936} \) with width 26. The small dissimilarities in the middle of the graph are due to a pan, and do not affect segmentation results. (c) A segment showing fade out centered at \( F_{2255} \) with width 19 and fade in centered at \( F_{2293} \) with width 20.

### 3.3 A Unified Approach

As stated before, the short term visual memory buffer has been previously used to model simple transitions between scenes. It assumed that scene transitions occurred between two consecutive shots. When modelling shot transitions, the length of the transition needs to be considered. We distinguish between cuts, which are transitions that occur between two consecutive frames, and gradual transitions which occur along several frames. In order to use the same model for all shot transitions, some modifications were made to the previous model. The new model
is unique in the sense that along with determining the different transition types, it could also be used to pick appropriate "key frames" for representing each shot.

3.3.1 Modeling a dissolve

To simplify further analysis, without loss of generality we adopt the following conventions. The unit of time is chosen to be the buffer size, $B$, to normalize all equations. We model dissolves (and therefore fades) so that the center of the dissolve occurs at time $t = 0$, and it extends from $t = -w$ to $t = w$; thus the dissolve is $2Bw$ frames long. The measured dissimilarity between the frame just before the dissolve and the frame just after the dissolve is given by $d$. We assume that a dissolve is linear, that is, the dissimilarity between any two frames within the dissolve is proportional to their temporal distance. The dissimilarity between the frames at times $i$ and $j$ within the dissolve, $-w \leq i \leq j \leq w$ is therefore given by $d \frac{j - i}{2w}$. We further assume that the shots just before the dissolve and just after the dissolve are almost constant in their measured feature ($D(F_i, F_j) \approx 0$ within the shot), and that the shots extend away from the dissolve for a time longer than $2w$.

On this basis, it is possible to compute the values of $Rec(i, j)$. The analysis can be broken into as many as nine parts, depending on whether $F_i$ and $F_j$ independently occur before, during, or after the dissolve. However, by symmetry considerations, the function can be shown to be even, and only three cases need to be computed. We show only the development of the second case, as the rest are similar in reasoning and form. For $i$ in the dissolve but $j$ after, $-w \leq i \leq w \leq j$, the recent past of frame $F_j$ consists of frames before, during, and after the dissolve. The dissimilarity to frames before the dissolve is identically $d$; the dissimilarity
to frames within the dissolve is linear; but the dissimilarity to its fellow frames within the shot after the dissolve is close to zero. Thus the integration has only two significant parts:

\[
Rec(i, j) = \int_{-\infty}^{-w} \frac{w - (-w)}{2w} e^{-(j-t)} dt + \int_{-w}^{i} \frac{w - t}{2w} e^{-(j-t)} dt
\]

(3.3)

This formula and all previous parts result in closed formulae which can be derived mechanically.

The value of \(Dis(i)\) can now be computed from the multi-part definition of \(Rec(i, j)\), by integrating over all frames \(j\) that occur after frame \(i\). After applying several simplifications, the response of the even function \(Dis(i)\) for non-negative \(i\) is given by:

\[
Dis(i) = \begin{cases} 
\frac{d}{w}(1 - e^{-w \cosh(i)}) & \text{for } 0 \leq i \leq w, \\
\frac{d}{w}(e^{-i \sinh(w)}) & \text{for } i \geq w.
\end{cases}
\]

(3.4)

The function and its derivative are continuous at \(w\), where the derivative reaches its global minimum, and where the curve has an inflection point.

### 3.3.2 Dissolves, fades and cuts

The response within the dissolve is a rounded peak with a flattish top near \(i = 0\) whose height is a function of both the frame difference \(d\) and the half-width of the dissolve \(w\), \(Dis(0) = \frac{d}{w}(1 - e^{-w})\); outside the dissolve \(Dis(i)\) follows a pure exponential decay. One can also show that as \(w\) approaches 0, that is, as the dissolve approaches a cut, the function approaches \(Dis(i) = de^{-i}\), for \(0 \leq i\), a pure exponential decay from a sharp peak of \(d\) at \(i = 0\); all cuts appear with
the same shape, except scaled by \( d \). And, for extremely small \( B \) (extremely rapid memory loss), it can be shown that \( \text{Dis}(i) \) does indeed approach an impulse at \( i = 0 \) with height \( d = D(F_i, F_{i+1}) \), the comparison of adjacent frames. Conversely, when \( w \) is large, the function approaches the constant function \( \text{Dis}(i) = \frac{d}{w} \) within the dissolve, capturing the linear distribution of shot difference across the dissolve. In short, it is not hard to show that under the model of dissolve invoked above, where frame differences are proportional to frame distance, the theoretic response of the memory-based model of dissimilarity is equivalent to a single convolution of adjacent frame differences with a symmetric exponential decay kernel.

Figure 3.1(a) shows the theoretic response curves to increasing integer values of \( w \), with \( w = 0 \) showing the impulse response of the kernel. It also shows a portion of the result of applying the memory-based model to the video.

The response of \( \text{Dis} \) can be probed near each peak to determine the location of \( w \), by searching forward from \( i = 0 \) until certain conditions are met. Although the value of the maximum, \( M \), attained at \( \text{Dis}(0) \) is determined by both \( d \) and \( w \), it can be shown that \( \text{Dis}(i) \) always attains the value of its half-maximum, \( M/2 \) at the point \( i = h \) in the exponential decay portion. Therefore, the half-width of the dissolve, \( w \), can be computed from this value of \( h \), as \( w = \ln(e^h - 1) \).

Cuts, which have \( w = 0 \), attain this half-maximum uniformly at \( i = h = \ln(2) \).

Alternatively, it can be shown that \( w \) can be found by using the probing function \( P(i) = M(1 + e^{-i})/2 \); the values of \( P \) and \( \text{Dis} \) monotonically approach each other until \( P(i) = \text{Dis}(i) \) exactly when \( i = w \). In practice, the detected value of \( w \) can also serve as a type of sanity check on badly detected peaks, since dissolves are generally limited to a total of about three seconds.
3.4 Coarse-to-Fine Method

A common problem that arises when applying shot detection techniques (ours as well as other known methods [Boreczky and Rowe, 1996]), is the compensation between the evaluation measures recall and precision, where

\[
\text{Recall} = \frac{\text{Correct}}{\text{Correct} + \text{Missed}}
\]

\[
\text{Precision} = \frac{\text{Correct}}{\text{Correct} + \text{FalsePositives}}
\]

For high recall values the precision drops, whereas high precision lowers recall. We handle this problem with a coarse-to-fine approach, by sampling the video every five frames. The coarse shot boundaries detected using the same procedure described earlier are then confirmed and refined by scanning all frames in the vicinity of the coarse boundaries. If a finer shot boundary is detected in the four-frame neighborhood of a coarse boundary, we shift the boundary label to the correct position. If a dissolve is detected in the neighborhood of a coarse boundary, we change its label from abrupt transition to gradual transition. An important goal of this sampling is to eliminate false detections. If there is no finer shot boundary in the neighborhood of a coarse boundary, this boundary is eliminated, hence reducing the number of false positives. Finally, coarse boundaries that resulted from smaller and not well defined peaks in the dissimilarity graph are either confirmed or rejected by the finer segmentation. Although this resulted in one additional missed detection in each of the video sequences, it eliminated many false positives.
3.5 Experimental Results

With large buffer size $B$, more frames are recalled, and the video tends to be perceived as being smoother. Conversely, smaller values of $B$ make it more unlikely that incoming frames have similar frames to recall, and local frames differences predominate in the measure. We have found that values of $B = 4$ give a reasonable tradeoff between smoothness and precision.

We hand-segmented two situation comedy video sequences, both 21 minutes long. There were 285 and 235 shots in the two sequences, respectively, which were manually labeled. We then applied our shot transition detection method. We used the DC coefficients of MPEG encoding [Yeo and Liu, 1995] and generated $RGB$ color histograms with $8 \times 8 \times 4$ bins. We derived the pair-wise frame-to-frame distances using the $L_1$ norm. These distances were computed using equation 3.2 with a finite window size $K$. We experimented with different window sizes ranging from 60, 120 to 240 and decided to use $K = 120$, so every frame was compared to the $K/2 = 60$ preceding and $K/2 = 60$ following frames. This range covered the dissolves and fades of length up to 100 frames. We computed the accumulated dissimilarity measure for each frame $F_i$ using a buffer size of $B = 4$ frames, and detected all maxima on the resulting responses. Figure 3.1 (b) and (c) shows a segment of the results. The sharp peak in (b) represents an abrupt change between shots and the smoother peak represents a dissolve. The smooth peaks in (c) represent fade out and fade in.

We have tested several methods for comparing $RGB$ color histograms before concluding with the $L_1$ norm. We used different quantizations of the $RGB$ color space, of $8 \times 8 \times 4$, $6 \times 6 \times 3$, and $4 \times 4 \times 2$. We then derived pair-wise frame-to-frame
distances under both the $L_1$ norm and the EMD (Earth mover’s Distance [Rubner et al., 1998]) norm for five of these six possible histograms (EMD on $8 \times 8 \times 4$ was too costly). Comparison of our cut detection results is shown in Figure 3.2. For $6 \times 6 \times 3$ and $4 \times 4 \times 2$ quantizations the EMD norm performed better than the $L_1$ norm. However, when comparing the results using $L_1$ norm for $8 \times 8 \times 4$ quantization and EMD norm for $6 \times 6 \times 3$ quantization, they perform almost equally well. We therefore decided to use the cheaper method of $L_1$ norm with $8 \times 8 \times 4$ quantization, assuming that the EMD norm with this quantization will not perform significantly better. The complexity of our method is linear in the number of frames $N$, but is also influenced from the window size parameter $K$. The complexity is therefore $O(KN)$.

![Figure 3.2: The graphs shows the cut detection results for 5 cases: $L_1$ norm and EMD norm for $6 \times 6 \times 3$ and $4 \times 4 \times 2$, and $L_1$ norm for $8 \times 8 \times 4$ (EMD on $8 \times 8 \times 4$ was too costly).](image)

Detecting shot transitions is a straightforward self-adjusting task that does not depend on a threshold, although it does depend on a single free parameter (the window size). We look for local maximas in $Dis(i)$. A frame $F_i$ is a candidate cut if $Dis(i)$ is greater than all other values of $Dis(j)$ in a neighborhood of $2N$. 


\( j \in [i - N, i + N] \). If within that neighborhood we detect a monotonic increase
to \( D_{i}(i) \) followed by a monotonic decrease, than \( F_{i} \) is declared a peak. Further
processing eliminates peaks that were measured to be wider than 80 frames (these
would have been later detected as false dissolves). For detecting dissolves, we
measure the width of the peak at half-maximum. Peaks whose half-width are
approximately \( \ln(2) \) are classified as cuts, and wider peaks are classified as dissolves.
In our experiments \( N = 6 \) gave best results.

Our results using the coarse-to-fine approach are better than other reported
results, including the classic experiments by Boreczky and Rowe [Boreczky and
Rowe, 1996]. We chose their work as reference since they tested their algorithms
on the same video genre as ours (the same situation comedy as ours). Their best
results were achieved using color histogram differencing on TV programs. Yet when
the recall is high the precision drops significantly. Our results and samples from
their results are shown in Table 3.1, where: For the first video sequence we had
97\% recall and 98\% precision. For the second video sequence we had 97\% recall
and 97\% precision.

We also note that the method accurately detected 13 of 14 dissolves in each
sequence. Typically, it detected and measured a dissolve of 29 frames long as being
31.9 frames, and another of 28 frames as being 28.6.

### 3.6 Conclusion

Examination of frames where \( D_{i}(i) \) was locally minimal appeared to intuitively
find frames that were very representative of their shots. This leads to a relatively
fast and simple solution for determining key-frames for shot representation. With
Table 3.1: Upper part of table shows results from [Boreczky and Rowe, 1996] on TV programs, including situation comedies (note: the numbers are approximations, copied from a graph). Lower part of table shows our results on the two video sequences.

this observation, our method unifies the concept of cut, dissolve, key-frame and scene change.

We note, as have others [Aigrain and Jolly, 1997], that for some applications our definitions for a “shot” or a “scene” are not appropriate. The method tended to confuse long high-speed tracking pans with dissolves. However, for several applications these changes should be marked as a scene change anyway. We also note that our reported results for segmentation of situation comedies were exceptionally good, which might be due to this type of video genre. Physical settings in situation comedies are often characterized by well distributed pronounced colors (e.g. green walls, purple door) making the frame comparison as well as shot clustering easier.
Chapter 4

Mosaic-Based Shot

Representation and Comparison

This chapter introduces our method for mosaic comparison, which we directly use to compare single shots. Each shot could be represented by a single mosaic, and if we generate mosaics for all shots in a video, we can generate a shot distance matrix using the mosaic comparison method described below. In the next chapter we show how sampled shots (and their corresponding mosaics) are further used to represent scenes and physical settings, and these higher-level structures are also compared using the mosaic comparison method described below.

Since in many video genres the main interest is the interaction between the characters, the physical location of the scene is visible in all of the frames only as background. The characters usually appear in the middle area of the frame, hence, the background information can only be retrieved from the borders of the frame. Oh and Hua noticed this and segmented out the middle area of the frame. They used color information of the borderlines of the frames for shot transition[Oh et al.,
2000b], and later scene transition [Oh et al., 2000a] detection. However, when key-frames are used to represent shots and to compare them, the information extracted from a single key frame is not sufficient. Most of the physical location information is lost since most of it is concealed by the actors and only a fraction of the frame is used.

Consider the example in Figure 4.1(a), which shows a collection of key-frames taken from a panning shot. In that shot an actress was tracked walking across a room. The whole room is visible in that shot, but the actress appears in the middle of all key-frames, blocking significant parts of the background. This effect becomes worse in close-up shots where half of the frame is occluded by the actor. The solution to this problem is to use mosaics to represent shots. A mosaic of the panning shot discussed above is shown in Figure 4.1(b). The whole room is
visible and even though the camera changed its zoom, the focal length changes are eliminated in the mosaic.

In order to construct color mosaics using either projective or affine models of camera motion, and in order to use mosaics to represent temporal video segments such as shots, we make the following assumptions (among others):

1. For the video genres for which we use the mosaic based representation, either:
   a) the 3D physical scene is relatively planar; or b) the camera motion is relatively slow; or c) the relative distance between the surface elements in the 3D plane is relatively small compared with their distance to the camera.

2. Cameras are mostly static, and most camera motion is either translation and rotation around the main axis, or zoom. Since the physical set is limited in size, both the cameras’ movement and varying positioning are constrained.

3. Cameras are positioned horizontally, that is, scenes and objects viewed by them will always be situated parallel to the horizon.

4. Cameras are usually placed indoors, so that lighting changes are not as significant as in outdoor scenes.

### 4.1 Construction of Color Mosaics

The use of mosaics for video indexing was proposed in the past by several researchers, and examples include salient stills [Massey and Bender, 1996], video sprites [Lee et al., 1997], video layers [Wang and Adelson, 1994] and [Vasconcelos, 1998]. However, there aren’t any examples of their use for shot comparison
and further use for hierarchical representation of video. We base our color mosaic construction technique on the grey level mosaic construction method proposed in [Irani et al., 1996], and will describe it here only in a brief manner. The first step is the generation of affine transformations between successive frames in the sequence (in this work we sample 6 frames/sec). One of the frames is then chosen as the reference frame, that is, as the basis of the coordinate system (the mosaic plane will be this frame’s plane). This frame is “projected” into the mosaic using the identity transformation. The rest of the transformations are mapped to this coordinate system and are used to project the frames into the mosaic plane. The value of each pixel in the mosaic is determined by the median value of all of the pixels that were mapped into it. We note that even though the true transformations between the frames in some shot sequences are projective and not affine, we still only compute affine transformations. This results in some local distortions in the mosaic, but prevents the projective distortion, as seen in Figure 4.1(c). This was found more useful for our mosaic construction method which divides the mosaic into equally sized blocks and compares them, as will be described in the following section.

In contrast to the method described [Irani et al., 1996], here we construct color mosaics, and therefore need to find for every color pixel in the mosaic the color median of all frame pixels projected onto it. Taking the median of each channel will result in colors which might not have existed in the original frames, and we would like to use only true colors. We therefore convert the frames to gray level images while maintaining corresponding pointers from each gray-level pixel to its original color value. For each mosaic-pixel, we form an array of all values from different frames that were mapped onto it, find the median gray value of that array, and use
the corresponding color value pointed to by that pixel in the mosaic. We also use outlier rejection, described in [Irani et al., 1996], to detect and segment out moving objects. This both improves the accuracy of the affine transformations constructed between frames as well as results in “clear” mosaics where only the background of the scene is visible, as shown in Figure 4.1(b).

4.2 Mosaic Comparison

Comparing mosaics is not a straightforward task. In video genres where a physical setting appears several times, as is the case for video genres we have tested - sitcoms and basketball games - it is often shot from different view points and at different zooms, and sometimes also in different lighting conditions. Therefore, the mosaics generated from these shots are of different size and shape, and the same physical setting will appear different across several mosaics. Moreover, different parts of the same physical scene are visible in different mosaics, since not every shot covers the whole scene location. An example of different mosaics generated from shots of the same scene is shown in Figure 4.2. The upper three mosaics show the physical scene from similar angles (they were all generated from shots taken by cameras located close to each other), whereas the lower two mosaics were generated from shots that were taken from cameras located in totally different locations than the ones used above, therefore show different parts of the physical scene. It would be easier to cluster the top three mosaics together and the bottom two mosaics together, based solely on these image properties. However, in order to cluster all these mosaics into one single cluster, information from other scenes where similar camera locations were used is crucial. Thus, comparing color histograms of whole mosaics is not
A solution to this problem is to divide the mosaic into smaller regions and to look for similarities in consecutive relative regions. When comparing mosaics generated from certain video genres, we can make assumptions about the camera viewpoint and placement, noting the horizontal nature of the mosaics. Cameras locations and movements are limited due to physical set considerations, causing the topological order of the background to be constant throughout the mosaics. Therefore, the corresponding regions only have horizontal displacements, rather than more complex perspective changes.

This work is, in a sense, similar to that in the area of wide baseline stereo matching, which relies on detecting corresponding features or interest points. Recent work [Schaffalitzky and Zisserman, 2001] has used texture features to match between corresponding segments in images, which are invariant to affine transfor-
mations and does not require extracting viewpoint invariant surface regions. Our method handles significant local distortions in the mosaics, which would cause feature detection techniques to be unreliable. For example, lines and corners might appear too blurred or might even appear twice. Moreover, the local region in the mosaic around a feature points will not necessarily confirm with the local invariants proposed since it might be a result of several super-imposed images used to construct the mosaic. Our method also has the advantages that it is not sensitive to occluding objects and global non-linear distortions. Similar to [Schaflitzky and Zisserman, 2001] it is insensitive to some lighting changes.

4.2.1 Rubber Sheet Matching Approach

We follow the idea of rubber-sheet [Gonzalez and Woods, 1993] matching, which takes into account the topological distortions among the mosaics, and the rubber-sheet transformations between two mosaics of the same physical scene. The comparison process is done in a coarse-to-fine manner. Since mosaics of common physical scenes cover different parts of the scene, we first coarsely detect areas in every mosaic-pair which correspond to the same spatial area. We require that sufficient portions of the mosaics will match in order to determine them as similar. Since the mosaic is either bigger or has the same size as the original frames in the video, we demand that the width of the corresponding areas detected for a matching mosaic pair should be not less than approximately the original frame width. The height of this area should be at least \( \frac{2}{3} \) of the original frame height (we use the upper part of the mosaic), a reason which is motivated by cinematography rules, concerning the focus on active parts [Arijon, 1976].
After smoothing noise in the mosaics with a $5 \times 5$ median filter (choosing color median was described in section 4.1), we divide the mosaics into relatively wide vertical strips and compare these strips. We determine an approximate common physical region in the two mosaics by coarsely aligning a sequence of vertical strips in one mosaic with a sequence of vertical strips in the second mosaic. Our method supports coarse alignment of a sequence of $k$ strips with a sequence of up to $2k$ strips, therefore allowing the scale difference between the mosaics to vary between $1:1$ and to be as big as $2:1$. An example of such mosaics is shown in Figure 4.7(a). This allows us to match mosaics which were generated from shots taken from different focal length. In our experiments there was no need to support a scale difference bigger than $2:1$, but with some modifications (see chapter 7) it will be possible to support bigger scale differences. The results of this coarse stage detect candidate similar areas in every mosaic-pair and are used to crop the mosaics accordingly. If no corresponding regions were found, the mosaics are determined to be dissimilar. In cases were candidate matching regions are found, we use a threshold, determined from sampled mosaic-pairs, to discard mosaic-pairs with poor match scores. We then apply a more restricted matching process on the remaining cropped mosaic-pairs. We use narrower strips to finely verify similarities and generate final match scores for each mosaic-pair. This second, finer step is necessary since global color matches might occur across different settings, but not usually in different relative locations within them.
4.2.2 Difference Measure and Color Space

We first explain how we define the distance measure between image regions. To address changes in lighting intensity, which cause variations along all three axes in $RGB$ space, we adopt a color space based on hue, chrominance (saturation) and lightness (luminance), where such changes correspond mainly to a variation along the intensity axis. The main advantage of these color spaces is that they are closer to the human perception of colors. We have investigated the use of several color spaces, and some evaluation of these color spaces is described in a more detailed manner in Appendix A. We eventually chose to use $HSI$ color space in polar coordinates, and we compute intensity channel as luminance (instead of brightness - average of $RGB$). The $HSI$ space, illustrated in Figure 4.3, forces non-uniform quantization (or: vector quantization, [Gersho et al., 1991]) when constructing histograms and does not capture color similarities as well as $CIE Lab$ color space, for example. However, the appropriateness of any such quantization
can be easily validated by converting the quantized $HSI$ values back to $RGB$ space and inspecting the resulting color-quantized images. Examples of converted mosaics are shown in Figure 4.4. This procedure allows us to tune the quantization and to predict the results of the three-dimensional $HSI$ histogram difference computations.

After some experimentation, we concluded that uniform quantization works well for hue values, and we used 18 values. Since both saturation and intensity behave badly for small values, we applied a non-uniform quantization for both.

![Figure 4.4](image1.png)

Figure 4.4: (a) Original mosaics constructed for two shows from different scenes. These mosaics were first filtered by a median filter, then $RGB$ values were converted to quantized $HSI$ color space and then these quantized values were converted back to $RGB$ space, resulting in the images in (b).

For saturation, a threshold was empirically chosen; for pixels with saturation values below this threshold (i.e., for grays), the hue values were ignored. Satura-
tion values above this threshold were equally divided into 5 bins. For intensity, another threshold was empirically chosen; for pixels with intensity values below this threshold (i.e., for black), saturation values were ignored. Intensity values above this threshold were equally divided into 2 bins. After determining the appropriate quantization, a simple $L_1$ norm between the $HSI$ color histograms was fast to compute and performed well. We also tried the use of quadratic form distance but it did not improve our results and significantly slowed running time, although our computation wasn’t fully optimized as in [Hafner et al., 1995]. Applying this $HSI$ histogram-based comparison on small regions of the mosaics, handled comparison of mosaics showing the same physical scene but from varying angels.

4.2.3 Finding Best Diagonal

All comparison stages are based on the same method of finding the best diagonal in a distance matrix, which corresponds to finding horizontal or vertical alignment. Example illustrations of such distance matrices could be found in Figures 4.5 (e), (f), and (h). Let $D(i, j)$ be an $N \times M$ matrix where each entry represents a distance measure. We treat this matrix as a four-connected rectangular grid and search for the best diagonal path on that grid. If $P\{(s, t) \rightarrow (k, l)\}$ is a path from node $(s, t)$ to node $(k, l)$ of length $L$, then its weight is defined by the average weight of its nodes:

$$ WP\{(s, t) \rightarrow (k, l)\} = \frac{1}{L} \sum_{(i,j)\in(s,t)\cdots(k,l)} D(i,j). \quad (4.1) $$

We search for the best diagonal path $P$ with the minimum weight of all diagonals, that also satisfies the following constraints:
1. $\text{Length}(P) \geq T_{\text{length}}$.

2. $\text{Slope}(P) \leq T_{\text{slope}}$.

The first constraint is determined by the width and height of the original frames in the sequence which determine the minimum mosaics’ size (352 × 240 pixels in our experiments), since we require that sufficient portions of the mosaics will match. For example, the width of the strips in the coarse stage (discussed below) was set to 60 pixels and $T_{\text{length}}$ was set to 5 for the horizontal alignment, so that the width of the matched region is at least 300 pixels. The second constraint relates to the different scales and angles of the generated mosaics due to different camera placements. We have found that the scale difference could be as big as $2:1$, resulting in a slope of approximately $26^\circ$. We therefore examine diagonals of slopes that vary between $25^\circ - 45^\circ$ in both directions (allowing either the first mosaic to be wider or the second mosaic to be wider). We use intervals of $5^\circ$, a total of 9 possible slopes. However, in order to determine the weight of each diagonal we have to interpolate the values along this straight line. Bilinear interpolation is time consuming and experiments have proved that nearest neighbor interpolation gives satisfactory results. With the straightforward use of look-up tables and index arithmetic, we were able to implement it more efficiently. We have altered the nearest neighbor algorithm and reordered operations not only to save computation time but also to have more flexibility on imposing additional slope and length constraints. There are multiple diagonals of the same slope, and for each slope we need to interpolate values along each diagonal by determining which entries should
be used and which indexes to repeat or to skip. Instead of repeating computation for each diagonal in each slope, we model the transformation of the distance matrix of a specified slope into a matrix of scale 1 : 1. We then only need to search for the best sub-sequence along the main diagonal and the diagonals parallel to it. For each slope we generate a list of interpolated indices which we use for all diagonals of that slope, and store them in a Look-Up-Table. We then scan all diagonals and for each diagonal generate a vector of values using the interpolated indices from the LUT. Since the slopes are symmetrical we only need to generate 4 LUTs for 25°, 30°, 35° and 40° and apply them once on rows and once on columns. The next step is to find the best subsequence along each diagonal. Instead of a brute-force search along the vector \( V[i] \) which holds all the interpolated diagonal values, we generate a partial sum vector \( P[i] = \sum_0^i V[i] \). Any sum of subsequence \( V[j]...V[k] \) can be computed simply as \( P[k] - P[j - 1] \) where \( P[-1] == 0 \).

4.2.4 Coarse Matching

In this stage we perform coarse horizontal alignment of two consecutive strip-sequences in a mosaic-pair in order to detect a common physical area. The width of the strips is set to be 60 pixels each, since we need no more that 5-6 vertical segments and we want a number that further divides into smaller segments (for the finer stage). In order to align two strip-sequences, we generate a strip-to-strip distance matrix \( S[i, j] \), where each entry corresponds to the distance between a strip \( s_i \) from one mosaic to a strip \( s_j \) in the second mosaic:

\[
S[i, j] = Diff(s_i, s_j). \tag{4.2}
\]
Figure 4.5: (a) Key-frames of the shot from which the mosaic in (c) was generated. (b) Key-frames of the shot from which the mosaic in (d) was generated. (c) Each strip-pair (example marked in white) is divided into blocks. The values of the $B$ matrix (e) for the marked strips is visualized by grey squares, where each entry represents the difference measure between a block in the example strip taken from (c) and a block in the example strip taken from (d). Coarse vertical alignment between the strips is determined by the “best” diagonal (thin white line); the average of the values along the diagonal defines the distance between the two strips. (f) The $S$ matrix for the two mosaics, with the strip score from (e) highlighted. Horizontal alignment between strip sequences is determined by the “best” diagonal path in $S$ (thin white line) and is used to (g) crop the mosaics to prepare for the verifying finer match in (h), which shows a similar $S$ matrix for the finer stage. Note that the mosaics are matched correctly even though there are scale and viewpoint differences and occluding foreground.
where \( Diff(s_i, s_j) \) is the difference measure between the strips discussed below. An example of \( S[i, j] \) is shown in Figure 4.5(f), where each grey level block corresponds to an entry in the matrix. Finding two strip sequences in the two mosaics that have a “good” alignment and therefore define a common physical area, corresponds to finding a “good” diagonal in the matrix \( S[i, j] \).

In comparing two strips that correspond to the same actual physical location, we cannot assume that they cover the same vertical areas, but we can assume that they both have overlapping regions. Therefore, in order to detect their vertical alignment, we further divide each strip into blocks and generate a block-to-block distance matrix \( B[k, l] \) for each pair of strips. Each entry in this matrix corresponds to the distance between a block \( b_k \) from the first strip and a block \( b_l \) from the second strip:

\[
B[k][l] = Diff(b_k, b_l). \tag{4.3}
\]

where \( Diff(b_k, b_l) \) is the distance defined in section 4.2.2. We look for the best diagonal (as explained in section 4.2.3) in the distance matrix \( B[k, l] \) and record its start and end points. The value of this diagonal is chosen to set the distance measure between the two strips:

\[
S[i, j] = \min_{d \in Diags} \left( \frac{\sum_{(k,l) \in d} B[k][l]}{\text{length}(d)} \right). \tag{4.4}
\]

Where \( Diags \) is the set of all allowable diagonals in the matrix \( B[k, l] \), and each diagonal \( d \in Diags \) is given by a set of pairs of indexes \( (k, l) \).

The comparison process is shown in Figure 4.5. Two mosaics are shown in Figure 4.5(c) and Figure 4.5(d). The strip comparison process (equation 4.3) is
graphically displayed in Figure 4.5(e) and the matrix $S$ is graphically displayed in Figure 4.5(f), with the result of the two strip comparison from Figure 4.5(e) highlighted. The height of this illustrated grey-level blocks matrix $S$ is the same as width of the mosaic in Figure 4.5(c) and its width is the width of the mosaic in Figure 4.5(d), such that each block represents the strip-to-strip difference between the two mosaics.

We next find the best diagonal in the matrix of strip-to-strip differences $S[i, j]$ in order to find a horizontal alignment between strip sequences, and we record its start and end points. The area of interest is represented by a thin diagonal line across the matrix along which entries have low values (corresponding to good similarity) in Figure 4.5(f). We use these start and end points to crop each mosaic such that only the corresponding matching areas are left. The start and end points of the diagonal in the $S$ matrix set the vertical borders of the cropped mosaics. In order to determine the horizontal borders, we inspect the results of the vertical alignment for every pair of strips along the diagonal path found in $S$. Since for every strip pair we recorded the start and end points of the best diagonals in its corresponding $B$ matrix, we take the average of these values and use it to set the horizontal border.

### 4.2.5 Determining Threshold

Once all mosaic pairs are processed, we check the distance values of the diagonal paths found for each pair. We expect that mosaic-pairs from different settings or with no common physical area will have high distance values, and we discard them according to a threshold. We determine the threshold by sampling several
distance values of mosaic-pairs which have common physical areas, which therefore should have the lowest distance values. We choose the highest of these sampled values as our threshold. This sampling method proved correct after inspecting all mosaic-pairs distances as shown in Figure 4.6.

Figure 4.6: Illustration of threshold determination for mosaics distance after the coarse stage. Values in the two leftmost groups belong to mosaic pairs with a known common background, and the rest of the values are on the right. The leftmost group of 10 distance values is the sample group. The horizontal line is the maximum value among this sample group.

To illustrate our results, we manually separated mosaic-pairs with common physical areas (two groups on the left) from the rest of the mosaic-pairs (group on the right). The sampled pairs are the leftmost group. The threshold chosen from the hand-labeled samples (horizontal line) quite accurately rejects mosaic pairs known to be from different physical areas, although it does permit a few false positive matches. If a diagonal path distance value exceeds this threshold, we determine that the two mosaics do not match. If we find a match, we continue to the finer step where we only use the cropped mosaics: those parts of the mosaics corresponding to the best sub-diagonal found, an example of which is shown in Figure 4.5(g).
4.2.6 Fine Matching

After discarding mosaic-pairs which had diagonal paths with large distance values, we refine the measure of their closeness, in order to detect false positives and to more accurately determine physical background similarity. In this finer stage we only compare the cropped parts of the mosaics, applying a more precise and restrictive method than the one used in the coarse stage. The cropped mosaics are displayed in Figure 4.5(h). Note that the mosaics are matched correctly even though there are scale and viewpoint differences, and the occluding foreground is cropped out. We now use thinner strips (20 pixels wide) and also take into account the scale difference between the two mosaics. Assuming that the cropped mosaics cover the same regions of the physical setting, if we divide the narrower of the two cropped mosaics into \( K \) thin strips, then the best match will be a one-to-one match with \( K \) somewhat wider strips of the wider mosaic, where each strip pair covers the exact physical area. Let \( \alpha \geq 1 \) be the width ratio between the two mosaics. We re-compute histograms of 20 \( \times \) 20 blocks for the narrower cropped mosaic, and histograms of 20\( \alpha \) \( \times \) 20\( \alpha \) blocks for the wider cropped mosaic. The best path in the new distance matrix should now have a slope of approximately 45°. Matching in the finer stage is less computationally expensive than the matching in the coarse stage. First, only a few mosaic pairs are re-matched. Second, having adjusted for the mosaic widths, only diagonals parallel to the main diagonal need to be checked. Third, since the cropped mosaics cover the same regions, only complete diagonals rather than all possible sub-diagonals need to be checked. Therefore, we only compute values for the main diagonal and its two adjacent parallel diagonals. These diagonals are automatically known to satisfy the diagonal length and boundary constraints.
These operations restrict the total number of strip and block comparisons to be performed and greatly lower computation time, even though there are more strips per mosaic. This final verification of mosaic match values is very fast.

The matching of mosaics could be further enhanced to support a more accurate alignment, by finding a coarse affine transformation between corresponding blocks. However, our coarse-fine approach yields a fast and simple matching method, which performs sufficiently accurately.

4.3 Conclusion

We have presented a method for general mosaic comparison and applied it to mosaics generated from sitcoms. Our method enables matching of mosaics which were generated from shots taken from different locations and with different focal length. This method has only few restrictions, in that it assumes a horizontal nature of mosaics and was only tested on shots taken indoors, where the lighting changes were generated artificially. However, our mosaic comparison method gave satisfactory results even for distorted mosaics when shots had parallax motion, or for mosaics with global or local distortions due to bad registration. It performs quite well for large scale differences, as shown in Figure 4.7.

By applying a median filter we were able to correct some of the noise which resulted from either inaccurate frame registration or inappropriate projection surface of the mosaic. However, we are aware of methods to improve the appearance of mosaics such as the ones suggested in [Zomet et al., 2000], [Szeliski and Heung-Yeung, 1997] or [Vasconcelos, 1998], if one is able to afford the additional computation time. Nevertheless, these methods are not able to determine the appropriate
Figure 4.7: (a) Two mosaics generated from shots in the same episode and (b) their corresponding strip-to-strip difference matrix. Note that the scale difference between them is close to 2 : 1.

surface type automatically (whether it is planar, cylindrical or spherical), hence will not guarantee accurate results for our example.

We are also aware of the benefits of the CIE color difference formula using $La^*b^*$ color space, and the quadratic form distance which will give more accurate results. Since these methods are more computationally expensive, we haven’t used them, but they might be preferable in testing our methods on mosaics generated from outdoor shots.

The comparison by alignment method could be made more robust if the strip width is not determined in advance. Instead of two levels of alignment, a more elaborate decreasing sequence of strip widths could be used, starting from relatively wide strips (the whole image width might be used) and converging to smaller strip width (up to pixel-wide strips). This process should converge automatically to the best matching strip width, by computing a parameter that evaluates the ‘goodness’ of the match at each level.
We believe that even though our comparison by alignment technique will never replace existing narrow baseline stereo matching methods, it could be made more robust to challenge the results of existing wide baseline matching techniques. In some cases (e.g. distorted images), it may also outperform them.

Another important aspect of our approach is the complexity of our comparison method. We regard the mosaic construction as a black box which is not included in our comparison complexity evaluation. Once the mosaics are constructed, the cost of the comparison is proportional to the square of the number of strips in each mosaic. In many video genres the mosaic width is limited due to restrictions on camera motion within the scene. Therefore, if the largest number of strips is \( m \), then there are \( m \times m \) strip comparisons. The comparison of each pair of strips is limited by the height of the mosaics, defined by \( n \) blocks. Therefore, the total cost of comparing two mosaics is \( O(m^2 \cdot n^2) \). In our experiments the values of \( m \) and \( n \) were small: \( m \leq 20, n \leq 6 \). This summarizes the cost for the coarse stage, which used strips and blocks that were 60 pixels wide. The cost for the finer stage is computed in a similar matter, and has the same order of magnitude.
Chapter 5

Hierarchical Video Abstraction
and Video Cross-Referencing

This chapter describes two applications of two different video genres for which we applied our mosaics comparison technique, and concludes the final levels of our tree-like video summary. In the first application, we cluster shots in basketball videos in order to classify shots and detect repeating characteristic events. In the second application, we complete the generation of hierarchical representation for sitcoms by segmenting shots into scenes and by further clustering scenes according to physical location. All the video sequences are first divided into shots using the shot boundary detection technique described in Chapter 3. For sitcoms the video is further divided into scenes. Then the scenes are clustered into physical settings, which are further used to cross-reference across episodes of the same sitcom. A video browser which uses this representation by mosaics and is based on the structure built here, will be described in the next chapter.
5.1 Clustering Shots in Sports Videos for Event Classification

A good example of the mosaic-based representation is in representing and comparing shots in sports videos. We used the coarse stage of our mosaic comparison method to cluster shots from basketball videos. This stage allowed us to easily distinguish between shots showing wide side views of the basketball court ("court shots"), close-up shots of people, shots taken from above the basket and shots showing the court from a close-up side view. The clustering of the mosaics led to a more meaningful categorization than clustering key-frames of the same shots, as shown in the dendrograms in the figures in Appendix B. Shots in sports videos tend to be very long. Therefore, an intelligent automatic key-frame selection algorithm is needed, but it would not always capture the correct side of the court; some shots have varying zoom, so if a key-frame is chosen when the camera has zoomed in, it will not match other key-frames showing a wide view of the court; key-frames do not easily distinguish between court shots which show the right court, the left court, or both. An example of a mosaic generated from a panned and zoomed court shot is shown in Figure 5.2(a), and the corresponding key-frames are shown in Figure 5.2(b).

Preliminary results from the clustering of basketball videos allowed us to classify shots and to determine temporal relations between clusters. Filtering rules, corresponding to video grammar editing rules, were manually determined to extract events of human interest. For example, in the video of a European basketball game, we detected all foul penalty throws, and used them as bookmarks in a quick
Figure 5.1: (a) First stage (coarse) comparison of shots from a European championship basketball video. The first cluster along the main diagonal represents court shots which cluster together in this stage, the following cluster represents foul penalty shots that were taken from above the basket. The rest of the shots are various close-ups of basketball players and the coach. (b) Screen captures of a video player that has been augmented with bookmarks. It indexes to those important events that have been derived using a grammar rule applied to mosaic clusters. At left, beginning of court shot of foul (where the bookmark is set), followed by a close-up shot, then the characteristic shot taken from above the basket.

browsing tool shown in Figure 5.1(b). Foul penalty throws were characterized by a three-part grammatical sequence: a court shot, followed by a close-up shot, followed by a shot from above the basket. The results from the shot clustering are shown in Figure 5.1(a). The figure shows the results of clustering mosaics after only the coarse stage. The clusters at the top-left (about $\frac{2}{3}$ of the matrix) are all court shots. The following cluster is of shots taken from above the basket, used for foul penalty shots. The rest of the data is of various close-up shots.

In another example, we analyzed several quarters of NBA basketball games and discovered that field goal throws were usually characterized by a court shot
Figure 5.2: (a) Example of a mosaic from an NBA basketball sequence with (b) corresponding key-frames. Note that the zoom in the key frames changes, making it harder to compare and classify them, whereas the zoom of the mosaics is uniform. (c) Screen captures of a video player that has been augmented with bookmarks for detected events, generated after analysis of the results of clustering the shots’ mosaics. On the left, a field shot of when a basket goal took place. On the right, a close-up shot temporally following the field shot. (d) First stage (coarse) comparison of shots from that game. The top-left cluster along the main diagonal represents court shots which cluster together in this stage, the following cluster represents various close-ups of basketball players and the coach.
followed by a close-up shot of the player making the goal. These instances were
easily detected after the mosaic clustering was used as input to classify court shots
and close-up shots. A dendrogram representing the clustering result of several
minutes of a game is shown in Figure B.1. The upper marked red cluster represents
close-up shots, and the lower marked blue cluster represents field shots. These
clusters could further be divided into smaller categories, such as field/audience
close-ups and whole/right/left field shots. In one example, the first quarter of one
of the first games of the season, we detected 32 sequences of a court shot followed
by a close-up shot, which were good candidates for representing field goals. Of
these, 18 were regular field goals, 7 were assaults, 2 occurred just before a time-
out and the rest of the 5 instances showed the missed field goals of a well-known
NBA player. (This happened to be Michael Jordan’s first game after retirement,
which explains why cameras switched to close-ups even though he missed). All of
these instances serve as interesting events in the game. They became bookmarks
for an augmented video player which allows regular viewing of the game as well as
skipping from one bookmark to the other. This video player was created at the
Weizmann Institute of Science in Israel, and we would like to thank Prof. Michal
Irani for its use, and Tal Hassner for his help in adjusting the browser for our needs.
Screen captures of this video player for a foul shot bookmark are shown in Figure
5.2(c). The preliminary clustering results which separated field shots from close-up
shots are shown in Figure 5.2(d).

A similar detection of field goal throws has been performed by [Nepal et al.,
2001], who relied on the analysis of the scoreboard image super-imposed upon the
frames and the audio detection of cheers from the crowd. We tested our field goal
throw detection technique on different basketball videos, and analyzed preliminary results using recall and precision. In one video we received recall of 72% and precision of 55%, and in the other the recall was 78% and the precision was 56%. We would like to note that these results do not rely on any prior information about the editing of the basketball game or on any audio information, but only use raw visual information of the filmed game.

Further analysis of the mosaics distinguishes between different categories of field goal throws and labels each with its corresponding team. This is done by adjusting the mosaic comparison method, which generally aligns sub-mosaics. It declares a good match the alignment of a left(right) field mosaic, for which only the left(right) basket is visible, with a whole field mosaic, in which both the left and the right basket are visible. By forcing the matching stage to align the whole mosaic instead of sub-mosaics, a left(right) field mosaic is not matched with a whole field mosaic. After separating whole, left and right field shots, the information about basket goals becomes more complete. Close-ups occurring immediately after left(right) field shots could infer basket goals for the right(left) team. For close-ups occurring immediately after whole-field shots, simple motion analysis of the whole field shots (which are panning shots) infers which side of the court was visible at the end of the shot, therefore infers which team made the basket goal.

5.2 Scene Transition Detection

We use the method described in [Kender and Yeo, 1998] and applied to sitcoms to temporally cluster shots into scenes. In this work the same model is used to detect scenes transitions, but instead of computing the frame recall value we compute the
Figure 5.3: Shot difference matrices, where each entry \((i, j)\) corresponds to the difference between shots \(i\) and \(j\). Only a band diagonal was computed, i.e. only the difference between a shot and its \(K\) neighbors was considered, where \(K\) corresponds to the characteristic number of shots within a scene. (a) Matrix generated using key frames, with similarity measure as \(L_1\) difference of \(RGB\) color histograms of frames. (b) Shot difference matrix generated using mosaics, with similarity measure as the mosaic comparison by alignment method described in Chapter 4. Note that the shot clusters are more enhanced in the right image.

shot recall value. [Kender and Yeo, 1998] have used information from all frames of a shot in order to define the dissimilarity measure between each pair of shots. The \(L_1\) norm between \(RGB\) color histograms of all frames in the shot was computed, and the minimum of these values was chosen as the shot dissimilarity value. In this work, instead of using raw frame information, we use the mosaics constructed for each shot. We have also changed the dissimilarity measure: we apply the mosaic comparison by alignment method described in Chapter 4 which not only uses a different color space, the \(HSI\) space in polar coordinates with vector quantization, but also considers the spatial information of the mosaic images and is robust to occlusion of foreground objects. The final scene segmentation results were somewhat improved, and the shot distance matrices for both method illustrate this. The two
matrices shown in Figure 5.3 show the distance computed between all shots in a single episode, along a time-constrained window (as described in [Kender and Yeo, 1998]). The figure illustrates how the mosaic-alignment based comparison method results in more pronounced shot clusters than the frame-based comparison method.

5.3 Hierarchical Summaries and Cross-Video Indexing

Our second example utilizes the mosaic-based representation to generate a compact hierarchical representation of video, and we demonstrate this representation using sitcoms. The video sequences are first divided into shots using the shot transition detection technique described in Chapter 3, and these shots are further divided into scenes using the method described in section 5.2. This hierarchical tree-like representation is illustrated in Figure 1.1. We represent shots and scenes with mosaics and use them to cluster scenes according to physical location.

We present a new level of abstraction of video, which concludes the top level of our tree-like representation, and call it physical setting. This high-level semantic representation does not only form a very compact representation for long video sequences, but also allows us to efficiently compare different videos (different episodes of the same sitcom). By analyzing our comparison results, we can infer the main theme of the sitcom as well as the main plots of each episode.
5.3.1 Mosaic-Based Scene Representation in Sitcoms

A scene is a collection of consecutive shots, related to each other by some spatial context, which could be an event, a group of characters engaged in some activity, or a physical location. However, a scene in a sitcom occurs in a specific physical location, and these locations are usually repetitive throughout the episode. Some physical locations are characteristic of the specific sitcom, and repeat in almost all of its episodes. Therefore, it is advantageous to describe scenes in sitcoms by their physical location. These physical settings can then be used to generate summaries of sitcoms and to compare different episodes of the same sitcom. This property is not confined to the genre of sitcoms alone. It could be employed for other video data which have the hierarchical structure of scenes and which are constrained by production economics, formal structure, and/or human perceptive limits to re-use their physical settings.

In order to represent a scene, we are only interested in those shots that have the most information about the physical scene. Once these shots are determined, their corresponding mosaics are chosen as the representative mosaics (“R-Mosaics”) of the scene. An illustration of a scene in a sitcom is shown in Figure 5.4. Shots that were photographed from different cameras are visually very different, even though they all belong to the same scene and were taken at the same physical setting. One example of a “good shot” is an extended panning shot, as shown in Figure 4.1. Another example is a zoom shot; the zoomed-out portion of the shot will most likely show a wider view of the background, hence expose more parts of the physical setting. Detecting “good” shots is done by examining the registration transformations computed between consecutive frames in the shot. By analyzing these transforma-
Figure 5.4: An illustration of a scene in a sitcom. Three cameras are used here. The bottom one is used to photograph the “establishing shot”, the first shot which is a zoomed-out shot introducing the setting and the characters participating in the scene. The other two cameras are used to photograph repeating zoomed-in shots that displaying a dialog. Note that even though all shots took place in the same physical setting, they are visually very different.

...
with significant zoom or pan. For mosaics generated from sitcoms, further processing is needed, since some shots have significant parallax motion. We segment such shots into two parts and construct two separate mosaics for each part. In our experiments, this was sufficient and the resulting mosaics were accurate enough for comparison purposes.

Figure 5.5: Examples of transformations analysis for different shot types. For each shot, the affine transformation computed were consecutively applied on an initial square, each transformed quadrilateral was colored differently (from a set of 8 colors) for view purposes. (a) Static shot. (b) Panning shot. (c) Zoom-in with pan. (d) Panning shot with strong parallax in the middle of the shot. This caused the transformation summary display to be divided into two groups.

However, some scenes are mostly static, without pan or zoom shots. Moreover, sometimes the physical setting is not visible in the R-Mosaic because a pan shot was close up. For these scenes, an alternative shot which better represents the scene has to be chosen. We observed that most scenes in sitcoms (and all the static scenes that we processed) begin with an interior “establishing shot”, following basic rules of film editing[Arjón, 1976]. The use of an establishing shot appears necessary to permit the human perception of a change of scene; it is a wide-angle shot (a “full-shot” or “long-shot”), taken to allow identification of the location and/or characters participating in that scene. An example of an “establishing shot” appears at the bottom of Figure 5.4. Therefore, we choose the first interior shot of each scene to be an R-mosaic; for a static scene, it is the only R-mosaic. Many
indoor scenes also have an exterior shot preceding the interior “establishing shot”, which photographs the building from the outside. We can easily detect this shot since it does not cluster with the rest of the following shots into their scene, and also does not cluster with the previous shots into the preceding scene. Instead, it is determined as a unique cluster of one scene (this was also found by [Hanjalic et al., 1999]). We detect those unique clusters and disregard them when constructing the scene representations. In our experiments, using these pan, zoom, and establishing shot rules leads to up to six R-Mosaics per scene.

Once R-mosaics are chosen, we can use them to compare and cluster scenes. The dissimilarity between two scenes is determined by the minimum dissimilarity between any of their R-Mosaics, as illustrated in Figure 5.6. This is due to the fact that different shots in a scene might contain different parts of the background, and when attempting to match scenes that share the same physical location, we need to find at least one pair of shots (mosaics) from the two scenes that show the same part of the background.

After determining the distance measure between each scene within an episode, we can construct a scene difference matrix where each entry \((i, j)\) corresponds to the difference measure between scene \(i\) and scene \(j\). An example is shown in Figure 5.7, where the entries in the scene difference matrix were arranged manually so that they correspond to the 5 physical settings of this episode. More specifically, scenes 1, 3, 7, 12 took place in the setting which we marked as “Apartment1”, scenes 2, 6, 9, 11 took place in “Apartment2”, scenes 5, 10 took place in “Coffee Shop”, scenes 4, 13 took place in “Bedroom1” and scene 8 took place in “Bedroom2”. 
Figure 5.6: Dissimilarity between two scenes is determined as the minimum dissimilarity of any of their R-Mosaics. This is due to the fact that different shots in a scene might contain different parts of the background, and when attempting to match scenes that share the same physical location, we need to find at least one pair of shots (mosaics) from the two scenes that show the same part of the background.

5.3.2 Video Representation by Physical Settings

Although the scene clustering process results typically in 5-6 physical settings, there are 1-6 scenes in each physical setting cluster, and about 1-6 mosaics representing each scene. Ideally, for the purposes of display and user interface, we would like to choose a single mosaic to represent each physical setting. However, this is not always possible. Shots of scenes in the same physical setting, and sometimes even within the same scene, are filmed using cameras in various locations which show different parts of the background. Therefore, two mosaics of the same physical setting might not even have any corresponding regions. We want the representation of a physical setting to include all parts of the background which are relevant to that setting. Therefore, if there isn’t a single mosaic which represents the whole background, we choose several mosaics which together cover the whole background.

We use the results of the matching algorithm’s finer stage, which recognizes
Figure 5.7: Example of clustering scenes for a single episode. On the left is the scene list in temporal order, and on the right is the similarity graph for the scene clustering results, in matrix form (darker = more similar). Scene entries are arranged manually in the order of physical settings. For example, there are 13 scenes in the episode represented in the middle grouping, scenes 1, 3, 7, 12 are in “Apartment1”, scenes 2, 6, 9, 11 are in “Apartment2”, scenes 5, 10 are in “Coffee Shop”, scenes 4, 13 are in “Bedroom1” and scene 8 is in “Bedroom2”.

corresponding regions in the mosaics, to determine a “minimal covering set” of mosaics for each physical setting. We approximate this set (since the real solution is an NP-hard problem) by clustering all the representative mosaics of all the scenes of one physical setting and choosing a single mosaic to represent each cluster. This single mosaic is the centroid of the cluster, e.g., it is the mosaic which has the best average match value to the rest of the mosaics in that cluster. This is achieved in two stages: first by identifying separate clusters, where each cluster corresponds to mosaics which show corresponding areas in the background of the particular physical setting; in the second stage a distance matrix between the mosaics of each
cluster is computed and the mosaic with the minimal average distance from all mosaics is chosen as the centroid of that cluster.

An example of R-Mosaics for one episode is shown in Figure 5.8. There are 5 physical settings and 13 scenes, and their relations are shown by the colored lines. For display purposes (see also chapter 6), we choose only one mosaic from the largest cluster to represent each setting. In our experience the largest cluster was always the most appropriate for choosing the single R-Mosaic.

Table 5.1 illustrates the compactness of our representation method using settings, scenes, shots and frames of a single episode. In this episode, there are 5 settings, each represented by a single R-Mosaic (the R-Mosaics are referred to by their corresponding shot number), and has 1-4 scenes. Each scene is represented by
<table>
<thead>
<tr>
<th>Video</th>
<th>Physical Settings</th>
<th>R-Mosaics (Settings)</th>
<th>Scenes</th>
<th>R-Mosaics (Scenes)</th>
<th>Interior Scene Shots</th>
<th>Frames</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Episode 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Apartment 1</td>
<td>225</td>
<td>1</td>
<td>2</td>
<td>2 – 14</td>
<td>1227 - 3012</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>41</td>
<td>41 – 57</td>
<td>5743 - 8112</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>7</td>
<td>121, 123</td>
<td>121 – 139</td>
<td>17802 - 20263</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>12</td>
<td>220, 225, 230</td>
<td>220 – 237</td>
<td>32284 – 36384</td>
</tr>
<tr>
<td></td>
<td>Apartment 2</td>
<td>31</td>
<td>2</td>
<td>15, 21, 31</td>
<td>15 – 40</td>
<td>3013 - 5742</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6</td>
<td>94, 114</td>
<td>94 – 120</td>
<td>14296 - 17784</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>9</td>
<td>155, 159</td>
<td>155 – 167</td>
<td>22652 - 24337</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>11</td>
<td>198, 204, 210</td>
<td>198 – 217</td>
<td>28961 - 32178</td>
</tr>
<tr>
<td></td>
<td>Coffee Shop</td>
<td>174</td>
<td>5</td>
<td>71, 79, 80, 83</td>
<td>71 – 92</td>
<td>11670 - 14238</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td>169, 174, 175</td>
<td>169 – 196</td>
<td>24450 - 28894</td>
</tr>
<tr>
<td></td>
<td>Bedroom 1</td>
<td>59</td>
<td>4</td>
<td>59</td>
<td>59 – 70</td>
<td>8163 - 11666</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>13</td>
<td>238</td>
<td>238 – 253</td>
<td>36385 - 37852</td>
</tr>
<tr>
<td></td>
<td>Bedroom 2</td>
<td>140</td>
<td>8</td>
<td>140</td>
<td>140 – 154</td>
<td>20216 - 22651</td>
</tr>
</tbody>
</table>

Table 5.1: Table summary for volume of data used for one episode. R-mosaics are referred to by their corresponding shot number. Interior scene shots are marked by start and end shot indices.
only 1-4 R-Mosaics, and has 11-26 shots, i.e. approximately 2400 - 11100 frames. More tables for the rest of the episodes that we gave processed are available in Appendix C.

The result of the mosaic-based scene clustering and the construction of physical settings for three episodes of the same sitcom are shown in Figure 5.9. The order of the entries in the original scene difference matrices were manually arranged to reflect the order of the reappearance of the same physical settings. Dark blocks represent good matches (low difference score). As can be seen in the left column of the middle sitcom, there are 5 main clusters representing the 5 different scene locations. For example, the first cluster along the main diagonal is a $4 \times 4$ square representing scenes from “Apartment1”. The false similarity values outside this square are due to actual physical similarities shared by “Apartment1” and “Apartment2”, for example, light brown kitchen closets. Nevertheless, it did not affect our scene clustering results, as can be seen in the corresponding diagram in the middle column in Figure 5.9(b), and the dendrogram generated by the results of applying a weighted-average clustering algorithm [Jain and Dubes, 1988] to the original scene difference matrix on its left.

5.3.3 Comparing Physical Settings Across Videos

When grouping information from several episodes of the same sitcom, we detect repeating physical settings. It is often the case that each episode has 2-3 settings which are unique to that episode, and 2-3 more settings which are common and recur in other episodes. We summarized the clustering information from three episodes of the same sitcom and computed similarities across the videos based on
Figure 5.9: Results from three episodes of the same sitcom. (a) Similarity graphs for scene clustering results, in matrix form as was shown in Figure 5.7. (b) Corresponding dendrograms (generated using the tools available in [Kleiweg, ]). Each cluster in each dendrogram represents a different physical setting. (c) Illustration of the results of inter-video comparison of physical settings. Settings appear in the same order they were clustered in (b). Lines join matching physical settings, which are common settings in most episodes of this sitcom: “Apartment1”, “Apartment2” and “Coffee Shop”. The rest of the settings in each episode identify the main plots of the episode, for example, the main plots in the first episode took place in the settings: jail, airport and dance class.
physical settings. We define the distance between two physical settings to be the minimum distance between their R-Mosaics. To compare between three long video sequences, each about 40K frames long, we only need to compare 5 settings of the first episode with 5 settings of the second episode and 6 settings of the third episode.

Comparing settings across episodes leads to a higher-level contextual identification of the plots in each episode, characterized by settings which are unique to that episode. For example, in the episode at the top of Figure 5.9 the main plots involve activities in a dance class, jail and airport. We have observed that most sitcoms do in fact involve two or three plots, and anecdotal feedback from human observers suggests that people do relate the plot to the unusual setting in which it occurs. This non-temporal indexing of a video within a library of videos is directly analogous to the Term Frequency/Inverted Document Frequency (TFIDF [Salton and McGill, 1983]) measure of information retrieval used for documents. That is, what makes a video unique is the use of settings which are unusual with respect to the library. Finally, the descriptive textual labeling of physical settings is done manually. Since there are usually only a few common settings for each sitcom (due to economic constraints on set design and to sitcom writing rules), there are only 2-3 additional settings for every newly added episode.

Another example, of a fourth episode that we have analyzed, is shown in Figure 5.10. There are 6 true physical settings here: “Apartment 1”, “Apartment 2”, “Coffee Shop”, “Party”, “Tennis game” and “Boss’s Dinner”. In this episode the clustering of scenes together into physical settings was not as straightforward as in the first three episodes. This is due to the fact that the setting of “Apartment 1” was not presented in the same manner, since its scenes either took place in the
Figure 5.10: Clustering results from a fourth episode: (a) Scene distance matrix with entries manually re-ordered to match physical settings. Here the clustering was different since the first cluster names “Apartment 1” is sub-divided into two clusters, one of scene 1 which took place in the living room, and the second of scenes which took place in the kitchen. (b) Dendrogram generated for the scene distance matrix of this episode.

kitchen or in the living room, but not in both. Mosaics which were generated from shots in scenes that took place in the kitchen of “Apartment 1” have no overlap with mosaics which were generated from shots in scenes that took place in the living room of “Apartment 1”. This separates scenes of “Apartment 1” into two clusters, as shown in the top left part of Figure 5.10(a): scene 1 belongs to the living room cluster, and scenes 7, 9 and 12 belong to the kitchen cluster. The clustering also doesn’t separates scene 11 well from the rest of the scenes, since this new setting took place in an apartment with similar colored walls and windows as the colors appearing in the setting “Coffee Shop”, into which it is loosely clustered. However, when these settings are compared with settings of different episodes, the splitting of the setting “Apartment 1” into two separate settings is corrected. In the context of all the episodes of this sitcom, the setting of “Apartment 1” includes mosaics of both the living room and the kitchen, causing the two different settings of this episode to be combined together. More specifically, the “Apartment 1” setting
cluster already contains mosaics that match both scene 1 and scenes 7, 9 and 12 from the new episode.

This example demonstrates how the non-temporal level of abstraction of a single video could be verified and corrected by semantic inference to other videos. Scenes and settings that otherwise would not have been grouped together are related by a type of 'learning' from the previously detected “physical setting” structure in other episodes.

5.4 Analysis and Evaluation

Since the concept of segmentation of scenes into physical setting is new and was not discussed in previous work, we evaluate the clustering results by comparing to ground truth. Determining the correct non-temporal segmentation of video into physical settings is a rather subjective matter. However, in sitcoms the physical settings structure is well defined and it is rather straightforward to distinguish between them. This is mainly due to cost considerations which limit the number of different sets used for each sitcom, as well as the very pronounced colors which are often used in different settings, probably to attract viewer attention. As can be seen in Figure 5.9, the scene dissimilarity measure is what determines the accuracy of the physical settings detection. Different clustering methods would result in the same physical settings cluster structure as long as the scenes distance matrix has the correct values. In our experiments, the scene dissimilarity did not result in perfect precision values even after the finer alignment stage. Some scene pairs that did not have any corresponding regions, had relatively high match values. This,
however, did not affect the clustering results for the first three episodes that we analyzed, which were perfect. In the case of the fourth episode, the inter-video comparison of physical settings managed to correct the clustering results for the first setting of “Apartment 1”, but the clustering threshold was not as pronounced as in the first three episodes. Depending on this threshold, for large values scene 11 could be wrongly clustered with scenes 14, 2 and 4, and for small values scene 15 will not be clustered with scenes 5, 8, 10, and 13, as it should.

The complexity of the scene clustering method is very low. Since all mosaic pairs are matched, if there are $M$ mosaics in an episode, then $M^2$ coarse match stages will be performed, after which only several mosaic pairs will be matched in the finer stage. In our experiments this number of pairs was on the order of $O(2M)$. Once the scene distance matrix is constructed, the physical settings are determined using any clustering algorithm, which we consider as a black box. Since the maximum number of scenes encountered in sitcoms was 15, there are up to 15 elements to cluster, causing every clustering algorithm to run very fast. Finally, when comparing physical settings across episodes, there are only 5 to 6 settings in each episode, each represented by no more than 3 mosaics, which also makes the comparison process very efficient.

\section*{5.5 Conclusion}

We show a compact approach for summarizing video, which allows efficient access, storage and indexing of video data. The mosaic-based representation allows direct and immediate access to the physical information of scenes. It therefore allows not only efficient and accurate comparison of shots and scenes, but also the detection
and highlighting of common physical regions in mosaics of different sizes and scales.

We also presented a new type of video abstraction. By clustering scenes into an even higher level representation, the physical setting, we create a non-temporal organization of video. This structured representation of video enables reliable comparison between different video sequences, therefore allows both intra- and inter-video content-based access. This approach leads to a higher-level contextual identification of plots in different episodes. This structured mosaic-based representation is not confined to the genres of situation comedies, and we show an example of shots clustering for event classification in basketball sequences.

The method described here would serve as a useful tool for content-based video access. Video sequences are represented in their multiple levels of organization in the following hierarchical structure: frames, shots, scenes, settings and plots. This allows both a temporal representation of video for fast browsing purposes as well as a non-temporal representation for efficient indexing. An example tree-like hierarchical representation is shown in Figure 5.11. We have used mosaics to represent the physical settings, scenes and shots of a single episode (we used key frames to represent the lower level of the video, the frames themselves).
Figure 5.11: Hierarchical display of video, illustrating both temporal (frames, shots and scenes) and non-temporal (physical settings) representation. Mosaics are used to represent each level (except frame level).
Chapter 6

A Video Browsing Tool

This chapter presents an example video browser specialized for sitcoms, which utilizes our proposed tree-like hierarchical structure. It uses the video data gathered from all levels of representation of the video. At the frame level, the mpeg video format of each episode is used for viewing video segments of the original video. At the shot level, it uses a list of all marked shots for each episode, including the start and end frame of each shot. At the scene level, it uses a list of all marked scenes for each episode, including the start and end shots of each scene. For each scene, a list of representative shots is kept, and their corresponding image mosaics are used for display within the browser. At the physical setting level, it uses a list of all detected physical settings for each episode, with their corresponding hand-labeled descriptions (e.g. “Apartment 1”, “Coffee Shop”). Each physical setting has a single representative image mosaic, used for display. The collection of all this data was described in previous chapter, and its use will be summarized and demonstrated in this chapter.
6.1 Hierarchical Vs. Temporal Browsing

We have constructed a browsing tool that combines the ability to index a library of videos by both compact semantic representations of videos as well as temporal representations. The compact visual summary enables cross-referencing of different episodes and fast main plot analysis. The temporal display is used for fast browsing of each episode.

This browser was programmed using Java, and we would like to thank Lijun Tang for his hard work.

The main menu is displayed as a table-like summary in a single window. Each row in the table represents one episode of the specified sitcom. The columns represent different physical settings that were determined during the clustering phase of scenes for all episodes. Each cell \((i, j)\) in the table is either empty (setting \(j\) does not appear in episode \(i\)) or displays a representative mosaic for setting \(j\), taken from episode \(i\). The order of the columns from left to right is organized from the most common to the non-common settings. In our example, the first 3 columns represent common settings which repeat in almost every episode of the specific sitcom. The rest of the columns are unique for each episode. In this manner, the user can immediately recognize the main plots for each episode, by looking for non-empty cells in the row of that episode starting from the fourth column. For example, for the episode marked as “Friends 2” in the top row of Figure 6.1(a), the main plots involve scenes taking place in settings “Bedroom1” and “Bedroom2” (columns 4 and 5 from left). In order to confirm the main plots quickly, it is sufficient to left-click on the representative mosaics for these settings (Figure 6.2(b)), which displays a window of a short list of scene mosaics that correspond to those settings.
(usually one or two), and if further needed, double-clicking on the representative mosaic for each scene will start playing the video from the beginning of that scene (Figure 6.3(b)).

The temporal representation of each episode is also accessed from the main menu and is used for fast browsing of that episode. By left-clicking on a certain episode name, as shown in Figure 6.1(b), a window of a list of all scene mosaics belonging to that episode appears. Each scene on that list is represented by a single mosaic and it is optionally expanded by left-clicking into a window of a list of representative mosaics (shots) for that scene (Figure 6.3(a)). The fast browsing is performed by scanning the scenes in order and only playing relevant video segments from chosen scenes by double-clicking on them, as shown in Figure 6.3(b).

6.2 Browser Evaluation

Our browser has the advantage of being both hierarchical in displaying semantically oriented visual summaries of videos in a non-temporal tree-like fashion as well as semantically relating different episodes of the same sitcom to each other. We have tested its usefulness by analyzing feedback from several subjects who follow the sitcom. We tested whether the temporal scene representation allowed meaningful fast browsing, as well as whether they were able to recognize the main plots of each episode using our menus, and how fast they performed. The feedbacks were encouraging: main plots of familiar episodes were recognized within 2-3 minutes, which included first browsing the scenes for a general impression of temporal flow, and then by clicking on non-common settings and viewing the video. One subject claimed that he might have been able to recognize the plots in an episode in the
same amount of time simply by performing fast browsing of the video. He admitted, however, that without the temporal ordering of scenes and the ability to jump and play the video at each scene, it would be possible to miss important scenes and possible plots. The most interesting result was that almost all of the main plots (except one) were recognized **without** playing the corresponding audio - only the video was played. This proved the highly semantic value of the physical setting in summarizing and then recalling a video.
Figure 6.1: Temporal Vs. Non-Temporal representation of episodes: (a) Single window showing all episodes and physical settings, displayed as a table where each episode has entries (mosaic images) only in relevant settings. This is a very compact non-temporal representation of a video. (b) By left-clicking on the middle episode “Friends3”, it is expanded to show its temporal representation of 13 scenes; each scene is represented by a single mosaic.
Figure 6.2: Example menus of browsing tool. (a) Right-click on a single mosaic (first row, second from left) to enlarge it. (b) Left-click on a single mosaic (leftmost mosaic on first row, representative mosaic for setting “Apartment1” of episode “Friends3” ) to expand that physical setting, showing all scenes of that setting. (c) By left-clicking on second mosaic from left in the menu shown in (b), scene #3 is chosen and its representative mosaics are enlarged and displayed. (d) By double-clicking on the third mosaic from left in the menu shown in (b), the movie clip of scene #7 starting from the first shot of that scene is displayed.
Figure 6.3: Example menus of browsing tool. (a) By left-clicking on second mosaic from left in the menu shown in Figure 6.2(b), scene #3 is chosen and its representative mosaics are enlarged and displayed. (b) By double-clicking on the third mosaic from left in the menu shown in Figure 6.2(b), the movie clip of scene #7 starting from the first shot of that scene is displayed.
Chapter 7

Conclusion

The scene’s physical setting is one of its most prominent properties in many video genres. Each scene takes place at one specific physical setting, which usually changes between two consecutive scenes. Also, there is a strong correlation between the scene’s location and its respective part in the general plot. If every shot was labeled according to its physical location the tasks of scene segmentation and video summarization in general would be much easier.

However, since we usually do not have this labeling into physical locations, an alternative is required. In this work we suggest the use of mosaics automatically constructed from the video instead. A mosaic is more powerful than a collection of key-frames since it also holds information about the spatial order of its constructing frames. It also eliminates much of the redundancy caused by overlap in the spatial domain, and is robust to many temporal changes (e.g. moving objects).

Mosaics are not an ideal representation - they are not well defined for complex settings and general camera motion (causing distortions in the mosaics). They are also not invariant to camera viewing point and even for the choice of the reference
frame. However, we claim that they are flexible enough to serve as a good representation and indexing of video, by using the mosaic alignment and comparison method we present here.

Our alignment method makes use of several assumptions about the video from which the mosaics are constructed. These assumption are based directly on the underlying grammar rules of common video genres. One assumption is the assumption of lateral camera motion. Another assumption is the assumption of controlled lighting. Both assumptions hold for video genres such as sitcoms and indoor sports as well as for other genres. By relying on these assumptions we were able to propose a fast alignment algorithm and an associated distance measures between shots, scenes and physical settings. Although the alignment algorithm is not an accurate registration algorithm (the transformation between mosaics of the same settings is not given by a limited number of parameters), we show that the distance measure is effective.

We believe that our comparison by alignment technique could be extended to more general image retrieval applications. In contrast to many image retrieval techniques that use global color features of single images, our technique incorporates spatial information by applying a coarse alignment between the images. It is robust to occluding objects and will match images for which only partial regions match (for example, top-left region of one image matches bottom-right region of the second image). This comparison method could be made more robust to lighting changes by investigating more elaborate vector quantization methods. It could also be improved by removing the horizontal constraint. (Currently our comparison method demands that two matched images are “horizontally” aligned, that is, it does not
support matching of rotated images as reported in other wide baseline matching algorithms [Schaalitzky and Zisserman, 2001]).

Using mosaic comparison as our engine, we have constructed a complete system which takes raw video data and produces video summaries. Our proposed summary is based on an hierarchical structure capturing the semantic structure of video. It enables efficient cross-referencing between different episodes of the same sitcom which further infers their main plots. In this work we describe all the stages of the system, from temporal segmentation into shots and scenes, throughout mosaic-based representation of each shot and scene, and reliable clustering into physical settings. We demonstrate the use of this hierarchical structure with a browser specialized for sitcoms, for the unprofessional user.

At the lower levels of the hierarchical structure, we proposed using a unified approach to scene and shot detection and key-frame selection, which is based on a leaky-memory based model. This model deduces a use of exponentially decaying weights for integrating frame differences over time. We used this combined measurement to determine shot transitions (at the maxima) and choose representative key frames (at the minima). Previous approaches for scene detection without using predefined thresholds used clustering for that purposes.

For our higher level representations, the use of clustering is essential. The underlying reason is that scenes and shots are temporal segments while physical settings are discrete, non-ordered segments. Therefore, integrating over time will not be meaningful for physical settings.
7.1 Future Work

Shot transition detection is a well explored subject, therefore we will focus on expanding other parts of this work.

A fundamental issue which is relevant not only for our applications is improving the mosaic comparison and alignment technique. This technique is a variant of the wide baseline matching technique, which is applied on images. It allows matching and aligning video segments taken from varying view points and under varying lighting conditions, by comparing the mosaics constructed for each video segment. In contrast to known wide baseline matching techniques applied to images, the proposed comparison technique across mosaics performed relatively well even for the following three cases:

1. mosaics with local and global distortions,

2. mosaics with parts occluded by actors,

3. mosaics of shots that were filmed under different indoor lighting conditions.

However, the alignment could be made more accurate and more robust to scale, orientation and illumination changes. One suggestion allows larger scale differences between the mosaics, by combining histograms from several strips/blocks from one mosaic into a single histogram to be matched with a corresponding histogram of a single strip/block from a second mosaic.

The mosaic comparison and alignment technique was based on the assumption that the mosaics generated from each shot are of horizontal nature and that the background has a distinctive spatial color layout. This holds both for sitcoms
as well as for basketball fields, for which we also applied mosaic comparison between shots. The segmentation of mosaics into strips and blocks could be improved by adjusting the strip width and orientation to match actual scene features. By analyzing the spatial layout of color segments within the mosaic it is possible to first adjust the strip width such that each strip will represent mostly uniform color regions. Edge and line detection techniques could assist in determining the true horizontal orientation for most scenes types. More care in constructing color histograms and computing histogram differences could be invested. This will make the mosaic comparison technique more robust and accurate.

Another important field is the generation of meaningful video abstracts. The display of our video summary is mainly based on visual information. Many video abstraction systems rely on textual information, either automatically extracted (usually from captions) or manually added. Our only use of text is in the final labeling of physical settings, after the clustering of settings across videos. After this stage only $2 - 3$ unique settings in each episode and $3$ common settings for the entire sitcom need to be labeled. If we limit ourselves to visual summaries, the use of an episode’s specific physical settings seems to be one of the best possible choices for very short summaries. It could be improved, for example, by adding some carefully chosen segmented characters to the representing image mosaic. A question left for future work is how to choose these added elements.
Appendix A

A.1 Comparison of Scene Clustering Results for Different Color Spaces

Table A.1 shows results from our mosaic matching by alignment method which was applied using different color spaces. 27 mosaics from a single episode were compared, resulting in a total of 351 mosaic pairs. There were only 37 mosaic pairs which had true corresponding regions.

Three different color spaces with different vector quantizations were tested. The table only shows results for the quantizations that gave the best results for each color space. $HSI$ space was used with polar coordinates and vector quantization as described in chapter 4. Best results were achieved for this space, using quantization of 3, 6, and 6 for the H, S, and I channels, respectively.

The first column in the table presents the number of missed detection (mosaic pairs that have corresponding regions but were not detected as matching). The second column in the table presents the number of false positives (mosaic pairs that have no corresponding regions but were determined as matching). The third
and fourth columns show recall and precision values, defined as follows:

\[
\text{Recall} = \frac{\text{Correctly detected matching mosaic pairs}}{\text{Total no. matching mosaic pairs}} \\
\text{Precision} = \frac{\text{Correctly detected matching mosaic pairs}}{\text{Total no. of mosaic pairs determined as matching}}
\]

<table>
<thead>
<tr>
<th></th>
<th>No. Missed</th>
<th>No. False Positives</th>
<th>Recall</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>HSI 3 6 6</td>
<td>0</td>
<td>61</td>
<td>100%</td>
<td>37.75%</td>
</tr>
<tr>
<td>HSI 3 18 6</td>
<td>4</td>
<td>43</td>
<td>89.18%</td>
<td>43.42%</td>
</tr>
<tr>
<td>RGB 18 18 18</td>
<td>1</td>
<td>148</td>
<td>97.29%</td>
<td>19.56%</td>
</tr>
<tr>
<td>RGB 8 8 8</td>
<td>0</td>
<td>293</td>
<td>100%</td>
<td>11.21%</td>
</tr>
<tr>
<td>YIQ 3 6 6</td>
<td>0</td>
<td>293</td>
<td>100%</td>
<td>11.21%</td>
</tr>
</tbody>
</table>

Table A.1: Summary of mosaic matching results for different color spaces, for a single episode. There were 27 mosaics generated for that episode resulting in a total of 351 mosaic pairs that were matched. Out of these, only 37 mosaic pairs had true corresponding regions. The evaluation measures are based on recall and precision. Best results were achieved for HSI space, using vector quantization of 3, 6, and 6 for the H, S, and I channels, respectively.

### A.2 Evaluation of Illumination Effects

Figure A.1 shows comparison results of fine alignment stage for mosaic pairs which correspond to the same physical setting. The setting was illuminated differently for the various shots from which these mosaics were constructed. There are only four sample images in the figure, and even though the lighting differences had not been quantified, the lighting differences are apparent to the human eye. which demonstrate that our use of HSI color histograms with vector quantization was
sufficiently accurate, since the interpolated values along the main diagonal in the strip-to-strip distance matrices infer good matchings.

Figure A.1: Comparison results of fine alignment stage for mosaic showing different illuminations of the same physical setting.
Appendix B

The following figures are dendrograms generated from the clustering results of shots in two basketball sequences, as described in chapter 5.

In these dendrogram only two main clusters were marked and distinguished: close-up shots and field shots. However, the cluster of close-up shots is constructed from several visually different clusters. These clusters represent close-up shots which show either the audience, the basketball field, or combinations of both. They are visually distinctive within the upper cluster in the dendrogram shown in Figure B.1. We have grouped all these close-up shots clusters together for shot labelling purposes.

Likewise, in the case of the cluster representing field shots, the coarse mosaic alignment did result in a single cluster, but the cluster could be further enhanced to distinguish between shots which either show only the left side of the field, the right side of the field, or the whole field. This could aid in distinguishing between events related to different teams.
Figure B.1: Dendrogram representing shot clustering results from the first basketball sequence. The upper marked red cluster represents close-up shots, and the lower marked blue cluster represents field shots.
Figure B.2: Dendrogram representing shot clustering results from the second basketball sequence. The upper marked red cluster represents close-up shots, and the lower marked blue cluster represents field shots.
Appendix C

<table>
<thead>
<tr>
<th>Video</th>
<th>Physical Settings</th>
<th>R-Mosaics (Settings)</th>
<th>Scenes</th>
<th>R-Mosaics (Scenes)</th>
<th>Interior Scene Shots</th>
<th>Frames</th>
</tr>
</thead>
<tbody>
<tr>
<td>Episode 3</td>
<td>Apartment 1</td>
<td>213</td>
<td>1</td>
<td>1, 6, 9, 11</td>
<td>1 – 20</td>
<td>65 - 2123</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>21, 22, 24</td>
<td>21 – 28</td>
<td>3210 - 4943</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>44, 53</td>
<td>44 – 55</td>
<td>7437 - 8872</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>8</td>
<td>138, 144</td>
<td>138 – 155</td>
<td>18280 - 20499</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>9</td>
<td>157, 165, 170</td>
<td>157 – 177</td>
<td>20600 - 22787</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>11</td>
<td>207, 211, 212, 213, 222</td>
<td>207 – 227</td>
<td>25364 - 28885</td>
</tr>
<tr>
<td></td>
<td>Coffee Shop</td>
<td>125</td>
<td>3</td>
<td>30, 33</td>
<td>30 – 42</td>
<td>5232 - 7221</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6</td>
<td>96, 97</td>
<td>96 – 105</td>
<td>13155 - 14595</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>7</td>
<td>108, 114, 123, 124, 125</td>
<td>108 – 136</td>
<td>14695 - 18249</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td>179, 199</td>
<td>179 – 206</td>
<td>22932 - 25363</td>
</tr>
<tr>
<td></td>
<td>Dance Class</td>
<td>57</td>
<td>5</td>
<td>57</td>
<td>56 – 95</td>
<td>8873 - 13154</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>13</td>
<td>257, 258</td>
<td>257 – 266</td>
<td>31904 - 32819</td>
</tr>
<tr>
<td></td>
<td>Jail</td>
<td>230</td>
<td>12</td>
<td>230</td>
<td>230 – 255</td>
<td>29020 - 31764</td>
</tr>
<tr>
<td></td>
<td>Airport</td>
<td>268</td>
<td>14</td>
<td>268</td>
<td>268 – 291</td>
<td>32982 - 36865</td>
</tr>
</tbody>
</table>

Table C.1: Table summary for volume of data used for episode 3. R-mosaics are referred to by their corresponding shot number.
<table>
<thead>
<tr>
<th>Video Settings</th>
<th>R-Mosaics (Settings)</th>
<th>Scenes</th>
<th>R-Mosaics (Scenes)</th>
<th>Interior Scene Shots</th>
<th>Frames</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Episode 4</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Apartment 1</td>
<td>266</td>
<td>5</td>
<td>93, 120, 122, 130, 132</td>
<td>93 – 137</td>
<td>11835 - 17234</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8</td>
<td>180, 186</td>
<td>180 – 198</td>
<td>20283 - 22428</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>217, 219, 230, 236</td>
<td>217 – 242</td>
<td>24252 - 26690</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12</td>
<td>264, 266, 285</td>
<td>264 – 292</td>
<td>29851 - 33227</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15</td>
<td>306, 308</td>
<td>306 – 308</td>
<td>34724 - 35196</td>
</tr>
<tr>
<td>Coffee Shop</td>
<td>4</td>
<td>1</td>
<td>1, 4</td>
<td>1 – 8</td>
<td>1 - 1444</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>10, 13, 25, 29</td>
<td>10 – 48</td>
<td>1498 - 6689</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>66, 74, 82</td>
<td>66 – 91</td>
<td>8832 - 11785</td>
</tr>
<tr>
<td></td>
<td></td>
<td>13</td>
<td>293, 295</td>
<td>293 – 297</td>
<td>33228 - 34043</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16</td>
<td>309</td>
<td>309 – 317</td>
<td>35197 - 36458</td>
</tr>
<tr>
<td>Apartment 2</td>
<td>299</td>
<td>3</td>
<td>50, 61, 62</td>
<td>50 – 64</td>
<td>6774 - 8778</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9</td>
<td>200, 206</td>
<td>200 – 216</td>
<td>22498 - 24251</td>
</tr>
<tr>
<td></td>
<td></td>
<td>14</td>
<td>298, 299, 300</td>
<td>298 – 305</td>
<td>34044 - 34723</td>
</tr>
<tr>
<td>Babysitter</td>
<td>139</td>
<td>6</td>
<td>139</td>
<td>139 – 151</td>
<td>17318 - 18381</td>
</tr>
<tr>
<td>Street</td>
<td>152</td>
<td>7</td>
<td>152</td>
<td>152 – 179</td>
<td>18382 - 20282</td>
</tr>
<tr>
<td>Bar</td>
<td>244</td>
<td>11</td>
<td>244</td>
<td>244 – 263</td>
<td>26749 - 29850</td>
</tr>
</tbody>
</table>

Table C.2: Table summary for volume of data used for episode 4. R-mosaics are referred to by their corresponding shot number.
<table>
<thead>
<tr>
<th>Video</th>
<th>Physical Settings</th>
<th>R-Mosaics (Settings)</th>
<th>Scenes</th>
<th>R-Mosaics (Scenes)</th>
<th>Interior Scene Shots</th>
<th>Frames</th>
</tr>
</thead>
<tbody>
<tr>
<td>Episode 1</td>
<td>Apartment 1</td>
<td>268</td>
<td>1</td>
<td>1, 5</td>
<td>1 – 15</td>
<td>770 - 2080</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>7</td>
<td>145, 153</td>
<td>145 – 154</td>
<td>19738 - 20900</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>9</td>
<td>201, 217</td>
<td>201 – 219</td>
<td>32839 - 35418</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>12</td>
<td>267, 268</td>
<td>267 – 272</td>
<td>40383 - 41170</td>
</tr>
<tr>
<td></td>
<td>Apartment 2</td>
<td>103</td>
<td>5</td>
<td>96, 103, 106</td>
<td>96 – 107</td>
<td>14892 - 16560</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>8</td>
<td>156, 162</td>
<td>156 – 199</td>
<td>21031 - 32775</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td>220, 221</td>
<td>220 – 240</td>
<td>35440 - 37174</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>13</td>
<td>273, 293</td>
<td>273 – 295</td>
<td>41171 - 44651</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>15</td>
<td>317</td>
<td>317 – 339</td>
<td>52685 - 55103</td>
</tr>
<tr>
<td></td>
<td>Coffee Shop</td>
<td>76</td>
<td>2</td>
<td>19</td>
<td>19 – 34</td>
<td>6774 - 8778</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>61, 72, 76</td>
<td>61 – 94</td>
<td>22498 - 24251</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>14</td>
<td>297, 313</td>
<td>297 – 317</td>
<td>34044 - 34723</td>
</tr>
<tr>
<td></td>
<td>Party</td>
<td>36</td>
<td>3</td>
<td>36</td>
<td>36 – 59</td>
<td>17318 - 18381</td>
</tr>
<tr>
<td></td>
<td>Tennis</td>
<td>109</td>
<td>6</td>
<td>109, 139</td>
<td>109 – 144</td>
<td>18382 - 20282</td>
</tr>
<tr>
<td></td>
<td>Boss Dinner</td>
<td>252</td>
<td>11</td>
<td>241, 252</td>
<td>241 – 266</td>
<td>26749 - 29850</td>
</tr>
</tbody>
</table>

Table C.3: Table summary for volume of data used for episode 1. R-mosaics are referred to by their corresponding shot number.
Bibliography


International Conference on Acoustics, Speech, and Signal Processing (ICASSP),
1997.

invariant texture matching and wide baseline stereo. In Proceedings of the Inter-

change detection. In In Proc. IS&T/SPIE Conf. Storage and Retrieval for Image
and Video Databases III 2420, 1995.

[Shahraray, 1995] B. Shahraray. Scene change detection and control-based sampling

[Shen and Delp, 1995] K. Shen and E.J. Delp. A fast algorithm for video parsing
using compressed sequences. In Proceedings International Conference on Image
Processing (ICIP), 1995.

acterization through the combination of image and language understanding. In
Proceedings of the IEEE International Workshop on Content-Based Access of Im-
age and Video Databases (ICCV'98), 1998.

robust features for classifying audio in the cuevideo system. In Proceedings of
the ACM Multimedia, 1999.


