UNIVERSITY OF MIAMI

SYNCHRONIZED AUDIOVISUAL SYNOPSIS USING
AUDIO SELF-SIMILARITY ANALYSIS, VIDEO SUMMARIZATION AND
MULTIMEDIA SEMANTIC DESCRIPTION TOOLS

By

Nicolas Betancur Ospina

A RESEARCH PROJECT

Submitted to the Faculty
of the University of Miami
in partial fulfillment of the requirements for
the degree of Master of Science in Music Engineering Technology

Coral Gables, Florida

May 2006
SYNCHRONIZED AUDIOVISUAL SYNOPSIS USING
AUDIO SELF-SIMILARITY ANALYSIS, VIDEO SUMMARIZATION AND
MULTIMEDIA SEMANTIC DESCRIPTION TOOLS

Nicolas Betancur Ospina

Approved:

_____________________    ____________________
Colby N. Leider     Dr. Edward P. Asmus
Professor of Music Engineering  Associate Dean of Graduate Studies

_____________________    ____________________
Ken C. Pohlmann     Dr. Mohammed A. Mottaleb
Assistant Professor of Music Engineering  Professor of Electrical Engineering
In a home video archival scenario, it is desirable to store only a summary of the best quality portions of the media. This equates to less required storage capacity and removal of regions which typically hinder the overall semantic intent of the document (scenes with excessive brightness or random, chaotic movement typical to home video). Towards constructing this summary, psychological studies have demonstrated that if the video is coupled with a professional-quality musical soundtrack to accompany the visual material, there is an overall increase in perceptual impact of the media. A novelty score according to self-similarity analysis is computed from any chosen audio track, which is segmented at maximum novelty points. Pictures together with the video material, constrained to the higher quality samples, are segmented temporally and aligned to match the audio segments, producing an initial audiovisual digest. A further enhancement is to semantically synchronize the specific media elements and events, based on the premise that artistic composition is the montage of passages that construct a tension/resolution progression. This is quantified using feature vectors for describing temporal and spectral features of the media and implemented as recommended by the MPEG-7 Standard. The result is a summary in which the different media elements are synchronized semantically, enhancing perception and providing a consolidated audiovisual experience.
DEDICATION

A mi Familia y a mi novia Margarita,

porque al recorrer este largo camino

nunca he caminado solo.
ACKNOWLEDGEMENTS

It is impossible to thank every person who contributed in some way to fulfilling this dream. I would first like to thank my Father for his incomparable wisdom and guidance. To my Mother and my Fiancée Margarita, who have always supported me, even when at times it meant giving up their self happiness. My sister for being the other side… the anti-engineering, chaotic and passionate side of beauty.

I would like to thank my friends back home, for showing me true and unconditional friendship, and for sharing the passion for music. It is you who motivated me to come here and follow this dream. To my friends in Miami, because I thought the gap of leaving such a treasure of my Colombian friends could never be filled… you more than proved me wrong. And of course… Oveal Walker, Arun Thirugnanasambandam and Karthik Palanichamy for sharing their knowledge, culture, friendship and all those silly jokes that didn’t let us study half the time.

Finally, I would like to thank my teachers and mentors both in Colombia and Miami for supporting my academic and professional realization. To Professors Ken Pohlmann, Colby Leider, Joe Abbati and Edward Asmus, for contributing their expertise towards providing me with the complete picture of Audio Engineering. Professor Abdel-Mottaleb for opening the doors to the world of Multimedia, which inspired this thesis. A special thanks to Mr. George Arriola at MoodLogic for helping me out with the perceptual attributes I needed for an adequate system profiling.

…And to all those who believe that Music Engineering is not just Engineering; it is also Music.
# TABLE OF CONTENTS

1 **INTRODUCTION** .......................................................................................................................... 1  
   1.1 **TECHNOLOGICAL MOTIVATION** .......................................................................................... 1  
   1.2 **RELATED WORKS** .................................................................................................................. 4  
      1.2.1 *Muvee Technologies* ........................................................................................................... 6  
      1.2.2 *Music Video Creation using Video Unsuitability and Audio Novelty* ....................... 7  
      1.2.3 *“Meta Script” Counterpoint Structure Description* ......................................................... 9  
   1.3 **INSTANTIATION OF THE PROPOSED SYSTEM** ................................................................. 10  

2 **CORRESPONDENCES OF MUSIC AND VISUAL MEDIA** ...................................................... 12  
   2.1 **VISUAL MUSIC BACKGROUND** ......................................................................................... 14  
   2.2 **AUDIO – VISUAL MAPPING** .............................................................................................. 16  
      2.2.1 *Three Different Flavors* .................................................................................................... 18  
      2.2.2 *The Tension/Resolution theory* ......................................................................................... 19  
      2.2.3 *Visual features that provide harmonic structure* ............................................................ 20  
   2.3 **MONTAGE** ........................................................................................................................... 23  
      2.3.1 *Metric Montage* ............................................................................................................... 24  
      2.3.2 *Rhythmic Montage* .......................................................................................................... 24  
      2.3.3 *Tonal Montage* ................................................................................................................. 26  
      2.3.4 *Multimedia Montage* ........................................................................................................ 27  
   2.4 **CONCEPTUAL PRESENTATION OF THE ALGORITHM** ..................................................... 28  

3 **MEDIA SEGMENTATION** ............................................................................................................ 32  
   3.1 **AUDIO** ............................................................................................................................... 32  
      3.1.1 *Self-Similarity Analysis* ...................................................................................................... 33  
      3.1.2 *Temporal Vs. Spectral Resolution* ..................................................................................... 35  
      3.1.3 *The Similarity Matrix* ....................................................................................................... 37  
      3.1.4 *Beat and Rhythm Detection* ............................................................................................... 42  
      3.1.5 *Audio Novelty Segmentation* ............................................................................................ 48  
   3.2 **VIDEO** .................................................................................................................................. 53  
      3.2.1 *HSV Color Space* .............................................................................................................. 54  
      3.2.2 *Video Unsuitability Segmentation* .................................................................................... 56  

4 **FEATURE EXTRACTION AND SEMANTIC PROFILING** .................................................... 60  
   4.1 **THE MPEG-7 MEDIA DESCRIPTION FRAMEWORK** .................................................... 60  
   4.2 **FEATURE EXTRACTION ALGORITHMS** ......................................................................... 66  
      4.2.1 *Audio* .................................................................................................................................. 66  
      4.2.2 *Video* .................................................................................................................................. 70  
   4.3 **MomLogic PERCEPTUAL ATTRIBUTES** .......................................................................... 71  
      4.3.1 *Genre Distribution* .......................................................................................................... 73  
      4.3.2 *Mood Distribution* ............................................................................................................. 74  
      4.3.3 *Perceptual Energy* ............................................................................................................. 74  
      4.3.4 *Perceptual Valence* .......................................................................................................... 74  
      4.3.5 *Perceptual Beat* ............................................................................................................... 75
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>IMPLEMENTATION</td>
<td>77</td>
</tr>
<tr>
<td>5.1</td>
<td>PREPROCESSING STAGE</td>
<td>77</td>
</tr>
<tr>
<td>5.2</td>
<td>FUNCTIONAL BLOCK DIAGRAM</td>
<td>79</td>
</tr>
<tr>
<td>5.3</td>
<td>SEGMENT CREATION BLOCK</td>
<td>81</td>
</tr>
<tr>
<td>5.4</td>
<td>MOOD/GENRE RECOGNITION BLOCK</td>
<td>82</td>
</tr>
<tr>
<td>5.5</td>
<td>PROFILES DATABASE</td>
<td>83</td>
</tr>
<tr>
<td>5.6</td>
<td>FEATURE VECTOR WEIGHTS PROFILING</td>
<td>84</td>
</tr>
<tr>
<td>5.7</td>
<td>VISUAL AND AUDIO SEGMENT DESCRIPTION BLOCKS</td>
<td>85</td>
</tr>
<tr>
<td>5.8</td>
<td>CLUSTERING OF MEDIA SEGMENTS</td>
<td>86</td>
</tr>
<tr>
<td>5.9</td>
<td>SUBSET MATCHING TO AUDIO TIMEFRAME</td>
<td>87</td>
</tr>
<tr>
<td>5.10</td>
<td>SYNCHRONIZATION TO RHYTHMIC ATTRIBUTES</td>
<td>88</td>
</tr>
<tr>
<td>5.11</td>
<td>REORGANIZATION OF AUDIOVISUAL STREAM</td>
<td>90</td>
</tr>
<tr>
<td>6</td>
<td>EXPERIMENTATION</td>
<td>92</td>
</tr>
<tr>
<td>6.1</td>
<td>PRELIMINARIES</td>
<td>92</td>
</tr>
<tr>
<td>6.2</td>
<td>METHODOLOGY AND PARAMETERIZATION</td>
<td>95</td>
</tr>
<tr>
<td>6.2.1</td>
<td>MoodLogic Queries</td>
<td>95</td>
</tr>
<tr>
<td>6.2.2</td>
<td>Profiles Definition</td>
<td>96</td>
</tr>
<tr>
<td>6.2.3</td>
<td>Instantiation of Thresholds</td>
<td>100</td>
</tr>
<tr>
<td>6.2.4</td>
<td>Visual test material</td>
<td>102</td>
</tr>
<tr>
<td>6.2.5</td>
<td>Programming language</td>
<td>103</td>
</tr>
<tr>
<td>6.3</td>
<td>RESULTS</td>
<td>104</td>
</tr>
<tr>
<td>6.3.1</td>
<td>Profiling</td>
<td>104</td>
</tr>
<tr>
<td>6.3.2</td>
<td>Visual Media</td>
<td>105</td>
</tr>
<tr>
<td>6.3.3</td>
<td>Soundtrack: “Wish You Were Here”</td>
<td>107</td>
</tr>
<tr>
<td>6.3.4</td>
<td>Soundtrack: “Galaxy Bounce”</td>
<td>110</td>
</tr>
<tr>
<td>6.3.5</td>
<td>Soundtrack: “Back To School”</td>
<td>113</td>
</tr>
<tr>
<td>6.3.6</td>
<td>Performance Measurements</td>
<td>116</td>
</tr>
<tr>
<td>6.4</td>
<td>ANALYSIS</td>
<td>118</td>
</tr>
<tr>
<td>7</td>
<td>CONCLUSIONS</td>
<td>121</td>
</tr>
<tr>
<td>7.1</td>
<td>FUTURE WORK</td>
<td>126</td>
</tr>
<tr>
<td>REFERENCES</td>
<td></td>
<td>131</td>
</tr>
</tbody>
</table>

APPENDIX A. MOODLOGIC QUERY

I “Wish You Were Here,” Incubus: | 135 |
II “Galaxy Bounce,” The Chemical Brothers: | 136 |
III “Back To School,” Danny Elfman: | 136 |

APPENDIX B. MATLAB IMPLEMENTATION

I TOP LEVEL | 137 |
II USER PROFILING | 140 |
III AUDIO ANALYSIS | 143 |
III VIDEO ANALYSIS | 153 |
IV IMAGE ANALYSIS | 160 |
# INDEX OF FIGURES

| Figure 1: | Extracting regions with the lowest Unsuitability Score | 6 |
| Figure 2: | Muvee Technologies Commercial logo | 6 |
| Figure 3: | Creating a Video Summary by synchronizing suitable video with novelty points in an audio soundtrack | 8 |
| Figure 4: | The evolution of shapes, colors and positions through time are useful for creating a visual web animation | 13 |
| Figure 5: | Music Video of Daft Punk’s “Around the World” | 16 |
| Figure 6: | John Whitney | 18 |
| Figure 7: | An inverted pyramid and its regular version, conforming a tension/resolution pair | 21 |
| Figure 8: | Mendelssohn’s Symphony No.4 – Italian; Second Movement | 25 |
| Figure 9: | Generic path of tension/resolution progression for a multimedia document | 30 |
| Figure 10: | Tension/Resolution progression including audio energy information | 31 |
| Figure 11: | Autocorrelation Analysis System | 33 |
| Figure 12: | Autocorrelation of a simple sine wave | 34 |
| Figure 13: | Similarity matrix computed for seconds 5-10 of the song “Galaxy Bounce” | 38 |
| Figure 14: | Block diagram for constructing a self-similarity matrix | 39 |
| Figure 15: | Parameterization of Audio Frames into MFCC’s | 40 |
| Figure 16: | Similarity matrix computed for seconds 0-10 of the song “Galaxy Bounce” | 42 |
| Figure 17: | Summation energy per column towards detecting meaningful audio events | 45 |
| Figure 18: | Beat Spectrum plot | 47 |
| Figure 19: | Similarity matrix computed for the song “Wish You Were Here” | 49 |
| Figure 20: | Checkerboard Kernel, which represents a situation of perfect Novelty | 51 |
| Figure 21: | Novelty Plot | 52 |
| Figure 22: | HSV Color Space | 54 |
| Figure 23: | Plot of HSV differential progression | 57 |
| Figure 24: | MPEG-7 Media Description Toolset | 65 |
| Figure 25: | Adaptive algorithm for pitch tracking | 68 |
| Figure 26: | Pitch following example | 69 |
| Figure 27: | Block Diagram for Basic Automatic Video Summary Creator | 78 |
| Figure 28: | Functional System Block Diagram | 80 |
| Figure 29: | Tension/Resolution ratio | 89 |
| Figure 30: | Tension/Resolution Curve for the Alternative Genre | 96 |
| Figure 31: | Tension/Resolution from Genre Distribution | 98 |
| Figure 32: | Basic Mapping of Low Level Media Descriptors | 100 |
| Figure 33: | Video Segmentation according to Unsuitability | 105 |
| Figure 34: | Histogram of elements in visual material (Clips & Images) | 106 |
| Figure 35: | Audio Analyses for “Wish You Were Here” | 108 |
| Figure 36: | Audiovisual Clustering | 109 |
| Figure 37: | Audio Analyses for “Galaxy Bounce” | 111 |
| Figure 38: | Audiovisual Clustering for “Galaxy Bounce” | 112 |
Figure 39: Analyses for “Back To School” ................................................................. 114
Figure 40: Audiovisual Clustering for “Back To School.” ......................................... 115
Figure 41: System Performance – Processor Load .................................................... 117
Figure 42: Memory Management ............................................................................. 118
Figure 43: Hard Drive Access .................................................................................. 118
INDEX OF TABLES

Table 1: MoodLogic Perceptual Attributes................................................................. 73
Table 2: Media Features and their Instantiation in the MPEG-7 Framework........... 76
Table 3: MoodLogic Query ..................................................................................... 95
Table 4: Instantiation of Thresholds ....................................................................... 101
Table 5: Anchor Genres Tension/Resolution Curves ............................................. 104
Table 6: Proportion of Video to Images. ............................................................... 110
Table 7: Proportion for “Galaxy Bounce” .............................................................. 113
Table 8: Proportion for “Back To School” ............................................................. 115
Table 9: Computer Testing Platform .................................................................... 116
1

INTRODUCTION

1.1 TECHNOLOGICAL MOTIVATION

The advent of digital technologies in the modern world creates the potential for any individual to become a multimedia content producer [1]. The average consumer, having access to systems such as digital cameras, personal computer systems and the Internet, now has the capabilities of creating multimedia documents that can be stored, published and distributed in the way only professional institutions were capable of less than a decade ago. Furthermore, this content may not only be stored and accessed from a local database but can also be made available on-line. However, producing content this easily poses the issue of managing it efficiently, given the vast and ever-increasing amount of information involved.

Considering this problem in the scope of amateur multimedia content production, the home user with the average digital camera and computer equipment can store thousands of pictures and hours of music and home video. This is made possible by standard compression technologies coupled with storage space presently in the order of hundreds of gigabytes. The result is a collection of documents of such size that even the person who created the content finds managing it a tedious and difficult challenge.

Video is increasingly being used towards documenting events in the home, such as vacations, ceremonies, family moments, and many other occasions. Outside, it is being used to document conferences, field work activities, etc. Most of it is created by users
who have no professional experience in recording or production. This creates the demand for video editor software that can handle these non-professional video scenarios [2].

A common fate for this kind of video collections is that it may be viewed once or twice, but then it is stored away and forgotten. A fundamental reason for this is that the interesting parts of the video are combined with much longer, uninteresting regions. Many sections are comprised of low-quality video resulting from over/underexposure, excessive and random camera movement or scene cuts, and simply static, boring video for long time spans. To watch these sections while waiting for only a few seconds of valuable material makes the experience of watching the video document a rather tedious one. Browsing for these exceptions is also a daunting task. Another issue is that the document takes a relatively large portion of memory for storage; something exponentially worse in an on-line scenario.

A solution is to segment the video in a way that only the interesting parts of the document are preserved, creating a summary commonly called a digest or “highlight reel.” However, this involves using video editors such as Adobe Premiere [3], Sony Vegas [4] and the like, which many users find too difficult and expensive to be worth the effort. Hence, a solution for performing this segmentation automatically is brought forth, providing the non-professional video creator with the following advantages [2]:

- Does not require the user to manually select start and end points for each useful clip.
- Provides a shorter, more interesting version of the video document.
- Produces a version of the document which is much smaller in size, making it more efficient towards storage, transmission and publication purposes (e.g., sharing with friends and family).

Another problem exists when considering the extremely low-quality audio that generally accompanies amateur video [5]. The microphones that are included in home video cameras, as well as the mechanisms for recording inside the device, are far from the quality of professional recording equipment. Furthermore, there is a lack of audio production in the recorded stream. This makes even professional-quality video be hindered in its overall perception; a fact supported by studies which show that perceived image quality is lowered by low-audio quality [6].

There is presently no method of automatically processing low-quality audio and increasing it to the level of a professional quality recording. There is however, the possibility to discard (or mix) the audio and replace it with a chosen soundtrack from the user’s personal music collection. This guarantees professionally produced audio to accompany the visual material. The result is a concise video digest that is accompanied by a soundtrack of the user’s choice. This could even enhance the perception of the video over that of its silent version [6].

A system can then be designed which analyzes the audio soundtrack for rhythm and overall temporal structure, meanwhile analyzing the raw, unedited home video for the highest quality visual regions [5]. Assuming the video track is longer than the audio, these regions may be extracted from the original and cut to match the temporal structure
of the audio soundtrack, which is a time-indexed novelty score indicating points in the audio where significant overall change is detected.

A “professional” looking video digest can thus be created. However, there exist many premises from the cross correlation of audio and visual media which have not yet been considered; media has been temporally synchronized without any regard to its content. If after initial segmentation, feature extraction is performed on the different media and linked according to certain audio-visual feature correspondences, levels of synaesthesia can be achieved which will at last provide, what is more than a good quality summary, a truly synchronized audiovisual document.

1.2 RELATED WORKS

In the field of video summarization, there are several alternatives available towards producing a valid document. The general approach to providing an accurate segmentation index is to extract features from the media stream and arrange them in a feature vector, assigning each one a specific weight factor representing the semantic contribution of each feature to the overall event situations that are to be detected. This feature vector provides a specific value (magnitude and angle) for each analyzed frame of media. The succession of values through time that this vector produces represents a curve with peaks which correspond to the maximum level of occurrence of the situation being searched for. Segmentation is consequently performed by assigning segment boundaries at peaks or valleys in the curve.

Several attempts have been made to extract highlights according to specific image and audio features or underlying textual annotations [7] [8] [9]. As an example [9], a
fully automatic soccer video abstraction system can be designed based on analysis for
pitch in audio, and dominant color plus motion on the visual domain. The proposed
approach is to base a feature vector on these three attributes, under the assumption that a
highlight of the game is generally accompanied by an excited commentary, cameras are
focused on the soccer field in a wide-angle shot (a green dominant color), and exhibits
considerable action, with an equivalent amount of camera motion. Hence, if these three
conditions are at a maximum simultaneously, this indicates a highlight in the game. The
system works with a good degree of accuracy, but it lacks two desired characteristics:

- It is based on a constrained media domain (i.e., soccer matches).
- It includes a professional grade audio soundtrack coming from the
television transmission, so it does not experience the problem of underlying
low quality audio.

This can be generalized towards all the systems in this category.

Other systems can provide video summaries using a generic approach; the media
features need not be constrained to a specific domain. They operate constructing the
feature vector from purely visual features. This makes them suitable for a home video
scenario, but implies lack of synchronism with a desired soundtrack. However, [2] is
relevant as a base for what is discussed in what follows, therefore its consideration is
worthwhile.

Hitchcock performs analysis on the video stream to determine an unsuitability
score. Unsuitable video represents low-quality artifacts found in home video such as fast
and erratic camera motion, or under/overexposure. Avoiding these artifacts is actually a
standard rule used in film edition. This score is used to identify candidate clips to include
in the final video specified automatically by their start and end points, corresponding to regions with minimum amounts of unsuitability, as shown in Figure 1. A user needs only to drag these clips into a storyboard interface, where final video is constructed. The user can also change the duration of specific clips or the document as a whole without ever needing to specify start and end frames manually.

![Figure 1: Extracting regions with the lowest Unsuitability Score [2].](image)

Moving on to systems that relate directly to the proposed algorithm, that is, they segment and summarize video according to features extracted from an audio soundtrack, there are mainly two [10] [5] to be considered.

### 1.2.1 Muvee Technologies

![Figure 2: Muvee Technologies Commercial logo](image)

Property of Muvee Technologies, All Rights Reserved.
Muvee [10] (logo of Figure 2) is a commercial software product designed for creating visual media summaries according to the analysis of video and a chosen audio soundtrack. It thus has a direct relationship with the proposed system. It is the pioneer in these kinds of technologies, with a history of 5 years of research. Details of the algorithms for feature extraction, mapping and segmentation are unavailable.

From what can be observed, Muvee acts as an editing system where the audio soundtrack’s temporal structure provides a rhythmic basis for segmentation. A user can profile the video outcome directly by choosing one of several predefined video styles. These styles are mapped into “editing rules” that control the pace of transitions and sensitivity to the rhythm of the soundtrack. This way, an “action” style video is full of transitions and is strongly linked to rhythmic components; on the other end of the spectrum a “calm” style is composed of smooth and slow transitions. Muvee can be considered a “state-of-the-art” home video production system if compared to the available technologies of today. However, no audio to visual mapping other than low level transient synchronization on the rhythmic structure of the soundtrack was identified.

1.2.2 Music Video Creation using Video Unsuitability and Audio Novelty.

This is a work which, parallel to the Muvee approach, addresses cross-media linking from a purely temporal standpoint. As opposed to Muvee, it does not represent a commercial venture but rather a compilation of research initiatives [11] based on similarity analysis in audio [12] and unsuitability analysis in video [2]. Details on algorithms and implementation are therefore available for this system. This is one of two reasons of why it was chosen as a pillar upon which the proposed system is constructed.
The other reason is that the audio analysis algorithms employed have a strong correlation to perceptual attributes; something extremely desirable in a cross-modal application such as this. The specific algorithms will be reviewed in Chapter 3.

Revisiting Hitchcock [2], the program was created by the same research group as the one that proposed this work. Therefore, the Unsuitability curve generated in automatic mode by this program is used as a valid segmentation algorithm for video media. As is discussed in Chapter 3, the chosen audio soundtrack is segmented according to a time-indexed Novelty score (i.e., the chosen feature for audio segmentation is a measure of Novelty). The peaks of the resulting curve represent maximum novelty peaks, which reflect a significant change in the audio autocorrelation at that specific point in time. Transitions in a song such as from verse to chorus, the beginning/end of a solo, a change in rhythm, etc, are identified by using this Novelty score. Video is linked to the audio by matching the video unsuitability and audio novelty curves, as shown in Figure 3.

![Figure 3: Creating a Video Summary by synchronizing suitable video with novelty points in an audio soundtrack [5].](image-url)
The audio is segmented at points of novelty peaks; this separates it by sections, as defined in classical music theory form. Much more about Novelty will be said in following chapters. The unsuitability curve in video is also analyzed to extract the regions which have the least amount of unsuitability, of equal duration to each respective audio segment. This guarantees usage of only the portions with highest video quality.

The results of this process are extremely accurate temporally, and actually use the best and most stable portions of video. However, pictures cannot be included in such an implementation, and again, as in Muvee, only temporal structure is considered. Semantic counterpoints and other audiovisual mappings discussed in Chapter 2 are not considered.

1.2.3 “Meta Script” Counterpoint Structure Description

This work is a research effort [13] [14] to quantify the idea of montage in multimedia documents. As with Novelty, Montage will be revisited throughout the following chapters. For now, the Wikipedia [15] definition for montage is sufficient: Musical montage (literally "putting together") is a technique where sound objects or compositions are created from collage. In the author’s scope, multimedia montage refers to the structural synthesis in time and space of multimedia components. This is applicable to the proposed system directly because the segmentation of video and subsequent pasting of a sequence of video clips and pictures aligned to an audio soundtrack is montage in the complete sense of the word.

The research work of the authors is focused on identifying Montage structures in multimedia analogous to the counterpoint structures found in music, subsequently defining a script language that describes multimedia documents according to this
parameter [14]. That this counterpoint concept can be quantified is fundamental to the proposition of the system, as it represents the psychological and semantic counterpart of existing applications (i.e., the ones discussed in the previous two sections). This is truly an inspiring work, as it provides validity to the theoretical concepts of cross-media semantic mapping upon which the proposed system is built. This is considered in detail in Chapter 2.

1.3 Instantiation of the Proposed System

After reviewing prior art, the proposed system can be identified and explained, as it uses the three reviewed systems to produce a synergetic combination. Its goal is to produce music videos which are a summary of raw unedited video and pictures taken by the home user, using an arbitrary chosen audio soundtrack to synchronize all the media. Up to this point, it conceptually tries to provide what [5] and [10] have already achieved. However, it goes further, building upon three pillars:

- It computes a self-similarity matrix as in [5], and uses this information in two ways: to create a Novelty curve and to construct a rhythmic structure for each section, which is another application of this matrix [16], to provide the temporal synchronization found in Muvee. It therefore uses the best of both approaches. It then segments the video according to unsuitability [2]. This is explained in depth in Chapter 3.
- It uses the concept of Montage [13] to construct a tension/release artistic multimedia piece. Furthermore, it provides a semantic and rhythmic link
between media low-level features such as pitch and energy for audio, and

color and brightness for video. This is discussed in Chapter 2.

- It also provides an enhancement to user profiling by integrating a query

for musical – video style matching made available by the MPEG-7

framework. This is explained in Chapter 4.

The result is an enhanced video summary creator, which synergistically combines

both rhythmic and semantic premises to finally create a truly consistent audiovisual
document.
CORRESPONDENCES OF MUSIC AND VISUAL MEDIA

The concept of interaction between audio and visual perceptions in art has been explored since ancient times by composers and artists, in forms such as theatre presentations and dance traditions, among many others. However, the exploration of the underlying psychological parameters which make an audiovisual composition coherent has been seldom considered, mainly because the initial number of variables which contribute to cognition in this area is enormous and still relatively unknown. Research has been focused on either the audio or the visual domains, but in the interaction of both media, studies are just beginning to emerge. The possibilities that computerized analysis has brought about have created interest in considering the parameters involved in these new multimedia forms.

In the scope of visual music, a parameter may be considered as the classical definition of “one of a set of independent variables that express the coordinates of a point” [17]. For example, a specific musical sound may (but does not have to) be described as a fixed point in the pitch, loudness and timbre space. In the visual domain, an image pixel can be instantiated using dimensions such as color, saturation, and brightness. Coordinates corresponding to more abstract features such as temporal and spatial information may also be used. Instantiating temporal progression as the
independent variable, the notion of visual animations and musical compositions consists of paths of evolution through the parameter space [17].

Figure 4 depicts this scenario for visual media. Supposing parameters are:

- Color: Yellow, blue and red
- Shape: Circle, quarter circle, curve, line
- Position: (X,Y) coordinates

A very simple visual composition can be created by an evolution of these parameters through time, as shown.

Figure 4: The evolution of shapes, colors and positions through time are useful for creating a visual web animation [18]
Having parameterized these attributes, an algorithmic approach to composition can now be adopted. If the evolution of parameters can be structured according to a rhythmic and harmonic basis, this creates the possibility of realizing a “visual music” [19]. This progression can then be mapped to coincide with the corresponding structure of a musical piece; this evolves the art to unprecedented levels of audiovisual synchronism, creating almost an art form in itself. The use of an automated system (presently a computerized platform) to aid in performing this synchronism more efficiently is what generally people understand as audio and visual multimedia synchronization.

2.1 **Visual Music Background**

The sub-domain of visual music found in multimedia is the scope of design that concerns this dissertation. For convenience purposes, this will be referred to from this point onward as simply “Visual Music.” Pioneering artwork would allegedly be the color organs of the 18th century, such as Castel’s Ocular Harpsichord [19]. These were organs that produced their typical sounds while simultaneously projecting light. A century later (1920), Thomas Wilfred performed on what he called a Clavilux, consisting of an electrical instrument that created clouds and streams of continuous color. The Art has of course enjoyed considerable growth throughout the 20th and 21st Centuries, supported by the technological revolutions of recent times.

The dominant approaches towards visual composition forms in recent history have followed Hollywood’s model of a narrative type storyboard creation; what is denominated as cinema or movies. Nonetheless, there has been a parallel development of
other compositions that are of non-narrative nature, and focus more on pure (audio)visual harmony. The earliest creators of such films were German filmmakers such as Walter Ruttmman, Viking Eggeling, Hans Richter and Oskar Fischinger [19]. The Art was approached initially in America by John and James Whitney, Mary Ellen Butte, Stan Brakhage, and Jordan Belson as well as Canadians Norman McLaren, Evelyn Lambart, and several others from the National Film Board of Canada.

In modern times, Visual Music has enjoyed a renewed interest from both the artistic and scientific communities. It is not uncommon to presence related works at any given performance in conferences all around the globe, such as the International Computer Music Conference (ICMC) [20]. Furthermore, other festivals are exclusively focused on visual music expressions; some examples are the 2003 Sonic Light Festival in Amsterdam, or the “Sons & Lumières, Une histoire du son dans l’art du 20e siècle” exhibit in France. In the United States, the exhibit “Visual Music: Synaesthesia in Art and Music Since 1900” has been in Los Angeles and Washington D.C. in recent years. The reader is encouraged to explore this vast world at www.iotacenter.org [21] and centerforvisualmusic.org [22], among many other sites on the internet.

An example encompassed in recent commercial western popular music is the video of French Electronic Music group Daft Punk’s “Around the World”, conducted by Director Michel Gondry. When discussing his artistic concept for creation of this video [23], he explained that each character in the video represented one of the instruments in the composition, and since all the instruments were sequenced, they could be translated into a human choreography such as the one depicted in Figure 5.
2.2 **Audio – Visual Mapping**

What is the semantic correspondence between musical and visual parameters that permits a coherent “Visual Music” composition? This question has in fact been posed not only by the composers just mentioned, but also by scientists, psychologists, philosophers… just about anybody interested in the Art. And as in any valid form of art, there is no single correct answer. There are only interpretations and approaches which provide aesthetic value to a given audience. A valid approach is to follow the simple premise that no matter what the art form, it is the resolution of tension what moves art
through time. According to Russian Filmmaker Serge Eisenstein (1949) [19], “Art is always conflict (...) In the moving image we have a synthesis – the spatial counterpoint of graphic art, and the temporal counterpoint of music”

Revisiting the first section of this chapter, if the evolution of media parameters through time is incorporated a structure of tension/resolution, this would suggest a rhythmic/harmonic progression of the media. The classical definitions of musical harmony can be then generalized to other media. To make this a valid relationship, consider what was proposed by Computer Art and Visual Music pioneer John Whitney (Shown in Figure 6) [25]:

*The foundation of my work rests first upon laws of harmony, then in turn, upon proof that the harmony is matched, part for part, in a world of visual design. (...) the attractive and repulsive forces of harmony’s consonant/dissonant patterns function outside the dominion of music. Attractions and repulsions abound in visual structures as they become patterned motion. This singular fact becomes a basis for visual harmony with a potential as broad as the historic principles of musical harmony.*

Based on this premise, to create a specific tension and release structure towards the evolution of media parameters is to adopt a composing style from an artist’s point of view, and to adopt a certain algorithmic set of rules for cross-modal correspondence (i.e., mapping) in an automatic multimedia analysis/synthesis scenario.
2.2.1 Three Different Flavors

Whitney’s definition is applicable to a vast diversity of artistic possibilities; something overwhelmingly difficult to encompass in a single automatic analysis and synthesis program. An algorithm design must, at least initially, be constrained towards a specific scenario. Three major possibilities exist:

- Visualization: Visual material is synthesized according to parameters extracted via the analysis of a pre-existent musical piece, conforming a “Sound to Light” application. This is Whitney’s realm, and probably the possibility that has historically been explored the most.
- Sonification: Control is generated by analysis of pre-existing visual material, towards synthesizing synchronized audio content in coherence with the visual composition.
- Synchronization: Both audio and visual media are pre-existent; the goal is to segment and/or synchronize one according to the parameters of the other. The case of both media being non-existent and being synthesized in conjunction is also a viable possibility [17].
A home video editing application such as the one proposed belongs to the third category. Therefore, the premises of Eisenstein, Whitney and any others must be accommodated to suit this type of media structure. This said, the exploration of audio-visual mappings continues.

2.2.2 The Tension/Resolution theory

Regarding Whitney’s works, his investigations of musical and visual harmony started by considering that a note by itself is not yet melody; a chord by itself is not yet harmony [25]. In other words, a note by itself or a chord by itself is not yet semantically valid. Music is motion (again, consider the evolution of parameters through time in this sense); it is the combinations of notes and chords which imply the musical progression. Furthermore, these combinations cannot be completely chaotic and disorganized; they must obey some kind of direction, or differential organization according to a certain parameter. He called this “Differential Dynamics,” and defined it as the basis of the patterns of which temporal art is constructed.

His conclusions constitute three important premises, useful in identifying these craved techniques, or algorithms, that will permit the creation of a Visual Music piece:

- Motion becomes a pattern if the objects in the visual space move differentially.
- A resolution towards order in the patterns of motion may be linked to points of resonance in the musical progression.
- Visual resolution at resonant events provides visual harmony. It is a way to resolve tension at meaningful moments.
These three conclusions define a structured motion; i.e., a harmonic and rhythmic progression of the visual material. At this point, a truism of music can now be generalized to the visual domain: “Structured motion begets Emotion” [25].

2.2.3 Visual features that provide harmonic structure

From the previous discussion, it is coherent to say that a set of visual parameters may be linked to a musical progression by evolving them according to differentials, mapped to a corresponding set of differentials in the music. Some parameters may be suitable semantically towards this purpose, some may not. If a specific one is suitable, then it will provide patterns that relate to the harmonic and rhythmic structure of music, creating visual tension that is resolved at resonant points in the musical progression.

A standard teaching in art and design schools is that of “visual consonance” [19], where the concepts of visually “right” and “wrong” expressions are explored. These terms do not correspond to their typical definitions of correct or incorrect; they refer more to the consonance and dissonance they produce in a given visual scenario. As an example, consider Figure 7. The first figure is a pyramid, oriented upside down. The second figure is the same pyramid, but in its upright orientation. The inverted version contains a much higher degree of visual tension than the regular one. This is because perceptually, the impression of the former is of a completely unstable structure while the latter is stable [28]. Showing the first and consequently the second in a movie is an example of a basic tension/resolution progression.
In this sense, evolving from “wrongness” to “rightness” coincides with moving from dissonance to consonance, or from tension to release. This moves the visual material within a musically structured composition. What are then, wrong and right visual constructs, and what are the underlying parameters?

Considering Whitney’s selections, motion is a valid parameter; it is extremely useful in a visualization scenario because an object can be moved around the screen, creating patterns that synchronize with the music. However, it does not directly relate to the proposed home video edition system because the visual material is already created. This bounds motion to whatever was recorded originally.

Another parameter is image proportion [19]. Western music theory is based on intervals, both spectrally and temporally. Frequency ratios and the relative consonance of intervals define musical scales as constructs upon which melodies and harmonies are based [29]. Temporal ratios are also employed in fundamental structures such as meter; a specific time signature is indicated by ratios or proportions, as in 4/4, 3/4, 9/8, etc. On the other hand, visual media also obeys its proportions. It has been common practice among diverse cultures and times to paint according to proportion. For example, the
golden ratio is geometrical proportion found in paintings that range from the pre-historic figures of the Lascaux caves in France, to Boticcelli’s “Birth of Venus” [19]. In more recent years, photography and film artists have employed ratios in defining camera angles and proportional views of recorded images.

Building tension and release according to proportion may be performed either in the spatial or temporal domain. Spatially, it consists of moving an image out of symmetry or balance (e.g., deviating from the center of the picture). Several techniques to enhance an image’s impact are commonly used in video edition, such as slow zoom or scrolling of images, as is standard in modern documentaries. In fact, Muvee Technologies [Muvee], a parallel implementation of this system discussed in Chapter 1, employs these rules of motion when exhibiting pictures. This would be something very desirable to include in this system.

Temporally, the building of tension and release may be even more significant. According to [19], “In temporal design, we move the viewer from visual dissonance to cadences of visually balanced, well composed moments. Time passes musically through patterns of tension/release.” Rhythmically, this can be achieved by including video cuts. It is observed that a cut is possibly one of the most visually violent moments in video, as it corresponds to a quantized “jump” between semantics. It is therefore logical to expect that video with a high density of cuts builds tension, and video with low density of video cuts resolves or relaxes tension. Once again considering Muvee [10], a “fast action” editing style is based on a high volume of fast and synchronized cuts, while passive styles consist of a lesser number of gradual transitions (i.e., crossfades). This functionality will be a basic point in the design considerations in later chapters.
Although not the opinion of Whitney [25], low level features such as color, saturation and brightness have been recently considered important in the construction of tension and release, if they are used correctly [19]. However, it is an inexact science and at the least, very confusing. The parameters considered up to now are absolute in the sense that every person perceives movements and scene cuts rather similarly. This does not occur with color perception. The first consideration is that the human visual system processes luminance (brightness) and chrominance (color) components separately [30]. Psychovisual studies go further and state that luminance is far more important perceptually than chrominance. But revisiting Whitney’s studies (performed at a much earlier time than when these facts about visual perception were known), he presents the following observation:

Experiment has indicated how ineffective full-screen passages for color, without other graphic form, can be – how ineffectual these are as a device to carry any rhythmic idea. Unless they are supported by a strong musical reinforcement, the effect is surprisingly ambiguous.

This is not implying that color is useless; it simply states that color attributes are not adequate for the rhythmic synchronization. Their function is therefore chosen as to provide other semantics coupled to the tension/release curve, which will be explained later.

2.3 MONTAGE

Eisenstein’s work [31] addresses tension/release based harmonic progressions by what is referred to in artistic terms as Montage [19]. A specific event or phrase is
denominated by Eisenstein as a “cell;” the connection or sequence of cells produces a gestalt, composing a semantically meaningful section, contributing to the piece’s Montage. Following what has been said in this chapter, “Montage is a multi-tiered construct of tension and release” [19]. Eisenstein defines different types of montage, which can be applied directly to visual music and the proposed system.

2.3.1 Metric Montage

Eisenstein defines Metric Montage as “pieces (which) are joined together according to their lengths in a formulaic scheme corresponding to a measure of music” [19]. Applied to the proposed system, this approach deals exclusively with the duration of each shot, and how cuts are synchronized to the musical rhythmic structure. Visual meter can even be established by providing scenes of the same duration according to the rhythmic meter of the song. However, according to developments in home video editing products, it is more perceptually meaningful in this scenario to bound video and pictures by specific sonic events without regard to such high level structures. Field for exploration is left open as a promising alternative.

2.3.2 Rhythmic Montage

A concept directly related to Metric Montage, although at a higher level of abstraction. Here, visual material is synchronized to the music at variable durations, not necessarily following every sonic event in detail, but rather accommodating to the rhythmic concept of the song. Music may be described at high level according to its form [32]. This is a term typically used for description of Montage for classical music, but it
can be generalized to other music genres in that it defines an introduction, development stages, and finalization, and how these are structured to create a whole. As an example, consider the first 35 measures of Felix Mendelssohn’s “Andante con Moto,” Symphony No.4 in A major [33]. Its form, which in turn depicts the montage of the piece, is shown in Figure 8. The reader is encouraged to listen to the song while looking at this diagram.

Figure 8: Mendelssohn’s Symphony No.4 – Italian; Second Movement: Andante con Moto.
Form for measures 1-35; “T” indicates tension, “R” indicates resolution.

If the sections in a piece can be identified, then each specific section can be analyzed for Metric Montage in more detail. Note that transition between sections in music is generally a tension/resolution relationship, as indicated in Figure 8. This way, a question is answered; a new topic is embarked upon… whatever the music poses may be addressed this way. In terms of purist music theory, the tension/release curve follows the binary rounded form ABA’ [32].

Rhythmic montage can be applied analogously to visual media. Relating to the proposed editing application, visual material consists of two sources: photographs and video. Revisiting temporal proportion and the observation of the techniques used in
Muvee [10], rapid transitions between images creates tension; video without cuts resolves this tension. Therefore, for each song section contributing to Rhythmic Montage, a section of video is included, followed by a succession of pictures synchronized to the Metric Montage of specific audio events in that section. The more pictures included in the section, the higher the density of cuts. Therefore, the proportion of video duration vs. photograph progression can be related directly to the level of tension and resolution in the music.

2.3.3 Tonal Montage

Up to this point, aspects of art which translate exclusively to temporal information have been discussed. Semantics, i.e., the product of attribute relationships which convey coherency and meaning to the audience need also be considered. Eisenstein provides a third definition of montage, which encompasses the visual elements that provide melody and emotion. Here is where features such as color, brightness and saturation play a fundamental role.

The primary emotional feel of the shots in a movie section is called the Dominant [19]. Higher-level semantics such as thematic content of the film and the meaning of the images it contains amount to this term. Since this is a very personal and application specific set of parameters, Eisenstein considers only those which are universal to all visual documents: graphic or plastic aspects of the moving image (e.g., colors and shapes). He denominates this set “graphic tonality”— the design elements of color, line, and shape, and what they can represent.
Low-level color attributes specified in the HSV space (Hue, Saturation and Brightness), are integrated into this concept of tonal montage by providing consistency among specific song sections. That is, histograms of these quantities for each specific image are kept as similar as possible along the same section to provide semantic consistency in the visual composition. As an example, consider half of a collection of images and video shot in the daytime in an afternoon reunion, and half shot at night. In a summary video, montage may require semantic consistency which benefits from grouping images according to these two events. Therefore, it is desirable that the summary in each section include either exclusively daytime or exclusively nighttime shots. This can be done easily by keeping the brightness (V) parameter consistent. More about the HSV color space will be said in Chapter 3.

Eisenstein continues on to describe two more abstract forms of montage, specifically overtonal and intellectual montage. These however, are rather philosophical considerations and cannot be approached by an algorithm in a straightforward manner. They pose an interesting alternative for future work.

2.3.4 Multimedia Montage

All the artistic concepts discussed up to this point can be translated into rules for an algorithm, through the use of what has been defined as “Multimedia Montage” [13]. Researcher Ryotaro Suzuki defines it as “…the structural synthesis between time and space of multimedia components.” This process of collecting, organizing and quantifying different forms of media into a coherent document provides the backbone to the implementation of the system; the different engineering designs considered in the
following chapters are all derived from this concept. The translation of these premises is as follows [13]:

1) There is a Montage; a collection of parts which are autonomous and independent which amount to the complete document. This corresponds to audio sections, visual segments and photographs.

2) There are temporal relationships among the different events. Audio has a defined Metric and Rhythmic Montage in its structure of sections, notes and beats. Visual material is synchronized to this structure. This is implemented algorithmically by Media Segmentation techniques, explained in Chapter 3.

3) There is a harmonic relationship among the different events, providing melody and emotion, or Tonal Montage. Audio, being the dominant media, contains a defined harmonic structure. Visual material is described as to provide semantic consistency, and assigned to the audio sections to produce a meaningful passage. This is performed via media descriptors, discussed in Chapter 4.

4) There is a question/relationship, or tension/release progression throughout the piece; this is a common parameter for both music and visuals, and provides the final cross-media link.

2.4 CONCEPTUAL PRESENTATION OF THE ALGORITHM

Finally, a set of rules has been provided to construct a language; a mapping structure that can be implemented through an algorithm, which can synchronize the audio
and visual media both temporally and semantically. These rules are now presented according to what was listed in the previous section, but specific to the proposed system:

1) A home user provides the program with three types of media: raw video, a collection of photographs, and a soundtrack he chooses from his personal music collection. This soundtrack is considered the dominant media for the application, meaning that the other two visual collections are segmented and synchronized according to audio parameters. This synchronism is performed according to the precepts of the counterpoint theory in music, which creates a multimedia summary with artistic montage validity.

2) From analysis of the audio, both rhythmic and semantic structures are extracted to reveal specific temporal events as well an information on the different sections in a song. This is mapped to provide synchronism of the visual material, matching cuts with the audio events.

3) Each section extracted in the audio carries information about its specific plastic features, contained in a feature vector that may include magnitudes of energy, pitch, timbre, etc. Analogously, the visual collections are described by an average feature vector containing color, saturation and brightness information, among any other characteristics. Even though the temporal synchronization has taken place in the previous step, consistency of these feature vectors along the specific elements in each visual collection guarantees semantic consistency throughout the set.

4) The ratio of video to images dictates the density of cuts performed in each section, which is directly proportional to the level of tension in the multimedia
document. Therefore, a tension/release progression can be constructed to match that of the counterpoint representation of the soundtrack.

In this last point, the specific video/image ratio is determined by two factors. The first is the specific position of the section within the overall soundtrack’s structure. Revisiting the concepts of form and for the scope of this research, the overall structure of tension/resolution of the multimedia is assumed to follow the approximate path of Figure 9, generally valid for contemporary and western music:

![Figure 9: Generic path of tension/resolution progression for a multimedia document](image)

The overall energy of a specific audio section may also be associated with the tension/resolution theory. Considering the application domain, and supported by the approaches that commercial ventures such as Muvee [10] have implemented successfully, in a song with considerable dynamic fluctuations it is desirable to link fast-paced edition
of visual media with the louder passages in the soundtrack, while adopting a more relaxed approach over melodic and soft sections. This adds an envelope to the previous pre-defined curve, to produce a tension/resolution curve for visual synchronization, characteristic to each particular song. It is depicted in Figure 10.

Energy per frame can be generalized towards a feature vector that may also include pitch and timbre descriptor components. This finally provides the specific information for the video/images ratio, defining media segmentation. Specific visual events only need to be indexed by keyframes and their temporal positions relative to the soundtrack. A truly synchronized, “visually musical” summary has been achieved.

Figure 10: Tension/Resolution progression including audio energy information
MEDIA SEGMENTATION

As was discussed in the previous chapters, the nature of the proposed system demands that media with a temporal dimension be segmented into coherent sections. Therefore, algorithms for segmentation of both audio and video need to be implemented. The theories behind the different chosen techniques for media segmentation are presented in this chapter.

3.1 AUDIO

Audio must be segmented in two ways. In the scope of the song as a whole, it is desirable to segment the song into its different sections that contribute to the montage of the piece; the introduction, verses, choruses, etc. A measure of audio novelty with peaks at the song’s transition points can be used efficiently. At a later stage in the system which is concerned with more specific audio events, it is desirable to synchronize the media temporally according to lower level attributes such as rhythm and beats. Therefore segmentation must be performed on a more detailed scale for each individual section. This requires some sort of rhythmic analysis. Self-Similarity analysis through use of the Similarity Matrix can serve both purposes with great accuracy [12].
3.1.1 Self-Similarity Analysis

The concept of self-similarity is closely related to that of autocorrelation. Typically, autocorrelation is found in sound analysis literature as a method for pitch detection; however, it is useful for performing a wide variety of music analyses wherever some sort of pattern or periodicity detection is desired. Self-similarity and autocorrelation both provide a measure of how similar a signal is to versions of itself, delayed at different time intervals, or lags [34]. Figure 11 shows the functional block diagram of a signal autocorrelation analysis system.

Figure 11: Autocorrelation Analysis System

If the correlation function (the diamond block in the figure) is simply the product between the two quantities, this diagram can be translated into the following equation [34]:

\[
\text{autocorrelation}[\text{lag}] = \sum_{n=0}^{N} \text{signal}[n] \times \text{signal}[n + \text{lag}]
\] (1)
where $n$ is the sample index and $0 < \text{lag} \leq n$. This measures, for different lags, the degree of similarity between the signal and the delayed versions of itself.

To understand the concept of autocorrelation, it is useful to start by considering the simplest case of a sinusoidal. This explanation can be found in [34], towards computing a method for pitch detection, using (1). Refer to Figure 12 and what follows.

![Autocorrelation of a simple sine wave](image)

**Figure 12: Autocorrelation of a simple sine wave [34]**

For case (a), $\text{lag} = 0$. This implies that the signal is correlated with an identical version of itself, which in a normalized computation yields a result of 1, or perfect similarity. In case (b), delaying by one-quarter period, the inner product of the samples of a wave for one period yields an output of 0, or no correlation. In the case of (c), the delay is half a period and therefore correlation is -1, or complete dissimilarity. Case (d) is analogous to that of (b), and finally for case (e), a delay of one complete period yields again perfect similarity, or a correlation of 1. If the obvious result of case (a) is ignored,
then the maximum of the autocorrelation function will be attained at precisely one period of lag, which corresponds to the fundamental frequency of the signal.

For more complex signals, this autocorrelation function produces a series of peaks that hint of tonal components existing at various fundamental frequency points in the spectrum. The important thing to keep in mind is that this analysis is based in finding the moments of repetition in the signal, via a similarity measure. This concept can be generalized under the term of self-similarity analysis.

3.1.2 Temporal Vs. Spectral Resolution

 Depending on the temporal resolution with which Self Similarity analysis is being conducted, in other words the time scale at which signal patterns are being observed, audio features that range from the frequency value of a fundamental to the rhythmic structures present in a song can be extracted. In the example of the previous section, analysis was being performed at sample level; each sample was delayed and compared with a previous sample, and at certain delay amounts, correlation was maximum. This way, pitch value was detected and possibly tracked.

 Generalizing this concept towards comparing collections of consecutive samples (i.e., frames), the specific magnitudes produced by the analysis at different points in time (corresponding not to a sample but to a frame) are compared one with the other according to a chosen measure. Consequently, the details of variations within each frame (amongst the samples in a collection), are averaged for the entirety of the frame. This renders the analysis useless for minute variations, but it creates new possibilities for observing other features in the musical progression. This reflects the compromise in the duality between
time and frequency: temporal resolution is inversely proportional to spectral resolution. This is a truism of signal processing that holds for any media form.

Revisiting the concept of autocorrelation, the results for comparison at frame intervals are maximized when there is high correlation between the two frames, and minimized when statistics are reciprocal for the two frames. If there is no correlation, then the output of the comparison is small in magnitude. This kind of self-similarity analysis can be performed successfully on music, based on the precept that music is generally self-similar [12]. That is, nearly all music in existence has some kind of repetitive structure; it is based on patterns.

At relatively fine and detailed time levels (short frames), repetition constitutes acoustical patterns that dictate specific audio events in the piece (e.g., individual percussive events). A broader scope makes extracting specific event information difficult, but reveals repetition and structure in melodic and rhythmic patterns. On an even larger scale, similar sections pertaining to the song’s form become evident, such as verses, choruses, codas, among others. This is explained in depth in the following sections. Up to this point, the following features are important to point out:

- Self Similarity analysis is not constrained by specific spectral assumptions such as spectral envelopes (e.g., tonal, noise-like, etc.). Neither does it rely on temporal features, such as silence, periodic energy peaks, or specific time signatures. Because it is based on self-similarity, the only required features are repetitive events (even silence) in the source audio [16].

- The duality between temporal and spectral resolution is of extreme importance in autocorrelation analysis. By controlling the size of the
analysis frame, autocorrelation can be computed over fine to coarse temporal representations of the audio. This provides essential functionality, because fine temporal resolution is adequate for extracting low-level audio features, while detailed descriptions over longer spans of time are useful for computing higher level semantics.

3.1.3 The Similarity Matrix

The specific method used in this dissertation for performing self-similarity analysis is presented. It analyzes the structural aspects of music in the temporal domain, with the intent of visualizing this structure according to acoustic (dis)similarities of the music in time [12]. An example of a Self-Similarity Matrix is shown in Figure 13.
This visualization is interpreted as follows: a pixel in the matrix represents the correlation between two specific frames. Pixel brightness represents the amount of correlation; a 100% bright pixel implies perfect correlation, while a 100% dark pixel implies perfect dissimilarity. Frames advance in time from left to right as well as from top to bottom. Having chosen a frame size of 0.01 seconds (because the Frames-Per-Second i.e., FPS is 100), the time in seconds is simply the index divided by 100.

The Similarity Matrix is symmetrical in nature. The top left pixel represents the correlation of the first frame with itself; it is consequently maximum (its color is white). Generalizing this feature, there is a bright white diagonal in the matrix where the frames...
are compared with themselves, because this implies perfect similarity. As another example, the coloring of a pixel halfway through the top (or symmetrically left) gives a measure of the similarity between the first frame and that of the midpoint of the analyzed piece.

The functional block diagram of this self-similarity analysis system is shown in Figure 14. Note the direct relation to that of Figure 12, which pertains to generic autocorrelation analysis.

![Figure 14: Block diagram for constructing a self-similarity matrix](image)

According to [12], after the audio is read into frames, it is parameterized using a spectral representation, resulting in a set of feature vectors per frame. Subsequently, a distance measure is used to calculate the similarity between all combinations of feature vectors. If each frame is considered an instant of the audio in time, this implies that every instant in the audio is analyzed against every other instant in the piece. The result is arranged into a two-dimensional representation which conforms the Self Similarity matrix.

More specifically, in the first block of Figure 14 audio is read at a frame rate dictated by the desired time scale, and each frame is tapered using a Hamming window.
In the next block, the audio is parameterized by computing the Mel-Frequency Cepstral Coefficients [36] for each frame, as shown:

![Diagram of audio parameterization](image)

Figure 15: Parameterization of Audio Frames into MFCC’s [12]

For each frame, the samples are transformed into a spectral representation by computing the Discrete Fourier Transform (DFT) of the signal. The log of the power of this spectrum is computed, and the resulting coefficients are perceptually weighted by a non-linear map of the frequency scale, in what is denominated as Mel-scaling. This is used to emphasize mid-frequency bands, proportionately to their psychoacoustic importance [12]. The DFT is applied again on these coefficients, producing the “cepstrum” of the signal [34].

Cepstral analysis is useful because it de-convolves two components found in audio spectra: excitation and resonance. Excitation is the pure sound generated by an instrument, produced by its original vibratory impulses where it includes its attack, decay and sustain components; it dictates the fundamental pitch and temporal information for
the sound. The resonance component can be considered the impulse response of the body
of the instrument, where the original sound is filtered and delayed, coloring the signal. It
pertains mostly to timbre and modal information in the audio. Analogous psychoacoustic
processes occur in human auditory perception; if the Mel-frequency cepstral coefficients
of two frames are similar, a person would probably judge them as similar. It is therefore
very adequate to use these feature vectors towards self-similarity analysis. Of this vector,
only the first 12 MFCC’s are considered, as it is standard in the literature to consider the
higher order coefficients as perceptually insignificant. Each frame is hence represented
by this 12 dimensional feature vector.

The next step is to compute the correlation between the feature vectors pertaining
to the different frames. This is done by initially performing zero-mean calculation on the
parameter vectors and then computing the cosine of the angle between them, according to
the geometric relation:

\[ D_C(i, j) = \frac{v_i \cdot v_j}{\|v_i\| \|v_j\|} \]  \hspace{1cm} (2)

The advantage of this analysis is that by computing angular distance, the level of
similarity is obtained disregarding the magnitude of the feature vectors. This is generally
desirable so that even if similar regions have low energies, they will be judged highly
similar. It is this value \( D_C(i, j) \) that is embedded in the two-dimensional representation
of the Similarity Matrix.
3.1.4 Beat and Rhythm Detection

Employing a relatively small time scale (around several frames every 0.1s), then the Similarity Matrix will exhibit patterns corresponding to specific audio events or rhythmic information [16]. Consider the similarity matrix of Figure 16:

![Similarity Matrix](image)

Figure 16: Similarity matrix computed for seconds 0-10 of the song “Galaxy Bounce” by The Chemical Brothers [35]

It is recommended that the audio be heard while the following explanation is considered (remember that for 100FPS, time index is obtained by simply dividing the frame number by 100). The song begins with an electronic scratch of two movements for the first quarter second. The pairs of vertical and horizontal lines close to the top left
corner represent this. It then follows with a solo sustained voice up to 2.3 seconds, where percussion begins and another note in the voice is also produced. At 3.65 seconds, a short snare progression consisting of three syncopated strikes is performed. This progression ends at 4.5 seconds, and the initial progression described up to now, with the addition of percussion under the first voice is repeated until almost the end of the analyzed audio.

It is also noted that the matrix shown in Figure 13 is contained in this last matrix. The following rhythmic events can be identified from this matrix [16]:

- “Regions of high audio similarity, such as silence or long sustained notes, appear as bright squares on the diagonal.”
  
  o The Bright squares on the main diagonal follow the silence and voice progressions.

- “Repeated figures will be visible as bright off-diagonal rectangles.”
  
  o The repeated figures can be clearly seen in the off-diagonal rectangles, for example the one which upper left corner corresponds to 6:30 sec. Vertical. This means that this specific section occurs at both 2:30 and 6:30 seconds.

- “If the music has a high degree of repetition, this will be visible as diagonal stripes or checkerboards, offset from the main diagonal by the repetition time”
  
  o Consider the bright diagonals that start at approximately 4 seconds on either of the axes. In reality, the loop of the song repeats itself every 4 seconds!!!
Two other geometric forms can be identified clearly in this matrix:

- Symmetric vertical and horizontal lines, which make a cross at their meeting point on the main diagonal. These represent meaningful individual audio events, as the crosses are of a dark color, which implies that at the special times corresponding to the lines, audio is uncorrelated from every other instant of the song.

- Larger scale checkerboard structures. These correspond to the fundamental feature of audio Novelty. However, analysis of these structures will be the focus of the next section, because it pertains to a different time scale.

Having identified these features in the audio, a mathematical analysis can be performed on the similarity matrix in order to have a quantitative measure of rhythmic musical components and specific audio events. The most basic analysis can produce accurate results of the latter. Considering the dark lines that conform crosses along the diagonal, to identify these it is a matter of simply computing the sum of each individual column or row. Choosing columns as the summing direction, this calculation is performed as follows:

$$B(i) = \sum_{j=1}^{L} S(i, j) \quad (3)$$

where $S(i,j)$ is the similarity matrix, $i$ is the column index, $j$ is the row index, and $L$ is a convenient distance along which to sum the columns. $L$ is necessary because some
similar exceptional regions (such as the main diagonal) tend to average the values that have produced this dark line, plus it provides computational efficiency as opposed to summing over the totality of the matrix. This produces data which is plotted in Figure 17.

![Summation Energy per Column](image)

Figure 17: Summation energy per column towards detecting meaningful audio events. Local minima “X” values are the time indexes that indicate these events (over 100). Some were omitted in the plot for visualization purposes.

The results of this analysis were better than expected. Not all the loud audio events in the progression appeared as strong minima in this plot. Only the “meaningful” audio events, i.e., the ones that contribute significantly to the rhythm, exhibit low values. As an example, consider a fast progression of consecutive snare hits in a percussive sequence. Only the first and last hits contribute to the overall rhythmic montage of the
song. This coincides with the approach taken by similarity analysis; the second snare hit is similar to the first one. Therefore, it does not create a strong line of low energy, since it is not providing any novelty to the progression. At the end, the silence that follows the snare hits becomes something novel, creating a meaningful similarity energy dip. This occurs at 4.86 seconds.

Another approach to identifying metric montage, applied to music that is based on a strong percussive presence, is the beat spectrum [16]. To derive it, the diagonals offset from the main are considered. Each diagonal in the matrix is summed towards computing an energy value; diagonals representing meaningful rhythmic components are bright, yielding high values. This way, peaks corresponding to the amount of autocorrelation for each specific beat are achieved. The computation is performed according to:

\[
B(i) = \sum_{i,j} S(i, j)S(j, i + j)
\]  \hspace{1cm} (4)

The beat spectrum is plotted for the continuing example of the matrix of Figure 16 and shown in Figure 18.
This plot is read in a different manner from the meaningful audio events plot. Since it is a “spectrum” type plot, it indicates at what frame lag values the music is most repetitive. Three peaks are clearly identified. As was discussed previously, the music repeats itself at every 400 frames (i.e., 4 seconds). It therefore repeats again every 800 frames. The peak at 81 frames corresponds to the actual BPM value of the song (i.e., the BPM value of the song is the inverse of the time it takes for 81 frames to progress). Minor peaks correspond to other rhythmic components. Two features can be extracted via the beat spectrum; temporal relations (beats) can be identified by the lag time of corresponding peaks, while the relative amplitudes of different peaks are related to the strength of each rhythmic component.
3.1.5 Audio Novelty Segmentation

Considering Figure 16 from the previous section, the matrix exhibits large-scale checkerboard patterns. This depicts a very high-level feature; audio Novelty. It is useful towards segregating different structural parts of a song’s form: the introduction, verses, choruses, codas, etc. To perform this analysis, the time scale is changed to a frame rate of 5 frames per second (20 times wider than the previous analysis).

Even though novelty analysis works with great success for the previously considered song, it is fairly complex in rhythmic transitions and complicates an introductory analysis such as this one. Therefore, a new song is considered from this point on. It is the song “Wish You Were Here” by Incubus. Its similarity matrix is shown in Figure 19. Readers curious to see the matrix for the entire “Galaxy Bounce” song can refer to Chapter 6 (the figures already considered will coincide with the top-left corner section of the entire matrix).
Figure 19: Similarity matrix computed for the song “Wish You Were Here” by Incubus [37].

Note that the matrix, at this scale level, does not exhibit diagonals (except the main one), nor specific beats. It mainly illustrates the checkerboard patterns. Revisiting what was said in the last section, sustained notes or patterns appear as bright squares on the diagonal, while regions of low cross-similarity constitute dark regions whose vertices touch the main diagonal [12].

Steering for the moment to the tangent topic of psychoacoustics, consider a perceptual definition of novelty. A transition point between two distinct regions occurs when the two following conditions are met:

- Each individual region is highly self-similar.
The cross correlation between the two regions is considerably low. These two conditions translate into the Similarity Matrix as follows:

- Each Individual Region is a bright rectangle on the diagonal.
- The cross computation region in between the two sections is a dark rectangle off the main diagonal. Since the matrix is symmetric, the rectangle is reflected by the diagonal.

Therefore, a Novelty point constitutes a checkerboard, with the crux as the point of transition. To detect these points, $S(i,j)$ (the similarity matrix) can be correlated with a kernel which itself looks like a checkerboard [12]. The simplest “checkerboard” kernel that can be implemented is:

$$
C = \begin{bmatrix}
1 & -1 \\
-1 & 1
\end{bmatrix} = \begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix} + \begin{bmatrix}
0 & -1 \\
-1 & 0
\end{bmatrix}
$$

(5)

where the checkerboard kernel is constructed by the addition of a “coherence” and an “anti-coherence” kernel, depicting the aforementioned psychoacoustic premises. The next step is to accommodate to the large time scale of the analysis. This is done by making the kernel large in size, so it spans several frames, therefore averaging out any “noise” present in the matrix. This is done by computing the kronecker product of the basis kernel (5) with a kernel of ones as shown:

$$
Ck = \begin{bmatrix}
1 & -1 \\
-1 & 1
\end{bmatrix} \otimes \begin{bmatrix}
1 & 1 \\
1 & 1
\end{bmatrix} = \begin{bmatrix}
1 & 1 & -1 & -1 \\
1 & 1 & -1 & -1 \\
-1 & -1 & 1 & 1 \\
-1 & -1 & 1 & 1
\end{bmatrix}
$$

(6)
Finally, to prevent edge effects, the kernel is tapered using a Gaussian Radial Basis Function, resulting in the structure shown in Figure 20:

![Checkerboard Kernel](image)

**Figure 20:** Checkerboard Kernel, which represents a situation of perfect Novelty.

Correlating this kernel along the main diagonal of $S(i,j)$ produces a measure of audio novelty as a function of lag. This way, the specific transition points in time can be extracted from the audio. This novelty function is computed by multiplying the kernel $C(m,n)$ with the matrix $S(i,j)$ as follows:

$$N(i) = \sum_{m=-L/2}^{L/2} \sum_{n=-L/2}^{L/2} C(m,n)S(i + m, i + n)$$  \(7\)

Where $i$ is the index for the audio frames. In this equation, $i$ is centered in the kernel, but the computation can easily be done taking the reference at the start or endpoints of the kernel, corresponding to the top-left or bottom-right corners (this will
permit analysis of the song from absolute start to absolute end, but care must be taken to accommodate for the frame lag that results. This analysis yields for the considered song a plot of novelty vs. time, as shown in figure 21.

![Novelty Measure (FPS = 5)](image)

**Figure 21:** Novelty Plot.

The peaks which are clearly observed in this plot correspond to frame numbers which, if divided by 5 (the FPS value), coincide to transition time instants in the song’s form. Hence the introduction is separated from the verse, verses from chorus, and so on. Audio can be segmented at these peaks, and metric montage is completely defined.
3.2 Video

With the analysis of the Similarity matrix discussed in the last sections, the temporal aspects of montage have been instantiated. Video is another type of media that depends on a temporal dimension; nonetheless, consideration of the video parameters is not essential to the temporal montage of the final piece, because audio is the dominant media.

Considering tonal montage of the piece however (i.e., that which contributes to semantic consistency of the final creation), video segmentation does play an important role. Video input presents one fundamental difference from the audio that needs to be addressed: the audio soundtrack is a professional piece that has undergone prior mastering and production processes. On the other hand, video input consists of unedited media shot by the typical user, inexperienced in film techniques. If it is assumed the complete unedited video plus the picture slide show amount to be longer than the audio soundtrack duration, then video must be segmented so that sections of it are discarded. The goal is then to segment video towards retaining only its highest quality portions.

There are certain common premises followed by professional video editors which can be translated into a basis for automatic video edition according to semantic quality [2]:

- A clip should not be too long or too short.
- Fast and erratic camera movement is generally bad quality, perceptually meaningless video.
- Excessive brightness/darkness levels also typically hint of low quality video clips.
Another characteristic observed in home video is that material close to scene cut points is not of good quality and is also not focused on a meaningful subject or event. It is common to find clips which focus on a scene for a period of time, present some erratic movement, and then focus on another scene (e.g., when filming one subject and turning to film another). It is convenient to detect these kinds of transitions and segment video by removing corresponding parts.

3.2.1 HSV Color Space

The color space that addresses human psychovisual perception most directly is the HSV color space. It is represented by Figure 22.

![HSV Color Space](image)

Figure 22: HSV Color Space [38]
The totality of colors can be theoretically indexed in this color space, according to cylindrical coordinates of three parameters. Even though the classical definition of the space calls for it to be depicted by a cylinder, dark values that tend towards black constrain the parameters, given the color gamut of current video technologies as well as human vision. Its reality is better depicted by a conical form such as Figure 22. As its name implies, The HSV color space is composed of the following three parameters:

- **Hue (H):** It is the angular coordinate of the system; its function is to specify color family. As angle progresses, the entire visual spectrum is traversed from red to yellow, green, blue and finally purple.

- **Saturation (S):** Specifies color purity. Its minimum value (0) corresponds to gray-scale shades, while its maximum (1) depicts pure colors (no white component).

- **Value (V):** Depicts the brightness of a color. Its minimum (0) is completely dark, perceptually equivalent to “black.” The color becomes increasingly brighter until it reaches the maximum value, 1.

The first transformation of the video, which is originally composed of true color RGB representations, is to convert its frames to the HSV color space. This is done according to a non-linear psychovisually modeled mapping procedure [30]. Once an image or frame is converted, a feature vector is constructed from the averages of these values for all the pixels in the frame:

\[
HSV_{vect}[i] = aH[i] + bS[i] + cV[i] \quad (8)
\]
The weights $a, b, c$ are determined by user intervention, where specific video styles can be adopted, giving more or less importance to any of the attributes, to make the video more dependent on either color, saturation or brightness.

### 3.2.2 Video Unsuitability Segmentation

Revisiting prior considerations on tonal montage and the HSV color space, together with the visual concepts for semantic consistency discussed in Chapter 2, a set of rules can be applied algorithmically to the progression of the defined feature vector. Raw video can then be segmented according to these rules:

- Color progressions within a section must be kept as consistent as possible.
- Textural attributes contained in the saturation value should maintain homogeneity; if there is a rapid transition, it hints of undesired movement or a scene cut.
- Brightness should also exhibit stability; additionally it should be constrained to level thresholds that are neither too high nor too low.

Each of these degrees of freedom is addressed in turn by its respective parameter in the feature vector, if changes in their magnitudes from frame to frame are considered. Consequently, the magnitude for a specific HSV feature vector is not as important as the difference, or distance, between two subsequent vectors. A differential HSV feature vector is therefore defined, consisting of three differential functions for each of the three parameters. A first computational approach is to simply subtract the magnitudes of each pair of consequent feature vectors; it was found to suffice for unsuitability segmentation.
purposes. However, other distances such as the cosine angle used to construct the similarity matrix in the audio analysis section can be accommodated easily. More complex video analysis techniques can also be applied successfully [2] [5].

Since the three features require stability, they can be linearly combined according to magnitude averages between them to form a unified representation of the change of the three semantic parameters. This produces a data progression such as the one plotted in Figure 23:

![Figure 23: Plot of HSV differential progression](image)

This plot is the result of analysis from a raw home video sample. Note the time resolution at which the analysis is being conducted. Frames are decimated to only include
one every second; this is sufficient for the application, and it also permits efficient video manageability using MATLAB [38]. Time in seconds is equivalent in this case to frame number. Peaks in this figure coincide with significant changes in the feature vector. According to the hypothesis that random and rapid change of the features implies unsuitable video, then this constitutes an unsuitability function. It does hold true; the example video that was analyzed to produce this curve has random movements for approximately the first third of video. It then stays relatively still filming a scene, or doing very gradual transitions between views. A scene cut is produced at 30 seconds, hence the corresponding spike. Very random movement also occurs at 40 seconds.

To segment the video according to this curve, two thresholds are defined: a suitability threshold and a duration threshold. The suitability threshold measures the maximum amount of unsuitability that is admissible in the clip. If the curve surpasses this threshold, it is removed from the pool of eligible video because it is considered low quality. It is exemplified in the plot to be a y axis value of 0.12. Consequently, for Figure 23, the spikes above this threshold are detected and removed.

At this point, the pool of available video includes clips which start and end frames are indicated by the boundaries of the video just removed. The duration threshold specifies minimum and maximum duration times for the clips. Revisiting the video editing rules explained before, a clip should not be too long or too short. For time differences computed between frame numbers, if this difference is too small, it lies below the minimum duration, and the corresponding regions are removed. In the figure, this minimum threshold was set to be approximately 8 seconds. This way, several regions occurring mainly in the first half of the original video are discarded. In reality, these
regions would coincide with random good quality instants immersed in generally bad quality video, which is generally of no semantic importance (although as future work, making this threshold dynamic could provide a way to save the exceptional instants which are truly meaningful).

For the maximum threshold, if the difference of frames is longer than this value, then the region is split in half until all resulting regions lie below the duration threshold. The final video clips are cut to match the temporal span dictated by the audio analysis, respecting the metric and rhythmic montage premises. The actual video that is retained is indexed from the halfway point of the useful clip, with equal duration before and after this midpoint. This implies that the used video is kept as far away from unsuitable regions as possible.
4

FEATURE EXTRACTION AND SEMANTIC PROFILING

Semantic description systems rely on extraction of features that can be psychologically mapped to meaningful perceptual parameters in the media document. Towards addressing Tonal Montage, semantic description of the media via carefully chosen attributes is essential. This complements the Media Segmentation stages of the proposed system, which have up to this point completely specified the temporal structure of the final audiovisual document.

However, before analyzing these concepts in more depth, it is convenient to present the chosen standard for feature extraction and media description: MPEG-7. The design of this system follows this standard closely in its normative and recommended practices, as it provides the perfect framework for multimedia information systems such as the one proposed [1].

4.1 THE MPEG-7 MEDIA DESCRIPTION FRAMEWORK

MPEG-7, formally called “Multimedia Content Description Interface,” is an initiative that started officially in 1997, conducted by the ISO/MPEG [39] consortium as an addition to its MPEG-1, MPEG-2 and MPEG-4 portfolio of multimedia standards (MPEG 21 is a standard which appeared later). It is however, completely different from these technologies both in its conception and functionality. MPEG-1 and 2 are concerned
with compression of media documents for efficient storage and transmission purposes; as an example, an Mp3 (MPEG-1 layer 3) encoded audio file is only a fraction of the size of its original PCM digital version. MPEG-4 deals with the modeling of a multimedia scene as a composition of natural and synthetic objects, with which a user may interact. This enables the expansion of the multimedia capabilities of emerging infrastructures such as mobile and web-based content. These standards have been highly successful and are recognized worldwide by the Multimedia industry.

Motivated by the evolution of multimedia information systems (much of which was a product of the aforementioned technologies), MPEG-7 moves on from the data aspect of a document to the area of metadata for multimedia descriptions [1]. This addresses the need to manage large collections of multimedia content, providing filtering, searching and summarization capabilities, among countless others. Although nearly a decade ago its scope was more oriented to multimedia entertainment providers, currently it is gaining an unprecedented level of importance with the need for organization of large multimedia databases required by even the average home computer user, and the quantity of documents growing at an ever-faster rate.

According to [40], MPEG-7 provides a portfolio of tools to describe multimedia content completely and efficiently. Although content management is an essential application scenario for the technology, the standard was designed to encompass a broad range of other multimedia applications. It can provide a detailed description of the media’s content attributes, which can be extracted via analysis and processing, in other words, feature extraction algorithms. MPEG-7 is intended to be generic, not targeted to a specific application or application domain.
One of the main characteristics of the MPEG-7 standard is that it supports several abstraction levels, which range from low-level signal data characteristics to high-level semantic information. As an example, consider low-level descriptors for audio material: power, pitch, timbre, etc. For visual material, low-level attributes such as color, brightness and texture, among others, can also be extracted. On the other end of the spectrum, high-level descriptions can also be constructed, such as the following scene description: “This is a scene with a barking brown dog on the left and a blue ball that falls down on the right, with the sound of passing cars in the background [40]”. At an intermediate abstraction level, music can be described according to genre, rhythm, or segmentation information, such as the product of the novelty method described in the previous chapter. Visual material can be described by shape, motion and position attributes, together with any other descriptions relevant to the application.

To encompass all the possible scenarios on which MPEG-7 can be applied, a large number of descriptive features of multimedia content must be considered. Therefore, MPEG-7 encourages the use of only a subset of its application-specific features in any given implementation. Producing such a description starts with feature extraction performed by analysis of the media. Low-level features are extracted automatically from the data stream, whereas semantic content generally needs some sort of human interaction. Whatever the case, the specific feature extraction algorithms used are not dictated by the MPEG-7 standard; only their description format. This leaves room open for selection of the most convenient algorithms as pertaining to the specific application.

MPEG-7 architecture is based on the following basic structures, as specified in the Requirements Document [40] [41]:


- **Data:**

Multimedia information that will be described using MPEG-7, regardless of storage, coding, display, transmission, medium, or technology. Furthermore, a Feature is a distinctive characteristic of the data [that] signifies something to somebody.

- **Descriptor:**

A representation of a Feature. A Descriptor defines the syntax and the semantics of the Feature representation.

- **Description Scheme:**

The structure and semantics of the relationships between its components, which may be both Descriptors and Description Schemes.

- **Description Definition Language (DDL):**

A language that allows the creation of new Description Schemes and, possibly, Descriptors. It also allows the extension and modification of existing Description Schemes.

- **Systems Tools:**

Tools to support multiplexing of descriptions, synchronization of descriptions with content, delivery mechanisms, and coded representations (both textual and binary formats) for efficient storage and transmission and the management and protection of intellectual property in MPEG-7 Descriptions.

- **Description:**

A Description consists of a Description Scheme (structure) and the set of Descriptor Values (instantiations) that describe the Data.
Since a description is conformed by a Description Scheme, which in turn is conformed by other Description Schemes and Descriptors, these are the key description tools that include the actual media semantics. The other blocks (DDL and Systems Tools), conform the interface of the MPEG-7 document and, although they are essential for the output of a compliant version of the system, they are not the focus of this chapter. For further information on the Standard, please see [40] through [42]. Concentrating on Descriptors and Description Schemes, these are defined in three main sections of the MPEG-7 standard [42]:

- Part 3 (Visual): “Standardizes the descriptors related to visual features that apply to images and/or videos”
- Part 4 (Audio): “Standardizes the description tools related to audio features, covering areas from speech to music”
- Part 5 (Multimedia Description Schemes [MDS]): “Standardizes the description tools related to features applying to audio, visual, and audio–visual content”

In these three parts, the set of MPEG-7 description tools is defined. It is then instantiated according to Figure 24.
This figure reveals the categorization of the various MPEG-7 description tools; each of the basic rectangles depicted in the figure consists of a Description Scheme. Although all are relevant to the application in a commercial scenario, there are two categories which from a researcher’s point of view represent the essence of this project and directly relate to its implementation: The Content Description and User Interaction Toolsets. The User Interaction toolset will be discussed at the end of this chapter. The Content Description Toolset is divided into three classes:

- Spatio-Temporal Structure: Describes the content in terms of its temporal and spatial structure, permitting the indexing and localization of its elements. This is crucial for this application, as it provides the means for synchronism in between the various media objects that conform the music video. Revisiting Chapter 2, it is the algorithmic structure for providing the temporal (i.e., metric and rhythmic) montage of the piece.
- Audio and Visual Features: Conforms the attributes description framework for the multimedia document; it is here that low-level descriptors such as color, brightness, audio energy, pitch, etc. as well as higher-level features such as rhythm and novelty are defined. It provides the tonal montage structure of the final document.

- Semantic Structure: Represents the conceptual part of the descriptors, as it relates to the perceptual meaning that a multimedia document conveys; it describes objects, events, abstract concepts and relationships. It is also useful for cross-linking the structures and features; it provides a direct relationship with the audio-visual mappings discussed in Chapter 2.

4.2 Feature Extraction Algorithms

There are a considerable number of descriptors defined in the MPEG-7 media description toolset. The techniques for extracting their respective features are typically a recommended part of the standard. Hence, only a subset of these descriptors, accommodated to certain feature extraction algorithms, is implemented in this project. Extraction of mid and high level audio and visual descriptors (e.g. rhythm, novelty, color consistency) has been discussed in the previous chapter. Concerning Low-Level signal descriptors, they are proposed in what follows.

4.2.1 Audio

Audio Power (AP): A Low Level descriptor that describes the temporally smoothed instantaneous power of an audio signal [43]. Instantaneous power refers to the
power of each audio frame; if the power on a sample basis is to be calculated, the frame size may be equal to one sample. It is calculated according to the recommended MPEG-7 standard method of the following equation:

$$AP(l) = \frac{1}{N_{\text{hop}}} \sum_{n=0}^{N_{\text{hop}}-1} |s(n + lN_{\text{hop}})|^2 \quad (0 \leq l \leq L - 1)$$  \hspace{1cm} (9)$$

where $L$ is the total number of time frames, $l$ is the specific frame for which AP is calculated, and $N_{\text{hop}}$ is the number of time samples per frame. This provides a moving-average progression of the overall audio power level for different moments in a song.

Audio Fundamental Frequency (AFF): Provides estimations of the fundamental frequency in a frame, assuming that the contained audio in the frame is periodic. For these tonal regions in the audio, the raw stream of samples is input into a frame, which uses the right half of a Hamming window of 20ms. An initial estimation of the frequency is performed via cepstrum analysis [36] for the window, which yields a preliminary frequency value. This is then used to construct a comb filter of order 2, with a notch at precisely that frequency value.

The next step in the algorithm is to take this filter as the initial condition for an adaptive comb filter. The purpose of this adaptive filter is to minimize the energy output as much as possible; this implies that it tries to track the fundamental value of frequency to block it; this is the value where energy is most concentrated for a tonal frame of audio. Therefore, the adaptive filter is configured as a system identification mechanism with
desired response zero and an output prediction error that is minimized in the least squares sense. The concept is portrayed in Fig. 26.

For speed and accuracy purposes, an LMS filter with a sign difference (SDLMS) filter was chosen [45]. Results were as expected; this system works well for ambient music or speech where tonal frames are the common, but it is not suitable for percussive music. Results are shown in Figure 26, where a Choir is signing at a dramatic scene and the tension slowly builds up, resulting in a meaningful scene where two people meet. If the audio portion is played, for the graph, the variation of the first singular value (norm), in other words general envelope, of the graph does actually follow the pitch variations from the choir’s melody.
Audio Spectral Spread (ASS): Refers to how the spectral representation for a frame is distributed around its centroid. It is useful for differentiating tonal from noise-like signals; it presents a strong correlation with timbre information. For a frame, this feature is computed by calculating the second moment of the RMS value of the deviation magnitude that a spectrum has from its centroid. This is done by first calculating the Audio Spectral Centroid (ASC) descriptor and then computing according to the following equations:

$$ASC = \sum_{k=0}^{(N_f/2) - K_{low}} \log_2 \left( \frac{f'(k')} {1000} \right) P'(k')$$

(10)
\[
ASS = \sqrt{\sum_{k=0}^{(N_T/2)-K_{low}} \left[ \log_2 \left( \frac{f'(k')}{1000} \right) - ASC \right]^2 P'(k')}
\]

With these three low-level features for audio, a feature vector can be constructed that will add/subtract tension from the tension/resolution curve described in Chapter 2 (refer to Figure 10). The generic form of this feature vector is:

\[
A_i = w_{0i} AP + w_{1i} AFF + w_{2i} ASS
\]

where \(i\) denotes the specific frame being analyzed. The specific values of the weights assigned to the feature vector are the result of user profiling, discussed in the next section. This provides a coherent way of mapping the audio to the visual parameters throughout the document.

4.2.2 Video

Visual Counterparts consist of the Dominant color Descriptor (DCD), which provides a description of the representative colors in an image or image region. It is useful for similarity retrieval in image databases; therefore, it can provide a measure of similarity between the elements in the pool of images as well as the specific video sections for clustering according to color. Although MPEG-7 recommends extraction of this descriptor using the Generalized Lloyd Clustering Algorithm [1], for the desired
evolutional stage in this application it is enough to compute the color, saturation and brightness values through transformation to the HSV color space as discussed in Chapter 3, in order to cluster the visual material into semantically meaningful representations. Therefore, the corresponding feature vector for video is:

\[ V_i = w_{0i}H + w_{1i}S + w_{2i}V \]  

(13)

This is also supported by the fact that this system deals with audio visualization. This means that the audio track is the dominant media that will dictate how the visual material is segmented and clustered. For sonification applications, where audio is manipulated according to visual cues, the emphasis would probably be on visual descriptors.

4.3 MOODLOGIC PERCEPTUAL ATTRIBUTES

As was previously stated, another relevant semantic structure described in Figure 24 is the User Interaction Toolset. It serves as a bridge between user profiling and the semantics that control the feature extraction algorithms. Without user intervention, the music video exhibits a temporal synchronism between the audio and the visual material as well as a semantic link between the basis feature vector for the audio and the basis feature vector for the video. However, up to this point, the semantics are static without regard to soundtrack genre or video editing style. An immediate enhancement to the system is to profile the video editing style according to the soundtrack’s musical style. Both of these attributes are of an extremely high level of abstraction, and automatic
extraction of these parameters is a complex task. A possible solution to the problem is to provide user interaction through a database of user-rated musical style attributes, created from listening tests for the different soundtracks that could be used in such a system.

MoodLogic is an application implemented over the MPEG-7 standard [46] that proposes a set of high level semantic attributes as an extension to the already defined descriptors. These attributes not only describe the content of a song, but are targeted towards indicating the most likely reactions or feelings of the audience while listening to the song. Therefore, these receive the name of “Perceptual Attributes” [47].

The process for creation of MoodLogic’s perceptual attributes has been to gather a large number of listener subjects (thousands of customers) and let them describe their perceptions while listening to a particular song. This is done using a listening test type interface deployed through the internet, working over each listener’s personal music library. The songs are consequently profiled according to these perceptual attributes. It is therefore a convenient approach towards defining highly abstract description levels of musical genres and styles.

In conformance to the MPEG-7 Standard, these perceptual attributes are structured in an Attributes Description Scheme. It defines a complete portfolio of attributes that can be used to profile a song with MoodLogic. A subset of these has been chosen to profile the music video style that the proposed system creates. They are specified in Table 1 and discussed in what follows.
Table 1: MoodLogic Perceptual Attributes [47]

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Perceptual Attribute Type</th>
<th>Distribution Type</th>
<th>Min</th>
<th>Max</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Genre Distribution</td>
<td>Distribution</td>
<td>Closed</td>
<td>1</td>
<td>3</td>
<td>Alternative, rock, electronica, Classical, Pop</td>
</tr>
<tr>
<td>SubGenre Distribution</td>
<td>Distribution</td>
<td>Closed</td>
<td>0</td>
<td>5</td>
<td>Grunge, Funk, House, Orchestral, Acid Jazz</td>
</tr>
<tr>
<td>Mood Group</td>
<td>Distribution</td>
<td>Closed</td>
<td>1</td>
<td>2</td>
<td>Fun/Cheerful, Loving/Sensitive, Calm/Chill</td>
</tr>
<tr>
<td>Mood</td>
<td>Distribution</td>
<td>Closed</td>
<td>1</td>
<td>3</td>
<td>Happy, Yearning, Hypnotic</td>
</tr>
<tr>
<td>Perceptual Energy</td>
<td>Linear</td>
<td>Open</td>
<td>1</td>
<td>1</td>
<td>Values ranging from 0 to 255</td>
</tr>
<tr>
<td>Perceptual Valence</td>
<td>Linear</td>
<td>Open</td>
<td>1</td>
<td>1</td>
<td>Values ranging from 0 to 255</td>
</tr>
<tr>
<td>Perceptual Beat</td>
<td>Linear</td>
<td>Open</td>
<td>1</td>
<td>1</td>
<td>Values ranging from 0 to 255</td>
</tr>
</tbody>
</table>

4.3.1 Genre Distribution

Genre information is an extremely useful parameter towards profiling musical style. However, describing a song with one fixed genre type only goes so far. For example, rock music may be upbeat or downbeat, romantic or aggressive, may be “bluesy” or may be more towards the side of classical music. To address this issue, MoodLogic defines the concept of Genre Distribution, where a song can exist in several genres at various degrees. Within this genre distribution, sub-genre distributions can also be defined. This approach describes a song with much more accuracy than the standard
genre classification. The Distribution types are closed, meaning that the sum of all the specified genre/subgenre information must be exactly 255. As a result, the song’s “membership” in a particular genre/subgenre is its measure over 255.

### 4.3.2 Mood Distribution

It consists of a Mood Group distribution and specific child Moods, in a structure analogous to that of genre/subgenre distributions. It tries to address directly the feelings that a song provokes in an audience. It may be linked to the system by defining any video editing rule that may affect the overall mood of the piece. For example, a mellow song may be deemed to consist of slow scene transitions, while a happy, upbeat song may require more rhythmic and aggressive cuts.

### 4.3.3 Perceptual Energy

Defines the song’s energy as perceived by an audience. It does not represent a linear relation to the specific magnitude of audio power, because it depends on how a listener perceives this energy. A high energy value describes a song as dynamic, while a low energy value would describe it as lethargic.

### 4.3.4 Perceptual Valence

This attribute tries to describe a song as to how negative (angry, dark, sad, etc.) or positive (cheerful, happy, peaceful, etc.) it is perceived. It is a parameter that is of a very high level of abstraction, but certain rules pertaining to color spaces in video can be created, therefore it is mentioned as relevant to the system.
4.3.5 Perceptual Beat

This information depicts how much an audience perceives the music to rely on an underlying beat pattern. It ranges from a “light” and “faint” qualification to a “heavy” or “potent” beat presence. It is directly related to the judgment of a song as being of a rhythmic or a melodic nature. Therefore it controls the rhythmic feature extraction algorithms described in Chapter 2 and modeled in the last stage of the system’s flow diagram (refer to Chapter 5) as to synchronize the video with the beat spectrum information if it is considerably rhythmic, or to specific events encountered randomly in a musical piece.

The collection of all of these perceptual attributes constitutes the User Interaction category of Description Schemes, as defined in Figure 24. It provides the highest-level semantics needed by the system and their mapping to the low-level descriptors of the media streams by the Semantic Structure Description Scheme.

In an effort to consolidate what has been discussed in this dissertation up to this point, Table 2 provides a view of the proposed system as pertaining to the MPEG-7 framework.
Table 2: Media Features and their Instantiation in the MPEG-7 Framework

<table>
<thead>
<tr>
<th>Media Feature</th>
<th>MPEG-7 Descriptor</th>
<th>Semantic Attribute</th>
<th>Description Scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Audio Power</strong></td>
<td>Audio Power (AP)</td>
<td>Loudness</td>
<td>Content Description - Audio Low Level Descriptors (LLD)</td>
</tr>
<tr>
<td><strong>Audio Pitch</strong></td>
<td>Audio Fundamental frequency (AFF)</td>
<td>Tone</td>
<td>Content Description – Spectral Descriptors</td>
</tr>
<tr>
<td><strong>Audio Tonality</strong></td>
<td>Audio Spectral Spread (ASS)</td>
<td>Tonal Vs. Noise-Like</td>
<td>Content Description – Spectral Descriptors</td>
</tr>
<tr>
<td><strong>Visual Parameters</strong></td>
<td>Dominant Color Descriptor (DCD)</td>
<td>Visual Counterparts for Audio</td>
<td>Content Description – Color Spaces and Dominant Colors</td>
</tr>
<tr>
<td><strong>Audio Rhythm and Novelty</strong></td>
<td>Tempo Music Description Tools</td>
<td>Temporal indexing for segmentation</td>
<td>Content Description – Spatio-Temporal Structure</td>
</tr>
<tr>
<td><strong>User Interaction</strong></td>
<td>MoodLogic Perceptual Attributes</td>
<td>User Profiling of style</td>
<td>User Interaction Description Scheme</td>
</tr>
</tbody>
</table>
5

IMPLEMENTATION

Automatic video digest creation using an arbitrary audio soundtrack is a relatively novel approach to video summarization; the design space for this kind of applications is yet unexplored [5]. However, as was observed in previous chapters, there is a rich set of technological tools available today which can be used, although spread out amongst several fields of Multimedia technology. If integrated creatively, these can be applied effectively to the proposed concept.

5.1 PREPROCESSING STAGE

A basis for devising such a system is presented in Figure 3 of Chapter 1. As an input, a user of this system provides any musical piece of preference as a source soundtrack for the video summary. Both a collection of still pictures and a raw video recording may be provided for the visual component. The latter does not need to be related to the chosen audio in any way. If no pictures are provided, it is necessary that the source video be of equal or greater duration than the chosen soundtrack. Two separate analyses follow: one for the soundtrack and another for the video.

Audio is segmented according to a time-based degree of Novelty, computed from the previously mentioned self-similarity analysis. This approach has the advantage that depending on the chosen duration of the analysis time window, correlation may be found anywhere from specific audio events to structural relationships among entire form
components of a musical piece. Peaks in the novelty score appear at whatever interval scale is desired, depending on the size of the kernel for correlation. The actual data resulting from this analysis is a time-based signal which magnitude quantifies audio novelty at any given moment. Segment boundaries for the audio stream correspond to peaks along this curve; the higher the peaks, the more dramatic the change.

Concerning video, if the raw video track is longer in duration than the chosen audio, then it must be truncated in order for both to coincide in the temporal dimension. Initial segmentation is performed by detecting scene cuts and low quality video material. Since video is computationally intensive, then the video stream is decimated to only analyze frames at 1 second intervals. The result of this video analysis is a time-based signal which magnitude quantifies video unsuitability at any given moment. Suitable video regions are indexed by the time intervals in between unsuitability regions.

This way, the chosen audio soundtrack is segmented, and its boundary points serve as indexes for constructing a video with only the highest quality portions. A basic video clip could now be created at this point, according to Figure 27.

![Block Diagram for Basic Automatic Video Summary Creator](image)

Figure 27: Block Diagram for Basic Automatic Video Summary Creator
The shaded portion of this diagram is for showing that this approach is only implemented up to the novelty and unsuitability score blocks as part of a preprocessing stage; the shaded portions of the diagram are not implemented (their usage would imply the process depicted in Figure 3). Rather, the latter stages are replaced by algorithms which instantiate the concepts mentioned in the previous chapters.

5.2 FUNCTIONAL BLOCK DIAGRAM

Data is fed into the system which flow diagram is portrayed in Figure 28 (on the following page). The critical path of this diagram is indicated by the thicker arrows. At the initial stage of this path, the following inputs are received from the preprocessing stage, also shown in the diagram:

- A curve for video unsuitability. Boundaries for suitable regions are indexed using keyframe information.
- A structured index of the candidate photographs to be included in the final video.
- A curve for audio novelty of the chosen soundtrack audio. It corresponds to the segmented audio as a result of similarity analysis. Time information both for the total audio progression and the segment boundaries is specified in samples. Together with Video decimation (FPS) information and \( Nh_\) (Frame Time span) for audio, time is organized and indexed so that it can be converted easily from the audio to the video domain. This follows the MPEG-7 Standard’s \( Time DS \) (Description Scheme) reference [1].
Figure 28: Functional System Block Diagram
Note that at this point (after the preprocessing stage), for video and audio content, only time indexes are being considered; the actual media content may be left aside at this stage. In case of the photograph information, this also holds true; indexes can be considered a measure analogous to time in the sense that it denotes a sequence of pictures, therefore giving the collection a defined organization for sequential I/O in a system. This is also in accordance with MPEG-7 guidelines.

5.3 **SEGMENT CREATION BLOCK**

Time index data for indicating segment boundaries in sequential media, such as the three media streams that are being dealt with, is enough to create a set of temporal descriptors which can be used as “containers” for the media segments they represent. The collection of these descriptors amounts to the complete temporal montage of the piece. This approach is perfectly compatible with the MPEG-7 abstract Multimedia Content Description Schema, specifically addressing descriptors that deal with structural information. This way, a set of the *Segment DS* family is created. This set comprises two types of relevant descriptors:

- **Video Segment DS**: Suitable for describing a collection of frames, which can be assigned semantic attributes using the MPEG-7 Visual and Temporal Description Tools discussed in Chapter 4. This descriptor addresses the collection of video segments, both towards tonal and temporal types of montage.

- **Audio Segment DS**: Suitable for describing a collection of audio samples, which can be assigned attributes using the MPEG-7 Low Level and
Temporal Description Tools. It therefore describes the different audio segments.

The critical path of the system inputs data into the “Clustering of Media Segments” Block. It is here that indexing information from the prior analysis is first related to the media content itself; this resulting from inclusion of semantic information into what up to this point has been exclusively structural content description. Before analyzing this block, it is necessary to describe the parallel analysis chains that operate towards extracting the semantic information needed for profiling this stage.

5.4 MOOD/GENRE RECOGNITION BLOCK

Soundtrack audio is also analyzed for semantic content hinting of mood or artistic intent, this time using the MPEG-7 User Interaction DS. The specific toolset chosen for this project is the subset of descriptors used by the MoodLogic [46] approach, mentioned in Chapter 4.

MoodLogic descriptors are created by considering the soundtrack as a whole, without segmentation. MoodLogic features are important in the sense that they can be conceptually related to visual media by linking them to specific video editing styles. The processing for these descriptors is performed in parallel to all the other processes which have been mentioned up to this point. Per document, they constitute a query to a MoodLogic MPEG-7 profiles database, which was emulated with the collaboration of the Company. Towards testing purposes, several popular songs were sent to MoodLogic staff, which kindly collaborated on the initiative by providing feedback with the
descriptors for each of the songs. The specific values can be observed in the Results, Chapter 5. The emulated process would consist of this same procedure, but done automatically.

5.5 Profiles Database

According to the values of the MoodLogic Perceptual Descriptors, a certain profile that dictates a specific video editing style is chosen. This style is equivalent to an MPEG-7 Semantic DS multimedia descriptor. A profile can be thought of as a descriptor with a very high level of abstraction; required to be high enough to be a cross-modal semantic feature, common to all considered media.

As an example, consider two different possible soundtrack choices: a Punk-Rock track and a classical track. The former is typically a high-energy, rhythmic audio piece which, in general consensus would benefit from an editing style which includes bright images and fast changes ideally synchronized with the soundtrack’s rhythm. A profile could be created and can be called “dynamic,” “energetic,” etc. On the other hand, classical music, a more melodically oriented, generally less energetic musical genre, calls for an editing style with smoother transitions, benefiting from a sublime digest, probably linked to color histograms and pitch dynamics, among other characteristics. The corresponding profile could be labeled as “dramatic” or “emotional.”

In the scope of this system, the profile this database provides is used towards producing two pieces of information. First, its genre distribution dictates the original tension/release curve to be used in the construction of temporal montage forms. This is related to Figure 9, in Chapter 2. Revisiting this concept, a song exhibits a certain path of
tension/resolution from beginning to end. This is directly related to the video editing style, as it defines the ratio of video to images in each section. The higher this ratio, the more cuts present, therefore the more tension in the video. This way, in the prior example, the tension/resolution path for a Punk Rock track would be “higher” and “flatter” than that of an orchestral piece. Second, the mood and perceptual profile is mapped directly into defining the weights used in the audio and visual feature vectors discussed in Chapter 4. This affects the tonal montage progression of the piece. For example, if more significant weights are assigned to the loudness parameter in audio, and the brightness parameter in video, the resulting multimedia would be consistent towards loudness and brightness measures. This may be desirable in percussive music. On the other hand, if more importance is given to pitch and color, this could provide color and tonal consistency; something allegedly desirable for melodic music.

5.6 Feature Vector Weights Profiling

Although there are three different types of media being analyzed in the system, from a sensory point of view, the application can be described as containing two types of material: visual and auditory. This implies that if temporal information is unnecessary for a certain visual content analysis, which may likely be the case when extracting semantic descriptors, the same process may be performed both on the video frames and the photographs. The only important thing is not to confuse the structural descriptors that contain each of the analyzed subjects, which both belong to the Video Segment DS structure. In other words, the analysis performed by a visual feature extraction algorithm uses a subset of the MPEG-7 Visual Description Tools set, which is common to both
types of media. Analogously, the analysis performed by an audio feature extraction algorithm uses the MPEG-7 *Audio Description Tools* set.

This being said, the functionality of this block is straightforward; it consists of a decision structure that, according to the specific MPEG-7 profile presented at its input, decides on the specific weights to be assigned to each of the Audio and Visual description tools (i.e., Feature Extraction Algorithms) to be used throughout the system for that particular clip realization. Consider the generic feature vectors defined for audio and video respectively in Chapter 4. For example, a suitable profiling for these vectors in a percussive track belonging to Punk Rock is a bright, dynamic video editing style. Therefore, emphasis is assigned to loudness and brightness information. This would profile the vectors to have these values:

\[
\begin{align*}
A_i &= 0.7 AP_i + 0.1 AFF_i + 0.2 ASS_i \\
V_i &= 0.3 H_i + 0.1 S_i + 0.6 V_i
\end{align*}
\]

The specific weight values for this profile were chosen out of trial and error. Future work is proposed towards conducting perceptual tests to define more accurate weight profiles.

### 5.7 Visual and Audio Segment Description Blocks

Using the *Visual Description Tools* set, low-level feature extraction is performed on images which correspond either to the pictures or to the decimated video frames (the same frames that were used to compute the unsuitability score). The computed HSV
features instantiate a specific visual feature vector value for each video frame following the template of (13).

Similarly, the audio is analyzed towards extracting the subset of low-level descriptors of the Audio Description Tools set mentioned previously. Again, as a result, there is an instantiation of the feature vector (12) for each audio frame.

5.8 Clustering of Media Segments

Returning to the critical path of the flow diagram, this block receives the indexes for the Video, Audio, and Still Images segments (index information), as well as the actual descriptions of images and audio. It reorganizes the segments into collections by clustering media objects which are similar according to the considered high level feature of the profile, which is common to all media. This high-level information was integrated in the “Feature Vector Weights Profiling” Block, by defining the specific weights accompanying each low-level attribute. Feature consistency of every section is the main priority in this stage; tonal Montage is directly addressed. This in turn is mapped to the mood of the piece and the feelings of the audience.

Consistency is achieved through computing distances between either the magnitudes or the angles of the feature vectors. The first step is to cluster together samples from the same media which are most similar. This can be done by simply performing a sort operation on the magnitudes of the vectors. Another approach is to use the cosine angle distance between the vectors, and cluster according to this parameter.

This is then applied on a cross-media scenario. The different samples of the same media belonging to each cluster are averaged, and a representative feature vector is
created. A simple averaging of each of the components (e.g., an average for H, another for S, another for V in video) is sufficient. Each is then assigned their cross-media counterparts by selecting the vector which is closest in distance or angle to it.

The output of this subsystem is contained in a structure equivalent to the Collection DS MPEG-7 descriptor, which represents the clusters by a linkage structure (i.e., a tree hierarchy) of all the segments in each cluster.

5.9 **Subset Matching to Audio Timeframe**

In the final clip, visual material in each audio section will have a total duration of:

\[ T_i = V_i + \sum_k P_{ki} \]  \hspace{1cm} (16)

where \( i \) is the audio section (corresponding also to cluster number), \( T_i \) is the total duration of the visual material, \( V_i \) is the duration of the corresponding video segment, and \( P_{ji} \) is the duration for display of each picture inside the cluster (this duration is not fixed; it depends on the specific audio events detection i.e., metric montage). Depending if \( T \) is longer or shorter that the duration of the accompanying audio segments, minus the minimum quantity of pictures to be included in each section (towards following the desired video/images ratio that corresponds to the specific tension level for the section), then it is either cut down or complemented by more pictures.

Making \( T_i \) of the same exact duration as the audio section ensures that the total duration of the visual edition is equal to the duration of the audio segment in each cluster,
which guarantees coherent reconstruction when the final clip is to be conformed. Again, in this block, the media content does not need to be considered, as it is sufficient to provide the set of structural descriptors to perform the analysis. The output of this block is another descriptor, similar to the SegmentRelation DS. The specific function of this block is to provide these temporal relationships.

Additionally, it is closely linked to the following block; therefore there is a feedback loop returning the absence or surplus information from the section, to pull more pictures from the available pool of images, or store extra images and video that may be required later to fill up another starved incoming section.

5.10 SYNCHRONIZATION TO RHYTHMIC ATTRIBUTES

The next step is to synchronize the edited video in order to complete the temporal aspects of Montage. Again, semantics are introduced for this block; synchronization is done according to the tension/release premises discussed in Chapter 2. Consequently, the video/image cut ratio is changed on-the-fly. Hence, the need for the feedback loop towards the previous block, which feeds the visual material’s indexes as required, and fetches or stores images that are needed or not, respectively.

As an example, consider Figure 17, Chapter 3, where meaningful audio events coincide with minima on the plot. The rhythmic montage for such a section of audio is shown in Figure 29, for three different tension/release levels. It is clear that, the higher the level, the more density of cuts and saturation present in the final visual material. The specific level of tension and release comes from the User Interaction DS, together with a deviation proportional to average audio power (AP) for the respective section.
Figure 29 (a): Tension/Resolution ratio = 0.3

Figure 29 (b): Tension/Resolution ratio = 0.6
Figure 29 (c): Tension/Resolution ratio = 0.9

In the case of Figure 29 (c), the high density of cuts may require more pictures than what was initially assigned to the section; it becomes starved for pictures. The feedback loop then returns the information to the previous block and pulls more pictures from the pool of available documents.

The output of this system is equivalent to yet another MPEG7 descriptor; the AudioVisualSegment DS.

5.11 Reorganization of Audiovisual Stream

As its name implies, the last functional block in the system reorganizes the different audiovisual segments into a whole document. It is capable of doing this, because with the aid of the Time DS descriptor, absolute time in samples for each of the audio
segments and its transformation ratios towards video frames was relayed across the different blocks of the system. It has only to concatenate the segments in a chronological manner, and the produced output is the finalized video clip.
6

EXPERIMENTATION

6.1 PRELIMINARIES

Contrary to typical engineering situations, the classical definition of the Scientific Method is not suitable towards the analysis of some problems in multimedia description. The main reason for this is the inclusion of topics such as the ones discussed in Chapter 2. Certain concepts such as Montage, even though they can be quantified to drive deterministic algorithms, are not just psychological considerations; they constitute artistic premises, and as art, they cannot escape their subjective natures.

This dissertation poses a means of constructing a video creation system which follows certain common practices found in the world of art, but it does not constitute a fundamental truth in that a video has to be constructed according to these guidelines. Parallel implementations founded on completely different considerations are equally valid and possible.

A way of accommodating the Scientific Method to the analysis is to consider the parameters from a stochastic point of view, ignoring the currently unknown physical processes that govern the brain in perception, and basing the “truth” in the collection of experiences of an audience. Even though the premises involved are subject to people’s opinions, there is a general consensus among different groups on what is aesthetic and what is not. Recollection of these general opinions is translated into a quantity – something tangible that can be addressed scientifically, through the collection of user’s responses to subjective tests.
The approach taken towards examining the algorithm’s performance is chosen to review the same visual input according to different soundtracks. This provides a reference point for comparison in that in all cases, the visual input remains the same. The stochastic input is integrated by choosing different soundtracks which have been judged by the general audience as highly dissimilar, based on metrics that will be discussed in what follows. The hypothesis is that the system will create music video digests which are different in both their rhythmic and tonal montage components, but coherent with each specific soundtrack. The tension/release progressions, as well as the consistency of the low-level attributes, are unique for each soundtrack, even though the source visual material is the same. This guarantees that it is the edition process, not the input material alone, which creates the semantic progression that is not addressed by already existing products.

The choice of soundtracks for system testing is done in a way that a strong degree of dissimilarity between the candidates is guaranteed. Musical descriptions that support this idea are employed. While humans find it difficult to describe music in absolute terms, valid descriptions can be created by comparisons to already existent media by using anchor parameters [48]. For example, the Rhapsody Music Subscription Service [49] describes the progressive rock band “Dream Theater” as:

*The Berklee-bred quintet have fashioned a style that feeds off remnants of Rush, Speed Metal, and arena-ready prog rock (think Styx), all mixed in with a not-so-small dose of 1980s hair-band perfumery. Most commonly referred to as "Progressive Metal," Dream Theater's music isn't the sort that inspires headbanging. Marveling at the band's clinically precise execution...*
This description never focuses on absolute musical structures such as form, rhythm, etc. but rather compares the music to a series of subjective parameters and pre-existing similar music. In other words, it draws on parameters from parallel anchors. In this example, anchors are genre information, music bands, and historical periods. Even though the analysis in [49] proceeds to employing neural networks and decision trees, it is clear from this that a music from a band such as Dream Theater would fall in a time cluster belonging to the hard rock genres, influenced by music and culture from the 1980’s, and so on.

For experimentation, three songs are considered. The anchors to judge the soundtrack candidates as dissimilar are the following:

- **Genre Information**: Three genres which are subjectively dissimilar may be rock, electronic music and classical music.

- **Perceptual Beat**: Three dependencies on beat can be considered; strong (percussive), medium (both percussive and melodic) and soft (melodic).

- **Timbre**: Three types of music can be identified; Tonal, percussive and textured.

Three songs which are categorized inside the intersection of the preceding anchor points are:

- “Wish You Were Here” by Incubus [37]. Its predominant genre is Rock, has a medium perceptual beat (meaning that it has drums and also melodic progressions), and a textured sound produced by vocals and electric guitars.
- “Galaxy Bounce” by The Chemical Brothers [35]. The genre is Electronica, relies heavily on beats and its sound is predominantly percussive.
- “Back To School” theme by Danny Elfman [50]. It is a classical/orchestral piece, with nonexistent percussive beats, comprised of mainly tonal instruments.

6.2 METHODOLOGY AND PARAMETERIZATION

6.2.1 MoodLogic Queries

The three songs just mentioned were sent as a query to the MoodLogic Database (actually, this was emulated, as was explained in Chapter 4). This returned the files for perceptual attributes specified in Annex A. Chosen relevant descriptors were Genre Distribution (Disregarding Sub Genre), Perceptual Energy, Perceptual Beat and Perceptual Valence. A summary of these values is provided in Table 3:

<table>
<thead>
<tr>
<th>Song</th>
<th>Genre Distribution</th>
<th>Perceptual Energy</th>
<th>Perceptual Beat</th>
<th>Perceptual Valence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wish You Were Here</td>
<td>Alternative: 212 Rock: 43</td>
<td>170</td>
<td>160</td>
<td>244</td>
</tr>
<tr>
<td>Galaxy Bounce</td>
<td>Electronica: 245 Pop/Dance: 10</td>
<td>244</td>
<td>244</td>
<td>244</td>
</tr>
<tr>
<td>Back To School</td>
<td>Easy Listening: 213 Jazz: 19 Pop/Dance: 23</td>
<td>244</td>
<td>244</td>
<td>117</td>
</tr>
</tbody>
</table>

Table 3: MoodLogic Query
6.2.2 Profiles Definition

The profiles database structure of the system’s block diagram, depicted in Figure 28, Chapter 5, is responsible for analyzing values returned from the MoodLogic query. It instantiates the tension/release curve and feature vector weights that in turn control the montage aspects of the final multimedia document. At the present point of system evolution, it consists of look-up table structures in which input descriptors push a pre-defined series of values. For the preliminary tension/release curves, there is a fixed database of curves for each genre anchor value, for all the genres that were returned by MoodLogic, consisting of ten points of data. This data is normalized from 0 to 1. As an example, a representative curve of the “alternative” genre is shown in Figure 30.

![Tension/Resolution Curve - Alternative](image)

Figure 30: Tension/Resolution Curve for the Alternative Genre
This figure shows the tension/resolution progression of a “typical” alternative music track. It is based on the progression of an introduction, verse, chorus, verse, chorus, variation, CODA. The conception of this curve comes from the observations of typical songs from the genre; the majority allegedly follow a progression in the form AABA (i.e., Rounded Binary Form) [19]. At the evolitional stage of this dissertation, this is considered to have enough semantic correlation to portray the idea of the T/R ratio; there is obviously an immense opportunity for finer implementations on this topic.

To construct a curve for one of the given input soundtracks, the genre distribution is assigned a curve for each one of the genres returned in the distribution, and each is weighted by its respective relevance to the overall distribution of the soundtrack. This is achieved by dividing the magnitude of the attribute by 255 (the total of the genre distribution). It is then added/subtracted a small deviation coming from the average audio power (MPEG-7 AP descriptor), to adapt to the specific form of the song. For further explanation, refer to Figure 10 in Chapter 2. Continuing the example, consider the distribution for the “Wish You Were Here” soundtrack in Table 3. This weights the Alternative curve by 0.83 and the Rock Curve by 0.16, and “fine tunes” it according to the significant dynamic variations of this particular song. The average curve actually depicts the form of the music, as shown in Figure 31.
Figure 31 (a): Tension/Resolution from Genre Distribution

Figure 31(b): Tension/Resolution affected by Audio Power
Perceptual energy and perceptual valence values define instantiation of the Feature Vector weights. Energy and valence can be mapped to the overall dynamics of a musical piece. The variations from soft to loud sections of a composition affect its consistency in tonal montage. Certain musical styles (often encountered in orchestral music) exhibit abundant loudness variations and accentuations, and their transitions in energy are carefully composed and are generally accompanied by melodic progressions. Other musical styles follow more binary approaches, where loud passages are even aided by significant amounts of compression and soft passages are generally their minimal counterparts; they also achieve this relying heavily on rhythmic components.

From the three chosen tracks, it is observed that the amount of dependence on these energy variations for a song is inversely proportional to the average of these two values. This is a logical consequence of dynamic music being generally more dramatic than music which is constantly high energy (e.g., this is a truism of a fundamental difference between classical and modern popular genres). Considering the analyses conducted in Chapter 2 and other visual music resources [34], a mapping of audio and visual low level attributes can be constructed, not towards semantic progression but rather towards providing consistency, as shown in Figure 32. The two perceptual attributes are averaged and, the higher the result, the more dependent is the feature vector in audio power and brightness. On the other end, there is more dependency on color and tonal information. In the case of the chosen soundtracks, the music is more oriented towards color attributes, except for the song “Galaxy Bounce,” which is very energetic.
A further functionality that the MoodLogic Query provides is to define the Music Type depending on the dependence of the music to a percussive beat. This is addressed directly by the Perceptual Beat attribute. One way of employing it is to cluster music into “melodic” or “rhythmic” style by choosing a threshold which if surpassed implies that music is heavily relying on percussion and beats. This permits the choice of synchronizing to specific audio events, or more complex temporal structures such as meter and BPM measures, which may be applied quantitatively only to music with a very salient rhythmic structure. This is also implemented at a basic level for this project, although it leaves again a lot of ground for future work.

6.2.3 Instantiation of Thresholds

Concerning thresholds, their functions and specific values are shown in Table 4.
Table 4: Instantiation of Thresholds

<table>
<thead>
<tr>
<th>Threshold</th>
<th>Block</th>
<th>Function</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Novelty</td>
<td>Novelty Preprocessing</td>
<td>Ignores insignificant local maxima in the novelty curve; controls the amount of sections for a song’s form. If local maximum falls below threshold, it is ignored.</td>
<td>0.2 in a normalized curve.</td>
</tr>
<tr>
<td>Novelty Energy</td>
<td>Novelty Preprocessing</td>
<td>Ignoring specific audio events that still appear at large temporal scales; this may be fundamental at later stages but may hinder detection of true novelty transitions. Such a peak is extremely transient; therefore its integral is small. If sum is below threshold, it is ignored.</td>
<td>1.5 for integral in normalized curve.</td>
</tr>
<tr>
<td>Audio Duration</td>
<td>Novelty Preprocessing</td>
<td>Audio sections may not be too long or too short, towards preventing overly long or short video clips from occurring. This is in accordance with the rules discussed in Chapter 3. If section is short, merge it with next section. If too long, split at maximum novelty point inside section.</td>
<td>3 seconds; 20 seconds</td>
</tr>
<tr>
<td>Video Duration</td>
<td>Unsuitability Preprocessing</td>
<td>Analogous to Audio Duration; discussed in Chapter 3.</td>
<td>8 seconds; 30 seconds</td>
</tr>
<tr>
<td>Suitability Threshold</td>
<td>Unsuitability Preprocessing</td>
<td>Measures the maximum amount of unsuitability that is admissible in the clip. If the curve surpasses this threshold, it is removed from the pool of eligible video because it is considered low quality. Chapter 3, Fig. 24.</td>
<td>0.25 in a normalized curve</td>
</tr>
<tr>
<td>Image Sample Quantity</td>
<td>Visual Segment Description</td>
<td>Number of sample frames in a candidate video clip to be analyzed and averaged towards description of the whole clip with the low-level HSV feature vector. Must be large enough to ignore noise but small enough to prevent edge effects and computational complexity.</td>
<td>5 images per segment</td>
</tr>
<tr>
<td>Beat threshold</td>
<td>MoodLogic Profiling</td>
<td>A simple decision structure based on the Perceptual Beat attribute, if its magnitude is above this threshold, the music is considered predominantly percussive. Else, it is considered predominantly melodic</td>
<td>180 out of 255</td>
</tr>
</tbody>
</table>
6.2.4 Visual test material

Visual input consists of a collection of 100 pictures from the author’s personal collection. They were chosen as to provide a ground truth according to these perceptual anchors:

- Color: Subsets of pictures which have an evident homogeneity in color were chosen, representing blue, yellow and green hues. It tests consistency in the hue (H) value of the feature vector.

- Brightness: Images taken at daytime and nighttime were included, towards testing of such consistency by using the value (V) parameter in the visual feature vector.

- Saturation: There is also a textured gray color corresponding to urban images and another one from images of trees. This is included as a wildcard for color and brightness values. It is interesting to include this subset and observe where these images are placed in the final video.

For video, a home video was recorded in the DV format, with different scenes happening at environments exhibiting a variety of colors and brightness values. They were chosen as to correlate with the photograph set, towards examining if the system can provide consistency between the video and the images. These scenes were recorded:

- A marina and sailing boats
- The flight of birds with blue sky as background
- A dog playing in a green field
- Driving through Downtown Miami, FL
6.2.5 Programming language

The system was implemented in the MATLAB [38] programming environment, towards effective creation of a signal processing prototype. Consequently, audio was transformed from MP3 format to uncompressed Microsoft wave (i.e., .wav audio). This presented no problems, as self similarity analysis is generally robust towards a certain degradation of audio quality. Video was recorded from the DV format into an MPEG2 document and rendered to .AVI format using Adobe Premiere [3] software. This was necessary for input into MATLAB. The original frame rate of 29.97 FPS was decimated to approximately 1 FPS for efficient computation. Degradation of quality and temporal resolution do pose a significant cause for analysis error, but it is tolerable in the light of video being edited, not analyzed (audio is the “dominant” media).

The final output of the MATLAB system is a collection of sample times corresponding to the music, which index start and end points of visual material. These are matched in a data structure by video absolute frame indexes, which map the audio sample times into corresponding frame indexes that point to the boundaries in included video. The sample times from the MATLAB output are instantiated as markers for the video document, and implemented in the Sony Vegas [4] video creation suite.
6.3 Results

6.3.1 Profiling

The curves corresponding to Tension/Resolution progressions, as defined by the available genre distributions from the MoodLogic query are shown:

Table 5: Anchor Genres Tension/Resolution Curves

<table>
<thead>
<tr>
<th>Anchor Genre</th>
<th>T/R Curve Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alternative</td>
<td>[0.8 0.8 0.4 0.4 0.8 0.4 0.3 0.2 0.2 0.6]</td>
</tr>
<tr>
<td>Classical</td>
<td>[1 0.7 0.5 0.6 0.8 0.6 0.5 0.4 0.2 0.5]</td>
</tr>
<tr>
<td>Easy Listening</td>
<td>[1 0.8 0.7 0.7 0.5 0.4 0.5 0.6 0.4 0.7]</td>
</tr>
<tr>
<td>Electronic</td>
<td>[0.6 0.4 0.5 0.4 0.3 0.3 0.1 0 0.2]</td>
</tr>
<tr>
<td>Jazz</td>
<td>[0.8 0.5 0.5 0.3 0.4 0.5 0.3 0.4 0.6 0.8]</td>
</tr>
<tr>
<td>Pop</td>
<td>[0.6 0.4 0.2 0.2 0.4 0.4 0.2 0.2 0.2 0.4]</td>
</tr>
<tr>
<td>Rock</td>
<td>[0.5 0.5 0.3 0.6 0.5 0.4 0.3 0.2 0.2 0.5]</td>
</tr>
</tbody>
</table>

Additionally, the perceptual Beat threshold for distinguishing melodic and percussive music was chosen to be 180/255. This is high enough so as to guarantee that only music with a very dominant beat is classified as percussive.
6.3.2 Visual Media

Figure 33 shows the result for unsuitability analysis.

![Video Unsuitability (FPS = 1)](image)

Figure 33: Video Segmentation according to Unsuitability. The white regions represent candidate video clips; shadowed regions are unsuitable and therefore ignored.

This curve was computed by the use of HSV differentials as discussed in Chapter 3.

Input photographs were included and analyzed together with representative frames from the extracted video clips. Their HSV values were weighted according to whatever parameter was to be held for color consistency (from user profiling). They were then categorized according to the feature vector’s magnitude, and organized into
preliminary clusters. Results for a feature vector relying heavily on color (i.e., HUE) are shown in Figure 34.

Figure 34: Histogram of elements in visual material (Clips & Images). Each color bar represents the centroid of a visual cluster; all images Fall into one of these clusters, as shown.

The visual clusters arrange images into groups, in this case corresponding to dominant color. This provides consistency in media organization. For example, there is a large population of images in the ninth cluster (light blue). This corresponds to identifying all images taken with the ocean as background. Taking images from this cluster into the video implies that the specific passage will be concerned mostly with scenes around the ocean.
6.3.3 Soundtrack: “Wish You Were Here”

The first song to consider is “Wish You Were Here” by Incubus [37]. For convenience, the various graphs pertaining to audio analysis were included in one figure, shown on the following page. Initially, the right hand portion of the figure is read from bottom to top. It depicts the Self-Similarity matrix performed on preprocessing towards detection of Audio Novelty. The corresponding Novelty Curve is linked to this matrix in the figure above it.

Further up, the profiled Tension/Resolution curve, adjusted to the rhythmic montage of the song, is also linked to the Novelty. Different sections in the song are identified, and a summary of each is provided in the table to the right. Finally, at the bottom-right corner of the figure, a progression for the corresponding Audio Feature vector is shown. It depicts the magnitude of this vector per song section. This song was profiled by MoodLogic as Melodic and having a medium-level Perceptual Energy-Valence average (Refer to Table 3.) therefore, the Feature Vector is instantiated as follows:

\[ A_i = 0.8AP + 0.3AFF + 0.6ASS \]  \hspace{1cm} (17)

The reader is encouraged to listen to the song while looking at these figures towards a better understanding of the concept.
Figure 35: Audio Analyses for “Wish You Were Here”
Figure 36 depicts the audiovisual arrangement, where finally video and music come together. Having chosen the visual feature vector as predominantly color oriented, the figure shows how color consistency is mapped into the music’s progression. However, the clustering of color depends more on the evolution of the audio feature vector (bottom-right corner of Figure 35), but it is more informative to show its relation to the Tension/resolution Progression, as it depicts the complete montage creation of the document.

![Tension/Resolution Progression](image)

**Figure 36:** Audiovisual Clustering. Rhythmic Montage, defined by the T/R curve, is complemented by Tonal Montage, where every section is assigned an individual color cluster.
The T/R ratio is directly related to the proportion of video to images in every section. The more video, the less tension. Results for this track are shown in Table 6.

<table>
<thead>
<tr>
<th>Section</th>
<th>Duration (s)</th>
<th>Video Portion</th>
<th>Image Qty</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10.6</td>
<td>85%</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>23.6</td>
<td>71%</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>32.8</td>
<td>47%</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>23.4</td>
<td>39%</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>11.2</td>
<td>53%</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>22.4</td>
<td>72%</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>15.6</td>
<td>35%</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>9.2</td>
<td>25%</td>
<td>9</td>
</tr>
<tr>
<td>9</td>
<td>21.2</td>
<td>28%</td>
<td>7</td>
</tr>
<tr>
<td>10</td>
<td>9.8</td>
<td>33%</td>
<td>8</td>
</tr>
<tr>
<td>11</td>
<td>17.6</td>
<td>22%</td>
<td>9</td>
</tr>
<tr>
<td>12</td>
<td>10.2</td>
<td>72%</td>
<td>2</td>
</tr>
</tbody>
</table>

6.3.4 Soundtrack: “Galaxy Bounce”

“Galaxy Bounce” by The Chemical Brothers [35] is analyzed in a similar manner as the preceding song. The Audio Analysis figure is shown in Figure 37. This song was profiled by MoodLogic as Percussive and having a very high Perceptual Energy-Valence average (Refer to Table 3.) therefore, the Feature Vector is instantiated as follows:

\[ A_i = 0.9AP + 0.0AFF + 0.1ASS \]  \hspace{1cm} (18)

The reader is encouraged to listen to the song while looking at these figures towards a better understanding of the concept.
Figure 37: Audio Analyses for “Galaxy Bounce”
Figure 38 depicts the audiovisual arrangement. Note that for section 10, three clusters were used. This is actually a section which is 51 seconds long, needing 27 images (The Duration threshold did not apply probably because there was no significant novelty present throughout that section). Therefore, other clusters had to be searched for images.

Figure 38: Audiovisual Clustering for “Galaxy Bounce”
Table 7: Proportion for “Galaxy Bounce”

<table>
<thead>
<tr>
<th>Section</th>
<th>Duration (s)</th>
<th>Video Portion</th>
<th>Image Qty</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>18</td>
<td>72%</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>41</td>
<td>40%</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>31</td>
<td>54%</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>16.6</td>
<td>36%</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>12.8</td>
<td>24%</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>3.6</td>
<td>21%</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>7.4</td>
<td>23%</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>42%</td>
<td>3</td>
</tr>
<tr>
<td>9</td>
<td>11.4</td>
<td>5%</td>
<td>5</td>
</tr>
<tr>
<td>10</td>
<td>51.2</td>
<td>8%</td>
<td>27</td>
</tr>
<tr>
<td>11</td>
<td>6.4</td>
<td>46%</td>
<td>4</td>
</tr>
</tbody>
</table>

6.3.5 Soundtrack: “Back To School”

This is a theme soundtrack by classical composer Danny Elfman [50]. The Audio Analysis figure is shown in Figure 39. This song was profiled by MoodLogic as Percussive and having a low Perceptual Energy-Valence average (Refer to Table 3.) therefore, the Feature Vector is instantiated as follows:

\[ A_t = 0.5AP + 0.8AFF + 0.2ASS \]  \hspace{1cm} (19)

The reader is encouraged to listen to the song while looking at these figures towards a better understanding of the concept.
Figure 39: Analyses for “Back To School”
Figure 40 shows the Multimedia Montage for this piece. Something peculiar about this composition is that colors not necessarily match high or low tension levels. Again, this is something coherent because colors are not chosen according to the T/R curve, but from the Audio Feature Vector progression depicted on the lower right corner of Figure 39.

![Tension/Release Progression](image)

**Figure 40:** Audiovisual Clustering for “Back To School.”

<table>
<thead>
<tr>
<th>Section</th>
<th>Duration (s)</th>
<th>Video Portion</th>
<th>Image Qty</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>21</td>
<td>78%</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>3.4</td>
<td>74%</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>7.2</td>
<td>58%</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>19.6</td>
<td>42%</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>5.4</td>
<td>50%</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>4.2</td>
<td>50%</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>14.2</td>
<td>47%</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>11.3</td>
<td>34%</td>
<td>4</td>
</tr>
</tbody>
</table>

**Table 8:** Proportion for “Back To School”
After reviewing the resulting compositions for each of the soundtracks, the system then proceeds to assign image and video boundaries on specific audio events, as was discussed in Chapter 3. An example of results on this topic can be observed in Figure 29, Chapter 5, where some sections for the song “Wish You Were Here” are shown.

These three results are very informative in that they are useful for identifying several advantages and flaws of the system, as will be discussed in the following Chapter.

6.3.6 Performance Measurements

It is interesting to analyze how a computer system is affected in its runtime by execution of the program. Towards quantifying this performance, the system was tested in a computer with specifications stated in Table 9.

Table 9: Computer Testing Platform

<table>
<thead>
<tr>
<th>Computer Model</th>
<th>Dell Latitude D600</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>Intel Pentium M @ 1.6 GHZ</td>
</tr>
<tr>
<td>Memory</td>
<td>512 MB DDR</td>
</tr>
<tr>
<td>HD</td>
<td>80 GB</td>
</tr>
<tr>
<td>OS</td>
<td>Microsoft Windows XP SP2</td>
</tr>
<tr>
<td>Software</td>
<td>Matlab 7 R 14</td>
</tr>
</tbody>
</table>

The test of section 6.3.5 was chosen for analysis of system performance. A recapitulation of the characteristics of input is in order.

- Soundtrack:
Experimentation

- Duration time: 01:26 mm:ss
- Format: WAV mono (downmixed), 16 bits, 44100 bps

- Unedited Video:
  - Duration time: 17:10 mm:ss
  - Format: AVI, 360x240, 32 bits, 30 fps

- Photographs:
  - Quantity: 137 pictures
  - Format: JPG, 320x240, 24 bits

Figure 41 shows the performance of the system towards percentage of processor load, at execution time for the program.

![Processor Load](image)

**Figure 41:** System Performance – Processor Load

Analysis of System Performance towards memory management is shown in Figure 42.
Finally, a graph showing Hard Drive access is shown in Figure 43.

6.4 ANALYSIS

Both the Media Segmentation and Media Description stages for the visual material output what was expected; a curve for video unsuitability with corresponding thresholds was constructed, and the HSV feature vector which described both the pictures
and video sets grouped the media into coherent clusters, as shown in the example of a dominant Hue Feature Vector (Figure 34).

As can be observed from the audio figures corresponding to each specific summary creation test (Figures 35, 37 and 39), the Tension/Resolution curve (shown at the top left corner) is computed according to two parameters. The most influential is the preliminary curve produced by the Genre Distribution query. This has the effect of producing gradual increases in tension to reach a maximum point at approximately 80% of the song, to consequently provide a resolution for a coherent ending section. On the other hand, the details of the specific sections which characterize each individual song are included in this curve by means of the audio low-level feature vector (shown at the bottom right corner). This last description completely coincided with the subjective descriptions stated in the accompanying tables for each Figure.

Finally, the audiovisual clusters were shown in Figures 36, 38 and 40, respectively. In the case of a dominant Hue feature vector, it can be seen that for each individual section there is one main cluster that is addressed (with some exceptions in “Galaxy Bounce”), guaranteeing that there is similar thematics throughout each specific section. In the case of dominant Brightness feature vectors, an analogous process occurred; however, the common parameters for consistency were different, like clusters of day vs. night, etc.

Concerning performance analysis, there is a very interesting behavior from the system. The first portion of the execution commits a considerable load on CPU; on the second portion, the processor is released and the emphasis is now on memory. This implies that in this application, audio demands processing power while video demands
memory allocation. This is a logical consequence of the fact that the deep analysis and feature extraction is performed on the audio, which is the dominant media. Video is not analyzed as deeply, as it only needs to be synchronized with what audio may dictate. However, visual media is a considerable memory consumer, since each uncompressed frame consists of a cube of matrices, as opposed to audio media where per sample there is only two individual values (in a stereo scenario).
Multimedia applications are an essential part of modern life. The digital approach to information has become standard for every technology that conveys to the senses. Analog has taken the role of purist and exotic applications, but is no longer a practical alternative. On the other hand, Digital technology is extremely efficient and practical. In fact, it is so practical that it offers to the typical computer user tools which enable the storage of thousands of documents, and the parallel processing of several different documents at any one time. The ground has been prepared for addressing a new issue in multimedia: the efficient management of this overwhelming collection of information.

Video is no exception to this tendency; the average user has access to a digital camera for recording such media, a computer to process and playback, and vast memory space on which to store documents. Unfortunately, the average user is also not a professional film director or visual editor. The consequence is that this home-produced video is generally of very low quality and allegedly somewhat boring.

Solutions in this digital scenario have been proposed, are successful and continue to evolve. Towards implementing a completely unsupervised application, the consensus has been to adopt an approach of segmenting the video and pictures that the user recorded, basing the temporal aspect of the media on an existing soundtrack from the user’s personal library. This guarantees professional audio quality, and a patterned time structure to support the sequenced playback of only the best quality portions of the raw video. These solutions have nonetheless ignored the parallel
Conclusions

Betancur, N. May 2006, University of Miami

122

The evolution of media description theories and technologies which go further than a one-to-one synchronization with time. There is a huge possibility on expanding synchronization towards semantics and artistic concepts; things that are also followed actively by the film director in conventional manual video compositions. This work tries to open the door towards that realm of possibilities.

It does so by implementing a system which follows rules stated in audio and visual art theories, transforming these rules into algorithms that analyze and process the soundtrack, the video and the images accordingly. Throughout the preceding chapters, both the conceptual and systematic considerations to make this a functional system have been discussed. The final result has been a basic, but nonetheless synchronized AND coherent multimedia document.

The nature of a system such as this is considerably subjective; the standard Scientific Method cannot be applied directly as a proof to its success. The most important functionality of this system is to enhance the systems that exist towards automatic video summary creation, by providing a higher aesthetic value through the use of Temporal and Tonal Montage forms. From the analyses in the previous chapter, having produced the output that was expected, two major novelties that have been proven to be successful can be mentioned:

- The temporal structure of the music has been addressed in an efficient way, towards detecting both the fine rhythmic structures (Metric Montage) in specific audio events and the higher-level musical form descriptions (Rhythmic Montage) through the use of Novelty. They were proven to be successful with the obtained results. Therefore, the concept that “Art is
Conflict” [19], which creates a Tension/Resolution pair, is a key enhancement to simple fixed one-to-one audiovisual mapping in the temporal structure. This correlates psychologically to the interest level that an audience will have towards the video, as provides a rhythmic structure to the multimedia document. This was implemented successfully in the system through the instantiation of the Tension/Resolution progression, providing coherent rhythmic Montage.

- Addressing temporal relations is necessary, but not enough to produce a truly coherent document. There has to be a sort of order, or consistency, to the sequence of the media; this is called Tonal Montage. A cross-mapping of audio and visual low-level features can provide this successfully, as was shown by the consistency in clusters on the audiovisual mapping results. Blue stayed blue, red stayed red, and so on. Even though mapping color attributes directly to audio features has been proven not be achievable in a one-to-one basis, again it is the consistency of the features which is important. As the audio soundtrack may not be broken into a disorganized sequence of sections (i.e., move the verse to the CODA and the chorus to the verse, and so on), visual material loses its meaning if it is shown inconsistently. This has been implemented successfully by clustering audio and visual media into similar feature vectors, which retain their values throughout every section, preserving similarity among the low-level features of the sequence.
Other conclusions derived from this system which are desirable towards producing the multimedia document are:

- Adopting MPEG-7 as the framework for implementation of such a system is also something which has not been addressed in prior art. This is something completely logical from a technological standpoint; the media descriptors and schemes proposed in MPEG-7 are applicable to the system. It would be expected that, as MPEG-7 gains popularity in the Multimedia community, related technologies are going to migrate towards adoption of this standard.

- An enhancement also produced from adopting MPEG-7 (implying interoperability with other technologies) is the possibility created by MoodLogic [46] to integrate user profiling to the system. Several features coming from genre and general opinions of music are “impossible” to be extracted automatically. MoodLogic perceptual attributes provide this information conveniently.

Revisiting the results mentioned in the previous chapter, there are several specific precepts which have been identified as important to maintain the semantic multimedia link:

- When defining a Tension/Resolution progression, both the pre-defined curve produced from MoodLogic Profiling and the fine adjustments provided by Audio Power descriptions are important for coherency. For example, the song “Wish You Were Here” presents tension buildup almost
until the end of the song, with its corresponding resolution afterwards; something typical to rock music. However, the variations in loudness encountered since practically the start of the piece cannot be ignored, as they provide a specific semantic signature to this piece.

- Audio power, more than being the simplest low-level descriptor to implement, is of extreme importance in both profiling the T/R curve and also providing consistency to the visual material. It must always hold a significant weight within the audio feature vector. For example, low-energy sections may contain tranquil pictures of the sea, while high energy passages may contain only pictures of a city. This can be identified visually with retaining similar hue and saturation values.

Several conflicting issues have also been encountered along the way. Some characteristics of the output documents were identified, which at first glance conflict with the synchronism and semantic coherence being sought.

- Looking at Figure 34, where the entire Hue spectrum is depicted, not all colors are shown. For example, purple is not in the picture. This was initially considered wrong until observing the fact that there were no purple pictures in the set. Including some predominantly purple pictures would probably correct this issue.

- In the song “Galaxy Bounce,” there is a 51 second passage requiring 27 images. The effect is that the section used up every available picture in its cluster, and had to switch clusters to keep on pulling images into the
composition. This implies inconsistency in the visual material. However, for this specific case, inconsistency may be desirable. The only way a 51 second passage can occur is if there is no significant novelty point for the span of the region. This may render the visual composition somewhat uninteresting. According to [19], color inconsistency produces an immediate raise of visual tension. Hence, switching visual clusters heightens tension in an allegedly boring passage, so this may actually be convenient.

7.1 Future Work

Throughout the different discussions in this dissertation, it is evident that this is only a peek into the vast world of visual music, as applied to automatic video editing applications. Hence, future work on the subject is vast and promising. Concerning what was discussed in Chapter 2, basically any visual or musical artistic premise that can be applied creatively to the application can produce interesting results. Some standard techniques in film which have not been implemented in this prototype can enhance Montage [19], like slow pan and zoon procedures over the images.

Additionally, Eisenstein’s conclusions yet deal with two other forms of montage: Overtonal and Intellectual. He considers these two types as an integration of Metric, Rhythmic and Tonal Montage forms, towards providing meaning to the visual composition as a whole. This way, the movement of thought rather than of light structures is addressed. This is undoubtedly something worth investigating, as it promises yet a greater enhancement to semantic coherency.
Similarly, the works of John Whitney [25] on which much of the premises of this system are based on, are only implemented in their basic structures. Much work has been done by Whitney on the Tension/Resolution approach to audiovisual composition; his definition of “Differential Dynamics” is a complicated set of rules which, as the name implies, deal with feature vector differentials which hint of tension/resolution pairs in semantic structures. A more intricate implementation of the Differential Dynamics concept, not only constrained to the lower level attributes but also addressing semantic differentials is undoubtedly an enhancement.

Both Eisenstein and Whitney strongly believed though, that the best art comes from that artist who is able to break the rules in a beautiful way. This is a very important consideration towards defining the market to which a system such as this is going to be marketed. A film director would probably appreciate more control over the Montage features of the document, especially for Tonal Montage. For a talented film director, it may actually be the greatest creation to provide scenes which exhibit immense tension in one media form, while providing the opposite in the other media. Anything artistic must not be bound to such rules as the ones proposed in this dissertation. However, the temporal mappings and description framework are tools of extreme value to the artist. An important enhancement is hence to provide several modes of operation with the system, where the typical home user sets every parameter to automatic, as this dissertation proposes. Additionally, the existence of more manual and “professional” modes, which enable a user to specify the TR curve manually (e.g., by drawing) or to specify certain video regions to be included in the final output, would greatly expand the scope of this initiative.
The topic of a system being supervised or automatic is one of much relevance, due to the fact that higher-level semantics are sometimes impossible to be extracted by an algorithm. This is the reason of why the MoodLogic query was incorporated in the first place. Considering this idea further, MoodLogic offers a broad portfolio of perceptual attributes that can be used other than the ones proposed here. A generalization of this is that the more perceptual or user-created data (with a high degree of confidence) is integrated into the system, the more semantics can be addressed.

The work performed on Multimedia Montage is also interesting for further investigation [13] [14]. The researchers involved actually have devised an algorithmic structure towards scripting the fine details that contribute to Montage in a multimedia document. Integrating this Meta-Script with the MPEG-7 framework, towards controlling the parameters of this application is an interesting possibility.

Work on Self-Similarity also poses a good alternative for further exploration [12] [51] [52]. The concept of Self-Similarity is by no means constrained to music or even audio. Self-Similarity can be performed on any progression; on any media which involves an evolutionary dimension such as time. Work has been done on performing this kind of analysis in video, and probably good results will be achieved. The same research group that works on these topics [11] also provides a means of analyzing video unsuitability in a much more efficient way than what is proposed herein; to incorporate visual structures such as motion vectors and macro blocks into the Video Unsuitability computation.

Along these lines, MPEG-7 is a standard which offers ways to describe media that are much more efficient than basic color differentials. Video analysis which includes
information on shapes and patterns is desirable to include. For example, if face recognition is implemented efficiently, then for every major character a separate video can be produced towards having a specific person as the main character of the video. It could also be implemented in such a way that each audio section involves one main character. Any other possibility that MPEG-7 currently supports can be adapted.

On certain application scenarios, unsuitability may not be the best way to perform video segmentation. As an example, consider the case of soccer matches. Goals, which include high movement, could be judged as unsuitable. An approach such as the one described in [9] is much more adequate than unsuitability, to guarantee that every highlight (e.g., every goal) is included in the music video. These description algorithms can be used to contribute to the montage aspects of the piece, especially the overtonal and intellectual aspects discussed previously.

Concerning audio, many description algorithms are available for enhancing synchronization, especially for each specific musical section. The Beat Spectrum idea [16] is mentioned and implemented very basically in this work. If this is explored further towards obtaining efficient meter and tempo descriptions which relate directly to the song’s form and perceptual attributes, then specific structures for each song can be addressed in a much more coherent way. Consider for example a rock music song in 4/4 time. If this 4/4 structure can be identified, then not only could visual material be made to match the specific audio events, but also the time signature of the song, (e.g., for two bars, a video for one whole bar, and then four pictures; one for every beat). This would imply a much more coherent way of addressing rhythmic multimedia montage.
Finally, a call is made for a more thorough evaluation for the aesthetic value of the output, which can be achieved by performing qualitative tests of results and comparing them to existing products such as Muvee Technologies [10]. It would be extremely interesting to inquire the opinion of audiences with different backgrounds and especially ages. What advantages/disadvantages does an adult observe? How about a child? The most attractive market could be actually the teenager population. Such a test would definitely point out where this idea is successful and where it is weak.

Multimedia Information Systems are becoming more standardized in both the professional and home scenarios. A truism of this field of technology is that techniques from one type of media can be applied to other types creatively. This automatic video summary creator is a good example of such integration. With the advent of new standards such as MPEG-7, the computational power of modern devices and the renewed artistic interests for cross-modal applications, the future for visual music is surely a bright one.
REFERENCES


References


[29] Cook, P.R., Music, Cognition and Computerized Sound, MIT Press, Boston, 1999


[53] Slaney, M., mfcc.m – Compute Mel Frequency Cepstral Coefficients, (MATLAB). Interval Research Corporation, 1998


APPENDIX A.
MOODLOGIC QUERY

The following pages present the actual data of the MoodLogic Query Simulation. It is observed that data other than genre distribution for the song “Back To School” was not available. This was substituted by the data of the song “The Simpsons,” as it is also a song pertaining to a soundtrack theme, composed by the same artist (Danny Elfman), and exhibits similar melodic and rhythmic progressions, as well as moods.

This information is displayed exclusively for academic purposes, and it is subject to copyright from MoodLogic Inc., All Rights Reserved.

The scores within genre distributions and mood distributions total to 255, so the songs "membership" in a particular genre/sub-genre is the score / 255.

Energy, valence, and beat are also out of 255.

I  “Wish You Were Here,” Incubus:

<table>
<thead>
<tr>
<th>Genre Distribution:</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Genre: Alternative, Sub-Genre: AlternaPop) 31</td>
</tr>
<tr>
<td>(Genre: Alternative, Sub-Genre: Grunge/Post Grunge) 55</td>
</tr>
<tr>
<td>(Genre: Alternative, Sub-Genre: Punk/Pre-Punk) 19</td>
</tr>
<tr>
<td>(Genre: Alternative, Sub-Genre: Rap/Funk/Metal) 107</td>
</tr>
<tr>
<td>(Genre: Rock, Sub-Genre: Adult Alternative) 43</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mood Distribution:</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Mood Group: Fun / Cheerful, Mood: Happy) 54</td>
</tr>
<tr>
<td>(Mood Group: Fun / Cheerful, Mood: Upbeat) 68</td>
</tr>
<tr>
<td>(Mood Group: Loving / Sensitive, Mood: Romantic) 58</td>
</tr>
<tr>
<td>(Mood Group: Loving / Sensitive, Mood: Yearning) 75</td>
</tr>
</tbody>
</table>

Energy: 170  
Valence: 244  
Beat: 160
II  “Galaxy Bounce,” The Chemical Brothers:

Genre Distribution:
(Genre: Electronica, Sub-Genre: Beats & Breaks) 232
(Genre: Electronica, Sub-Genre: House) 7
(Genre: Electronica, Sub-Genre: Triphop) 6
(Genre: Pop/Dance, Sub-Genre: Acid-Jazz) 4
(Genre: Pop/Dance, Sub-Genre: Eurodance) 6

Moods:
(Mood Group: Fun / Cheerful, Mood: Groovin) 59
(Mood Group: Fun / Cheerful, Mood: Happy) 66
(Mood Group: Fun / Cheerful, Mood: Upbeat) 68
(Mood Group: Calm / Chill, Mood: Hypnotic) 62

Energy: 244
Valence: 244
Beat: 244

III  “Back To School,” Danny Elfman:

Note: Perceptual attributes correspond to the similar song “The Simpsons.”

Genre Distribution:
(Genre: Easy Listening, Sub-Genre: Classical Crossover) 57
(Genre: Easy Listening, Sub-Genre: Orchestral) 79
(Genre: Easy Listening, Sub-Genre: Standards) 77
(Genre: Jazz, Sub-Genre: Free/Avant Garde) 19
(Genre: Pop/Dance, Sub-Genre: Mainstream) 23

Mood Distribution:
(Mood Group: Fun / Cheerful, Mood: Happy) 85
(Mood Group: Fun / Cheerful, Mood: Humorous) 34
(Mood Group: Fun / Cheerful, Mood: Upbeat) 84
(Mood Group: Fun / Cheerful, Mood: Uplifting) 52

Energy: 244
Valence: 244
Beat: 117
APPENDIX B.
MATLAB IMPLEMENTATION

I Top Level

close all; clear all; clc;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Synchronized Audiovisual Digest Creation using Similarity Analysis
% and MPEG-7 Media Description Tools

% Abstract
% In a home video archival scenario, it is desirable to store only a
% summary of the best quality portions of the media. This equates to
% less required storage capacity and removal of regions which typically
% hinder the overall semantic intent of the document (scenes with
% excessive brightness or random, chaotic movement typical to home
% video). Towards constructing this summary, psychological studies have
% demonstrated that if the video is coupled with a professional-quality
% musical soundtrack to accompany the visual material, there is an
% overall increase in perceptual impact of the media. A novelty score
% according to self-similarity analysis is computed from any chosen
% audio track, which is segmented at maximum novelty points. Pictures
% together with the video material, constrained to the higher quality
% samples, are segmented temporally and aligned to match the audio
% segments, producing an initial audiovisual digest. A further
% enhancement is to semantically synchronize the specific media
% elements and events, based on the premise that artistic composition
% is the montage of passages that construct a tension/resolution
% progression. This is quantified using feature vectors for describing
% temporal and spectral features of the media and implemented as
% recommended by the MPEG-7 Standard. The result is a summary in which
% the different media elements are synchronized semantically, enhancing
% perception and providing a consolidated audiovisual experience.

% Nicolas Betancur
% University of Miami - Music Engineering Thechnology - GMUE
% Thesis Project.

% November 2005 - May 2006

% NOTE: this code is by no means optimized for speed, as it frees
% memory only to load the same media variables again. This is
% done on purpose, to optimize for RAM usage throughout the
% system. It is clearly at the expense of more processing
% complexity, therefore requiring of more total runtime.

% ------------------------------
% ***** Uses adapted (modified) code from the following, exclusively
% for academic purposes:
% mfcc.m - Mel frequency cepstrum coefficient analysis.
% Malcolm Slaney, August 1993
% (c) 1998 Interval Research Corporation
% LMAX.m [lmval, indd]=lmax(xx,filt). Find local maxima in vector XX
% Serge Koptenko, Guigne International Ltd., 06/03/97
% ASSdesc.m - MPEG7 Audio Spectral Spread Descriptor
% Written by Melanie Jackson
% Version 1.0 12 Jan 2001
% Modified 19/04/2002 by Thibaut Sacreste
% Modified 03/05/2003 by Holger Crysandt
% h_mpeg7init.m - creates a structure of the default values to be
% used throughout the descriptors
% Written by Melanie Jackson
% Version 1 15th March 2001
% Modified 30/04/2002 by Thorsten Kastner
% h_mpeg7getspec.m - Specify MPEG7 compliant audio representations
% Written by Melanie Jackson
% Version 1 15th March 2001
% Modified 30/04/2002 by Thorsten Kastner

%--------------------------

%INITIALIZATION
%general
draw = 1; %Flag for output of graphs

%AUDIO
AudioFileName = 'BackToSchool.wav'; %File name for chosen soundtrack
AudioFPS = 5; %Frames per second for similarity
KSize = 32; %desired size for analysis kernel; must be a power of 2
NovTh = 0.2; %Novelty Threshold for filtering
NovNgyTh = 1.5; %Novelty Energy Threshold for further filtering of Novelty Maxima
FeatVect = 1; %Feature vector for audio semantic correlation
Appendix B. MATLAB Implementation

AudDurTh = [3 20];

% Visual
VidFileName = 'nick1234.avi'; % File name of unedited video
VidFPS = 0.5; % Frames per second for decimation
ColorSubspace = 'HSV'; % Subspace for unsuitability processing; H or S or V or HSV
DurTh = [10 50]; % Duration in seconds of a candidate video clip
SuitTh = 0.2; % Maximum level of permitted unsuitability
Nave = 5; % Number of sample images taken from each clip in order to compute semantic features

% --------------------------
% AUTOMATIC MEDIA PROFILING

[GenDist, Energy, Beat, Valence] = MoodLogic(AudioFileName);
[MusType, AudFeatVect, VisFeatVect, ProfData, TR] = profiler(GenDist, Energy, Beat, Valence);

% --------------------------
% MEDIA SEGMENTATION AND INDEXING

% Segment and index audio soundtrack according to similarity and Novelty
[AudNovIndx, AudSampIndx] = novelty(AudioFileName, AudioFPS, KSize, NovTh, NovNgyTh, AudFeatVect, AudDurTh, MusType, ProfData, TR, draw);
save('BSNovelty1_3.mat');

% Segment and index video soundtrack according to unsuitability (HSV)
[VidSuitIndx, VidSampIndx] = unsuitability(VidFileName, VidFPS, ColorSubspace, DurTh, SuitTh, Nave, VisFeatVect, draw);
save('BSSuit1_3.mat');

% Index photo pool
ImIndx = ImageOrg(VisFeatVect);
save('nick1234_BS_1_3_medSeg.mat');

% --------------------------
% MEDIA CLUSTERING

% Cluster audio
AudioSets = cluster(AudNovIndx);

% Cluster video
VideoSets = cluster(VidSuitIndx);
Appendix B. MATLAB Implementation

%Cluster Images
ImageSets = cluster(ImIndx);

save('nick1234_BS_1_3_medClust.mat');
% %---------------------%
% %MEDIA LINKING
% % Reorganize Video Sets
VideoSets = SetOrg(VideoSets,AudioSets);
% % Reorganize Image Sets
ImageSets = SetOrg(ImageSets,AudioSets);
% % Sort sets according to semantic families
SuperSets = families(AudioSets, AudSampIndx, ImageSets, ImIndx,
                      VideoSets, VidSampIndx);
% save('nick1234_BS_1_3_medLink.mat');
%---------------------%

%RHYTHMIC SYNCHRONIZATION
MmaSets = MedAssign(AudioFileName, SuperSets, MusType, draw);
save('nick1234_BS_1_3_final.mat');

### USER PROFILING

function [GenDist Energy Beat Valence] = MoodLogic(AudFileName);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% MoodLogic Profiling
% This module emulates a query to the MoodLogic Database, where
% according to the perceptual attributes for a specific song, a profile
% for the ratios involved in the analysis is defined.
% [GenDist Energy Beat Valence] = MoodLogic(AudFileName);
% AudFileName      -   File name of the analyzed audio soundtrack
% GenDist          -   Output returned Genre Distribution of the
%                     soundtrack
% Energy           -   Output returned Perceptual Energy Attribute
% Beat             -   Output returned Perceptual Beat Attribute
% Valence          -   Output returned Perceptual Valence Attribute
% Nicolas Betancur

Betancur, N.  
May 2006, University of Miami
function [MusType AudFeatVect VisFeatVect ProfData TR] = profiler(GenDist, Energy, Beat, Valence)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Feature vector and TR Curve Profiling
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This module receives information from the MoodLogic Query and
% translates it into profiles for temporal and semantic video editing
% styles. Specifically, if instantiates preliminary tension levels in
% the audio which may be translated to video cut density for each
% musical section, and also instantiates the set of weights which
% emphasize certain semantic descriptors of the media.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% [MusType AudFeatVect VisFeatVect ProfData TR] = profiler(GenDist,
% Energy, Beat, Valence)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% GenDist          -   Genre Distribution of the soundtrack
% Energy           -   Perceptual Energy Attribute
% Beat             -   Output returned Perceptual Beat Attribute
% Valence - Output returned Perceptual Valence Attribute
% MusType - Output on Musical type, if percussive or melodic
% AudFeatVect - Output with the weights of the audio feature vector
% VisFeatVect - Output with the weights of the video feature vector
% ProfData - Output Structures containing the weighted generic TR curves for each genre from the distribution
% TR - Output weights for the audio feature vector that fine-tunes the generic TR curves towards giving it the unique signature of the specific song.

% Anchor Genres:

alternative = [0.8 0.8 0.4 0.4 0.8 0.4 0.3 0.2 0.2 0.6];
classical = [1 0.7 0.5 0.6 0.8 0.6 0.5 0.4 0.2 0.5];
EasyList = [1 0.8 0.7 0.7 0.5 0.4 0.5 0.6 0.4 0.7];
electronic = [0.6 0.4 0.5 0.4 0.3 0.3 0.3 0.1 0 0.2];
jazz = [0.8 0.5 0.5 0.3 0.4 0.5 0.3 0.4 0.6 0.8];
pop = [0.6 0.4 0.2 0.2 0.4 0.2 0.2 0.4];
rock = [0.5 0.5 0.3 0.6 0.5 0.4 0.3 0.2 0.2 0.5];

% Profiling of the preliminary T/R curve
for i = 1:length(GenDist)
    switch GenDist(i).gnr
        case 'Alt'
            ProfData(:,i) = alternative.*(GenDist(i).val/255); % Case 'Alt'
        case 'Cls'
            ProfData(:,i) = classical.*(GenDist(i).val/255); % Case 'Cls'
        case 'Esy'
            ProfData(:,i) = EasyList.*(GenDist(i).val/255); % Case 'Esy'
        case 'Ele'
            ProfData(:,i) = electronic.*(GenDist(i).val/255); % Case 'Ele'
        case 'Jzz'
            ProfData(:,i) = jazz.*(GenDist(i).val/255); % Case 'Jzz'
        case 'Pop'
            ProfData(:,i) = pop.*(GenDist(i).val/255); % Case 'Pop'
        otherwise
            ProfData(:,i) = rock.*(GenDist(i).val/255); % Case 'Rck'
    end;
end;

% Beat Relevance categorization
if Beat > 200
    MusType = 'Rtm';
else
    MusType = 'Mld';
end;
% Feature Vector Weights Profiling (right now only magnitude, to be
% compared according to angle
% distance later!)
Ngy = (Energy+Valence)/2;
for i = 1:length(GenDist)
    switch GenDist(i).gnr
        case 'Alt'
            if Ngy > 180
                AudFeatVect = [0.5 0.1 0.4];      %[AP AFF ASS]
                TR = [0.8 0 0.2];                % for the TR curve
                VisFeatVect = [0.7 0.1 0.2];    %[H S V]
            else
                AudFeatVect = [0.6 0.4 0];
                TR = [0.5 0.2 0.3];                % for the TR curve
                VisFeatVect = [0.7 0.1 0.2];
            end
        case 'Ele'
            if Ngy > 200
                AudFeatVect = [0.65 0 0.35];
                TR = [0.3 0 0.7];                % for the TR curve
                VisFeatVect = [0.5 0.2 0.3];
            else
                AudFeatVect = [0.3 0.6 0.1];
                TR = [0.5 0.3 0.2];                % for the TR curve
                VisFeatVect = [0.5 0.2 0.3];
            end
        case 'Esy'
            if Ngy > 150
                AudFeatVect = [0.3 0.6 0.1];
                TR = [0.6 0.4 0];                % for the TR curve
                VisFeatVect = [0.3 0.1 0.6];
            else
                AudFeatVect = [0.1 0.6 0.3];
                TR = [0.3 0.7 0];                % for the TR curve
                VisFeatVect = [0.3 0.1 0.6];
            end;
        otherwise
            continue;
    end;
end;

III AUDIO ANALYSIS

function [AudNovIndx AudSampIndx] = novelty(FileName, Fps, Ksize, NovTh, NovNgyTh, FeatVect, AudDurTh, MusType, ProfData, TR, draw);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% AUDIO NOVELTY DETECTION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% In this function, an audio soundtrack is segmented according to a
% novelty measure derived from self-similarity analysis. It is then
% analyzed according to a chosen feature (Power, Fundamental Pitch, or
% Tonality), and indexed accordingly.
%
% [AudNovIndx AudSampIndx] = novelty(FileName, Fps, Ksize, NovTh,
NovNgyTh,FeatVect, AudDurTh, MusType, ProfData, TR, draw);
%
% FileName         -   File name for the original soundtrack
% Fps              -   Desired FPS rate.
% Ksize            -   Size of correlation kernel with similarity
% matrix
% NovTh            -   Novelty Threshold for detecting only significant
% peaks in the curve
% NovNgyTh         -   Novelty Energy Threshold for post-filtering of
% the novelty data
% FeatVect         -   Basis Feature Vector that represents the desired
% semantics to be correlated to.
% AudDurTh         -   Specifies minimum and maximum admissible lengths
% for each audio section
% MusType          -   Identifies music as being melodic or percussive
% ProfData         -   Structures containing the weighted generic TR
% curves for each genre from the distribution
% TR               -   Set of weights for the audio feature vector that
% fine-tunes the generic TR curves towards giving
% it the unique signature of the specific song.
% draw             -   Flag for output of figures (1 is yes, 0 is no,
% default is yes)
%
% AudNovIndx       -   Output is an array of audio segments and a
% corresponding Feature Vector Magnitude
% (Semantic Description)level.
% AudSampIndx      -   Output Array of start and end sample numbers for
% each segment of audio.
%

if nargin < 10 draw = 1; end;

%input audio
[audin Fs] = wavread(FileName);

%compute similarity matrix
sim = simtx(audin,Fs,Fps,draw);

%construct kernel
[kern crosskern] = kernel(Ksize,draw);

%compute audio novelty by sliding kernel along similarity matrix
%diagonal and computing correlation at each point of the upper right
%hand proton
Lk = length(kern);
Li = length(sim) - length(kern);
% Initialize Novelty Score
N = zeros(1,Li);
for i = 1:Li
for j = 1:(Lk)
    for k = 1:Lk
        N(i) = N(i) + (kern(j,k)*sim(i+j,i+k));
    end;
end;
end;

% the computed novelty score has a lag due to the indexing of the
% correlation kernel. Correct for this lag
lag = zeros(1,Lk/2);

% Output final audio Novelty score
Novelty = cat(2,lag,N);

if draw ==1
    figure;
    plot(Novelty)
end

% Segment the audio track in between maximum Novelty Points
[SegCell AudSampIndx]= NovPts(audin,Novelty,NovTh,NovNgyTh,Fs,Fps,
AudDurTh);
clear audin;  % raw input samples are not needed further; free memory

% Compute structure of novelty points and their specific Feature Vector
% magnitudes
[AudNovIndx AvAp]= audiodesc(SegCell, Fs, FeatVect, MusType, TR, draw);

if draw == 1
    stem(AvAp);
    hold on;
    plot(AvAp,:')
end

%Profile AABA curve according to Musical Style
AudSampIndx = TRcurve(AudSampIndx, ProfData, AvAp, draw);

% draw TR curve
if draw == 1
    TRCindx = AudSampIndx(1,:).*(Fps/44100);
    figure;
    plot(TRCindx,(1-AudSampIndx(2,:)));
end;

function sim = simtx(audin,Fs,Fps,draw);

% Appendix B. MATLAB Implementation
% AUDIO SELF-SIMILARITY COMPUTATION
% This function computes a self-similarity matrix for an audio
% soundtrack. Refer to [J.Foote - 2004] for theoretical explanation.
% sim = simtx(audin,Fs,Fps,draw);
% audin            - Raw audio input data
% Fs               - Sampling frequency
% Fps              - Desired FPS rate of analysis (1/FPS = HopSize)
% draw             - Flag for output of figures (1 is yes, 0 is no,
% default is yes)
% sim              - Output is a self-similarity matrix of length LxL

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%normalize input audio stream
in = audin(:,1)./max(audin(:,1));

%calculate MFCC's
ceps = mfcc(in,Fs,Fps);

%zero-mean MFCC vectors
for i = 1:length(ceps)
    meani = mean(ceps(:,i));
    ceps(:,i) = ceps(:,i) - meani;
end;

%build similarity matrix
for i = 1:length(ceps)
    j = 1;
    while j <= length(ceps)
        %calculate distance between vectors using cosine angle measure
        num = ceps(:,i)' * ceps(:,j);
        den = norm(ceps(:,i)) * norm(ceps(:,j));
        sim(i,j) = num/den;
        j = j+1;
    end
end

if draw == 1
    figure;
    imagesc(sim);
    colormap(bone);
end;
function [kern crosskern] = kernel(Ksize, draw);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CORRELATION KERNEL CREATION
% Creates kernel according to the corresponding visual features
% observed in a similarity matrix - a checkerboards signifies novelty
% - a cross signifies a major perceptual event. Refer to [J.Foote -
% 2004/05].
% [kern crosskern] = kernel(Ksize, draw);
% Ksize            -   Desired size for kernels; the smaller the size,
% draw             -   Flag for output of figures (1 is yes, 0 is no,
%                       default is yes)
% kern             -   Output 1 is the checkerboard kernel for novelty
%                       detection
% crosskern        -   Output 2 is the cross kernel for specific event
%                       detection
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Basis matrix for the analysis kernels
BaseKernel = [1 -1;-1 1];
CrossKernel = [1 -1 1; -1 1 -1; 1 -1 1];

Matrix to be used in the kronoecker product to expand the size of the
analysis kernel
grower = [1 1;1 1];

% Compute total number of growing steps to conform desired kernel
growsteps = log2(Ksize);

% Compute kernel
if Ksize > 1
    kern = kron(BaseKernel,grower);
    crosskern = kron(CrossKernel,grower);
    for i = 2:growsteps
        kern = kron(kern,grower);
        i = i < growsteps - 2
        crosskern = kron(crosskern,grower);
    end
end

% Compute Gaussian window for preventing edge effects
gx = gausswin(length(kern))';
for i = 1:length(kern)
    kern(i,:) = kern(i,:).*gx;
end;
gxc = gausswin(length(crosskern))';
for i = 1:length(crosskern)
crosskern(i,:) = crosskern(i,:).*gxc;
end;

for i = 1:length(kern)
kern(:,i) = (kern(:,i)'.*gx)';
end;

for i = 1:length(crosskern)
crosskern(:,i) = (crosskern(:,i)'.*gxc)';
end;

function [SegCell SampIndx] = NovPts(audin, Novelty, NovTh, NovNgyTh, Fs, FPS, AudDurTh);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% NOVELTY POINTS DETECTION AND AUDIO SEGMENTATION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% This function detects local maxima in the novelty curve, accommodated
% to include influence of thresholds, and segments audio between these
% novelty points.
% [SegCell SampIndx] = NovPts(audin, Novelty, NovTh, NovNgyTh, Fs, FPS, AudDurTh);

% audin - Input original audio
% Novelty - Complete curve of audio novelty
% NovTh - Threshold of Novelty - a qualified novelty point
% must be above this threshold.
% NovNgyTh - Threshold of Novelty Energy - after cutting
% curve with NovTh, some small peaks that don't
% constitute meaningful points still appear; get
% rid of these.
% Fs - Sampling frequency of audio
% FPS - Frame rate of audio similarity analysis
% AudDurTh - Specifies minimum and maximum admissible lengths
% for each audio section
% SegCell - Output 1 is the segmented audio between the
% novelty points
% SampIndx - Output 2 is the index vector of novelty points
% at the sample level
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Normalize Novelty Curve
NoveltyN = Novelty./max(Novelty);
NovL = length(NoveltyN);
NoveltyN(NovL) = 0;
% Compute Novelty Energy, Frames for start and end of Novelty Peaks
for frame = 1:NovL
    if (NoveltyN(frame) > NovTh) && (NoveltyN(frame) ~= NaN)
        NovIndx(frame) = NoveltyN(frame);
    else
        NovIndx(frame) = 0;
    end
end

engy = 0;
inci = 0;
i = 1;
while i <= NovL
    if NovIndx(i) ~= 0
        engy = engy + NovIndx(i);
        if inci == 0;
            inci = i;
        end;
    else
        if inci ~= 0
            if engy <= NovNgyTh %Regions with low nov. energy measure
                NovIndx(inci:i) = 0;
            end;
            engy = 0;
        end
        inci = 0;
    end
    i = i+1;
end

% Cut First novelty point... this coincides with song's start
NovIndx = NovIndx(2:(NovL-1));

% Find Local maxima in the novelty curve
[maxi FNov] = lmax(NovIndx,10);

FNov = FNov + 1;

% Cut novelty points below the duration threshold
FL = length(FNov);
j = 1;
while j < FL
    if FNov(j+1) - FNov(j) < (AudDurTh(1)*FPS)
        FNov = cat(2,FNov(1:j-1),FNov((j+1):FL));
        FL = FL-1;
    end
    j = j+1;
end;

% resegment regions above duration threshold
FL = length(FNov);
j = 1;
$$OldF_{Nov} = 1;$$

while $$j < FL$$

if $$F_{Nov}(j+1) - F_{Nov}(j) > (AudDurTh(2)\cdot FPS)$$
    $$reg = NoveltyN(F_{Nov}(j):F_{Nov}(j+1));$$
    $$[\text{max2 RegF}_{Nov}] = \text{lmax}(reg,10);$$
    $$[\text{newmax NewF}_{Nov}] = \text{max}(\text{max2});$$
    $$\text{NewF}_{Nov} = RegF_{Nov}(\text{NewF}_{Nov}) + F_{Nov}(j);$$
    $$F_{Nov} = \text{cat}(2,F_{Nov}(1:j),\text{cat}(2,\text{NewF}_{Nov},F_{Nov}((j+1):FL)));$$
    $$FL = FL + 1;$$
    if $$\text{NewF}_{Nov} ~= \text{OldF}_{Nov}$$
        $$j = j-1;$$
        $$\text{OldF}_{Nov} = \text{NewF}_{Nov};$$
    end;
end;
$$j = j+1;$$

%Return from Frame information to sample indexes, segment input raw audio into the chunks dictated by this index
$$\text{SampIndx} = F_{Nov}.*(Fs/FPS);$$
$$\text{SampIndx} = \text{cat}(2,0,\text{SampIndx});$$
$$\text{SampIndx} = \text{cat}(2,\text{SampIndx},\text{length(audin)});$$

for $$i = 1:(\text{length(SampIndx)}-1)$$
    $$\text{SegCell}(i) = \text{audin}((\text{SampIndx}(i)+1):\text{SampIndx}(i+1));$$
end;

function $$[\text{AudNovIndx AvAp}] = \text{audidesc(SegCell, Fs, FeatVect, MusType, TR, draw)}$$

% % % AUDIO SEGMENTS ORGANIZATION ACCORDING TO SEMANTIC FEATURES
% % This function recieves segments of audio, computes their correlation
% % magnitudes with a basis feature vector, and organizes them according
% % to this value.
% % $$[\text{AudNovIndx AvAp}] = \text{audidesc(SegCell, Fs, FeatVect, MusType, TR, draw)}$$
% % % SegCell - Cell containing all the audio segments
% % Fs - Sampling frequency of audio
% % FeatVect - Input Basis Feature Vector for correlation
% % MusType - Identifies music as being melodic or percussive
% % TR - Set of weights for the audio feature vector that fine-tunes the generic TR curves towards giving
% % it the unique signature of the specific song.
% % % AudNovIndx - Output is an array of magnitude values per segment, indexed from the original input
% % AvAp - Output audio feature vector that fine-tunes the
% generic TR curves towards giving it the unique
% signature of the specific song.

Compute Low Level MPEG-7 Descriptors for each segment

AP = zeros(1,length(SegCell));
AFF = zeros(1,length(SegCell));
ASS = zeros(1,length(SegCell));
for i = 1:length(SegCell)
    segment = SegCell{i};
    %Audio Power (AP)
    if FeatVect(1) ~= 0
        AP(i) = sum(abs(segment))/length(segment);
    end;
    %Audio Fundamental Frequency (AFF)
    if FeatVect(2) ~= 0
        AFF(i) = AFFdesc(segment, Fs);
    end;
    %Audio Spectral Spread (ASS)
    if FeatVect(3) ~= 0
        ASS(i) = ASSdesc(segment, Fs);
    end;
end

Profile features according to dominance
for j = 1:i
    ProfFV(j) = AP(j)*FeatVect(1) + AFF(j)*FeatVect(2) + ASS(j)*FeatVect(3);
end;

[AudNovIndx(:,1) AudNovIndx(:,2)] = sort(ProfFV,'descend');

Compute Tension/Resolution fine tuning vector

%Normalize Feature Vector Curves
if AP ~= 0
    APN = (AP./max(AP)).*TR(1);
else
    APN = 0;
end;
if AFF ~= 0
    AFFN = (AFF./max(AFF)).*TR(2);
else
    AFFN = 0;
end;
if ASS ~= 0
    ASSN = (ASS./max(ASS)).*TR(3);
else
    ASSN = 0;
end;

AvAp = APN + AFFN + ASSN;

function ProfIndx = TRcurve(RawIndx, ProfData, AvAp, draw)

% Tension/Resolution Curve Instantiation
% this function instantiates a Tension/Resolution level for audio
% according to the description offered by an audio feature vector
% analysis which takes into consideration Audio Power, Audio
% Fundamental Frequency and Audio Spectral Spread, all compliant with
% the MPEG7 framework.
% ProfIndx = TRcurve(RawIndx, ProfData, AvAp, draw)
% RawIndx          -   Start and end sample numbers for each segment
%                       of audio.
% ProfData         -   Structures containing the weighted generic TR
%                       curves for each genre from the distribution
% AvAp             -   Audio feature vector that fine-tunes the generic
%                       TR curves towards giving it the unique signature
%                       of the specific song.
% draw             -   Flag for output of figures (1 is yes, 0 is no,
%                       default is yes)
% ProfIndx         -   Index of normalized values per audio section,
%                       indicating the specific levels of musical
%                       tension.

% Computation of the preliminary TR curve according to MoodLogic Genre
% Distribution
IndxL = length(RawIndx);
for i = 1:IndxL
    if (i/IndxL) <= 0.1
        for j = 1:size(ProfData,2)
            PreCurve(j,i) = ProfData(1,j);
        end
    elseif (i/IndxL) <= 0.2
        for j = 1:size(ProfData,2)
            PreCurve(j,i) = ProfData(2,j);
        end
    elseif (i/IndxL) <= 0.3
        for j = 1:size(ProfData,2)
            PreCurve(j,i) = ProfData(3,j);
        end
    elseif (i/IndxL) <= 0.4

Betancur, N.  

May 2006, University of Miami
for j = 1:size(ProfData,2)
    PreCurve(j,i) = ProfData(4,j);
end
elseif (i/IndxL) <= 0.5
    for j = 1:size(ProfData,2)
        PreCurve(j,i) = ProfData(5,j);
    end
elseif (i/IndxL) <= 0.6
    for j = 1:size(ProfData,2)
        PreCurve(j,i) = ProfData(6,j);
    end
elseif (i/IndxL) <= 0.7
    for j = 1:size(ProfData,2)
        PreCurve(j,i) = ProfData(7,j);
    end
elseif (i/IndxL) <= 0.8
    for j = 1:size(ProfData,2)
        PreCurve(j,i) = ProfData(8,j);
    end
elseif (i/IndxL) <= 0.9
    for j = 1:size(ProfData,2)
        PreCurve(j,i) = ProfData(9,j);
    end
else
    for j = 1:size(ProfData,2)
        PreCurve(j,i) = ProfData(10,j);
    end
end;
end;
for i = 1:size(PreCurve,2)
    Curve(i) = sum(PreCurve(:,i));
end;

% Fine Tuning of TR curve for each audio section according to
% descriptions from audio feature vector
ProfPow = AvAp - median(AvAp);
NewPow = (sqrt(abs(ProfPow)))/2;
for i = 1:length(NewPow)
    if ProfPow(i) < 0
        NewPow(i) = NewPow(i)*(-1);
    end
end
NewPow = cat(2,NewPow,NewPow(end));
Curve = Curve - NewPow;

% Add the TR curve to the section information
ProfIndx = cat(1,RawIndx,Curve);

III VIDEO ANALYSIS

function [VidSuitIndx VidSampIndx] = unsuitability(FileName, FPS, ColorSubspace, DurTh, SuitTh, Nave, VisFeatVect, draw);
% Video Segmentation according to Unsuitability
%
% In this function, a video track is segmented according to
% unsuitability level. This level is computed according to standard
% rules in video edition, which state that abruptly moving video or
% video which is too dark/bright is bad quality. It is then analyzed
% according to a chosen feature (Hue, Saturation, Value of brightness),
% and indexed accordingly.
%
% [VidSuitIndx VidSampIndx] = unsuitability(FileName, FPS,
ColorSubspace, DurTh, SuitTh, Nave, VisFeatVect, draw);
%
% FileName - File name for the original video clip
% FPS - Desired FPS rate for decimation.
% ColorSubspace - Feature Vector for Video; %Subspace for
% unsuitability processing; H or S or V or HSV
% DurTh - Vector with minimum and maximum admissible times
% for segmented clips
% SuitTh - Maximum level of permitted unsuitability
% VisFeatVect - Weights of the visual feature vector
% Nave - Number of sample images taken from each clip in
% order to compute semantic features
% draw - Enables drawing of figures
%
% VidSuitIndx - Output is an array of video segments and a
% corresponding
% Feature Vector Magnitude (semantic description)
% level.
% VidSampIndx - Output Array of start and end frame numbers for
% each segment of video.
%
% Cut start and end of raw video that is simply DV black screen
ClipPos = [500 25000];

minDurTh = DurTh(1)*FPS;  % Minimum segmented eligible clip duration
maxDurTh = DurTh(2)*FPS;  % Maximum segmented eligible clip duration

DiffReg = 1;  % Span of differential calculation

Nave = Nave+2;  % Necessary to ignore clip boundaries in averages
% calculation

% Read video clip specified by FileName
[movVect FrameHop FrameRate]= MovieRead(FileName, FPS, ColorSubspace,
ClipPos);

% Compute average value per frame of each ColorSubspace Parameter
for i = 1:length(movVect)
    for j = 1:3
        aveV(j,i) = mean(mean(movVect{i}{:, :, j}));
    end
end
clear movVect;       % Liberate memory as soon as possible

save('UnsuitHalfway.mat')

% Construct Differentials Vectors
for i = (DiffReg+1):length(aveV)
    for j = 1:3
        DiffV(j,i) = abs(aveV(j,i) - aveV(j,i-DiffReg));
    end;
end;

% Conform Visual Feature Vector for Unsuitability Segmentation
% H is 0.5, S is 1, V is 0.5
SegVect = (DiffV(1,:).*0.5) + (DiffV(2,:).*1) + (DiffV(3,:).*0.5);

if draw == 1
    figure;
    plot(DiffV(1,:));
    hold all;
    plot(DiffV(2,:));
    plot(DiffV(3,:));
    figure;
    plot(SegVect);
end

% Compute frame indexes for segmented regions
SuitRegIndx = VidSuitPts(SegVect, SuitTh, minDurTh);
    IndxVect{1} = 0;
    for i = 1:size(SuitRegIndx,2) % Rearrange into a vector of frames
        IndxVect{1} = cat(2,IndxVect{1},SuitRegIndx(1,i):SuitRegIndx(2,i));
    end;

SuitFrameSet = IndxVect{1};

% Compute start and end frames for each video section
SegIndx = [0;0];
    j = 2;
    for i = 2:length(SuitFrameSet)
        if (SuitFrameSet(i) - SuitFrameSet(i-1)) > 1
            SegIndx(1,j) = SuitFrameSet(i); % Assign Start Frames
            SegIndx(2,j-1) = SuitFrameSet(i-1); % Assign End Frames
            j = j+1;
        end;
    end;
    SegIndx(2,j-1) = SuitFrameSet(i); % Assign Last End Frame

% Take regions with duration above maxthreshold and split
i = 1;
while i <= length(SegIndx)
    dur = SegIndx(2,i) - SegIndx(1,i);
    if dur >= maxDurTh
newIndx = SegIndx(1,i) + (dur/2);  %create new index atmidpoint
temp1 = SegIndx(:,1:i);
    temp1(2,i) = newIndx;
    temp2 = SegIndx(:,i:length(SegIndx));
    temp2(1,1) = newIndx;
    SegIndx = cat(2,temp1,temp2);
i = i-1;
end;
i = i+1;
end;

% Teake regions below minthreshold and eliminate
i = 1;
while i <= length(SegIndx)
    if size(SegIndx,2) == 1 break; end;
    if (SegIndx(2,i) - SegIndx(1,i)) < minDurTh
        SegIndx(:,i) = [];  % For final regions, make indexes correspond to the original framerate
    end;
i = i+1;
end;

% For final regions, make indexes correspond to the original framerate
% (Teake for lag of 2)TOOK OUT -2 from (SegIndx - 2)
VidSampIndx = round(SegIndx.*FrameHop);

% Sort the video clips according to chosen HSV Colorspace
VidSuitIndx = ClipIndex(FileN, VidSampIndx, VisFeatVect, 7);

function [movVect FrameHop FrameRate] = MovieRead(FileName, FPS, ColorSubspace, ClipPos)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% AVI VIDEO CLIP READER
%% This function reads an AVI video clip, returning a frame vector for
%% the movie specified in the HSV colorspace. It can optionally decimate
%% the clip at a lower FPS rate, or only output a certain parameter of
%% the HSV colorspace.
%% [movVect FrameHop FrameRate] = MovieRead(FileName, FPS, ColorSubspace, ClipPos)
%% FileName    -   File name for the original AVI clip
%% FPS          -   Desired FPS rate.
%% ColorSubspace - Specify parameter (H, S or V) for an individual
%% output, or HSV for the complete colorspace. Be careful if the output is going to be the whole
%% HSV colorspace, read at a lower FPS rate.
%% ClipPos      - Specify only a region of the clip to be read
%% with a
%% [StartFrame EndFrame]
Appendix B. MATLAB Implementation

%% movVect         - Output is a vector of frames in the HSV colorspace or a subset.
%% FrameHop        - Second Output is the number of frames to hop due to decimation.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Initialization

% pull video parameters out of file
VidInfo = aviinfo(FileName);

if nargin<4
    StartFrame = 1;
    EndFrame = VidInfo.NumFrames;
else
    StartFrame = ClipPos(1);
    EndFrame = ClipPos(2);
end

% Conform vector of indices for decimating AVI file at specified FPS
FrameRate = VidInfo.FramesPerSecond;
FrameHop = round((FrameRate)/FPS);
FrIndx = StartFrame:FrameHop:EndFrame;

% Read AVI File
mov = aviread(FileName,FrIndx);

% convert frames to HSV space; retain only the relevant SubSpaces
for i = 1:length(mov)
    movVect{i} = rgb2hsv(mov(1,i).cdata);
    mov(1,i).cdata = [];
    % clear memory as soon as possible;

    switch ColorSubspace
        case 'H'
            movVect{i}(:,:,2:3) = [];
        case 'S'
            movVect{i}(:,:,3) = [];
            movVect{i}(:,:,1) = [];
        case 'V'
            movVect{i}(:,:,1:2) = [];
        case 'HSV'
            % continue
        otherwise
            error('Color Subspace not Specified')
    end;
end;

function SuitRegIndx = VidSuitPts(SuitCurve, SuitTh, DurTh);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% VIDEO SUITABILITY REGION ANALYZER
% This function reads a suitability curve and extracts start and end frames for video with most suitability.

SuitReg = VidSuitPts(SuitCurve, SuitTh, DurTh)

SuitCurve     -   Input curve of suitability
SuitTh        -   Threshold of suitability of which regions above it are ignored
DurTh         -   threshold of duration: Specifies minimum duration of a suitable region
SuitRegIndx   -   Output that Specifies Start and End Frames for each suitable region

.Normalize Suitability Curve
SuitN = SuitCurve./max(SuitCurve);

SuitL = length(SuitN);
SuitN(SuitL) = 0;

%Initialization of Index Vector
SuitRegIndx = [0 ; 0];  %First Row is StartFrames, Second Row is EndFrames

% Perform extraction of suitable regions
i = 1;
frame = 1;
inci = 1;
while frame <= SuitL
    %If the analyzed frame falls inside a suitable region
    while (SuitN(frame) < SuitTh)
        if inci == 1;
            %Assign the frame index to the start frame and block consequent frames below the threshold
            SuitRegIndx(1,i) = frame;
            %Increase dimension of Index Vector for next region
            SuitRegIndx = cat(2,SuitRegIndx,[0 0]');
            inci = 0;
        end;
        frame = frame + 1;
        if frame > SuitL break; end;
    end;

    % If the Region is not suitable anymore
    if inci == 1
        frame = frame + 1;
    else
        %Assign EndFrame and block consequent unsuitable frames
        SuitRegIndx(2,i) = frame;
    end;
end;
Appendix B. MATLAB Implementation

\[ i = i + 1; \]
\[ inci = 1; \]
\[ end \]
\[ end \]

% Take regions with duration below the minimum out of the index
\[ i = 1; \]
\[ while i <= length(SuitRegIndx) \]
\[ if size(SuitRegIndx,2) == 1 break; end; \]
\[ if (SuitRegIndx(2,i) - SuitRegIndx(1,i)) < DurTh \]
\[ SuitRegIndx(:,i) = []; \]
\[ i = i - 1; \]
\[ end; \]
\[ i = i + 1; \]
\[ end; \]

function VidImData = ClipIndex(FileName, SegIndx, HSVvect, Nave);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% VIDEO CLIP INDEXER ACCORDING TO HSV FEATURES

% This function reads a collection of representative image values of a
% video clip, specified in the TrueColor RGB colormap, converts them to
% the HSV colormap. Subsequently, according to the feature vector
% specified, it sorts the images according to feature magnitude.

% VidImData = ClipIndex(FileName, SegIndx, HSVVect, Nave);

% FileName         -   File name of analized video clip
% SegIndx          -   Vector of Input Images to Sort Out.
% HSVvect          -   Feature vector that will be used to calculate
%                       the magnitudes.
% Nave             -   Number of images to compute average for each
%                       clip.

% VidImData(:,1)   -   Output is a vector energy values for the clips,
%                       according to their correlation with the base
%                       feature vector.
% VidImData(:,2)   -   Second Output is the index as related to the
%                       original order of the clips.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Extract index of segments with representative values computed from an
% average of Nave images
movVect = struct('H', {}, 'S', {}, 'V', {});
for i = 1:size(SegIndx,2)
    IndxSpace = linspace(SegIndx(1,i),SegIndx(2,i),Nave);
    ImMov = aviread(FileName, IndxSpace(2:(length(IndxSpace)-1)));
    % convert images to HSV space; retain only the relevant SubSpaces
    for j = 1:length(ImMov)
Appendix B. MATLAB Implementation

Betancur, N. May 2006, University of Miami

mov = rgb2hsv(ImMov(j).cdata);
HmovVect(:,:,j) = 1 - mov(:,:,1);  %Invert H value
SmovVect(:,:,j) = mov(:,:,2);
VmovVect(:,:,j) = mov(:,:,3);
end;
%Median so stray sample doesn't offset the general scene
movVect(i).H = median(HmovVect,3).*HSVvect(1);
movVect(i).S = median(SmovVect,3).*HSVvect(2);
movVect(i).V = median(VmovVect,3).*HSVvect(3);

% AN INTERESTING GRAPH
imx(:,:,1) = movVect(i).H./HSVvect(1);
imx(:,:,2) = movVect(i).S./HSVvect(2);
imx(:,:,3) = movVect(i).V./HSVvect(3);
imx = hsv2rgb(imx);
image(imx)
end;

%Compute Energy as correlated to base Feature Vector for each image
for i = 1:length(movVect)
  %For hue, the median is perceptually better for clustering
  %For sat, the median is perceptually better for clustering
  %Try norm but mean may be interesting too for bright
  imD(i) = mean([median(median(movVect(i).H)),
                 median(median(movVect(i).S)),
                 norm(norm(movVect(i).V))]);
end;

%Sort images in descending order according to previously calculated
%energy
[VidImData(:,1) VidImData(:,2)] = sort(imD,'descend');

IV IMAGE ANALYSIS

function ImIndx = ImageOrg(HSVvect);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% IMAGE INDEXING SYSTEM
% This system reorganizes images according to a basis feature vector
% derived from the HSV colorspace, which provides a structured image
% pool to the music video creation algorithm.
% ImIndx = ImageOrg(HSVVect);
% HSVVect - Feature vector of the HSV colorspace for image
% organizing
% ImIndx - Array of organized pictures with their
% corresponding magnitudes of correlation with
Appendix B. MATLAB Implementation

% the HSV feature vector

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Read Images and convert to HSV colorspace
for i = 1:137
    Image = imread(strcat('pics (',num2str(i-1),').jpg'));
    im = rgb2hsv(Image);
    HimVect = im(:,:,1).*HSVvect(1);
    SimVect = im(:,:,2).*HSVvect(2);
    VimVect = im(:,:,3).*HSVvect(3);

    % For hue, the median is perceptually better for clustering
    % For sat, the median is perceptually better for clustering
    % Try norm but mean may be interesting too for bright
    imD(i) = mean([median(median(HimVect)),
                    median(median(SimVect)),
                    norm(norm(VimVect))]);
end;

% Index Image Space
[ImIndx(:,1) ImIndx(:,2)] = sort(imD,'descend');

V MULTIMEDIA PROCESSING

function sets = cluster(IndxData);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% MEDIA CLUSTERING
% In this function, each media is conformed into clusters that follow
% similarity in its description according to the specific feature
% vector instantiated in previous stages. This helps preprocess the
% media towards semantically clustering different media forms at a
% later stage.
% sets = cluster(IndxData);
% IndxData   - Feature vector sith specific descriptor values
%             for each individual section
% Sets       - Output of sets containing each of the clusters
%             of the individual media type
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Compute histogram of input data
[HistD hVals] = hist(IndxData(:,1));

j = 1;
for i = 1:length(HistD)
    if HistD(i) ~= 0
        indd(j) = i;
        j = j+1;
    end
end

% Assign initial cluster centroids
L = length(indd);
sets = struct('centroids', {}, 'elements', {});
for i = 1:L
    sets(i).centroids = hVals(indd(i));
    if i > 1
        % Find midpoints for separating data
        meds(i-1) = (sets(i).centroids + sets(i-1).centroids)/2;
    end;
end;

% Cluster all data around centers
last = 1;
for i = 1:size(IndxData,1)
    for j = 1:L-1
        if IndxData(i,1) <= meds(j)
            if j > 1 & (IndxData(i,1) <= meds(j-1))
                % continue
            else
                k = length(sets(j).elements);
                sets(j).elements(k+1) = IndxData(i,2);
                last = 0;
            end
        end
    end;
    if last == 1
        k = length(sets(L).elements);
        sets(L).elements(k+1) = IndxData(i,2);
        last = 1;
    end;
end;

function NewSets = SetOrg(Sets,SetsRef);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% MEDIA SETS REORGANIZATION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Video or image sets are reorganized according to the results of audio
% segmentation. The quantity of clusters is made equal.
% NewSets = SetOrg(Sets,SetsRef);
% Sets             -   Input sets to e reorganized (Video - Image Sets)
% SetsRef          -   Reference sets to be reorganized to
% NewSets          -   Output 1 are the reorganized sets.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Desicion tree for sets reordering

while length(Sets) ~= length(SetsRef)
    Ls = [];
    if length(Sets) > length(SetsRef)
        for i = 1:length(Sets)
            %Look at the population in each sets
            Ls(i) = length(Sets(i).elements);
        end;
        [MinLs ILS] = min(Ls);
        if ILS == 1
            %compute centroid median
            Sets(1).centroids = ((Ls(1)*Sets(1).centroids)+(Ls(2)*Sets(2).centroids))/(Ls(1)+Ls(2));
            %unite elements
            Sets(1).elements = cat(2,Sets(1).elements,Sets(2).elements);
            Sets(2) = [];
        else
            Sets(ILS-1).centroids = (Ls(ILS-1)*Sets(ILS-1).centroids+(Ls(ILS)*Sets(ILS).centroids))/(Ls(ILS-1)+Ls(ILS));
            Sets(ILS-1).elements = cat(2,Sets(ILS-1).elements,Sets(ILS).elements);
            Sets(ILS) = [];
        end;
    else
        for i = 1:length(Sets)
            %Look at the population in each sets
            Ls(i) = length(Sets(i).elements);
        end;
        [MaxLs ILS] = max(Ls);
        Newset = struct('centroids',{},'elements',{ });
        Sets(length(Ls)+1).centroids = Sets(ILS).centroids;
        halfele = ceil(Ls(ILS)/2);
        Sets(length(Ls)+1).elements = Sets(ILS).elements(1:halfele);
        lnele = Ls(ILS);
        Sets(ILS).elements = Sets(ILS).elements((halfele+1):lnele);
    end;
end;
NewSets = Sets;

function SuperSets = families(AudioSets, AudSampIndx, ImageSets, ImIndx, VideoSets, VidSampIndx);
Appendix B. MATLAB Implementation

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% MULTIMEDIA CLUSTERS CONSTRUCTION
% This function takes the sets of media and combines them according to
% semantic similarity (from the descriptions produced by their feature
% vector analyses in previous stages). The result is multimedia
% supersets that are ready to be synchronized temporally.
% SuperSets = families(AudioSets, AudSampIndx, ImageSets, ImIndx,
% VideoSets, VidSampIndx);
% AudioSets        -   Clusters of audio media
% AudSampIndx      -   Start and end samples for each audio section
% ImageSets        -   Clusters of photographic media
% ImIndx           -   Indexes for identifying each specific image
% within the pool
% VideoSets        -   Clusters of Video Media
% VidSampIndx      -   Start and End frames for each video section
% SuperSets        -   Clusters of Multimedia sets which will be
% synchronized temporally. Ideally, the
% synchronizing stage will only use media from
% just one SuperSet per each audio section.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Create an empty SuperSets structure to hold the different families.
SuperSets = struct('Audio',{},'Video',{},'Images',{});

%convert indices into segment boundaries
for i = 1:length(AudSampIndx)
    if i < length(AudSampIndx)
        %TR curve goes to third row
        AudSampIndx(3,i) = AudSampIndx(2,i);
        AudSampIndx(2,i) = AudSampIndx(1,i+1);
    else
        AudSampIndx(:,i) = [];
    end
end

%Sort the image elements to coincide semantically with the audio
%Sorting guarantees that the correct magnitudes coincide, since the
%sets contain equal number of samples for each media case
for i = 1:length(AudioSets)
    VidCntVect(i) = VideoSets(i).centroids;
    ImgCntVect(i) = ImageSets(i).centroids;
end
[VidSort VidIndx] = sort(VidCntVect,'ascend');
[ImgSort ImgIndx] = sort(ImgCntVect,'ascend');
%assign clusters to SuperSets Structure
for i = 1:length(AudioSets)
    for j = 1:length(AudioSets(i).elements)
        SuperSets(i).Audio(:,j) = AudSampIndx(:,AudioSets(i).elements(j));
    end
    for j = 1:length(VideoSets(VidIndx(i)).elements)
        SuperSets(i).Video(:,j) = VidSampIndx(:,VideoSets(VidIndx(i)).elements(j));
    end
    for j = 1:length(ImageSets(ImgIndx(i)).elements)
        SuperSets(i).Images(j) = ImageSets(ImgIndx(i)).elements(j);
    end
    SuperSets(i).Images = sort(SuperSets(i).Images,2,'ascend');
end

function PopDif = sizes(SuperSets)

%****************************************************************************
% % Cluster Size Analysis
% % This function analyzes the media populations within each cluster
% % towards defining the way in which the pool of available media will be
% % queried.
% % PopDif = sizes(SuperSets)
% % SuperSets        -   Multimedia clusters
% % PopDif           -   Structure which categorizes each SuperSet as
% %                       having either more audio sets than video sets,
% %                       equal number of audio sets as video sets, or
% %                       more video sets than audio sets
% %****************************************************************************

PopDif = zeros(3,length(SuperSets));       %There is one video element
%per audio element
%
%extract sizes of populations on media sets.
for x = 1:length(SuperSets)
    AudElem(x) = size(SuperSets(x).Audio,2);
    VidElem(x) = size(SuperSets(x).Video,2);
    ImgElem(x) = size(SuperSets(x).Images,2);
    if AudElem(x) == VidElem(x)
        PopDif(1,x) = 1;
    else if AudElem(x) > VidElem(x)
        PopDif(2,x) = 1;
    else
        % Handle the case when AudElem(x) < VidElem(x)
    end
end
Appendix B. MATLAB Implementation

Betancur, N. May 2006, University of Miami

PopDif(3,x) = 1;
end
end
end

function MmaSets = MedAssign(AudioFileName, SuperSets, MusType, draw)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% % CROSS MEDIA RHYTHMIC SYNCHRONIZATION
% % This function synchronizes visual with audio material within each
% % multimedia cluster. It does so by including a ratio of video to
% % images according to the level of musical tension in each specific
% % section, and since it works within only one cluster, semantic
% % consistency of the visual material is guaranteed.
% % MmaSets = MedAssign(AudioFileName, SuperSets, MusType, draw)
% % AudioFileName    -   File name for the original soundtrack
% % Supersets        -   Multimedia Clusters
% % MusType          -   Classifies music as melodic or rhythmic
% % draw             -   Flag for output of figures (1 is yes, 0 is no,
% %                       default is yes)
% % MmaSets          -   Ouptut is one synchronized multimedia set per
% %                       each audio section. Start and end spamles for
% %                       the audio are provided, as well as start and
% %                       end frames for the corresponding video segment,
% %                       and a relation of audio sample vs. image index
% %                       for the subsequent image progression. The
% %                       progression of all the MmaSets completely
% %                       instantiates the audiovisual summary.
% % Nicolas Betancur
% % University of Miami - Music Engineering Thechnology - GMUE
% % March 2006
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

warn = 1;

%MmaSets = struct('AudBds', {}, 'VidBds', {}, 'PicPos', {}, 'PicInd', {});

%Create Empty structure that will show audio, video and image indexes
%and times
MmaSets = struct('AudBds', {}, 'VidBds', {}, 'PicPos', {}, 'PicInd', {});

% Calculate proportion of audio sets vs. video sets within each multimedia cluster
PopDif = sizes(SuperSets);

% Perform rhythmic linking and reorganization
k = 1;
LastVidRes = [];
for i = 1:length(SuperSets)
    [maxi DifIndx] = max(PopDif(:,i));
    switch DifIndx
    % Audio sets and video sets are equal in quantity
    case 1
        j = 1;
        while j <= size(SuperSets(i).Audio,2)
            % Fetch one audio sample, one video sample, and the available pictures for the set
            SingleSet = struct('Audio',{SuperSets(i).Audio(:,j)},'Video',{SuperSets(i).Video(:,j)},'Images',{SuperSets(i).Images});
            % Perform Synchronization to specific audio events
            [MmaSets(k).AudBds MmaSets(k).VidBds MmaSets(k).PicPos MmaSets(k).PicInd ResImg NLV NeedPics] = rhythm(AudioFileName,SingleSet,MusType,draw);
            % If the system is starved for pictures
            if NeedPics ~= 0
                for n = 1:length(SuperSets)
                    ImQty(n) = length(SuperSets(n).Images);
                end;
                % Fetch pictures from the super set with the most availability
                [maxim maximindx] = max(ImQty);
                SuperSets(maximindx).Images = SuperSets(maximindx).Images((maxim-NeedPics):maxim);
            else
                SuperSets(i).Images = ResImg;
            end;
        end;
    end;
end
if i == length(SuperSets) break; end;
% Assign residual images to available image pool
% Take residue to next cluster in case it gets starved
SuperSets(i+1).Images = cat(2,SuperSets(i+1).Images,ResImg);

% There are more video than audio sets; therefore, construct video residuals
case 3
    loops = size(SuperSets(i).Audio,2);
    j = 1;
    % Loop until the total quantity of audio sections is analyzed
    while j <= loops
        % Fetch one audio sample, one video sample, and the available pictures for the set
        ...
SingleSet =
struct('Audio',{SuperSets(i).Audio(:,j)},'Video',{SuperSets(i).Video(:,j)});
% Perform Synchronization to specific audio events
[MmaSets(k).AudBds MmaSets(k).VidBds MmaSets(k).PicPos] =
rhythm(AudioFileName,SingleSet,MusType,draw);
% If no suitable video has been found,
if NLV == 1
    LastVidRes =
cat(2,LastVidRes,SuperSets(i).Video(:,j));
    VidQty = size(SuperSets(i).Video,2);
    for jj = j:(VidQty - 1)
        SuperSets(i).Video(:,jj) =
            SuperSets(i).Video(:,(jj+1));
        VidQty = VidQty -1;
        if VidQty < loops
            sprintf('warning: starved video out of
inadequate clips, have %d, need %d, press any key to continue', VidQty, loops)
            pause;
            % end
        end;
    end;
else
    %If the system is starved for pictures
    if NeedPics ~= 0
        for n = 1:length(SuperSets)
            ImQty(n) = length(SuperSets(n).Images);
        end;
        [maxim maximindx] = max(ImQty);
        SuperSets(i).Images =
            SuperSets(maximindx).Images((maxim-NeedPics):maxim);
        SuperSets(maximindx).Images =
            SuperSets(maximindx).Images(1:(maxim-NeedPics));
    else
        SuperSets(i).Images = ResImg;
        k = k+1
        j = j+1;
    end;
end;

if i == length(SuperSets) break; end;
%Allocate residual images and video to next set
%Take residue to next cluster in case it gets starved
SuperSets(i+1).Images =
cat(2,SuperSets(i+1).Images,ResImg);
ResVid =
SuperSets(i).Video(:,((size(SuperSets(i).Audio,2)+1):size(SuperSets(i).Video,2)));
SuperSets(i+1).Video =
cat(2,SuperSets(i+1).Video,cat(2,ResVid,LastVidRes));
LastVidRes = [];

otherwise %there are more audio than video sets; therefore,
%fill up with residuals from other clusters
LenVid = size(SuperSets(i).Video,2);
while LenVid < size(SuperSets(i).Audio,2);
    for w = 1:size(PopDif,2)
        if PopDif(3,w) == 1
            LenVidN = size(SuperSets(w).Video,2);
            SuperSets(w).Video = SuperSets(w).Video(:,1:LenVidN-1);
            break;
        end;
    end;
    LenVid = size(SuperSets(i).Video,2);
end

%Finally Audio Sets and Video Sets Match
j = 1;
while j <= size(SuperSets(i).Audio,2)
    % Fetch one audio sample, one video sample, and the % available pictures for the set
    SingleSet = struct('Audio',{SuperSets(i).Audio(:,j)},'Video',{SuperSets(i).Video(:,j)},'Images',{SuperSets(i).Images});
    [MmaSets(k).AudBds MmaSets(k).VidBds MmaSets(k).PicPos MmaSets(k).PicInd ResImg NLV NeedPics] = rhythm(AudioFileName,SingleSet,MusType,draw);
    %If the system is starved for pictures
    if NeedPics ~= 0
        for n = 1:length(SuperSets)
            ImQty(n) = length(SuperSets(n).Images);
        end;
        [maxim maximindx] = max(ImQty);
        SuperSets(i).Images = SuperSets(maximindx).Images((maxim-NeedPics+1):maxim);
        SuperSets(maximindx).Images = SuperSets(maximindx).Images(1:(maxim-NeedPics));
        else
            SuperSets(i).Images = ResImg;
            k = k+1
            j = j+1;
        end
    end
    if i == length(SuperSets) break; end;
    %allocate residual images to next cluster
    SuperSets(i+1).Images = cat(2,SuperSets(i+1).Images,ResImg); %Take residue to next cluster in case it gets starved
end
PopDif = sizes(SuperSets); %Recalculate SuperSets Populations
function [AudSt VidInd PicPos PicInd ResImg NLV NeedPics] = rhythm(FileName,MmaSupSet,MusType,draw);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% MEDIA SYNCHRONIZATION ACCORDING TO RHYTHMIC ANALYSIS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
NLV = 0;
NeedPics = 0;
warn = 0;
AudSampBds = [MmaSupSet.Audio(1,1) MmaSupSet.Audio(2,1)];

for i = 1:length(AudSampBds)
    if AudSampBds(i) == 0
        AudSampBds(i) = 1; %Prevent start of song from provoking an
        %indexing error
    end;
end;

%input audio
[audin Fs] = wavread(FileName, AudSampBds);

%reach of rhythmic calculation
reach = 50;

FPS = 15;
dist = 1;

%compute similarity matrix
sim = simtx(audin,Fs,FPS,0);
for i = 1:length(sim)
    if sim(i) == NaN
        sim(i) = 0;
    end;
end;

%perform energy correlation calculation for rhythm matrix
Benergy = zeros(1,length(sim)-dist);
for i = 1:(length(sim)-dist-1)
    for j = 1:(reach-1)
        % if MusType == 'Rtm'
        %     Benergy(i) = Benergy(i) + sim(j,i+j);
        % else
        %     Benergy(i) = Benergy(i) + sim(i,j);
        %end
    end
end

if draw == 1
    figure;
    plot(Benergy)
end

%find local maxima for the beat spectrum
InvBer = Benergy(1:length(Benergy)-1).*-1; 
[lmval,indd]=lmax(InvBer,8);
indd = indd + FPS;  %Correct for lags

% Audio has been analyzed. Time to load visual material
VidPts = MmaSupSet.Video;
Images = MmaSupSet.Images;

AudioLen = length(audin)/Fs;        %length of Audio in seconds

%Flag to check if a point for video sync was found above the TR point
found = 0;
for i = 1:(length(indd)-1)
    %point of the video to change for pics
    if indd(i) >= AudioLen*(MmaSupSet.Audio(3,1)*FPS)
        VidLen = round(((indd(i)/FPS)*30)/2);
        found = 1;
        break;
    elseif MmaSupSet.Audio(3,1) == 0
        VidLen = 0;
        break;
    else
        found = 2;
    end
end
% Total duration of clip is video
if found == 2
    VidLen = round((AudSampBds(2) - AudSampBds(1))*FPS/Fs);
end;
if VidLen*2 >= VidPts(2) - VidPts(1);
    NLV = 1;
    if warn == 1
        sprintf('Warning: Call for longer length than current video
section!!! (has %f, needs %f, images will fall out of sync), press any
key to continue', (VidPts(2) - VidPts(1)), VidLen*2)
        pause;
    end;
end;

%center video duration around the video clip midpoint
VidCnt = round(mean(VidPts));
VidInd = [(VidCnt-VidLen) (VidCnt+VidLen)]';

%no video included, just pictures on all the rhythmic indexes
if found == 0
    i = 1;
end;
for j = i:length(indd)    %available points for inserting pictures
    PicPos(j-i+1) = (indd(j)*Fs/FPS) + AudSampBds(1,1);
end
PicPos = cat(2,PicPos(2:(length(PicPos)-1)),AudSampBds(1,2))';
%PicPos = PicPos-(0.2*Fs);  %correct for approx 30fps calculations
AudSt = AudSampBds(1,1);
i = 2;
LPP = length(PicPos);
%Specify a maximum duration threshold for picture duration
while i < LPP
    if PicPos(i+1) - PicPos(i) <= 5000;
        PicPos = cat(1,PicPos(1:(i-1)),PicPos((i+1):LPP));
        LPP = length(PicPos);
    end
    i = i+1;
end
PicPos = PicPos';
if found ==2
    PicPos = [];
end;
if length(PicPos) > length(Images)
    NeedPics = length(PicPos);
    if warn == 1
        sprintf('Warning: Starved for Images in rhythm module!!! hit any
key to continue')
        pause;
    end;
end;
if NeedPics == 0
    PicInd = Images(1:length(PicPos));
    if length(PicInd) == 0     % The clip was too short and all is video
        PicInd = [];
    end;
    ResImg = Images((length(PicInd)+1):length(Images));
    AudSt = AudSt';
else
    PicInd = [];
    ResImg = [];
    AudSt = [];
end;