

**Segmentation, Structure Detection
and
Summarization of Multimedia Sequences**

Hari Sundaram

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

COLUMBIA UNIVERSITY

2002

© 2002

Hari Sundaram

All rights reserved

ABSTRACT

Segmentation, Structure Detection and Summarization of Multimedia Sequences

Hari Sundaram

This thesis investigates the problem of efficiently summarizing audio-visual sequences. The problem is important since consumers now have access to vast amounts of multimedia content, that can be viewed over a range of devices.

The goal of this thesis is to be able to provide an adaptive framework for automatically generating a short multimedia clip as a summary, when given longer multimedia segment as input to the system. In our framework, the solution to the summarization problem is predicated on the solution to three important sub-problems — *segmentation*, *structure detection* and *audio-visual condensation* of the data.

In the segmentation problem, we focus on the determination of *computable scenes*. These are segments of audio-visual data that are consistent with respect to certain low-level properties and which preserve the syntax of the original video. This work does not address the problem of semantics of the segments, since this is not a well posed problem. There are three novel ideas in our approach: (a) analysis of the effects of rules of production on the data (b) a finite, causal memory model for segmenting audio and video and (c) the use of top-down structural grouping rules that enable us to be

consistent with human perception. These scenes form the input to our condensation algorithm.

In the problem of detecting structure, we propose a novel framework that analyzes the *topology* of the sequence. In our work, we will limit our scope to discrete, temporal structures that have a priori known deterministic generative mechanisms. We show two general approaches to solving the problem, and we shall present robust algorithms for detecting two specific visual structures — the dialog and the regular anchor.

We propose a novel entity-utility framework for the problem of condensing audio-visual segments. The idea is that the multimedia sequence can be thought of as comprising entities, a subset of which will satisfy the users information needs. We associate a utility to these entities, and formulate the problem of preserving the entities required by the user as a convex utility maximization problem with constraints. The framework allows for adaptability to changing device and other resource conditions. Other original contributions include — (a) the idea that comprehension of a shot is related to its visual complexity (b) the idea that the preservation of visual syntax is necessary for the generation of coherent multimedia summaries (c) auditory analysis that uses discourse structure and (d) novel multimedia synchronization requirements.

We conducted user studies using the multimedia summary clips generated by the system. These user studies indicate that the summaries are perceived as coherent at condensation rates as high as 90%. The study also revealed that the measurable improvements over competing algorithms were statistically significant.

Contents

1	INTRODUCTION	1
1.1	SEGMENTATION, STRUCTURE DETECTION AND SUMMARIZATION.....	1
1.2	WHY IS THIS PROBLEM INTERESTING?.....	3
1.2.1	<i>The use of films as our dataset</i>	5
1.3	PROBLEMS ADDRESSED	6
1.3.1	<i>Audio-visual segmentation</i>	6
1.3.2	<i>Structure detection</i>	9
1.3.2.1	An intuition	9
1.3.2.2	Defining deterministic structure	11
1.3.3	<i>Audio visual summarization</i>	13
1.3.3.1	Conceptual difficulties in prior work.....	14
1.3.3.2	A Chinese puzzle	15
1.3.3.3	Semantics of the syntax vs. the semantics of the data.....	18
1.3.3.4	Tasks, entities and utilities	20
1.3.3.5	An adaptive summarization framework	20
1.3.3.6	The scope of this work on summarization.....	21
1.4	SUMMARY OF CONTRIBUTIONS.....	22
1.4.1	<i>Audio-visual scene segmentation</i>	22
1.4.2	<i>Structure detection</i>	23

1.4.3	<i>Audio-visual condensation</i>	24
1.5	ORGANIZATION OF THE THESIS	25
2	THE SEGMENTATION PROBLEM	28
2.1	INTRODUCTION	28
2.2	THINKING ABOUT SHOTS AND SCENES.....	31
2.3	RELATED WORK: VISUAL ANALYSIS.....	35
2.3.1	<i>Scene transition graphs</i>	35
2.3.2	<i>Adaptive clustering</i>	38
2.3.3	<i>Video coherence</i>	39
2.3.4	<i>Discussion</i>	41
2.4	RELATED WORK: AUDITORY ANALYSIS	42
2.4.1	<i>Segment classification</i>	42
2.4.2	<i>Discussion</i>	44
2.4.3	<i>Computational Auditory Scene Analysis</i>	45
2.5	THE COMPUTABLE SCENE FRAMEWORK	49
2.5.1	<i>Why the emphasis on computability?</i>	49
2.5.2	<i>The idea of memory</i>	50
2.5.2.1	A model with infinite persistence.....	51
2.5.2.2	A FIFO memory model	52
2.5.3	<i>Insights from film-making</i>	53
2.5.4	<i>The computable scene</i>	55

2.5.4.1	Validation of C-scene definitions.....	56
2.5.4.2	Detecting the video and audio scenes	58
2.5.4.3	Detecting structure and silence	59
2.5.4.4	Multi-modal fusion.....	60
2.6	SUMMARY.....	61
3	THE COMPUTABLE VIDEO SCENE	63
3.1	INTRODUCTION.....	63
3.2	THE V-SCENE DEFINITION.....	64
3.3	VISUAL RECALL: MEASURING INTER-SHOT SIMILARITY.....	65
3.4	COHERENCE: MEASURING INTER-SEGMENT RELATIONSHIPS.....	67
3.4.1	<i>What are shot-lets?</i>	69
3.5	DETECTING V-SCENES.....	71
3.5.1	<i>Comparisons with shot detection</i>	73
3.6	EXPERIMENTS.....	74
3.6.1	<i>Labeling the data</i>	75
3.6.2	<i>Results</i>	76
3.6.3	<i>Comparison with related work</i>	78
3.7	NEW CONCEPTUAL MEMORY MODELS	78
3.7.1	<i>Improving the basic model</i>	79
3.7.1.1	Experiments using the new model.....	82
3.7.2	<i>Complexity based models</i>	85

3.7.2.1	Coherence formulated using Kolmogorov complexity.....	86
3.7.2.2	Coherence formulation with perceptual distortion	88
3.8	SUMMARY.....	90
4	THE COMPUTABLE AUDIO SCENE	92
4.1	INTRODUCTION.....	92
4.1.1	<i>Summary of our approach</i>	<i>92</i>
4.2	THE COMPUTABLE AUDIO SCENE DEFINITION.....	94
4.2.1	<i>Relating computable scenes to semantic scenes.....</i>	<i>95</i>
4.3	MODELING THE DATA.....	96
4.3.1	<i>The features</i>	<i>97</i>
4.3.1.1	Scalar sequences.....	97
4.3.1.2	Vector sequences	99
4.3.1.3	Scalar points	99
4.3.2	<i>The scalar sequence signal model.....</i>	<i>100</i>
4.3.3	<i>Estimating components of the scalar sequence model.....</i>	<i>102</i>
4.3.4	<i>Trends</i>	<i>103</i>
4.3.5	<i>Periodic components</i>	<i>104</i>
4.3.5.1	Matching pursuits	105
4.3.5.2	Subspace pursuits	106
4.3.6	<i>Noise.....</i>	<i>108</i>
4.4	COMPUTING THE FEATURE DISSIMILARITIES.....	108

4.4.1	<i>The scalar case</i>	109
4.4.1.1	Trends.....	109
4.4.1.2	Periodic components	110
4.4.1.3	Noise	111
4.4.2	<i>Vectors</i>	113
4.4.3	<i>Point data</i>	115
4.5	THE SEGMENTATION ALGORITHM	116
4.5.1	<i>Determining correlations from memory</i>	116
4.5.2	<i>Estimating the distance increase rate</i>	118
4.5.3	<i>β sequence analysis</i>	119
4.5.4	<i>Combining the β sequences</i>	120
4.6	SILENCE DETECTION.....	123
4.7	EXPERIMENTS.....	124
4.7.1	<i>Labeling the data</i>	124
4.7.2	<i>Results</i>	125
4.7.3	<i>Comparison with related work</i>	128
4.8	SUMMARY.....	128
5	MULTI-MODAL FUSION.....	130
5.1	INTRODUCTION.....	130
5.2	THE THREE RULES FOR DETECTION	131
5.3	A PROCEDURE FOR DETECTING C-SCENES.....	132

5.3.1	<i>Additional post-processing</i>	135
5.4	EXPERIMENTAL RESULTS	136
5.4.1	<i>The ground truth</i>	136
5.4.1.1	Is it any easier to label c-scenes?	138
5.4.2	<i>Segmentation results</i>	139
5.4.2.1	Shot detection errors.....	141
5.4.2.2	A-scene location uncertainty.....	142
5.4.2.3	The c-scene algorithm has low precision	142
5.4.3	<i>Comparison with related work</i>	144
5.4.4	<i>C-scene detector breakdowns</i>	145
5.4.4.1	Sudden change of scale	146
5.4.4.2	Widely differing backgrounds.....	146
5.4.4.3	Change in the axis of action	147
5.5	SUMMARY.....	148
5.5.1	<i>Improvements</i>	149
6	DETECTING STRUCTURE.....	150
6.1	INTRODUCTION.....	150
6.2	WHAT IS STRUCTURE?.....	152
6.3	THE TOPOLOGY OF VIDEO SHOTS.....	153
6.4	THE TOPOLOGICAL GRAPH.....	154
6.5	TOPOLOGICAL STRUCTURES.....	156

6.5.1	<i>Dialogs</i>	156
6.5.2	<i>Regular anchors</i>	157
6.5.3	<i>A note on the end conditions</i>	159
6.6	DETECTING DIALOGS.....	160
6.6.1	<i>The dialog presence condition</i>	161
6.6.2	<i>The sliding window algorithm</i>	162
6.7	DETECTING REGULAR ANCHORS.....	163
6.7.1	<i>An intuition</i>	164
6.7.2	<i>The regular anchor detection algorithm</i>	165
6.7.2.1	Indistinguishable permutations.....	165
6.7.2.2	Detecting the three element regular anchor.....	166
6.7.2.3	Detecting arbitrary sized regular anchors.....	168
6.8	EXPERIMENTS.....	171
6.8.1	<i>Dialog detection</i>	171
6.8.2	<i>Regular anchor detection</i>	172
6.9	RELATED WORK.....	174
6.10	SUMMARY.....	175
7	THE SUMMARIZATION PROBLEM	177
7.1	INTRODUCTION.....	177
7.1.1	<i>Scope of the problem examined in this thesis</i>	179
7.2	RELATED WORK: VIDEO.....	180

7.2.1	<i>Image based storyboards</i>	181
7.2.1.1	Video Manga	182
7.2.1.2	Scene transition graphs	184
7.2.2	<i>Slide shows</i>	185
7.2.3	<i>Video skims</i>	185
7.2.3.1	Prior work.....	186
7.3	RELATED WORK: AUDIO.....	187
7.3.1	<i>Speechskimmer</i>	188
7.3.2	<i>Discourse structure segmentation</i>	189
7.4	AN ENTITY-UTILITY FRAMEWORK.....	190
7.4.1	<i>What is an entity?</i>	192
7.4.1.1	Causes of predicates	194
7.4.1.2	Types of predicates	195
7.4.2	<i>Skims and entities</i>	196
7.4.2.1	Factors that affect skims.....	196
7.4.2.2	Entities, tasks and the device	197
7.4.2.3	Taxonomy: skims come in different flavors	198
7.4.2.4	Entities and utilities.....	200
7.4.3	<i>Skim generation goals</i>	201
7.5	SUMMARY.....	202
8	SKIMS: VISUAL ANALYSIS.....	204

8.1	INTRODUCTION.....	204
8.2	VISUAL COMPLEXITY	205
8.2.1	<i>Insights from film-making and experimental psychology</i>	205
8.2.1.1	The shot and its apparent viewing time	206
8.2.1.2	Concept complexity and learnability.....	207
8.2.2	<i>Defining visual complexity</i>	209
8.2.2.1	Bounds on Kolmogorov complexity	209
8.2.2.2	Estimating visual complexity	211
8.2.3	<i>Complexity and comprehension</i>	212
8.2.4	<i>Time-complexity bounds</i>	214
8.2.4.1	Notes on the time complexity relationship.....	216
8.3	SYNTAX ANALYSIS	217
8.3.1	<i>What are the elements of syntax?</i>	217
8.3.2	<i>Understanding the role of syntax</i>	218
8.3.3	<i>Rules for syntax based removal</i>	219
8.3.4	<i>Dealing with shot detector uncertainty</i>	222
8.4	DISCUSSION.....	225
8.4.1	<i>The comprehension time experiment</i>	225
8.4.2	<i>Video segment selection</i>	226
8.5	SUMMARY.....	228
9	SKIMS: AUDITORY ANALYSIS.....	230

9.1	INTRODUCTION.....	230
9.2	WHY IS SUMMARIZING AUDIO A HARD PROBLEM?	231
9.2.1	<i>Our approach to summarization</i>	232
9.3	AUDIO SEGMENT CLASSIFICATION	234
9.3.1	<i>Features used</i>	234
9.3.2	<i>The approach to segmentation</i>	235
9.3.2.1	<i>The duration dependent Viterbi decoder</i>	236
9.4	DETECTING SIGNIFICANT PHRASES	239
9.5	RESULTS.....	241
9.5.1	<i>Segment classification</i>	241
9.5.2	<i>Significant phrase detection</i>	243
9.6	SUMMARY.....	245
10	UTILITY MAXIMIZATION	247
10.1	INTRODUCTION.....	247
10.1.1	<i>Notation</i>	249
10.2	WHAT ENTITIES WILL WE PRESERVE?	249
10.3	UTILITY FUNCTIONS	251
10.3.1	<i>Video</i>	252
10.3.1.1	<i>What happens when we drop a shot?</i>	255
10.3.1.2	<i>The sequence utility function</i>	258
10.3.2	<i>Audio</i>	258

10.3.2.1	The audio segment utility loss function	260
10.3.3	<i>A note on dropped segments</i>	262
10.4	PENALTY FUNCTIONS	263
10.4.1	<i>Film rhythm</i>	263
10.4.2	<i>Audio slack</i>	264
10.5	CONSTRAINTS	266
10.5.1	<i>Tied multimedia constraints</i>	266
10.5.2	<i>Duration bounds</i>	268
10.6	MINIMIZATION VIA CONSTRAINT RELAXATION	269
10.6.1	<i>Algorithm bias</i>	269
10.6.2	<i>Constructing tied multimedia segments</i>	270
10.6.3	<i>Solution overview</i>	271
10.6.4	<i>Ensuring a feasible video solution</i>	273
10.6.5	<i>Ensuring a feasible audio solution</i>	275
10.6.6	<i>The mathematical formulation</i>	276
10.7	EXPERIMENTS.....	277
10.7.1	<i>Introduction</i>	278
10.7.2	<i>Three algorithms compared</i>	278
10.7.3	<i>The user study</i>	280
10.7.4	<i>Results</i>	281
10.7.5	<i>Discussion</i>	282
10.7.5.1	What about other skim forms?.....	284

10.8	SUMMARY.....	284
11	CONCLUSIONS AND FUTURE WORK.....	286
11.1	INTRODUCTION.....	286
11.2	RESEARCH SUMMARY.....	286
11.2.1	<i>Segmentation</i>	286
11.2.2	<i>Structure detection</i>	289
11.2.3	<i>Summarization</i>	290
11.3	IMPROVEMENTS TO THE FRAMEWORK.....	292
11.3.1	<i>Complex memory models</i>	292
11.3.2	<i>Film-making constraints</i>	293
11.3.3	<i>Joint audio-visual experiments</i>	294
11.3.4	<i>Fine-grained estimates of complexity</i>	295
11.4	FUTURE DIRECTIONS.....	295
11.4.1	<i>Summaries: anytime, anywhere</i>	296
11.4.2	<i>Extensions to current work on summarization</i>	297
11.4.3	<i>Working with raw data</i>	297
11.4.4	<i>Structure discovery</i>	298
11.4.5	<i>Multimedia information visualization</i>	299
11.4.6	<i>Environment attentive algorithms</i>	300
12	REFERENCES.....	301

13	APPENDIX	315
13.1	KOLMOGOROV COMPLEXITY	315
13.1.1	<i>Estimating $K(x y)$</i>	316
13.1.2	<i>Estimating $K(x A)$</i>	317
13.2	THE PHASOR DISTANCE.....	318
13.3	ENHANCING IMAGE STORYBOARDS	320
13.4	SKIMS: THE FIRST USER STUDY.....	321
13.4.1	<i>Experimental set-up</i>	322
13.4.2	<i>Results</i>	325
13.4.3	<i>Discussion</i>	325
13.4.4	<i>Conclusions</i>	327
13.5	SKIMS: THE SECOND USER STUDY.....	327
13.5.1	<i>Experimental set-up</i>	328
13.5.2	<i>Results</i>	329
13.5.3	<i>Discussion</i>	330
13.5.4	<i>Conclusions</i>	330

List of Figures

Figure 1.1: How do the methods of production affect the data that we are trying to analyze?.....	8
Figure 1.2: The familiar dialog sequence, is an example of visual structure.....	12
Figure 1.3: The figure shows the original Chinese document to summarized at the top and the resulting summary comprising topic sentences, at the bottom. The topic sentences have been highlighted in red. Note that the summary only contains the topic sentences.	17
Figure 2.1: A shot detector based on color and motion, segments the video into piece-wise constant approximations with respect to the two features.....	32
Figure 2.2: The computable scene overview.....	34
Figure 2.3: A scene transition graphs is segmented into scenes by detecting cut-edges. These edges divide the graphs into disconnected sub-graphs.....	36
Figure 2.4: The attention span T_{as} is the most recent data in the memory. The memory (T_m) is the size of the entire buffer.	52
Figure 2.5: The director will place cameras on the same side of an imaginary line, called the line of interest. This ensures consistent left-right orientation across shots.	53
Figure 3.1: (a) Each solid colored block represents a single shot. (b) each shot is broken up into “shot-lets” each at most δ sec. long. (c) the bracketed shots are	

present in the memory and the attention span. Note that sometimes, only fractions of shots are present in the memory.69

Figure 3.2: Each figure shows a plot of coherence against time. The three figures show the (a) normal (b) strong and (c) weak coherence minima cases. Figure (a) shows the two coincident windows W_o (outer window) and W_1 (inner window) that are used for coherence minima detection.71

Figure 3.3: Let us assume that the attention span contains a single shot, while the rest of the memory contains six different shots.....79

Figure 3.4: The figures shows the precision vs. memory size (a) and recall vs. memory size for the two models. The green curve represents the coherence values from the new model, while the red curve represents the coherence values from the old model. The memory size is varied from 8sec. to 48 sec. in increments of 4 sec.....84

Figure 3.5: Memory with two buffers: A and B. B is just the attention span, while A constitutes the rest of the memory.....87

Figure 3.6: The variance of the blur function increases with distance Δt from the boundary between the two buffers A and B.88

Figure 4.1: We extract three types of features from each section in memory: vector sequences, scalar sequences and single point features.96

Figure 4.2: We model the scalar sequence as a sum of three components — trend, periodic components and noise.100

Figure 4.3: The overall computational architecture for analyzing an input scalar sequence $x[n]$. Trends are represented using low order polynomials, periodic elements are represented using a conjugate symmetric basis. The noise is not explicitly modeled.	102
Figure 4.4: The features are extracted from two different sections of memory, a and b and then compared.	108
Figure 4.5: The two trends from sections a and b are extrapolated and then compared.	109
Figure 4.6: The solid blocks in the two sections represent phasors. Each phasor is an ordered triple: $\{a, \omega, \theta\}$ representing the amplitude, frequency and the initial phase of the periodic component.....	111
Figure 4.7: We compute a circular distance between two equal length vector sequences. The distance is computed by (a) arranging both sequences in a circle (b) rotating the outward sequence by one element each time and computing the average pair-wise distance between the corresponding elements. (c) the minimum of these rotated distances is the circular distance.	114
Figure 4.8: We compute each feature within the attention span. Then, we shift back the analysis window by δ sec. and re-compute all the features. We proceed this way till all the memory has been covered. Then we compute the distances between the feature values in the attention span with the rest of the segments.	116

Figure 5.1: Remove a-scene and v-scene boundaries and detected silence, when detected in structured sequences.	133
Figure 5.2: Tight synchronization is needed between weak v-scenes and non-weak a-scenes.	133
Figure 5.3: Do not associate normal v-scenes with a-scenes marked as silent.	134
Figure 5.4: Non-weak video coherence minimum intersecting silences (gray boxes) causes a c-scene boundary.	135
Figure 5.5: The first and the last frames show the irregular anchor. This shot will appear sporadically over the whole scene.	135
Figure 5.6: There is a sudden change of scale that is not accounted by the model.	145
Figure 5.7: a right angled camera pan, between two very different backgrounds.	146
Figure 5.8: A circular tracking shot will establish a new axis.	147
Figure 6.1: In the familiar dialog sequence, the sequence has interesting topological properties — (a) adjacent shots differ (b) alternate shots are alike and (c) this is independent of the duration of each shot.	154
Figure 6.2: The idealized topological graph for a dialog subsequence, that is induced by associating a metric d on the on the entire video sequence \mathbf{I}	155
Figure 6.3: A three image regular anchor.	157
Figure 6.4: A three element anchor surrounded by two end nodes.	159
Figure 6.5: A dialogue scene and its corresponding periodic analysis transform. Note the distinct peaks at $n = 2, 4 . . .$	162

Figure 6.6: The top sequence shows a three element regular anchor surrounded by the boundary elements. The lower sequence is the result of randomly permuting the first sequence. The intuition behind this permutation is that it will destroy the original topological structure, if such a structure was indeed present.164

Figure 6.7: This is a three element regular anchor, shown with the two boundary images. Permuting the first and the last image in the sequence does not alter the regular anchor pattern.165

Figure 6.8: A five element regular anchor. The anchor images are black, the internal non-anchor images are light gray while the end images are dark gray. The figure shows three five element templates: initial (i), extension (c), and end (e).....169

Figure 6.9: The figure shows an example dialog, where misses in the key-frame detection prevented the dialog from being detected. Here, misses occur after the 2nd and the 4th frames.171

Figure 7.1: An image storyboard181

Figure 7.2: video and audio entities. The dotted line indicates a relation amongst the low-level entities.195

Figure 7.3: a skim depends on the user’s information needs and the device.....197

Figure 8.1: figure (a) shows a close up while figure (b) shows a long shot. It seems to take a little longer to comprehend the action in figure (b). Why does this happen?.....206

Figure 8.2: A typical image used in the experiment.....	213
Figure 8.3: (a) avg. comprehension time vs. normalized complexity (b) histogram plot of avg. comprehension time across different complexity bins (c) one slice from plot (b) suggests a Rayleigh distribution. Plots (a) and (b) show time against visual complexity, while in plot (c), we plot a slice of the density for a specific value of complexity.	214
Figure 8.4: Avg. comprehension time (sec.) vs. normalized complexity (x-axis) showing comprehension (upper/lower) bounds. The dots in the figure represent the Rayleigh (95 th percentile) bounds.....	215
Figure 8.5: Three syntax reduction mechanisms. The black boxes are the minimal phrases and will not be dropped, while the gray shots can be dropped.	221
Figure 8.6: we drop the frames in a shot from the end.....	227
Figure 9.1: The audio data is processed in parallel, by the audio classifier and the significant phrase detection algorithm.	233
Figure 9.2: The tree structure for the audio segment classifier. Each gray dot represents a two-class SVM classifier.....	235
Figure 9.3: The duration utility plotted as a function of the segment duration.....	237
Figure 9.4: When classifying a candidate segment, we extract features from different parts of the candidate segment, as well as extract features from the pre-pausal segment. The duration of the preceding and the succeeding pauses are also used for classification.	239

Figure 9.5: The system for detecting segment beginnings (SBEG), which mark new topic phrases.	240
Figure 10.1: The shot utility function plotted as a function of the complexity (x-axis) and time (y-axis). The z-axis shows the function value.	255
Figure 10.2: A graph of the modulation function λ plotted against the proportional duration.	256
Figure 10.3: The audio utility function for the different classes, plotted as a function of segment duration. The plot has the y-axis scaled.	259
Figure 10.4: The plot shows the audio utility loss function as a function of the segment duration.	261
Figure 10.5: The figure shows two audio segments overlapping by ξ	264
Figure 10.6: the gray box indicates a speech segment and the dotted lines show the corresponding tied video segment.	266
Figure 10.7: Both figures show an artificial shot boundary, in a single shot, induced by a tie constraint. The gray boxes refer to the portions of the shot fragments to be dropped, while the arrows indicate the direction of the condensation. In figure (a) if we condense the two shots from the ends, we will create a new, visible shot boundary that didn't exist before. If we adopt the condensation scheme shown in figure (b), no new shot boundaries would be created. ...	267
Figure 10.8: searching for a feasible solution by successively relaxing the synchronization constraints.	271
Figure 10.9: The application for generating audio-visual skims.	278

Figure 10.10: The original video and the four skims: optimal, semi-optimal and proportional. segment types —red: video, green: audio; black: discourse segment. . Note that the semi-optimal and the proportional skims have the same proportional video data, and the optimal and the semi-optimal skims have the same optimal audio data.....279

Figure 10.11: The difference between the raw optimal score and the minimum of the other two scores. The differences are significant at 80% and 90% compressions rates.....281

Figure 10.12: The *blade runner* optimal skim showing the important elements captured by our algorithm. The gray box shows the location of the sig. phrase.283

Figure 13.1: Instantaneous distance in the complex plane, between two phasors $\{a, \omega_1, \theta_1(n)\}$ and $\{b, \omega_2, \theta_2(n)\}$ at time n.319

List of Tables

Table 2.1: Variation of precision and recall, for constant T (2000), with variations in δ .	37
Table 2.2: The four types of c-scenes that exist between consecutive, synchronized audio-visual changes. solid circles: indicate audio scene boundaries, triangles indicate video scene boundaries.....	56
Table 2.3: c-scene breakup from the film sense and sensibility.	57
Table 3.1: Ground truth v-scene data obtained from labeling the video data separately from the audio data.....	75
Table 3.2: Video scene detection results. In order: hits, misses, false alarms, correct rejection, number of shots, number of non-scene change locations, precision and recall.	77
Table 4.1: Ground truth a-scene data obtained from labeling the video data separately from the audio data.....	125
Table 4.2: Audio scene detection results. In order: hits, misses, false alarms, precision and recall.	126
Table 5.1: The table shows a list of icons used in the figures in this section and their semantics.	132
Table 5.2: Ground truth data obtained from labeling the audio and the video data separately. The c-scenes are broken up into constituent units: P-pure, Ac-V	

(audio changes, visuals consistent), A-Vc: audio consistent, visuals change, and MM: mixed mode.	137
Table 5.3: C-scene detector results. In order: hits, misses, false alarms, correct rejection, number of shots, number of non-scene change locations, precision and recall.	141
Table 6.1: The table shows the dialogue detector results for the three films. The columns are: Hits, Misses, False Alarms, Recall and Precision.	171
Table 6.2: The results for the regular anchor detection. The symbols are as follows: H: hits, M: misses, FA: false alarms, P: Precision and R: recall. The primed variables refer to the modified hits, and false alarms, after we take the results of the dialog detector into account.	172
Table 8.1: Three types of syntax reductions that depend on the element (dialog/progressive) and the number of shots k.	220
Table 8.2: Shot detector variations. Each row show the in order: the motion and color parameter thresholds, recall, precision, prob. of false alarm $P(F_a)$, $P(F_a $ $N_d)$: prob. of false alarm give a non-dialog scene. The 97.5% confidence upper bound for $P(F_a N_d)$	224
Table 9.1: the confusion matrix result of the speech (S) vs. Non-speech classifier ($\neg S$). The rows indicate the classification result for each ground truth class.	242
Table 9.2: the confusion matrix result of the speech (S) vs. noisy-speech classifier (N_s). The rows indicate the classification result for each ground truth class.	242
Table 10.1: The variation in λ and β with the class type.	259

Table 10.2: The table for determining the coupling factor η . The columns indicate the class of the preceding segment, while the rows indicate the class of the current segment.....265

Table 10.3: User test scores. The columns: algorithms (optimal, semi-optimal, proportional), the film, condensation rate, and the five questions. The table shows values of the optimal and the difference of the scores of the (semi / pr) skims from the optimal. Bold numbers indicate statistically significant differences.282

Table 13.1: Skims lengths in seconds for each clip and skim type. The numbers in brackets represent the percentage reduction. The films in order: *Blade Runner*, *Bombay*, *Farewell my Concubine*, *Four Weddings and a Funeral*.....324

Table 13.2: Test scores from five users. C: coherence, S_o: skip original? The last two rows represent best/worst preferences.325

Table 13.3: Test scores from five users. The columns: the algorithm used, the film, condensation rate, and the five questions.....329

Table 13.4: The result of the double-blind side-by-side comparison of the two skim algorithms. The new algorithm is preferred over the old one for all condensation rates. This improvement is statistically significant.330

Acknowledgements

I would like to thank those who have contributed to my life as well as to my education at Columbia.

First and foremost, I would like to thank my advisor, Shih-Fu Chang, who has been a source of great support and excellent guidance throughout my thesis. Shih-Fu gave me the rare freedom to pursue research on pretty much any topic that interested me, and I am extremely grateful. His sharp insight and his emphasis on demonstrability of the concept, has been a major influence on my research. His constant encouragement to look beyond improving the state of the art, has made working on the thesis fun, and a richly rewarding experience. Thank you, Shih-Fu.

I would also like to thank other members of my dissertation committee — Shree Nayar, Nevenka Dimitrova, Xiaodong Wang and Dan Ruebenstein, for their insightful comments on my research and for accommodating a tight defense schedule. I would also like to thank other faculty at Columbia — Alexandros Eleftheriadis, Dan Ellis and Yehoshua Zeevi for their support.

Many thanks to Shree Nayar, for his initial support at Columbia. My interactions with him, have played an important role in shaping how I conduct my research. Much of my methodology for analyzing problems, as well as the manner in which I communicate my ideas, stem from the time I spent working under him.

Nevenka Dimitrova's display of genuine excitement with my work, whenever I would present it to her, has been a source of great support and encouragement. Her careful reading of this thesis, has greatly improved this document.

I would like to thank Alexandros Eleftheriadis for being a wonderful mentor. His infectious joy at conducting research and his balanced perspective on academia has greatly influenced many of my own decisions on life. Thank you Alex, for being there.

Thanks to Lynn Wilcox, for giving me the opportunity to work at Fuji-Xerox Palo Alto Laboratories, during the summer of 1999. I would also like to thank other members at her team at FX: Jonathan Foote, Patrick Chiu, John Boreczky, Shingo Uchihashi and Andreas Girgensohn, all of whom made my stay fun and greatly rewarding.

Working with Shih-Fu had led me to interact with members of research laboratories around Columbia as well as abroad. I would like to thank John Smith (IBM) and Ajay Divakaran (MERL), for their encouragement and support of my work, throughout my interaction with them, Akiomi Kunisa (Sanyo) and Qibin Sun (LIT), for their constant support of my research and their encouragement for me to join academia.

I would like thank members of my own research group for making my stay at Columbia fun and memorable — Hualu Wang, Di Zhong, Ching-Yung Lin, Lexing Xie, Ana Benitez, Jianhao Meng, William Chen, Raj Kumar Rajendran, Shahram Ebadollahi, Alejandro Jaimes, Dongqing Zhang, Yong Wang and Winston Hsu. I would also like to

thank fellow Columbia PhD students: Kazi Zaman, Hari Kalva, Min-Yen Kan and Michael Theobald.

Many thanks to Lexing Xie for being a terrific collaborator and a wonderful friend. Collaboration with her has helped me extend my initial work on visual skim generation. Our joint work on auditory analysis for audio-visual summarization appears in Chapter 9. Her decision to ask me about a research problem whenever she found “my cpu idle,” kept me on my toes and probably ensured an earlier graduation!

Many friends have made my stay in New York memorable. I would like to thank Geeta Singh, Sucheta Sharma, Aparna Pappu, Sumeer Bhola, Smita Paul and Nitin Govil for sharing the joys that accrue from living in New York. I would also like to thank Ana Benitez and Masako Mori for their friendship and advice.

Montek Singh deserves special mention — he has been a close friend and confidant for well over a decade and I continue to seek his advice on all matters personal and professional.

My last invocation, to my family, is also necessarily the most important one — for without their unwavering support and encouragement, none of this research would have been possible.

To my teachers

1 Introduction

1.1 Segmentation, structure detection and summarization

We encounter the world that we live in, as a multi-modal sensory experience. As you read these lines with your eyes, you are also paying attention to the environmental sounds — the quiet of your home, the sounds of the street, at the same time, you can sense the weight of this thesis on your palms. If we were to collect your sensory inputs, sampled say every 100 ms., for say a couple of hours, we would be left with a tremendous amount of raw data. However, if someone were to ask you what you were doing at this moment, you would probably say “I am reading Hari Sundaram’s PhD thesis.” You would have effectively *summarized* your sensory experience over the past few hours into a single statement.

To take those raw streams of data from multiple sensors and produce a single sentence in English that communicates the essence of the multi-modal experience, must necessarily be one of the goals of multimedia summarization. This is however an

extremely difficult task, predicated as it is on many of the open issues in computer vision, linguistics and auditory analysis.

The goals of this thesis are then necessarily much more modest and limited in scope — we shall create algorithms for automatically analyzing *produced* multi-modal data, such as films, and the result of our work shall be a time-condensed multimedia clip, that shall provides a summary of the original data. Our solution to the problem is predicated on the solution of the following three sub-problems:

- **Audio-visual segmentation:** In this problem, we are interested in breaking up the original audio-visual stream into manageable chunks of data, where each chunk shares certain consistent properties.
- **Structure detection:** This problem tackles the issue of detecting a priori known structures in the data. We detect those structures specific to the film domain, that are important elements of the film syntax.
- **Audio-visual condensation:** The goal of this problem is to take a segment that was the result of the first algorithm, and *time-condense* the segment while taking into account the utility of the different audio visual segments detected, and the elements of syntax that were detected by our structure detection algorithm. The summary is generated using a constrained utility optimization.

We shall present novel approaches of each of the sub-problems outlined above. The result of this work is for example, to be able take a 50 min. multimedia clip and be able

to reduce it *automatically* to up 90% i.e. to a five minute video. These condensed videos are referred to as *skims*.

It is an important aspect of this work, that in our attempt to summarize the data, we make no attempt to understand the *semantics of the data*. Instead, we focus on summarizing the original data stream by understanding the *semantics of the syntax* of the domain and preserving this syntax in the resulting summary. The summary multimedia clips resulting from our algorithm are deemed coherent and intelligible by viewers, in user studies.

The rest of this chapter is organized as follows. In the next section, we shall discuss the case for automatically summarizing multi-modal data. In section 1.3, we shall present the motivation and the solution outline to the three problems mentioned in this section. In section 1.3.3.6 we shall summarize the contributions of this thesis and in section 1.5, we present the outline of this thesis.

1.2 Why is this problem interesting?

The entertainment industry has always attempted to replicate the multi-modal sensory experience of the everyday, into its products. Over the past few decades, with device costs falling due to more efficient means of production, as well as more processing power available, this has resulted in several noticeable trends:

- The *form* of entertainment has become much more sophisticated and incorporates more modes of information — we now routinely have sophisticated graphic overlays that accompany the audio and the video.
- There is much *more information* available to the average TV user than before. The advent of digital cable has meant that the user now has immediate access to hundreds of channels.
- The *content producer* has now come home¹ — the availability of cheap camcorders has now meant that ordinary consumers are now routinely producing many hours home video.
- The *devices* that enable access to multimedia content have increased; from initial access coming from movie theatres and television, people now access content from their PC / laptop and more recently on their cell phones (NTT DoComos 3G wireless network in Japan and the more recent Sprint PCS's 3G network in the United States.).

Hence, given the tremendous amount of information that the user has access to, it would be extremely useful to be able to summarize the video data in such a manner that not

¹ Note that this is form of content is not professionally edited or structured; however, as will be clear by the end of the thesis, we shall be able to take the ideas generated in this thesis and find applications to non-produced data as well.

only efficiently summarizes the information that the user needs, but is also device and network adaptable.

This thesis focuses on skim generation, as it is most closely approximates the original multimedia experience. While the thesis does not look at device and network adaptability, we solve the skim generation problem using a constrained utility maximization framework. This framework is general and is easily modified to incorporate the device and network characteristics.

1.2.1 The use of films as our dataset

This work focuses on using produced data, specifically films as a dataset. Films may seem like unlikely candidates for automatic summarization, as they are highly expensive to produce and it seems likely that the film makers would spend some more money to create movie previews.

A summary and a movie preview are not the same thing. The movie preview has a specific goal — to entertain and to entice the viewer into spending money into buying the product. The intent of what is shown and the emotional responses that it generates may be very different from the actual film itself. The goal of a summary is to preserve those multimedia segments from the original video data that satisfy the user needs. For example, a example of a valid summary is one which contains only those scenes of a particular actor making a cameo appearance in the film. Films make useful datasets for analysis for other reasons as well:

- They are produced content — there is an underlying grammar to the film form that influences the meaning.
- The syntactical elements of the film have specific meaning attached to them (e.g. dialogs)

The rich nature of the film content allows us to use the result of this work in other domains that have domain specific grammar and have specific structural elements (e.g. baseball videos etc.) present.

1.3 Problems addressed

In the following sections, we shall present the intuition behind our solutions to the problems of segmentation, structure detection and audio-visual summary generation by utility-based condensation of multimedia segments.

1.3.1 Audio-visual segmentation

The problem of segmenting audio-visual data into smaller manageable chunks is a basic problem in multimedia analysis, and its solution helps in problems such as video summarization. Traditional work on video segmentation [35] [43] [44] [49] [95] [100] [101] [103] has focused on partitioning the data into semantically consistent *scenes*. But, what exactly is a scene? *The Oxford English Dictionary* has many senses in which the word “scene” appears, and we reproduce below two of them that closely match what we are looking for.

The place where an action is carried on and people play their parts as in a drama; the scene of action, the place where events are actually happening or business being done.

A view or picture presented to the eye (or to the mind) of a place, concourse, incident, series of actions or events, assemblage of objects, etc.

Bordwell and Thompson, in their book *Film Art* [8] , define a scene as follows:

A segment in a narrative film that takes place in one time and space or that uses crosscutting to show two or more simultaneous actions.

Examples of such scenes include — a scene showing people on the beach, a marketplace, a segment showing a newscaster, a football game etc. Note that scenes are higher level abstractions than shots.

The problem of understanding the semantics of scene, without context is extremely difficult. The semantics in the data exist at multiple levels, and depend upon amongst others, the world knowledge of the viewer, the task of the viewer, the intentions of the creator. This is actually the familiar difficult computer vision problem of scene understanding, that can only be satisfactorily solved in a restricted domain.

The focus of this research is the detection of *computable* scenes. They are formed by looking at the relationships between elementary computable audio and video scenes and structure. The elementary audio and video scenes represent contiguous chunks of audio and video respectively. These scenes are termed *computable*, since they can be

automatically computed using low-level features in the data. But, if we are not interested in semantics of a scene, why bother with computability?

The focus on computability is really the idea of task based segmentation — the goal of the segmentation algorithm being to assist in the solution of another problem. In this work, the task of the segmentation algorithm is to generate segments that preserve the syntactical structure of the original video data. These syntactically correct segments form the input to our video summarization algorithm.

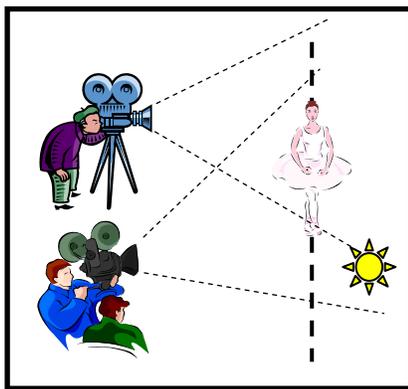


Figure 1.1: How do the methods of production affect the data that we are trying to analyze?

There are three key issues investigated in this current work:

- **Memory:** can we think of the problem of segmentation as the problem of trying to remember data?
- **Production:** how does the method of production constrain the audio-visual data that we are trying to analyze? (ref. **Figure 1.1**)

- **Structure:** when we watch video segments that contain structure, we seem to group that structure into a single entity (e.g. dialogs). Can we systematically discover other interesting temporal structures?

Central to the audio-visual segmentation framework is a model of a causal, finite human memory. Constraints on the model stem from *film making production rules* and from the psychology of audition. The final segmentation algorithm, which additionally depends on automatic structure discovery and silence detection, detects four types of computable scenes.

1.3.2 Structure detection

The solution to the problem of detecting structure in the multi-media data, plays an important role in the other two sub-problems outlined in the beginning of this chapter — segmentation, as well as audio-visual condensation. But before we discuss the problem in detail, we need to answer a basic question — when does a sequence said to contain structure? To answer this question, we must resort to an intuition.

1.3.2.1 *An intuition*

Let us assume that we have sequence generator that outputs two symbols — A and B with equal probability. Now, let us examine the following two sequences generated by the system:

A B A B A B A B A B

B B A B B A B A A A B A

Both have 12 symbols with an equal number of each symbol. Both sequences have the same probability of occurrence, and will *both* require 12 bits to encode using entropy coding. Why then, does the first sequence appear instinctively to be structured, while the second sequence appears to be more random? The answer lies in looking at Kolmogorov complexity [21] [51] .

A.N. Kolmogorov was the great Russian mathematician who established the foundations of the axiomatic approach to modern probability [51] . Later on in life, Kolmogorov was troubled by the idea of an axiomatic approach to the idea of randomness. For example, there seemed to be many instances of random sequences that appear just once in nature, that do not fit well with the idea of having been produced by a stochastic source. Conversely, there seemed examples of “random” sequences that are actually deterministic. For example, it is well known that the digits in the sequence of the expansion of π pass all known tests of randomness. However, a very simple program can generate the sequence.

Kolmogorov defined the idea of randomness using the model of the universal Turing machine. The Kolmogorov complexity² of a sequence is the length of the smallest program when input into a universal machine that outputs the string [21] [51] , (also see

² Also referred to as Algorithmic Information Complexity

appendix 13.1, for more details). Then, a sequence is defined to be random if the following condition holds:

$$\lim_{n \rightarrow \infty} \frac{K(x)}{n} = 1, \quad (1.1)$$

where $K(x)$ is the Kolmogorov complexity of the symbol sequence, n is the length of the symbol sequence. In other words, the length of the shortest program to print a random string is the print instruction followed by the string itself. Clearly, random strings are incompressible.

1.3.2.2 *Defining deterministic structure*

Structure (or structured-ness) in a deterministic sequence of symbols is then defined as the converse of randomness, as follows:

$$\frac{K(x)}{n} < 1, \quad (1.2)$$

for all sequences of arbitrary length. In other words, a structured sequence is compressible.

How does this definition help us in disambiguating between the two sequences presented at the beginning of this section? Clearly, the Kolmogorov complexity of the first sequence is much smaller than the second one, since there exists a simple program to generate the first string: PRINT “AB” 6 times.

Defining structure using Kolmogorov complexity is problematic since it is well known that the Kolmogorov complexity of an arbitrary number is non-computable, because this is the same as solving the halting problem [21] [51] . However, in the absence of any formal notions of structure, the definition using Kolmogorov complexity serves as a useful starting point to think about structured-ness of a sequence. Rather than focusing



Figure 1.2: The familiar dialog sequence, is an example of visual structure.

on computability, we use this definition of structure to guide us to think along the following two dimensions:

- Generative mechanisms for structure: The idea that sequence could be generated using a simple set of rules, or more generally as the output of a program.
- Structure detection as a compression problem.

Deterministic structures present in video sequences, or in discrete real sequences are characterized by their topological properties i.e. the metric relationship between the elements of the structure. In this thesis, we shall focus on detecting visual structure (ref. **Figure 1.2**) in discrete sequences that have associated metric space, via robust statistical tests.

We shall make a distinction between *macro-level* structures (structures that exist as a consequence of the relationships between scenes) and *micro-level* structures (structures

such as the dialog, that exist *within* a scene). In this work, we shall only look at detecting micro-level structure in the visual domain.

We shall limit the scope of the problem to the set of deterministic structures that have been generated via a priori known generative mechanisms.

1.3.3 Audio visual summarization

The *Oxford English Dictionary* defines the adjective form of the word summary³ as:

Of a statement or account: Containing or comprising the chief points or the sum and substance of a matter; compendious (now usually with implication of brevity).

The definition implies an understanding the semantics of the document (or video), and constructing a summary document (or video) based on this knowledge. Such a level of understanding for automatic analysis of the audio-visual data, requires access to a body of knowledge that is external to the data in the video. Since such access is difficult, summarization schemes that rely on the *semantics of the data* work well only on highly constrained data. In this work, we shall focus on using the *semantics of the syntax* and we shall revisit this dichotomy later on in this section.

³ i.e. the word is to be used as in a “summary video” or a “summary text,” which in colloquial English is used as “video summary” and as “text summary” respectively.

We summarize the original video by creating a short multimedia clip that attempts to satisfy the user's information needs. We determine the entities that need to be present that satisfy the user's information needs, and preserve those entities by using an optimization framework. In the next few sections, we shall present the intuition behind our approach to summarization, a section on the idea of using the semantics of the syntax. This is then followed by sections that discuss the idea of entities and the utility maximization. Finally, we shall discuss the scope of our work on summarization.

1.3.3.1 Conceptual difficulties in prior work

In our work on summarizing video data, we do not focus on determining the semantics of the data. Before we explain the intuition behind this approach, we briefly review the conceptual problems associated with prior work.

In prior work on generating audio visual skims [18] [52] , the focus was on using a small set of pattern recognition systems (e.g. faces, mountains, the sound of gunfire etc.) for *semantically* meaningful skim generation. There, segments that contained these privileged set of patterns⁴, were tagged as important, and were made part of the resulting skim.

There are two conceptual difficulties that arise with such an approach:

- The data in the videos (e.g. films, news etc.) is often unconstrained — hence by using a small set of pattern recognition algorithms we will find it difficult to

⁴The Informedia project [18] used transcript information as well.

satisfy many user queries. The summaries will always contain a subset of the *same* set of patterns that can be detected by the pattern detection algorithms, thus introducing a bias.

- The produced data contains a *grammar* that provides context to the shots present in the data. By detecting only a fixed set of patterns, and ignoring the grammatical context in which they occur, this will make the resulting skim incoherent.

The realization that one *cannot* possibly summarize unconstrained videos by only using a fixed set of pattern recognition sub-systems that attempt to understand the underlying semantics of the data, guides us to think in terms of the semantics of the syntax.

1.3.3.2 *A Chinese puzzle*

Is it possible to summarize a document without understanding it? This may seem like a strange idea, but let us consider the problem of summarizing a document written in a language that one does not understand. For example let us take case of a document written in Chinese, a language that I do not understand. I only have an orthographic relationship to the document i.e. I see the document as a sequence of visual patterns.

On closer scrutiny, we observe that the document is structured — into headings, paragraphs and sentences that are distinguishable using a stop symbol. Now, if we make the additional assumption that this is a well written technical document, then the

following principle applies — each paragraph begins with a topic sentence i.e. a sentence that summarizes the rest of the paragraph.

Now, if we make these assumptions about the *semantics of the syntax* i.e. about the meanings of the different elements of the document (i.e. headings, topic sentences etc.), and their relative importance, then it is possible to summarize the document as follows.

霍金在中國

史蒂芬·霍金為參加國際數學家大會來到了中國，一股“霍金熱”由之興起。從8月9日，“輪椅上的科學大師”霍金來到杭州參加弦理論國際會議以來，《時間簡史》和《果殼中的宇宙》在書店暢銷；印有霍金頭像的文化衫風靡了大街小巷；霍金的公眾演講票極為搶手，電視台為此專門搞了實況轉播……有人慨嘆：繼陳景潤之後，科學家已經很久沒有獲得這麼空前的轟動效應了。

霍金把自己受到人們“追星”般仰慕的原因概括成兩點。一、人類對自身來源的好奇；二、公眾總是尋找英雄。的確，霍金不僅探索了宇宙的極限，也挑戰了人類的意志力極限。由於患有盧伽雷氏症，霍金被禁錮於輪椅達20多年，全身惟一能動的是左手的3根手指和部分面部肌肉。近年，他又在一次手術後失去了發聲能力，與人交流需依靠計算機合成語音，但就是這樣，他還在理論物理領域取得了卓越的成就，被公認為是繼愛因斯坦之後最杰出的理論物理學家。

霍金對公眾發表了題為“膜的新奇世界”的講演，這正是霍金的科普新作《果殼中的宇宙》最後一個章節的名稱。正是在這個“新奇世界”中，凝聚了他近年來最具突破性、最受矚目的研究成果。因為愛因斯坦的廣義相對論在解釋宇宙開端、黑洞這些問題時遇到難題，顯示出理論的不完善，30多年來科學界一直在尋找一種更加宏大的理論框架，能夠將廣義相對論和量子力學完美統一起來；而霍金近年研究的“M理論”，極有可能就是這樣一種理論體系。霍金盡可能地以通俗的方式向大家解釋：我們也許就生活在多維空間中的一張“膜”上。我們在日常生活中看到周圍的空間是三維的，再加上一個維度標明時間，這樣，時空就是四維；而在霍金的理論中，時空是十維或十一維。膜世界模型是當今理論物理學研究的熱門課題，霍金自信地說：“我們將發現一個膜的新奇世界！”

史蒂芬·霍金為參加國際數學家大會來到了中國，一股“霍金熱”由之興起。霍金把自己受到人們“追星”般仰慕的原因概括成兩點。一、人類對自身來源的好奇；二、公眾總是尋找英雄。霍金對公眾發表了題為“膜的新奇世界”的講演，這正是霍金的科普新作《果殼中的宇宙》最後一個章節的名稱。

- Start at the lowest section level. Collect all topic sentences within a section, and

Figure 1.3: The figure shows the original Chinese document to summarized at the top and the resulting summary comprising topic sentences, at the bottom. The topic sentences have been highlighted in red. Note that the summary only contains the topic sentences.

concatenate them into a paragraph. This paragraph is meaningful as each sentence is a well formed topic sentence, and the paragraph will summarize the section as each topic sentence summarizes each paragraph in the section. Now attach the section heading to this summary paragraph. Repeat this procedure for all sections at this level.

- Now that we have summarized all the sections at the lowest level, repeatedly group the sections using the document structure.
 1. Attach the original document heading to this new document.

We have thus summarized the original document without understanding the meanings of any of the symbols. Crucially, we have determined the document structure and the meanings of the different elements of syntax of the document in order to summarize the document. See **Figure 1.3** for an example of this approach. This idea of using the syntax for summarization leads us to a class of skims, known as *syntax-preserving* skims. We shall make use of this principle of syntax preservation in our work. However, it should be noted that they are *not* a panacea to the general problem of summarization and work well only in specific contexts.

1.3.3.3 Semantics of the syntax vs. the semantics of the data

The approach used in this work — that the preservation of semantics can be achieved via the preservation of syntax, is contrary to traditional AI approaches to the problem. This is because traditional approaches focus on understanding the *semantics of the data*,

with bottom-up analysis with object detectors, often in conjunction with top-down rules. This work focuses on analyzing the *semantics of the syntax*. The difference is important and changes the way we look at the summarization problem.

By determining the elements of syntax, and preserving them in the final skim, we are making the following assumption — the presence of structure in the film (e.g. dialogs), indicates that the human director who has created the film, is using these elements to engender certain (but unknown to the algorithm) emotional responses in the viewer. Hence, a syntax preservation approach, will preserve those semantic concepts, even though the concepts remain unknown. Note the similarities of such an approach with the idea of a zero knowledge proof.

The focus on the semantics of the syntax, implies that for every domain, we have to use expert knowledge to identify the syntactical elements and their meanings in that domain. This often simplifies the problem since the rules for constructing syntactically correct expressions in a certain domain, are much fewer than the number of interesting patterns / objects in that domain.

However, object detection and syntax analysis can complement each other. In specific constrained domains, where the videos have considerable structure, and where one can successfully create specific object detectors, we must use both syntactical rules as well as the results from object detection when creating video summaries. Examples include sports videos such as baseball and news programs. We now introduce the idea of entities, and the discuss the relationship of the user task to the summaries generated.

1.3.3.4 *Tasks, entities and utilities*

The user's task affects the form of the summary. We distinguish between two types — *active* and *passive* tasks. In an active task (e.g. search) the user is looking for a specific piece of information (e.g. “how did Tiger Woods perform at the tournament?”), whereas in a passive task, the user has no specific queries in mind, but is more interested in consuming the information (e.g. a movie preview). This thesis focuses on the creation of passive skims.

The idea of an entity, is central to our summarization algorithm. An entity is a “real-world” thing, that has independent existence. An entity is associated with a set of attributes, a subset of which are selected to form the entity (e.g. a video shot, a segment of audio etc.). We attach a utility function to each entity that we are interested in preserving in the final skim. The problem of skim generation is now formulated as an utility maximization problem, where we maximize the utilities associated those entities that satisfy the user's information needs. We shall elaborate on this idea in chapter 7.

1.3.3.5 *An adaptive summarization framework*

The conceptual framework of utility maximization is a powerful one. Current work in summarization (e.g. [18] [32] [37] [52] [95] etc.) focuses on creating *static* summaries — summaries that have been created with certain user, device and network characteristics in mind (the user task, the nature of the user interface, bandwidth, screen resolution, the media that are supported by the device etc.). When one (or more) of these parameters

change, it has often resulted in researchers proposing a new summarization algorithm, to satisfy the new constraints.

The utility maximization framework offers *adaptability* to changes in parameters for which the summary was originally generated. These could be for example, the available bandwidth, the change in the task, the knowledge that the user has turned off the audio etc. These changes in the operating point of the summary can be readily incorporated into the utility maximization framework, by modulating the utilities of the entities affected by the change. For example, if it is known that the audio has been switched off, the summary generation algorithm has to simply set the utility of the audio segments to zero in the resulting framework. In recent work [17], we have introduced a formal framework modeling the relationships amongst entity, adaptation, resources, and utilities.

1.3.3.6 The scope of this work on summarization

We now indicate some of the issues related to summarization, that we did *not* consider in this thesis:

- We summarize the entire video, by individually summarizing each computable scene — i.e. we assume that each scene is independent of the other scenes. We do *not* exploit the inter-scene relationships that exist amongst scenes, at level of the whole video.
- We do not consider the use of text for the purpose of summarization. We decided against using text, since it is frequently unavailable in legacy video as well

as in many foreign language productions. However, when available, text is an important attribute and ought to be part of the summarization scheme.

- We do not consider scene-level syntax for summarization. An example of such syntax is parallel story development — where the director develops the plot over two locations in a rhythmic fashion, alternating between the two locations.

1.4 Summary of contributions

We now summarize the original contributions of this thesis when solving the three sub-problems of audio-visual segmentation, structure detection and audio visual skim generation.

1.4.1 Audio-visual scene segmentation

The goal of our work in segmentation was to segment of audio-visual data, such that each segment was consistent with respect to certain properties and preserved the syntactical elements in the data. We now briefly summarize the original contributions in our attempt to solve the segmentation problem:

1. A computational scene model that incorporates the synergy between audio, video, and structure in the data.
2. A finite, causal, memory model framework for segmenting both audio and visual data.

- a. Three new conceptual memory models, two of which incorporate Kolmogorov complexity and perceptual distortion.
3. The idea that features and the models used in segmentation should incorporate constraints that stem from film-making production rules as well as constraints due to the psychology of audition.
4. The idea that we need top-down structural grouping rules to improve segmentation results. This enables us to be consistent with human perception.
5. Constraints for merging information from different modalities (audio, video and within scene structure) to ensure that the resulting segmentation is consistent with human perception.

1.4.2 Structure detection

The goal of our work in structure detection was to detect a set of domain dependent structures that had generative mechanisms. We now summarize the original contributions in our attempt to solve the structure detection problem:

1. Defining structure in terms of Kolmogorov complexity.
2. A topological framework for the analysis of arbitrary discrete, deterministic sequences that have a metric associated with them.
3. Robust algorithms that incorporate statistical tests for detecting two specific structural elements — the dialog and the regular anchor.

1.4.3 Audio-visual condensation

The goal of our work in audio-visual condensation was to take a computable scene and given the target skim duration, reduce the duration of the scene till the time duration achieved. We now summarize the original contributions in our attempt to solve the condensation problem:

1. An entity-utility framework for skim generation:
 - a. The idea that a multimedia sequence can be thought of as comprising entities, a subset of which shall be needed to satisfy the users information needs.
 - b. A utility maximization framework that maximizes the utility of the entities required by the user. This framework is easily extensible beyond the specific skim problem addressed in this work [17] .
2. The idea that shot comprehensibility is related to its visual complexity.
 - a. We show that the duration of a shot can be reduced as a function of its visual complexity.
 - b. There exists a minimum duration below which a shot is incomprehensible.
3. The idea that skims need to preserve the visual syntax of the original sequence for them to be comprehensible.
 - a. We show specific syntax reduction mechanisms that preserve the essential semantics of these syntactical elements.

4. The use of prosody analysis to determine the discourse boundaries of speech, for determining the important speech segments.
5. The idea that summarization needs to be understood in terms of a holistic framework comprising the following:
 - a. The task that engages the user.
 - b. The user interface.
 - c. The user information needs
 - d. Network characteristics

Now, we provide an overview of the organization of the thesis.

1.5 Organization of the thesis

The rest of the thesis is organized as follows. In the next chapter, we introduce the segmentation problem. There, we shall discuss in detail the prior work on visual segmentation, audio segmentation and also review the important area of auditory scene analysis. In that chapter, we shall present our framework of the *computable scene*. We shall also introduce the idea of using memory as a central framework for audio-visual segmentation.

In chapter 3, we shall present our research on detecting computable video scenes. We shall introduce two key concepts of *recall* and *coherence* that are used in conjunction with the memory model to segment video data into scenes. In addition to presenting the

experimental results, we shall also discuss three new conceptual memory models that improve upon the basic model used in this thesis.

In chapter 4, we shall present our approach towards detecting the computable audio scene. There, we shall focus on three things: (a) metrics for the three types of features used (b) a new signal model for one of the classes of features and (c) using the correlations amongst the feature values to detect audio scene change locations.

In chapter 5, we shall present a multi-modal fusion framework for detecting computable scenes. The emphasis there is on developing a rule-based framework for fusing information from the computable audio and video computable scenes, silence and structure information, for detecting computable scenes.

In chapter 6, we present our approach towards detecting structure. We present a definition of structure and present the topological analysis framework for detecting structures with generative mechanisms. We also present specific algorithms for detecting the dialog and the regular anchor.

In chapter 7, we shall introduce the summarization problem. We shall also present extensive reviews of related work in video and audio summarization schemes. We shall also review the important area of discourse based segmentation. In this chapter, we shall also introduce the entity-utility framework that forms the basis for our skim generation.

In chapter 8, we shall present our work on visual analysis for skim generation. Here we shall introduce two concepts: the relationship between visual complexity and

comprehension time, and use of film grammar for ensuring that skims thus generated are comprehensible.

In chapter 9, we shall present our approach towards audio summarization. This involves two steps: classification of the audio segments using a robust SVM classifier, and the detection of significant phrases using prosody analysis.

In chapter 10, we shall show how to model the video skim generation problem as an utility optimization problem. We shall discuss the entities that we shall attempt to preserve, and the utilities associated with each entity. Then, we shall discuss the idea of tied multimedia constraints and shall present a detailed search strategy to arrive at the optimal solution.

In chapter 11, we shall present our conclusions as well discuss some future research directions.

The references are to be found in chapter 12, while we have the appendix in chapter 13. There, we shall present algorithms to estimate some specific forms of Kolmogorov complexity. We shall also present two early user studies done to evaluate visual skims (i.e. skims that contain no audio).

2 The segmentation problem

2.1 Introduction

This chapter introduces the segmentation problem and the framework that we propose to segment audio-visual data into scenes. We repeat the definitions from the *The Oxford English Dictionary*, and from Bordwell and Thompson's *Film Art* that we presented in the introduction, here:

The place where an action is carried on and people play their parts as in a drama; the scene of action, the place where events are actually happening or business being done. [OED]

A view or picture presented to the eye (or to the mind) of a place, concourse, incident, series of actions or events, assemblage of objects, etc. [OED]

A segment in a narrative film that takes place in one time and space or that uses crosscutting to show two or more simultaneous actions. [B & T, Film Art]

Examples of such scenes include — a scene showing people on the beach, a marketplace, a segment showing a newscaster, a football game etc.

The problem traditionally tackled in the multimedia community was one of segmenting audio-visual data into chunks that were consistent in the senses outlined above. However, this is a very challenging problem since this involves detecting segments that are consistent with respect to a high level semantic. The problem is normally tackled by constructing sophisticated models / object detectors using low-level feature information from the data.

In this work, we are interested in determining *computable scenes* and do not focus on determining the high-level *meaning* of the scene. We define a computable scene to be a segment of audio-visual data that possesses long-term consistency with respect to audio, video and structure. The term structure refers to the micro-level, within scene structure such as dialogs.

The computable scene comprises elementary computable video and audio scenes, as well as elements of structure, that are related to each other in specific ways. These scenes are deemed computable since they are consistent with respect to a certain property and can be reliably and automatically computed using low-level features alone. The computable scene problem is important for several reasons:

- Automatic scene segmentation is the first step towards greater semantic understanding of the structure of the film⁵, since we can then impose higher forms of knowledge on these elementary computable scenes.
- Breaking up the film into scenes will help in creating summaries, thus enabling a non-linear navigation of the film.

The absence of semantic consistency as an immediate goal in our approach to segmentation has a subtle but important effect. It forces us to think of the problem as *task-based* segmentation i.e. the generation of segments to assist the solution of other problems. In this work, the task of the segmentation algorithm is to generate segments that preserve the syntactical structure of the original video data. These syntactically correct segments form the input to our video summarization algorithm. There are four key components to our computable scene framework:

- A finite and causal memory model, to model short term memory.
- Algorithms for detecting computable audio and video scenes, that use the memory framework for segmentation.

⁵ We shall use the word “film” in this thesis to refer to movies i.e. commercially produced videos.

- The imposition of higher forms of knowledge in the form of structural constraints and silence.
- A multi-modal fusion framework for integrating the elementary computable scenes, structure and silence.

The rest of this chapter is organized as follows. In the next section, we present a discussion of the problem of shot detection and scene detection. In sections 2.3 and 2.4, we review prior work in visual and audio scene segmentation respectively. We present our framework for computable scene segmentation in section 2.5 and conclude by summarizing the chapter in section 2.6.

2.2 Thinking about shots and scenes

A video *shot* is a segment of video data, from a single camera take⁶. The *scene* in video data is normally defined (e.g. [100] [101] [106]) in terms of its semantics i.e. a group of shots that are characterized by a single semantic idea — for example, a group of shots from a trip to the beach. While these definitions may seem intuitive and seem to lend themselves towards analyzing data in a hierarchical fashion (i.e. the shot is an elementary unit of video, and the scene is a group of shots that share a single idea, and the whole film is a collection of scenes), the problem of analyzing visual data is complex.

⁶ i.e. it is a segment of video from the camera is switched on, till when it is switched off.



Figure 2.1: A shot detector based on color and motion, segments the video into piece-wise constant approximations with respect to the two features.

Current shot detection algorithms work very well, with accuracies up to 95% (e.g. [109]). This is because changes in shots are often accompanied by changes in color, spatial composition⁷ or motion. Shot detectors are usually based on these low-level features and are often tested on produced videos that show these visual changes. However, when confronted with a simple home video, many of these algorithms generate plenty of false alarms. For example, a one hour home video that is shot continuously, is really a single shot, but this gets fragmented by the shot detector. But, when we examine the resulting shot fragments, they look reasonable. The reason for this discrepancy is the fact that a shot actually encodes a semantic concept — the mechanical operation of the camera, in

⁷ According to the rules of film-making [8] , the viewing angle should change across shots by at least 30°, otherwise the shots are too similar, and this leads to a jerky visual effect. This is known as the 30° rule.

addition to the latent semantics in the visuals. In many produced videos, the changes in the visual properties coincides with the changes in the mechanical operation of the cameras.

Detecting segments that are consistent in the senses presented in the beginning of this chapter, is a challenging problem. The semantics of a scene is difficult to quantify in terms of the low-level features in a video. This is because the semantics of the scene depends upon many external, non-computable factors — (a) the particular method of visualization (i.e. the directorial style) of the semantic idea behind the shot (b) the relationship with preceding and following scenes and (c) the world knowledge of the viewer.

Clearly, the ability to segment data into semantically consistent scenes requires access to a body of knowledge that is external to the data in the video. Since such access is difficult, we need a scene definition that is (a) consistent (b) computable and (c) dependent only on the data within the video.

We define a shot to be a segment of video that is consistent with respect to a set of attributes. In this world-view, a shot-detector decomposes the video stream into segments that provide a first order approximation of the video with respect to a set of features (ref. **Figure 2.1**).

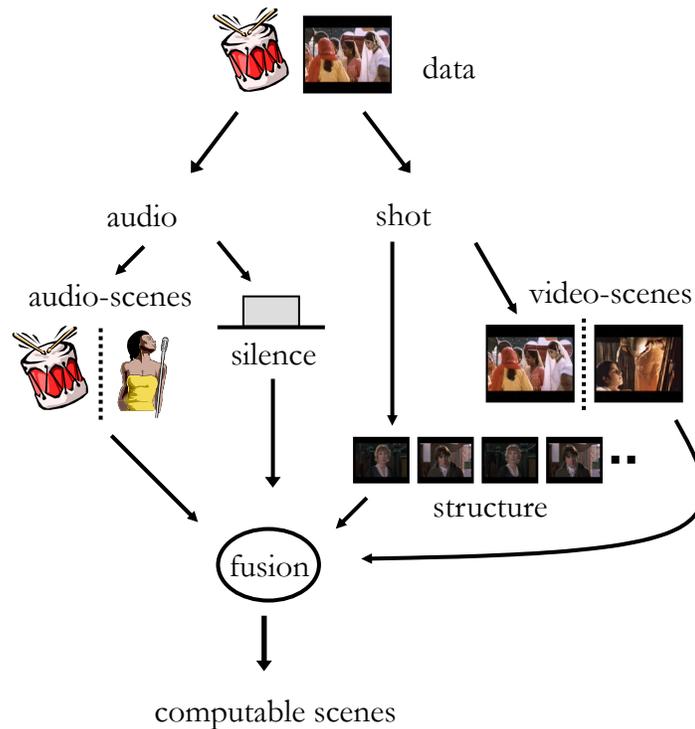


Figure 2.2: The computable scene overview.

There are constraints on what we see and hear in films, due to rules governing camera placement, continuity in lighting as well as due to the psychology of audition. In this work, we develop notions of video and audio computable scenes by making use of these constraints. A video computable scene (v-scene) exhibits long-term consistency with respect to two properties: (a) chromatic composition of the scene (b) lighting conditions. The audio computable scene (a-scene) exhibits long-term consistency with respect to the ambient audio.

We define a computable scene to be *a segment of audio-visual data that possesses long-term consistency with respect to audio, video and structure*. The computable scene comprises elementary computable video and audio scenes, as well as elements of structure, that are

related to each other in specific ways. We derive four types of computable scenes (c-scene) that arise from different forms of synchronizations between a-scene and v-scene boundaries. We term these scenes as *computable*, since they can be reliably computed using low-level features alone. **Figure 2.2**, presents a system overview; we shall discuss this idea in more detail in section 2.5.4.

Computability is a useful paradigm because it is a shift towards task based segmentation, where the primary task is *not* the semantics of the scene, but the preservation of the syntax of the video data. This preservation of syntax is central to our summarization project discussed in chapter 7.

Before we discuss the computational scene framework in detail in section 2.5, we shall now review prior work in visual and audio scene segmentation over the next two sections.

2.3 Related work: visual analysis

In this section we review three different approaches to visual scene segmentation: (a) Scene transition graphs, (b) Information theoretic clustering and (c) a model based on video coherence. The methods have been chosen since they represent three differing strategies for video scene segmentation.

2.3.1 Scene transition graphs

Scene transition graphs (STG) were first introduced by B.L. Yeo and M. Yeung [100] [101] [102] [103] [104] , as an attempt to segment video data into scenes, as well as a technique to compactly visualize the video data. We shall critique scene transition graphs in terms of their ability to summarize video data in section 7.2.1.

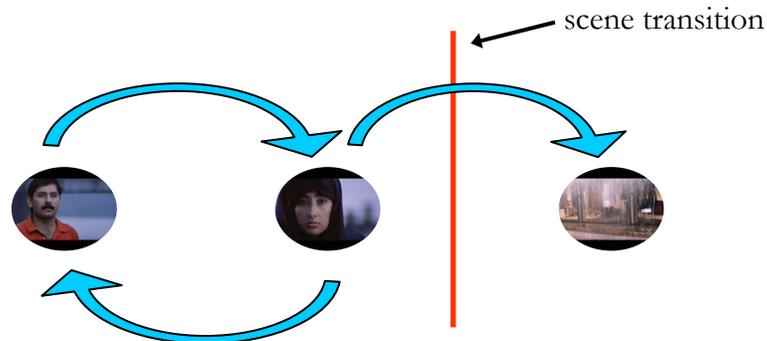


Figure 2.3: A scene transition graphs is segmented into scenes by detecting cut-edges. These edges divide the graphs into disconnected sub-graphs.

A scene transition graph [103] is a compact representation of a video that is a directed graph. Each node of the graph represents a cluster of similar shots (under a suitable similarity metric). Two nodes i, j are connected via an edge if there exists a shot in cluster i that precedes a shot in cluster j . In [103] , the authors perform time-constrained clustering on the shots using a time-window T to create the scene transition graph. Given two parameters δ and T , the maximum cluster diameter and the duration of the time window respectively, two shots belong to a cluster if they are within T sec. of each other and are within δ of each other with respect to the similarity metric [103] .

In **Figure 2.3**, we show three nodes and each node represents a time-constrained cluster. The presence of the cycle between the first two clusters indicates that these two nodes belong to the same scene. A scene transition occurs at a *cut-edge* i.e. edges when cut,

disconnect the graph into sub-graphs. This happens when there is a forward transition from an sub-graph to another sub-graph with no backwards transition.

An important concern in this work is the setting of the cluster threshold parameter δ and the time-window size T , both of which critically affect the segmentation result. Unfortunately, neither of these parameters can be set without taking into account the specific character of the data being analyzed.

Table 2.1: Variation of precision and recall, for constant T (2000), with variations in δ .

Cluster threshold: δ	Precision	Recall
0.3	0.33	1.00
0.5	0.75	0.88

The authors present their results on videos that are taken popular TV shows. The results from [103] seem to indicate the results vary, depending upon the specific parameter values for T and δ . While the authors do not provide precision-recall values, we can compute the precision and recall figures using the tables in [103]. The authors recommended values of $T = 2000$, $\delta \leq 0.5$ indicates sensitivity to the cluster threshold δ (ref. **Table 2.1**). The authors preferred the case of over-segmentation of the data as opposed to misses, since the segmented scenes could be post-processed to eliminate the false alarms.

2.3.2 Adaptive clustering

In this section we shall discuss the Video Manga project's approach to video segmentation [95]. While clustering is a popular technique for segmentation and summarization [32] [96] {more refs. needed!}, the Manga project incorporates a few novel techniques — (a) no shot segmentation (b) a measure of segment importance (c) key-frame *size* selection based on an importance measure.

In [95], the authors do not segment the video into shots. Instead, they begin with assigning *each* frame to a distinct cluster and then begin to hierarchically merge these clusters. Then they determine the cluster threshold as follows —

- keep merging the closest two clusters, till there is a single cluster.
- At every stage measure the average cluster diameter. A plot of average cluster diameter (d) versus the number of clusters (n) indicates the presence of a characteristic “knee” i.e. the value for n when the average cluster diameter starts to increase rapidly. Then set the cluster diameter threshold to be the average cluster diameter at the knee of the curve.

Once the cluster threshold is set, we have fixed the number of clusters in the approach. Now since in order to obtain the diameter vs. the number of cluster plot, we have already created a hierarchy of clusters, and where the details of the contents of each node have been stored. Now, given the cluster number, the segmentation result is immediately available, with the contents of each node representing the video segments.

The authors do not use time-constrained clustering and in its absence, the algorithm is likely to merge two similar sequences that are far apart in time, in the video. Also, the paper mentions no explicit results for segmentation.

The two other aspects of the Manga project — segment importance and size-based keyframe selection are not used in segmentation, but are important in the context of summarization. They shall be discussed in section 7.2.1.

2.3.3 Video coherence

John Kender and B.L. Yeo’s work on video coherence [44] , has influenced our own work on segmentation. In the video coherence formulation, the authors use a model of memory, to perform scene level segmentation. The idea being that segmenting video data can be viewed as the ability to recall the past, given the present data.

Briefly, in their model, a video frame entering a finite buffer of size B , has infinite persistence, i.e. for all time, the frame has a non-zero but exponentially decaying likelihood of survival. We shall discuss their memory model in more detail in section 2.5.2.1.

They define two parameters — recall and coherence, that are central to their segmentation framework. The recall between two shots P and Q , is defined as follows:

$$\begin{aligned}
R(P, Q) &= (1 - d(P, Q))(1 - e^{-l})e^{-m}(1 - e^{-r}), \\
l &= (b - a) / B, \\
m &= (g - b) / B, \\
r &= (b - g) / B,
\end{aligned} \tag{2.1}$$

where, shot P starts at a and ends at b , and shot Q starts at g and ends at h . d is the distance measure between the two shots and where B is the buffer duration in seconds. Coherence is defined at every shot boundary t as the pair-wise recall between *all* future shots with *all* shots from the past. Formally:

$$C(t) = \frac{\sum_{t_p \leq t} \sum_{t \leq t_q} R(p, q)}{C_{\max}(t)}, \tag{2.2}$$

where p and q are shots that precede and follow the shot boundary at t and where t_p and t_q refer to the timestamps associated with the shots. Note that *all* the shots in the video are used in this computation. C_{\max} is a normalizing constant obtained by evaluating the numerator of equation (2.2), with d the shot metric value set to zero. Note that the following:

- The system entailed by this formulation is non-causal (equation (2.2)).
- The coherence values are only computed at shot boundaries. When the shot lengths increase, we shall be sampling the true coherence function very coarsely.

In [44] , the authors first detect shot boundaries, via a shot detection algorithm. Then, they perform scene segmentation in the following way: they evaluate the coherence

across the all shot boundaries and declare scene boundaries at local minima locations.

The paper shows a relative comparison of the coherence based segmentation against scene transition graphs. In that comparison, the coherence method outperforms scene transition graphs.

2.3.4 Discussion

The three methods discussed in the preceding sections share the following issues:

- **Semantic scenes:** The scene definition in prior work was that of a *semantic* scene i.e. a collection of shots that share a common semantic property. It is important to keep this in mind while evaluating the performance of the algorithms discussed in the previous section, since the ground truth will be labeled with this criteria in mind.
- **No audio:** None of the three methods discussed below uses audio data to segment video data. While there has been some recent work [43] [44] that discusses audio-visual integration (more in section 5.4.3, when we discuss the performance of our joint model) most of the prior work focuses on visual data alone. Given the importance of audio in cognition, this anomaly is difficult to explain. Extenuating factors may include the relative ease of understanding and visualizing image data over auditory analysis.
- **Low-level:** The methods discussed in the previous sections do not make use higher forms of knowledge such as structural information, to aid segmentation.

Semantic level segmentation algorithm may be difficult with simple low-level feature based algorithms.

2.4 Related work: auditory analysis

In this section, we now discuss some related work on audio segmentation, as well as briefly review the important area of Computational Auditory Scene Analysis (CASA).

2.4.1 Segment classification

We begin this section by reviewing [74], one of the earliest papers and also one of most widely referenced papers in speech / music discrimination. There, the idea is to construct an algorithm to switch between radio channels so that user keeps hearing a particular genre of music or speech, perhaps according to a predefined user profile.

Noting that speech is a succession of syllables composed of short periods of frication followed by longer periods of vowels or highly voiced speech, the author proposes the use of the Zero Crossing Rate in order to make the distinction. The ZCR is a measure of the spectral center of mass. The dominant frequency principle shows that when a certain frequency band carries more power than other bands, it attracts the normalized expected zero crossings per unit time.

In the experiments, the incoming data is first split into non-overlapping frames that are 2.4 sec. long. Then, the author computes several statistical measures on the ZCR, per frame, such as the standard deviation of the first order difference of the ZCR etc. in

order to build a multi-dimensional feature detector. The test results are excellent, showing a 98% class separation.

In [75] , the authors perform an excellent, detailed analysis of a speech / music discriminator following up the work presented in [74] . They use three classification frameworks: A Gaussian MAP classifier, a Gaussian mixture of models (GMM) and a approximate k-d tree variant. The k-d tree is approximate in the sense that the authors only descend to the leaf node and avoid all backtracking (since the backtracking has an exponential cost associated with it).

The authors test their algorithms on a diverse collection of music and speech. All their data was recorded off local FM stations in the Bay area. They use a frame rate of 50Hz (i.e. the analysis frame was 20 ms. long). The results show that the Gaussian MAP classifier is better at speech than music (i.e. it is easier to misclassify music as speech than vice versa). The GMM also exhibits a similar behavior but the difference in performance is less marked. k-d trees show pretty much identical behavior with both classes. Of course, these results are strongly governed by the underlying data (i.e speech and music). This may also be an indicator that the underlying data is simple — i.e. the classes may be linearly separable.

In [50] , the authors adopt a two stage strategy to segment and classify the data. First they detect changes in energy to determine segmentation boundaries. Then, they classify 20ms frames using a trained Gaussian MAP classifier. They show a frame classification result of 85.1% on the test data (using MFCC). Then, they pool the classification results

of all the frames in the segment to classify the segment. The result of the pooling procedure boosts the classification accuracy to 90%.

In [107] the authors propose a heuristic approach to audio segmentation and indexing. They construct a system for real-time audio segmentation and classification. They attempt to classify the resulting segments into seven classes, including several mixture classes i.e. classes that contain more than one base class (e.g. environmental sounds plus speech).

For their on-line segmentation of data, the authors use three features — short time zero crossing rate, short time fundamental frequency and short time energy. They declare a segment boundary whenever an abrupt change is detected in *any* of these three features. A change is detected using two abutting sliding windows and features are computed in each window. A significant difference in any of the feature values in each window, signifies a scene boundary. They do not report segmentation results, but instead report classification accuracy. The average class classification accuracy is 90%. A similar technique is used in [84] , to classify audio data into speech, music and silence. There, they report classification accuracies of 75% averaged across classes.

2.4.2 Discussion

There are several patterns that emerge by analyzing prior work in auditory analysis.

- The focus is on *classification* of segments rather than segmentation. This difference is similar to the difference between an approach that first uses a technique for

general image segmentation, followed by grouping of labeled regions into objects, against an approach that detects objects directly in the image. While a classification approach to segmentation will not work well across data sets, in a data stream where the classes are a priori known and fairly well defined, a classification approach makes sense.

- The classification strategy uses *short* time-scales, typically 100ms to classify data. However, while human beings do perform short-term grouping of features, there is evidence that we also perform long term grouping before coming to a decision [9] [10] [26] .
- Finally, none of these algorithms take into account the generic generative properties of sound sources [9] , such as harmonicity, gradual changes in the sound characteristic etc. into the segmentation / classification algorithms.

We now review the important area of Computational Auditory Scene Analysis.

2.4.3 Computational Auditory Scene Analysis

The area of computational auditory scene analysis was pioneered by Albert S. Bregman [9] [10] whose work in this area is often compared to Marr's conceptual work in computer vision [55] . Bregman makes the case for using perceptual groupings in order to analyze sound in a manner akin to the analysis by computer vision researchers. Auditory scene analysis is a process whereby all the auditory evidence that comes over time from a single environmental source is put together as a perceptual unit. Bregman's observations on the process of audition stem from a series of psychological experiments

that he conducted on human beings. Bregman made many fascinating observations about the process of audition and we reproduce a few of them here.

- **Filtering in noisy environments:** We seem to be able to pick out particular sounds in an environment that is noisy in general. For example, we are able to readily recognize our name in a noisy environment. The reason for this could be that the schema to retrieve the particular thing (i.e. our name, in this case) that we filter out is in a very sophisticated state⁸. This filtering could also be voluntary, for example we are waiting in the doctor's office waiting for our name to be called out.
- A spectrogram of a sound mixture reveals that it is hard to "segment out" a particular sound from the composite mixture, even when we have spectrograms of the individual sounds in isolation. Clearly, we seem to be doing more than simple spectral analysis. The absence of any "clean" environment for learning these sounds indicates that we must have a low-level grouping scheme that helps us in our decision.
- **Old+New strategy:** The simple observation that unrelated sounds seldom begin and end at the same time, leads us to an *old+new* strategy. When a signal is suddenly (and briefly) replaced by another sound that shares some of the spectral characteristics of the "old" sound then the auditory system interprets it as the

⁸ Bergman also notes that we also seem to mistake a sequence of sounds for our name.

sum of two sounds. The old+new interpretation requires that the “new” sound be sudden. Determining what onset of sound counts as sudden is hard and intermediate onsets seem to give intermediate perceptual results.

- **Gradualness of change:** A single source tends to change its properties slowly and smoothly. A sequence of sounds from the same source tends to change its properties smoothly. e.g. if we are standing by the road and a car passes by, then the sound of the car engine will diminish gradually with distance. Hence, since the changes are gradual, samples from the same source taken in quick succession are likely to resemble each other.
- **Changes affect all components of the resulting sound:** For example, if we are walking away from the sound of a bell being struck repeatedly, the amplitude of *all* the harmonics will diminish gradually. Note that this change is brought about by a physical event. At the same time, the harmonic relationships and common onset⁹ are unchanged.
- **Multiple bases for segregation:** Bregman says that multiple grouping cues cooperate and compete for the users attention when the user is deciding whether a particular sequence (in time) is to be grouped in a particular way. However, it is not clear what role higher forms of knowledge (for example a person may want

⁹ Different sounds emerging from a single source begin at the same time.

to concentrate on the human voice in a sound clip) may have on the final grouping.

- **Regularity:** When a body vibrates with a repetitive period, its vibrations give rise to an acoustic pattern in which the frequency components are multiple of a common fundamental.

Different computational models have emerged in response to Bregman's observations [12] [20] [26] . While these models differ in their implementations and differ considerably in the physiological cues used, they focus on short-term grouping strategies of sound. Notably, Bregman's observations indicate that long-term grouping strategies are also used by human beings (e.g. it is easy for us to identify a series of footsteps as coming from one source) to group sound.

In [26] , the author makes a case for prediction driven computational auditory scene analysis as opposed to a data-driven analysis that attempts to "build-up" i.e. a bottom up approach to modeling data. A convincing argument against pure data-driven systems is the presence of auditory illusions such as the well documented continuity illusion as well as the phenomena of phoneme restoration.

In [26] , the goal is to use the mid-level representations and use them to build higher world models. These models are used to predict the subsequent input and the prediction is then reconciled with the actual input. This allows the parameters of the model as well the number of elements of the model to change. The change is achieved by having competing hypothesis exist for the current scenario and then one hypothesis is chosen by

computing the MDL [21] [36] [71] . The core of the reconciliation engine is a blackboard system (based on the IPUS system [15]).

2.5 The computable scene framework

We now present our framework for audio-visual scene detection, which we call the computable scene framework. There are four key aspects to our framework:

- A model for short-term memory that serves as a basis for audio and video scene detection.
- A computable scene definition in terms of relationships between elementary audio scenes and video scenes.
- An approach for detecting elementary audio-scenes, video scenes, structure and silence.
- A rule based framework for multi-modal fusion.

We begin with a discussion on computability vs. semantics and then in section 2.5.2 present a few insights obtained from understanding the process of film-making. We shall use them as well the insights gained from the psychology of audition (section 2.4.3), in creating our computational model of the scene.

2.5.1 Why the emphasis on computability?

The semantics of a normal scene within a film, are often difficult to ascertain and make sense only with regard to the context. The context is established due to two factors: the

film-maker and the viewer. The film-maker infuses meaning to a collection of shots in three ways: (a) by *deciding* the action in the shots (b) the selection of shots that precede (and follow) this scene (c) and finally by the manner in which he *visualizes* the scene. For example, in order to show a tense scene, one film-maker may have fast succession of close-ups of the characters in a scene. Others may indicate tension by showing both characters but changing the music.

All three methods affect the viewer, whose *interpretation* of the scene depends on his world-knowledge. Hence, if the meaning in a scene is based on factors that cannot be measured directly, it is imperative that we begin with a scene definition in terms of those attributes that are measurable and which lead to a consistent interpretation.

2.5.2 The idea of memory

We use the idea of memory in our approach towards video and audio scene segmentation. The idea is intuitive: human beings compare the present with the past, to decide if there is a scene change.

Formally, there are two forms of memory used in practice: when discussing memory — (a) declarative and (b) procedural memory. Both forms are true for arbitrary stimuli but we shall focus on visual examples in the discussion below.

- **Declarative:** In declarative forms of memory, we use an *exact* representation of the past to decide if we have seen the present data. For example, when we are

given a photograph of a face, we use a declarative form of memory to decide if we have seen that person earlier.

- **Procedural:** In procedural forms of memory we do not retain an exact representation of what has been seen earlier, but retain the process of seeing or sensing that stimulus. For example, let us assume that a student is learning to drive a car. Then, over an extended period, after driving for many miles, the student would have learnt how to drive. This form of memory, is procedural since it entails remembering the process of driving given any road, and is independent of any specific road that the student may have driven on to learn.

In this work, we shall focus on procedural forms of memory.

2.5.2.1 *A model with infinite persistence*

The work in [44] formulates the memory model as a “leaky buffer.” In that scenario, we first assume that we have a finite buffer B that can hold a limited number of images. Then, whenever a new frame enters the buffer, one of the frames in the buffer is removed. This is modeled as a random process, with a uniform distribution on all the frames in the buffer. i.e. all frames are equally likely candidates for removal. Then, it is easy to show [44] that the likelihood for a frame to survive after t sec. is given by:

$$L(t) = \exp\left(-\frac{t}{B}\right), \quad (2.3)$$

where L is the likelihood of survival at time t and B is the buffer length in sec. The likelihood function is used to compute recall between two shots in the following manner. The recall is determined by computing an integral over the frames of the two shots in the memory and the likelihood function (2.3) serves as the prior for each shot frame [44].

An important consequence of equation (2.3) is the following — once a frame enters the buffer, it has infinite persistence, since every frame has a non-zero likelihood of survival. Hence when one computes coherence according to equation (2.2), one must necessarily compute coherence with *all* of the past. The buffer size plays the role of a persistence modifier — with large values of the buffer implying that the likelihood remains large for long periods of time and vice-versa.

2.5.2.2 A FIFO memory model

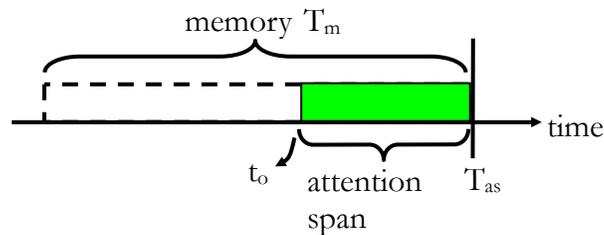


Figure 2.4: The attention span T_{as} is the most recent data in the memory. The memory (T_m) is the size of the entire buffer.

In order to segment data into scenes, we propose a causal, finite, first-in-first-out (FIFO) model of memory (**Figure 2.4**). In this scenario, whenever a new frame enters the buffer, the oldest frame in the buffer is removed. This model removes the issue of

infinite persistence in [44] (which makes the computation difficult), and makes the model causal. Intuitively, causality and a finite memory will more faithfully mimic the human memory-model than an infinite model. This will also make the model more readily amenable to real-time implementation. We shall use this model for *both* audio and video scene change detection.

In our model of a viewer (or listener in the case of audio memory model) two parameters are of interest: (a) memory: this is the net amount of information (T_m) with the viewer and (b) attention span: it is the most recent data (T_{as}) in the memory of the listener (typical values for the parameters are $T_m=32$ sec. and $T_{as}=16$ sec.). This data is used by the viewer to compare against the contents of the rest of the memory in order to decide if a scene change has occurred.

2.5.3 Insights from film-making

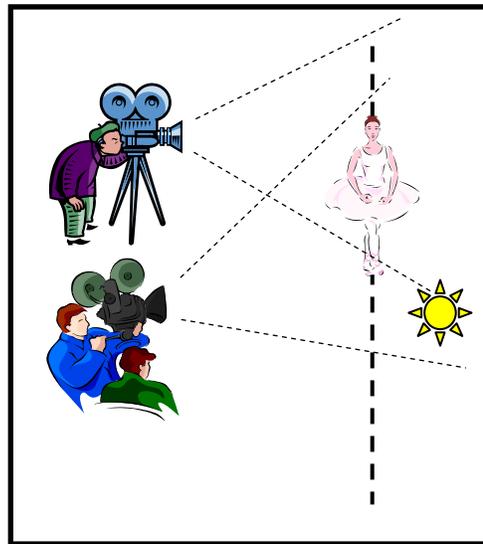


Figure 2.5: The director will place cameras on the same side of an imaginary line, called the line of interest. This ensures consistent left-right orientation across shots.

The line of interest is an imaginary line drawn by the director in the physical setting of a scene [8] [29] [77] . During the filming of the scene, all the cameras are placed on one side of this line (also referred to as the 180 degree rule). This has the following effect: the relative left-right orientations of the objects / actors in successive shots do not change. Transgression of this rule was so infrequent that directors who did were noted in the film theory community. e.g. Alfred Hitchcock's willful violation in a scene in his film *North by Northwest* [29] . Note that the critically acclaimed film *Memento*, that dealt with memory loss, does *not* violate the standard camera placement rules in film-making. Though the film is narrated *backwards* in time, each scene actually goes *forward* in time and also obeys the standard principles of film-making.

The 180 degree rule has interesting implications on the computational model of the scene. Since all the cameras in the scene remain on the same side of the line in all the shots, there is an overlap in the field of view of the cameras (see **Figure 2.5**).

This implies that there will be a consistency to the chromatic composition and the lighting in all the shots. Film-makers also seek to maintain continuity in lighting amongst shots within the same physical location. This is done even when the shots are filmed over several days. This is because viewers perceive the change in lighting to be indicative of the passage of time. For example, if two characters are shown talking in one shot, in daylight, the next shot cannot show them talking at the same location, at night.

2.5.4 The computable scene

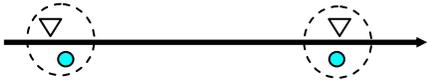
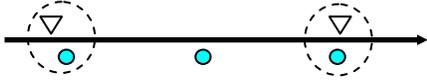
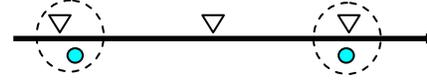
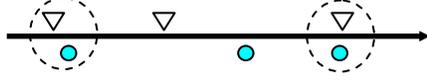
The constraints imposed by production rules in film and the psychological process of hearing lead us to the following definition of audio and video scenes. A video scene is a continuous segment of visual data that shows *long-term* consistency with respect to two properties: (a) chromaticity and (b) lighting conditions, while an audio scene exhibits a long terms consistency with respect to ambient sound. We denote them to be *computable* since these properties can be reliably and automatically determined using features present in the audio-visual data. The a-scene and the v-scenes represent elementary, homogeneous chunks of information. Analysis of experimental data (one hour each, from five different films) indicates that for both the audio and the video scene, a minimum of 8 seconds is required to establish the scene.

We define a computable scene (abbreviated as c-scene) in terms of the relationships between a-scene and v-scene boundaries. It is defined to be a segment between two consecutive, synchronized audio visual scenes. Note that in films, audio and visual scene changes will *not* exactly occur at the same time, since this is disconcerting to the audience. They make the audio flow “over the cut” by a few seconds [70] , [39] , thus making the transition between shots smooth.

2.5.4.1 Validation of C-scene definitions

The analysis of the kinds of synchronizations between audio and video scene boundaries leads us to hypothesize the existence of four types of computable scenes (see **Table 2.2**).

Table 2.2: The four types of c-scenes that exist between consecutive, synchronized audio-visual changes. solid circles: indicate audio scene boundaries, triangles indicate video scene boundaries

Type	Abbr.	Figure
Pure, no audio or visual change present.	P	
Audio changes but consistent visual.	Ac-V	
Video changes but consistent audio.	A-Vc	
Mixed mode: contains unsynchronized audio and visual scene boundaries.	MM	

We validated the computable scene definition, which appeared out of intuitive considerations, with actual film data. The data were from three one hour segments from three English language films (the films: *Sense and Sensibility*, *Pulp Fiction*, *Four Weddings and a Funeral*.) The definition for a scene works very well in many film segments. In most cases, the c-scenes are usually a collection of shots that are filmed in the same location and time and under similar lighting conditions (these are the P and the Ac-V scenes).

Note that the figures for Ac-v, A-Vc and MM type scenes, in **Table 2.2** show only one audio / visual change. Clearly, multiple changes are possible. We show only one change for the sake of figure clarity.

The A-Vc (consistent audio, visuals change) scenes seem to occur under two circumstances. In the first case, the camera placement rules discussed in section 2.5.1 are violated. These are montage sequences and are characterized by widely different visuals (differences in location, time of creation as well as lighting conditions) which create a unity of theme by manner in which they have been juxtaposed. Mtv videos are good examples of such scenes. In classic Russian montage, the sequence of shots were constructed from placing shots together that have no immediate similarity in meaning. For example, a shot of a couple may be followed by shots of two parrots kissing each other etc. The meaning was derived from the way the sequence was arranged.

Table 2.3: c-scene breakup from the film sense and sensibility.

C-scene breakup	Count	Fraction
Pure	33	65%
Ac-V	11	21%
A-Vc	5	10%
MM	2	4%
<i>Total</i>	51	100%

The second case of A-Vc scenes consists of a sequence of v-scenes that individually obey the camera placement rules (and hence each have consistent chromaticity and lighting). We refer to the second class as *transient scenes*. Typically, transient scenes can

occur when the director wants to show the passage of time e.g. a scene showing a journey, characterized by consistent audio track.

Mixed mode (MM) scenes are far less frequent, and can occur for example, when the director continues an audio theme well into the next v-scene, in order to establish a particular semantic feeling (joy/sadness etc.).

Table 2.3 shows the c-scene type break-up from the first hour of the film *Sense and Sensibility*. There were 642 shots detected in the video segment. The statistics from the other films are similar. Clearly, c-scenes provide a high degree of abstraction, that will be extremely useful in generating video summaries. Note that while this work focuses on computability, there are some implicit semantics in our model: the P and the Ac-V scenes, that represent c-scenes with consistent chromaticity and lighting are almost certainly scenes shot in the same location.

2.5.4.2 Detecting the video and audio scenes

We now present a brief overview of our approach to detecting computable video and audio scenes; details on our approach can be found in chapters 3 and 4 respectively. The algorithms for detecting the computable video and audio scenes both make use of the FIFO memory model (section 2.5.2.2).

In order to segment video data onto scenes, we proceed as follows. The video data comprises shot key-frames. The key-frames in the attention span are compared to the rest of the data in the memory to determine a measure of coherence between the

attention span and the rest of the memory. In order to compute coherence we need to include three factors: (a) the pair-wise dissimilarity between shots in the attention span and the rest of the memory, (b) the duration of all the shots and (c) the time separation between a shot in the attention span with a shot in the rest of the memory. Scene change locations occurs at local coherence minima.

In order to segment the data into audio scenes, we proceed as follows. We first compute a set of features from the audio data. These features differ in terms of type (scalar, vector and point) and the temporal resolution at which they are extracted. In this work, we also present a new signal model of the scalar sequence. As in the case of v-scene segmentation, we need to measure the degree of correlation amongst adjacent audio segments to determine scene boundaries. In order to do this, we compute a parameter that measures the rate of *increase* of the distance amongst adjacent segments. Local maxima of this distance increase rate, yield a-scene boundary locations.

2.5.4.3 Detecting structure and silence

Why is detecting structure and silence important? Let us first examine the case for structure. Assume that we have a dialog scene showing two people engaged in a conversation. Let us further assume that each shot in the dialog lasts 30sec. each. Then, the computational model for detecting video scenes cannot disambiguate between the case of two long and widely differing shots in a dialog sequence and the case of two adjacent v-scenes. It will thus detect a v-scene boundary between adjacent shots.

However, human beings recognize the structure in the overall sequence and thus group the shots together as a dialog.

Silence detection is useful in two contexts: (a) detecting the start of conversation by determining significant pauses [16] and (b) in English films, the transitions between computable scenes may involve silence.

We introduce a novel framework for detecting visual structure that assumes that the data containing the structure is discrete and can be associated with a metric. The framework seeks to exploit the topological properties of the structure (i.e. the metric relationships between the elements of the structural element). In this thesis we shall detect two structural elements that have simple generative rules.

We shall detect silence using an adaptive threshold on the energy histogram of the audio data. Details on structure detection can be found in chapter 6, while details on silence detection can be found in section 4.6.

2.5.4.4 Multi-modal fusion

In this work, we develop a rule based approach for determining computable scenes. This involves fusion of information (see **Figure 2.2**) from four sources: (a) v-scenes, (b) a-scenes, (c) structure and (d) silence. The use of silence and structure is just the idea of imposing higher levels of knowledge on the basic computable scene model. Another example of a higher-form of knowledge is the specific visualization (e.g. of emotions

such as happiness, anger etc.) style of the director; for example the director may present The details on the fusion algorithm can be found in chapter 5.

2.6 Summary

In this chapter we have introduced the segmentation problem and presented a computational scene framework for films. The framework was derived from camera placement rules in film-making and from experimental observations on the psychology of audition. We do not focus on deciphering the semantics of the scene.

The computational framework provides a consistent way to automatically generate scenes from low-level feature analysis. While the resulting scenes can form the basis for semantic analysis at a later stage, the specific task behind our computational approach was to preserve the syntactical properties of the original video data. These syntactically correct segments are important for our approach to video summarization.

We discussed in depth, prior work in visual scene analysis that dealt with three approaches to the problem of visual scene segmentation — (a) scene transition graphs, (b) adaptive clustering and (c) video coherence. We also reviewed prior work in audio scene segmentation, that primarily dealt with the classification of audio into predefined classes. We also discussed the area of computational auditory scene analysis and discussed some of Albert Bregman’s observations on the process of audition.

Finally, we presented our computable scene framework. The framework had four key components: (a) a memory model, (b) algorithms for v-scene and a-scene detection (c)

the use of structure and silence, as higher knowledge constraints and (d) a multi-modal fusion framework for the c-scene detection.

3 The computable video scene

3.1 Introduction

In this chapter, we shall describe our approach towards detecting computable video scenes (v-scenes). V-scene detection is the first step towards computable scene segmentation. A v-scene has certain consistent temporal properties that stem from camera arrangement constrains from film-making. We deem these scenes as computable since they can be automatically and consistently computed from the raw data. We shall not focus on deciphering the *semantics* of the scenes.

These video scenes are computed within a memory model framework and makes use of two computations on the data in the memory — *recall* and *coherence*. Recall models the relationships between two shots, while coherence measures the inter-segment similarity. We use this coherence measure to determine the relationship between the present data and the data stored in the rest of the memory. We declare v-scene boundaries to exist at locations of local coherence minima. These minima are obtained using a two-window strategy, that measures the strength of the minimum. Our experimental results indicate that our algorithm works well, with best case precision of 70% and recall of 91%.

In this chapter, we shall also introduce three new theoretical models to compute the coherence function. These models improve upon the basic model used in our

experiments. While the first model improves upon the basic model by introducing new functions, the other two models use an entirely new approach. They make use of the idea of Kolmogorov complexity [21] [51] (ref. appendix 13.1), a fundamental measure of information, to determine scene boundaries.

The rest the chapter is organized as follows. We begin by defining a v-scene and then follow that section with definitions of two functions central to v-scene segmentation — recall and coherence, in sections 3.3 and 3.4 respectively. Then in section 3.5, we discuss the procedure for detecting v-scenes and we follow that up with experimental results in section 3.6. In section 3.7, we present three new theoretical models that improve upon the model that we have used in our experiments, and we finally conclude the chapter by summarizing our contributions in section 3.8.

3.2 The v-scene definition

We define the v-scene as a contiguous segment of visual data that is chromatically coherent and also possesses similar lighting conditions. A v-scene boundary is said to occur when there is a change in the long-term chromaticity and lighting properties in the video. This definition stems from the film-making constraints discussed in section 2.5.1. Note that we have avoided bringing in the semantics of the segment into the definition, thereby allowing us to label the ground-truth, in a consistent manner.

The framework for v-scene detection that we propose, involves using the memory model framework discussed earlier (ref. section 2.5.2) and the shot key-frames. The framework

can conceptually work without having to detect shots, i.e. with raw frames alone. However, this would lead to an enormous increase in the computational complexity of the algorithm. Hence, the video stream is first converted into a sequence of shots using a sophisticated color and motion based shot boundary detection algorithm [109], that produces segments that have predictable motion and consistent chromaticity. A frame at a fixed time after the shot boundary is extracted and denoted to be the key-frame.

3.3 Visual recall: measuring inter-shot similarity

In this section we define *recall*, a measure of how one shot viewed in the past is similar to a shot that we are viewing at the present moment. In our visual memory model, the data is present in the buffer in the form of key-frames of shots and each shot occupies a definite span of time. The model also allows for the most recent and the oldest shots to be partially present in the buffer (see section 2.5.2.2, for details on the FIFO memory model). The idea of recall between two shots a and b is formalized as follows:

$$R(a,b) = (1 - d(a,b)) \bullet f_a \bullet f_b \bullet (1 - \Delta t / T_m), \quad (3.1)$$

where, $R(a,b)$ is the recall between the two shots a, b . $d(a,b)$ is a L^1 color-histogram based distance measure between the key-frames corresponding to the two shots, f_i is the ratio of the length of shot i to the memory size ($T_m > 0$). Δt is the time difference between the two shots.

The formula for recall indicates that it is proportional to the length of each of the shots in the memory. This is intuitive since if a shot is in memory for a long period of time it

should be recalled more easily. Also, the formula quantifies the intuitive observation that the ability of one shot to recall another should decrease, if they are further apart in time. Finally, as the formula notes, with an increase in distance between the two shots, the recall should decrease.

The product form of equation (3.1) is justified as follows.

- Intuitively, the recall between two shots should be zero if either of shots is of zero duration (i.e. it doesn't exist in memory!).
- If the distance between the two shots is infinite, the recall must drop to zero. In our case, the distances are normalized to lie between 0 and 1.
- If one of the shots lies out of the memory buffer (i.e. $\Delta t = T_m$), then the recall again must drop to zero, since we cannot compare shots that don't exist in the memory buffer.

The reason why the time durations present in the formula for recall (i.e. the shot lengths f_a and f_b , and the relative time separation Δt) are normalized with respect to the buffer length is as follows. If we think of the amount of information in each shot to be proportional to its shot length, then our decision on computing recall must be based on the relationship between the amount of information in a shot with respect to the net information in the rest of the memory.

A similar argument can be made for normalizing Δt . Let us assume that the time separation between two shots is 20 sec. Now, consider two cases when the buffer are sized 25sec. and 250 sec. respectively. In the first case, the viewer has very short term memory and is likely to forget the old shot (i.e. the shot that the viewer is trying to recall), while in the second case the viewer has very long term memory, and is likely to retain the old shot for much longer, hence increasing the recall.

The distance function $d(a,b)$ in equation (3.1) is computed as follows. First we use 232 bin histogram in the HSV (Hue, Saturation, Value) space. The use of the HSV space accomplishes two things: (a) perceptually similar colors are close in the metric used in this space, and (b) the lighting changes are now easily detected (via changes in value). We quantize the color space as follows: 18 hues, 4 saturation and 3 value points. We additionally use 16 shades of gray. The metric d used in the formula for recall is the L¹ color histogram difference, that is normalized to lie between 0 and 1.0.

Note that the definition of the term *recall* used in this section is different from the one used in information retrieval (precision / recall).

3.4 Coherence: measuring inter-segment relationships

In this section we define *coherence*, a measure of how two segments are similar to each other. V-scene boundaries occur when there is low coherence between two consecutive segments. Coherence is defined by making use of the definition of recall:

$$C(t_o) = \frac{\sum_{a \in T_{as}} \sum_{b \in \{T_m \setminus T_{as}\}} R(a, b)}{C_{\max}(t_o)} \quad (3.2)$$

where, $C(t_o)$ is the coherence across the boundary at t_o and is just the sum of recall values between all pairs of shot-lets (defined in the next section) across the boundary at t_o . $C_{\max}(t_o)$ is obtained by setting $d(a, b) = 0$ in the formula for recall (equation (3.1)) and re-evaluating the numerator of equation (3.2). This normalization compensates for the different number of shots in the buffer at different instants of time.

A scene transition boundary occurs at t_o if the shots that come after that point in time, do not recall [44] the shots prior to that point. This will then cause a coherence minimum to occur at the shot boundary.

While the current implementation of the coherence algorithm does use shot key-frames, conceptually, it will work without any changes on the continuous video stream. We have traded a slight decrease in performance for large decrease in computational complexity.

The coherence function is also useful after segmentation. Let us assume a skimming application where one wants to reduce (i.e. eliminate frames) from the shots after a scene segmentation boundary. Then, the coherence function can be modified to indicate the rate of change of information as we move away from the boundary. This can allow us to determine the amount of the scene to retain.

3.4.1 What are shot-lets?

We need to introduce the notion of a “shot-let.” A shot-let is a fraction of a shot, obtained by breaking individual shots into δ sec. long chunks. However, they could be smaller due to shot boundary conditions. Each shot-let is associated with a single shot and its representative frame is the key-frame corresponding to the shot. As will be clear shortly, this is done to reduce the computational complexity of the algorithm.

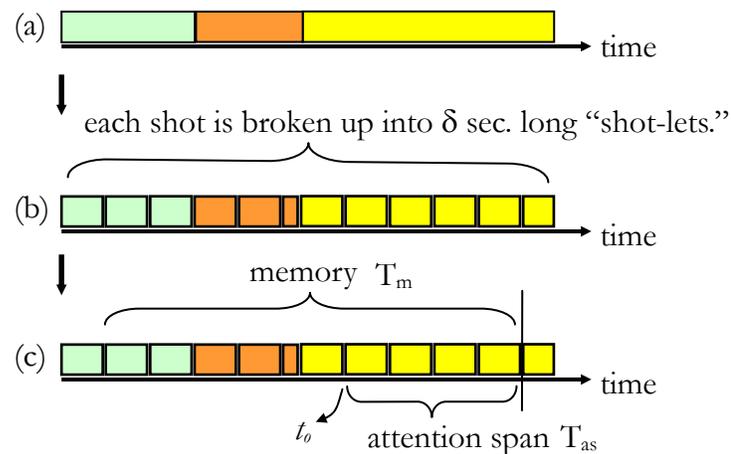


Figure 3.1: (a) Each solid colored block represents a single shot. (b) each shot is broken up into “shot-lets” each at most δ sec. long. (c) the bracketed shots are present in the memory and the attention span. Note that sometimes, only fractions of shots are present in the memory.

Why are shot-lets necessary? This becomes important when we have long shots. Since the coherence function is only evaluated at shot boundaries, the presence of long shots can cause the coherence function to be sparsely sampled, thus causing potential minima to be missed. For example, in a short video clip of 90 sec. with three shots of 30 sec.

each, the coherence function would only be evaluated at the two interior points. The shot-let formulation allows the coherence function to be smoothly sampled while preserving the original shot boundaries.

Another way to evaluate the coherence function is to sample the video at additional points (for example, by using additional key-frames per each shot) to increase the coherence density. However, since coherence evaluation is $O(N^2)$ in the number of shots, this technique will increase the evaluation time of the coherence function.

Shot-lets decrease the computational complexity of the algorithm by associating with the same key-frame as the original shot. Consider the case when we have computed the dissimilarity between two shots A and B . Then, once this is done, we do not have to reevaluate the dissimilarity between shot-lets from shot A , with shot-lets from shot B ¹⁰. Note that we will use shot-let duration information, and time-separation between shot-lets, in addition to the dissimilarity, when computing recall (ref. equation (3.1)). In our experiments, we find that $\delta = 1$ sec. works well. **Figure 3.1** shows how shot-lets are constructed.

In the sections that follow, it is to be understood that we are dealing with shot-lets, rather than shots. Note that the formulas for recall and coherence do not change with the

¹⁰ Implicit in this discussion is the assumption that the visual characteristics of the shot do not change appreciably over the duration of the shot.

introduction of shot-lets, since the key-frame associated with the shot-let is the same as that of the shot.

3.5 Detecting v-scenes

In this section we discuss the procedure for detecting v-scenes. We detect the local coherence minima for v-scene detection since v-scene boundaries only occur at coherence minima.

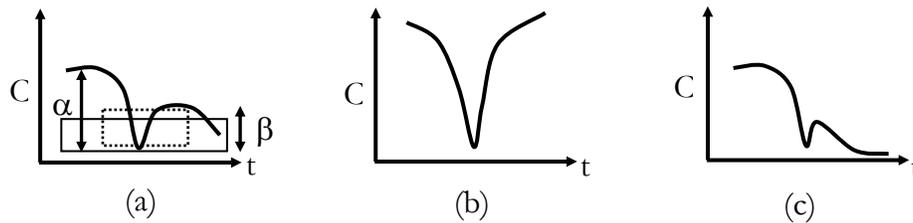


Figure 3.2: Each figure shows a plot of coherence against time. The three figures show the (a) normal (b) strong and (c) weak coherence minima cases. Figure (a) shows the two coincident windows W_0 (outer window) and W_1 (inner window) that are used for coherence minima detection.

We use a two-window approach to detect coherence minima. We define two windows W_0 and W_1 , where W_0 is a window of size $2k+1$ points and W_1 is a smaller window centered in W_0 of size $k+1$ (see **Figure 3.2 (a)**). For example, a typical value of k is 4. To determine if a minima exists, we first check if a minima exists within W_1 . If it does, we then need to impose conditions on this minima with respect to the coherence values in the larger window W_0 before we deem it to be a v-scene boundary.

First, we need to define three parameters (α , β , γ) relating to coherence values in W_0 .

α , β : they are respectively, the difference between the maximum on the left and the right half coherence windows and the minima value. γ : this is the difference between the minima and the global minima in W_0 . Then, on the basis of these three values, we classify the minima into three categories (see **Figure 3.2**):

- Strong : $S \equiv \min(\alpha, \beta) > 0.3 \vee (\min(\alpha, \beta) > 0.1 \wedge \max(\alpha, \beta) > 0.4)$
- Normal: $N \equiv (\max(\alpha, \beta) > 0.1) \wedge (\min(\alpha, \beta) > 0.05) \wedge (\gamma < 0.1) \wedge (\neg S)$
- Weak : $W \equiv \max(\alpha, \beta) > 0.1 \wedge (\neg S) \wedge (\neg N)$

The values above were determined using a 500 sec. training set obtained from the film *Sense and Sensibility*. The strong case is good indicator a v-scene boundary between two highly chromatically dissimilar scenes. The weak case (the two window technique is particularly useful in detecting the weak minima cases.) becomes important when we have a transition from a chromatically consistent scene to a scene which is not as consistent. These are the $P \rightarrow A-Vc$, or $Ac-V \rightarrow A-Vc$ (and vice-versa) type scene transitions (see **Table 2.2** for a definition of each type of scene). The categorization of v-scenes into strong, normal and weak becomes very useful when we are fusing information from a-scene boundary detection, v-scene boundary detection, silence detection and structure analysis. This is discussed in greater detail in chapter 5, where we discuss integration of different modalities.

How does the use of shot-lets affect the v-scene boundaries? Can it occur in the middle of the original shot? V-scene boundary locations obtained from shot-lets, can actually differ from the original shot boundaries. However in practice, these deviations are small compared to the original shot durations. Then, since the true v-scene can only occur at one of the original shot boundary locations, we simply time-stamp the detected v-scene boundary with the time of the nearest shot boundary.

3.5.1 Comparisons with shot detection

Now, we briefly comment on the differences between the shot detection algorithm used in this work and the new v-scene detection algorithm. The shot detection algorithm [109] operates on the MPEG-1 compressed stream. It uses the following features — average color and variance, motion statistics (ratio of intra coded blocks to motion predicted blocks, ratio of number of forward to backward motion vectors). The detection is done over two short windows (0.2 sec. and 2 sec.) with a decision tree to come up with a robust algorithm. The performance is excellent over a wide range of datasets (precision 91 % and recall 95%). We now highlight the key differences.

The shot detection algorithm compares *two* frames and picks the local minima of this measure over a small window (typically 0.2 ~2 sec.) to detect shots. However, the consistency of scene is a long-term (empirical analysis of video data indicates that in order for a shots to be grouped together as a scene, the duration of this group must at least be 8 sec.) *group* property, and is better determined by using the mutual information between two video segments (approximated by two groups of key-frames of shots).

The distance function for the v-scene detection takes into account the distance between the color-histogram two shots, the duration of each shot, and their temporal separation. There is no temporal weighting in the shot detection algorithm.

An important aspect of our v-scene algorithm is the imposition of top-down structural constraints. Human beings tend to group data that is highly structured into one semantic unit (e.g. dialogs, point-of-view shots¹¹). Such groupings will be missed by simple scene detection algorithms since the structural constraints are not considered. Chapter 6 discusses a framework to detect structure in sequences and the fusion of different modalities is discussed in greater detail in chapter 5.

3.6 Experiments

In this section we shall describe our experiments on v-scene detection. We first begin by discussing in detail our labeling procedure. Then, we present the results of our algorithm and this followed up by a comparison with related work.

¹¹ A point-of-view shot is a shot that shows the audience what the character sees. For example, the director may show a shot of the character, then show us what the character sees, and then will cut back to the character to show the reaction of the character to what he / she sees. For an example see **Figure 6.3**.

3.6.1 Labeling the data

The video data was labeled independently from the audio so as to prevent any influence of the audio track on the labeling. Only one person (the author) labeled the data. We attempt to label the video data into coherent segments (segments that possess long term consistency in chromaticity and lighting). From empirical observations of film data, it became apparent that for a group of shots to establish an independent context, it must last at least 8 sec. Hence all the v-scenes that we label must last more than 8 sec. Then, the labeling criteria were as follows:

- Do not mark v-scene boundaries in the middle of dialogs or regular anchors (these are elements of structure that we detect in our framework; structure detection is discussed in detail in chapter 6), instead mark structure end points at the beginning and end of the dialogs/regular anchors.
- When encountering montage sequences (see section 2.5.4.1), only label the beginning and end of the montage sequence. The result of using this labeling criteria is shown in the table below.

Table 3.1: Ground truth v-scene data obtained from labeling the video data separately from the audio data.

Film	V-Scenes
Sense and Sensibility	57
Four weddings and a funeral	61
Pulp Fiction	40

3.6.2 Results

In this section, we discuss the v-scene change detector results. First, we discuss the parameters that we need to set. For detecting video coherence, we set the attention span to be 8 sec. (in accordance with our labeling rule) and the size of the memory is set to 24 sec. Note that the attention span is the most recent data in memory; hence in this case this case, the rest of memory is 16 sec. long. In general, increasing the memory size reduces false alarms, but increases misses.

In evaluating our results, we shall compare the detected v-scenes against the total number of shots in the film, since they are all candidate v-scene change points. Secondly, it is important to note that we are dealing with an asymmetric two-class problem (scene change vs. non-scene change) where the number of ground truth scene change locations is typically less than 10% of the total number of shots. Hence it is important to correctly reject non-scene change points in addition to correctly detecting the scene change points.

We now need to define precision and recall, two standard measures of performance, that are primarily used in information retrieval. Precision: $\text{hits} / (\text{hits} + \text{false alarms})$, Recall: $\text{hits} / (\text{hits} + \text{misses})$, where hits, misses and false alarms refer to the detector performance.

The precision and recall metrics, focus on the performance of the algorithm with respect to the scene change locations only. They does not illustrate well, the performance of an algorithm when dealing with an asymmetric class distribution. Hence we present the

entire confusion matrix (i.e. hits, misses, false alarms and correct rejection), in addition to presenting the precision and recall. Correct rejection in this case refers to those shot boundary locations that were not v-scene change locations and were correctly labeled as such.

We now present results for v-scene detection in **Table 3.2**. These results are for the entire duration of the film (each film is one hour long) and for transitions between all types of scenes (see **Table 2.2** for definition of each type of scene). We use the following notation: H: hits, M: misses, FA: false alarms, CR: correct rejection. Shots is just the number of shots detected by the shot detection algorithm. NCL: non-scene change locations; this is just the number of shots less the number of ground truth scene change locations.

Table 3.2: Video scene detection results. In order: hits, misses, false alarms, correct rejection, number of shots, number of non-scene change locations, precision and recall.

Film	H	M	FA	CR	Shots	NCL	Precision	Recall
Sense and Sensibility	52	5	22	563	642	585	0.70	0.91
Pulp Fiction	38	6	20	414	478	434	0.64	0.86
Four Weddings and a Funeral	49	12	41	634	736	675	0.55	0.80

The result shows that the v-scene detector works well. The recall for the v-scene detector varies between 80~91% while the precision varies between 55 ~ 70%. Note that the correct rejection is excellent — around 95% across all cases. One important source of error is due shot detector misses that causes the v-scenes to be missed since the wrong

key-frame is present in the buffer. Other sources of error for computable scene segmentation are discussed later, in section 5.4.1.1.

3.6.3 Comparison with related work

While the thrust of this work is to come up with a joint audio-visual model for computable scene segmentation, it is interesting to compare our v-scene segmentation algorithm with prior work on video scene segmentation.

- Prior work on video scene segmentation was focused on detecting semantic scenes, and not computable scenes. Hence the ground truth labeling criteria and the resulting segment boundaries will differ from a c-scene labeling of the data.
- There has been prior work done in video scene segmentation using only visual features [102] , [44] . There, the authors focus on detecting scene boundaries for sitcoms (and other TV shows). However, since we expect the v-scenes in sitcoms to be mostly long, and coherent, we expect our visual detector to perform well.
- We have introduced two novel ideas in our approach:
 - The use of production models in creating the segmentation algorithm.
 - The use of higher forms of knowledge, in our case shot-level structure information, to improve the resulting segmentation.

3.7 New conceptual memory models

In the upcoming sections, we outline several new theoretical models for the memory-centric v-scene segmentation. We begin by showing how we can improve the basic

coherence model by introducing the idea of self-coherence. Then, we show two models based on the idea of Kolmogorov complexity [21] [51] (ref. appendix 13.1).

3.7.1 Improving the basic model

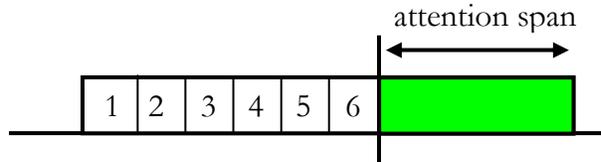


Figure 3.3: Let us assume that the attention span contains a single shot, while the rest of the memory contains six different shots.

We begin by pointing out a conceptual limitation of the coherence-based model that we have used in our work (equation (3.2)).

Let us assume that we have a situation where the attention span contains a single shot while the rest of the memory contains six distinct shots. Then if the recall between the shot and all the shots in the rest of the memory is *identical*, then a random permutation of the shots in the rest of the memory will not change the coherence value (equation (3.2)). This absence of a difference is surprising given that all the images are distinct.

When does this occur? If the images are represented by n bin histograms, then the space of all histograms is simply represented by the following equations:

$$\begin{aligned} \sum_{i=1}^n b_i &= 1 \\ 0 \leq b_i &\leq 1 \quad \forall i, \end{aligned} \tag{3.3}$$

where, h_i is the value of a particular histogram bin and n is the number of bins. This is just a hyperplane in a finite n dimensional space. Let us now define a hypersphere s of radius r , centered at \vec{h}_o . Then, when $n=2$ (i.e. the two bin histogram case), the hypersphere will intersect the line at two points, when $n=3$, the sphere will intersect the plane in a circle and so on. These points of intersection are distinct histograms that represent distinct images.

While the case of the distance between the shots in the attention span with the shots in the rest of the memory being *exactly* equal is unlikely, in practice, we do have the situation of low *dynamic range* of the L^1 color histogram distance. This can happen when shots in the attention span and the rest of the memory are from two different scenes, but both scenes have low-contrast (this can happen in dimly lit scenes). Thus, we have case where there will be no detectable coherence minima.

The problem with the current coherence formulation then is that it does not take into account the *self-coherence* of the shots in the attention span i.e. the relationship amongst the shot-lets within a scene. The self-coherence of a buffer of shot-lets is defined as follows:

$$\begin{aligned}
 r(i, j) &= (1 - d(i, j)) \delta^2, \\
 \mu_r &= \frac{1}{N(N-1)} \sum_{i, j \in B, i \neq j} r(i, j), \\
 C_{\text{self}}(B) &\triangleq \frac{1}{N(N-1)} \sum_{i, j \in B, i \neq j} (r(i, j) - \mu_r)^2,
 \end{aligned} \tag{3.4}$$

where, i, j are any two shot-lets in the buffer, r is the recall between any two shot-lets, δ is the duration of each shot-let, d is the metric defined over the images, μ_r is the mean of all the pair-wise recalls, $N(N-1)$ is just the number of distinct pairs, and where N is the number of shot-lets in the buffer. Note that since we normally use $\delta=1$ sec. in our work, C_{self} then just becomes the variance of the pair-wise distances.

Note that in equation (3.4), the formula for recall does *not* include the temporal separation between the shot-lets as a weighting factor. The reason for this is that in computing C_{self} , we are really interested in computing the spread of the pair-wise image distances in the buffer; hence the use of the temporal weighting factor here seems artificial.

Now, the modified coherence across two buffers A and B, is readily obtained as follows:

$$C(t_0) \triangleq \frac{C_{\text{mutual}}(A, B)}{C_{\text{self}}(A) + C_{\text{self}}(B)}, \quad (3.5)$$

where, the C_{mutual} is the coherence as defined in equation (3.2), and where the self coherence C_{self} is defined in equation (3.4) and where t_0 is the time coordinate of the boundary between the two buffers. Note the striking resemblance to the familiar Fischer discriminant [24]. The coherence is detected as before, using coherence minima detection, in section 3.5. We now present a comparison using synthetic data between the new formulation of coherence and the older model, in the next section.

3.7.1.1 *Experiments using the new model*

We conducted experiments on artificial data, using the new model presented here (equation (3.5)), and compared the performance against the model (equation (3.2)) used in our experiments on v-scene segmentation.

We modeled a shot to be a sequence of frames that lasted between 2sec. to 5sec. in duration. Each “frame” was represented using a real number between 0 and 255. Note that it is not important to represent a frame using a matrix, since we are only interested in the inter-frame dissimilarity — a real number. The dissimilarity between the frames was computed using the absolute difference metric. The frames within a shot have the same value. This stems from our notion that all the shot-lets of the shot are associated with the same key-frame.

We modeled a computable v-scene to a sequence of shots that could last between 15sec. to 45 sec. in duration. The scene has a mean frame value μ and standard deviation of the frames within a scene is σ (the same value of σ is used for all scenes). Then, the shots within the scene are generated using a Gaussian distribution i.e. we generate a single number using the distribution and assign it to a shot. All the shot-lets of this shot will have the same frame value. The durations of the shots and the durations of the scenes were distributed uniformly over their range. The final parameter of interest is the inter scene dissimilarity — Δ .

Now, the synthetic dataset comprising 1000 scenes, with fixed σ and Δ , was generated in the following way.

1. Pick a mean value (μ) at random, for the first scene.
2. Determine the scene duration, using a uniform distribution over $[15,45]$ sec. i.e. scenes can lie uniformly between 15 and 45 sec.
3. Now, for the scene, generate shots:
 - a. Determine the shot duration using a uniform distribution over $[2,5]$ sec. i.e. shots can lie uniformly between two and five sec.
 - b. For each shot, generate a frame value using a Gaussian distribution with parameters μ and σ . Assign all the shot-lets within the shot, the same frame value.
 - c. Keep generating shots, till the sum of the durations of the shots equals the duration of the scene.
 - d. Now generate the mean value for the next scene using a random walk and the parameter Δ . i.e. the mean value of the next scene is either $\mu+\Delta$ or $\mu-\Delta$, with equal probability. At the boundaries of the range for μ , (note that $\mu \in [0,255]$), we just “reflect” the values over the boundaries. For example, if $\mu+\Delta = 260$, with $\mu = 220$ and $\Delta = 40$, then the reflected value around 255, would be 250.

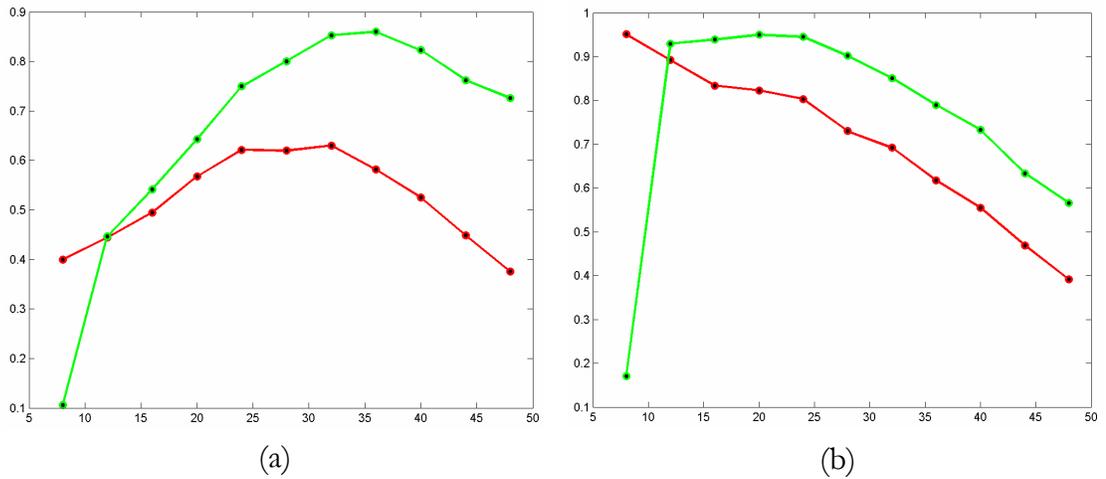


Figure 3.4: The figures shows the precision vs. memory size (a) and recall vs. memory size for the two models. The green curve represents the coherence values from the new model, while the red curve represents the coherence values from the old model. The memory size is varied from 8sec. to 48 sec. in increments of 4 sec.

- e. Repeat steps 2-4 until we have generated 1000 scenes.

Now, with the artificially generated scene, and using the L^1 metric between shots we computed the coherence values using the old and the new models presented in the previous section. We computed the precision and recall for the scene detection in both cases, by varying the size of the memory. The memory lies between 8 and 48 sec. and is incremented by 4 sec. The attention span is constrained to half the size of the memory.

We plot the results in **Figure 3.4**. The figures indicate the new model is an improvement over the old model. The precision and recall values for the new model show an improvement of about 15% on the average, over the old model.

Interestingly, precision and recall fall alarmingly when the size of the memory buffer is set at 8sec. Why does this happen? The reason that this happens is because the self coherence calculation (ref. equation (3.4)) becomes unreliable as there are too few points in the attention span and the rest of the memory (just four points each). Now since the shot sizes can lie between 2 and 5 seconds, if the attention span (or the rest of the memory) lies entirely in one shot, the self coherence of the memory segment will be zero, thus pushing the true coherence minima to a large value. Note that the self coherence is just the variance of the distances in the memory.

3.7.2 Complexity based models

The basic coherence model (ref. Section 3.4) and the improved model (see section 3.7.1) are both predicated on a simple idea — is the data viewed in the past related to the present data? It computes the segment boundaries using coherence minima, where the measure of “closeness” between two images is their metric distance.

A more basic view of the memory model is as follows: does the past data contain information that allows us to *predict* the present data more easily? If it does not, then the two segments must not be related. Rather than using entropy and mutual information, quantities that are natural for stochastic processes, we shall use the idea of Kolmogorov complexity [21] [51] (see section 13.1, for a brief review). We do this for two reasons:

- Kolmogorov complexity is a measure of complexity defined for deterministic sequences.

- Since the images in the buffer are natural images, it is unintuitive to impose an artificial generative model on these images.

It is also an elegant result of Kolmogorov complexity theory, that the expectation of the Kolmogorov complexity of a stochastic process is just the entropy rate of the process [21] [51]. Thus, Kolmogorov complexity theory agrees with the well known Shannon result on compressibility of a stochastic source. We now introduce two different approaches to modeling coherence, based on the idea of Kolmogorov complexity.

3.7.2.1 Coherence formulated using Kolmogorov complexity

We now need to define a few preliminaries. Let us consider a finite memory with the attention span defined as buffer B and with buffer A representing the rest of the memory. Let the boundary between the two buffers be located at t_0 . Analogous¹² to mutual information we need the idea of algorithmic mutual information [51]:

$$I_K(x : y) \triangleq K(x) - K(x | y), \quad (3.6)$$

where, x and y are two deterministic images, $K(x)$ is the Kolmogorov complexity of the image x (ref. section 8.2.2.2 on estimating visual complexity) and $K(x|y)$ is the

¹² However, unlike the mutual information that is defined using entropy, algorithmic mutual information is asymmetric [51].

conditional Kolmogorov complexity of image x when given image y (ref. appendix 13.1.1, on how to generate an estimate).

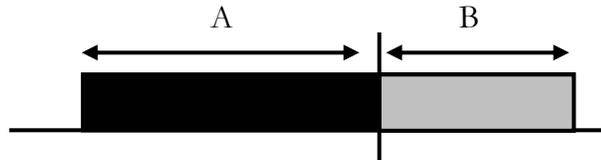


Figure 3.5: Memory with two buffers: A and B. B is just the attention span, while A constitutes the rest of the memory.

Now, let there be N_A shots in buffer A, where each shot is represented by a single key frame. We assume that each shot lasts a finite duration. We make similar assumptions about buffer B. We additionally assume that each frame is uncompressed. Now, the algorithmic mutual information of a frame in buffer B, with respect to the *entire* buffer A is defined as follows:

$$I_K(s_i^B : A) = K(s_i^B) - K(s_i^B | A), \quad (3.7)$$

where, $K(s_i^B | A)$ is just the conditional Kolmogorov complexity of the i^{th} frame in buffer B with respect to all the frames in buffer A (ref. appendix 13.1.2). Now the coherence C at t_0 is easily determined as follows:

$$\begin{aligned}
D(A) &= \alpha \sum_{i=1}^{N_A} K(s_i^A) t_i^A, \\
D(B) &= \alpha \sum_{i=1}^{N_B} K(s_i^B) t_i^B, \\
C(t_o) &= \frac{\alpha \sum_{i=1}^{N_B} I_K(s_i^B : A) t_i^B}{D(A) + D(B)},
\end{aligned} \tag{3.8}$$

where $D(A)$ and $D(B)$ are the amount in bits to represent each buffer A and B, s_i^A , s_i^B represent the shots in the two buffers and where t_i^A , t_i^B represent the durations of those shots. $K(s_i^B)$ and $K(s_i^A)$ are the Kolmogorov complexities of the shots respectively. And α represents the bitrate (i.e. it is the product of the frame-rate, the number of pixels per frame and the number of bits per pixel). It is easy to see that the scene boundaries are located at the coherence minima.

3.7.2.2 Coherence formulation with perceptual distortion

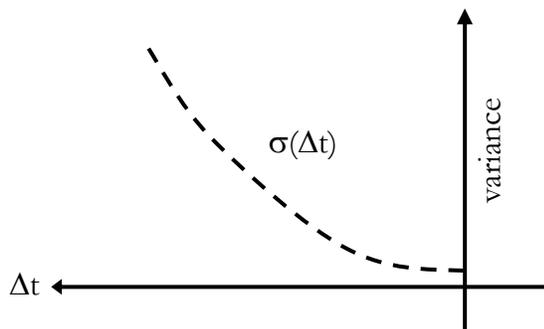


Figure 3.6: The variance of the blur function increases with distance Δt from the boundary between the two buffers A and B.

The coherence formulation on the previous section deals with the idea of information loss, when the present data is compared with the past data. However, while computing the estimates of Kolmogorov complexity, we use “perfect” representations of the images stored in the memory. However, this is non-intuitive — data in short-term memory fades with time.

One way to achieve this fading effect is to blur the past data with a two-dimensional, symmetric Gaussian kernel, whose variance changes with time (see **Figure 3.6**).

Formally:

$$\begin{aligned} \sigma(\Delta t) &= \beta \exp(\alpha \Delta t) & \Delta t \geq 0, \\ b(x, y, \Delta t) &= \frac{1}{\sqrt{2\pi\sigma^2(\Delta t)}} \exp\left(-\left(\frac{x^2 + y^2}{2\sigma^2(\Delta t)}\right)\right), \end{aligned} \quad (3.9)$$

where Δt is the distance from the boundary, σ is the variance in pixels, α and β are constants, and where b is the Gaussian kernel that will blur the images in the memory. Now, we blur all the shots in the rest of memory (i.e. all shot-lets except those in the attention span) with the Gaussian kernel, as follows:

$$s'_i{}^A = s_i{}^A * b(\Delta t_i{}^A), \quad (3.10)$$

where, s' is the modified shot-let, s is the original shot-let and Δt is the distance of this shot-let from the boundary t_b , and where $*$ is the familiar convolution operator. Then, the coherence is readily computed using the coherence formulation similar to (3.8):

$$\begin{aligned}
D(A) &= \alpha \sum_{i=1}^{N_A} K(s_i^A) t_i^A, \\
D(B) &= \alpha \sum_{i=1}^{N_B} K(s_i^B) t_i^B, \\
C(t_o) &= \frac{\alpha \sum_{i=1}^{N_B} I_K(s_i^B : A) t_i^B}{D(A) + D(B)},
\end{aligned} \tag{3.11}$$

where the significant change has been the use of the blurred shot-lets s_i^A . Then, as in the previous section, the v-scene boundaries are located at the coherence minima location.

3.8 Summary

In this chapter, we have discussed our approach to detecting computable video scenes. There were three key components to our work: (a) a memory model, (b) recall and (c) coherence. We defined recall to be a measure of inter shot similarity, that incorporated the shot durations, their temporal separation, in addition to their metric distance. Coherence was computed using the pair-wise recall of all shots in the buffer, across the time boundary where we wanted to compute recall. We modified the original coherence formulation using shot-lets (shot fragments) since the original formulation could lead to misses in the coherence minima. Our experimental results indicate that our algorithm works well, however there is room for improvement.

We additionally presented three new theoretical models for v-scene detection. The first model improved upon the original model by defining a new function on the shots —

self-coherence. The next two models began by reformulating the problem as one of successfully predicting the present with information in the past data. Then, we used the idea of Kolmogorov complexity to define the algorithmic mutual information distance between the shots in the buffers. This enabled us to compute coherence under two circumstances — (a) perfect image representation and (b) perceptual distortion of the images.

4 The computable audio scene

4.1 Introduction

In this chapter we shall present our approach towards detecting computable audio scenes (a-scenes). The computable audio scene is defined to be a segment of data with long term consistency with respect to ambient sound. Note that each scene can often contain multiple overlapping audio classes (e.g. speech + environmental sounds), thus making the a-scene *segmentation* problem qualitatively different from the traditional audio scene *classification* problem, where a segment of audio data is assigned membership to a single class. The work presented in this chapter is an important aspect of the thesis since it complements the v-scene segmentation, thus helping us detect audio-visual computable scenes. We shall adopt a memory model based approach to detecting a-scenes (ref. section 2.5.2).

4.1.1 Summary of our approach

Our approach to determine audio computable scenes can be summarized as follows:

1. Compute features for the raw audio data in the attention span. The features can be of three types — (a) scalar (b) vector and (c) point data.

2. Model the scalar feature sequence as the sum of three types: (a) trend (b) sinusoids and (c) noise.
3. A scene change is postulated to exist if the *features* computed in the attention span are not correlated to features computed from the past data. Note that the memory model has two parts — the attention span that has the most recent data in memory and the rest of the memory, that is used to determine if the present differs from the past. In order to determine if this is true, we do the following:
 - a. Compute the features in the attention span.
 - b. Move the time window for the computing the features back by δ sec. and recompute the features.
 - c. Compute the *distance* of the features just computed, with the features in the attention span.
 - d. Repeat steps b and c till we have exhausted the data in the memory. We now have distance sequence, per feature. This helps us estimate the distance increase rate parameter β .
 - e. Advance the original time window by δ sec. and repeat the steps a, b, c and d, until all the audio data has been analyzed. We now have computed the parameter β , per feature at every time instant. This parameter will have a local maximum at the scene change location.
 - f. After performing some SVD based smoothing, determine local maxima, in each β sequence, for each feature.

- g. Combine the local maxima using a majority vote to determine if a scene change point exists.

The rest of the chapter is organized as follows. In the next section, we present a definition of the computable a-scene. Then, in section 4.3, we present our approach towards modeling the audio data — we compute the different features, and present new signal models for a subset of the features. In section 4.4, we shall show how to compute the dissimilarities for the features. In section 4.5, we present our segmentation algorithm. Then, in section 4.6, we present a simple, but robust approach to silence detection. We conclude the chapter by presenting experimental results in section 4.7 and the chapter summary in section 4.8.

4.2 The computable audio scene definition

In this section, we develop the computable audio scene model that is informed by the insights obtained from Computational Auditory Scene Analysis (ref. section 2.4.3). We model the scene as a collection of sound sources. We further assume that the scene is *dominated* by a few of these sources. These dominant sources are assumed to possess stationary properties that can be characterized using a few features. For example, if we are walking away from a tolling of a bell, the envelope of the energy of the sound of the bell will decay quadratically. A scene change is then said to occur when the majority of the dominant sources in the sound, change.

We define a computable audio scene to be a continuous segment of audio data that exhibits long-term consistency in ambient sound. In section 4.5.1, we shall formally quantify this intuitive idea of consistency, in terms of the local correlation decay rate of the different features. We denote such a segment to be a *computable* since it can be reliably and automatically determined using low-level features present in the audio data.

4.2.1 Relating computable scenes to semantic scenes

The segmentation algorithm developed in this work is completely general, however since the test-set is culled from films, it is useful to examine the relationship between a computable audio scene and a semantic scene. Under usual circumstances, the ambient audio is indicative of location and changes in the ambient sound are indicative of changes in location. There are several interesting cases where a change in the audio data does *not* indicate a location change and where consistent audio does *not* imply the same physical location:

- **Part of a semantic scene:** Alice is seen in her room reading a book silently. We can hear the hum of the fan. Then, she gets up and turns on the radio. Clearly, there are two computable audio scenes within the single audio-visual semantic scene. Hence, such audio scene changes correspond to events within the semantic scene.
- **Transient scenes:** Suppose that we are watching a semantic scene depicting a journey. Then, the director will show several, differing visual segments, with some well chosen ambient music. Here, the audio data isn't indicative of the locations

that are seen, but is a consequence of directorial intent. He (or she) may want to develop a particular theme (or mood) while showing the journey.

- **Mtv/Montage scenes:** These scenes are characterized by widely different visuals (differences in location, time of creation as well as lighting conditions) which create a unity of theme by manner in which they have been juxtaposed. However, they are characterized by a long-term consistency in the audio track.

4.3 Modeling the data

In order to segment data into scenes, we adopt a causal, first-in-first-out (FIFO) model of memory (ref. section 2.5.2.2, **Figure 2.4**). The parameters of this model are identical to the one used for computing v-scenes.

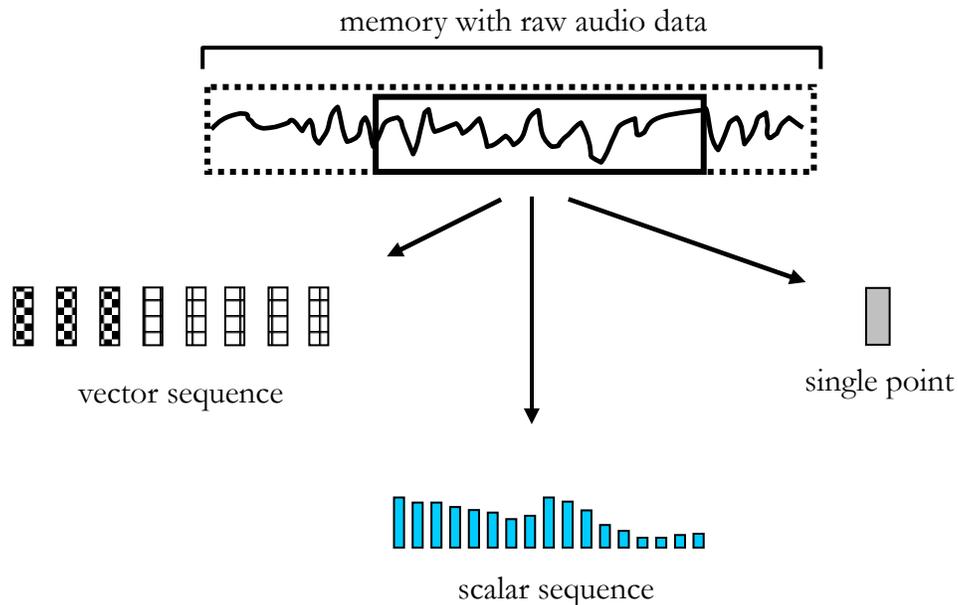


Figure 4.1: We extract three types of features from each section in memory: vector sequences, scalar sequences and single point features.

In this section we present our data representation framework. We model the audio data using three types of features: scalar sequences, vector sequences and single points (see **Figure 4.1**). Features are extracted per section of the memory, and each section is T_{as} sec. long (the length of the attention span).

The rest of this section is organized as follows. In section 4.3.1, we discuss the features used in this work,. In section 4.3.2 we show the detail signal representations for our scalar sequence feature representation. This incorporates ideas of trends, periodic elements and randomness.

4.3.1 The features

We now describe the features used in this work, grouped by the type (scalar, vector and point) of the feature.

4.3.1.1 *Scalar sequences*

The scalar sequences used in this work are as follows:

- **Zero-crossing rate:** The number of zero crossings in the raw signal, per 100ms frame. The zero crossing rate is indicative of the spectral content of the signal and is also an excellent speech/music discriminator [74] [75] .
- **Spectral Flux:** The norm of the frame to frame difference of the spectral magnitude: $\| |F_i| - |F_{i+1}| \|$. Music has a higher rate of change than speech [75] .

- **Cepstral Flux:** The norm of the difference between cepstra [75] of successive frames: $\|C_i - C_{i+1}\|$.
- **Energy:** This is simply the energy of the raw data per frame, where each frame is 100ms long.
- **Energy Sigma:** This feature measures the variance of the energy, per 1 sec. window.
- **Low energy fraction:** Computes the fraction of frames in each second of data that have energy that is less than a certain global threshold.

$$\begin{aligned}
 I(f, w) &= 1; \quad \text{if } E(f) < E(w)/2, \\
 &= 0 \quad \text{otherwise,} \\
 L_{ef}(w) &= \frac{1}{N} \sum_{i=0}^{N-1} I(f, w),
 \end{aligned} \tag{4.1}$$

where, each frame f is 20ms long and each window w is 1 sec. long. $E(f)$ is the mean energy within the frame, while $E(w)$ is the mean energy in the window w . $I(f, w)$ is just the indicator function, and L_{ef} denotes the low-energy fraction. For example, when the attention span is 10 sec. long, we would have 10 scalar values. This feature helps us determine whether a region is largely silent, with most of the energy concentrated in a small region.

Note that the features have been evaluated at different time scales. For the scalar sequences, we obtain one point per 100ms frame for the first four features, but for the last two, we obtain one point per second. Hence, for example, with a attention-span

window size of 16 sec. we will have scalar sequences of length 160 for the first four features and 16 point sequence for the last two features.

4.3.1.2 *Vector sequences*

Now, we describe the vector sequences determined in this work.

- **Multi-channel cochlear decomposition:** A ten dimensional vector per frame, it is derived from the output of a Patterson model of the cochlea. We use a ten channel filter-bank, and derive the envelope from each band-pass channel.
- **Cepstral vectors:** 18 dimensional cepstra from each frame for the duration of the analysis window.
- **Mel-frequency cepstral coefficients (mfcc):** In order to simulate the subjective spectrum we use a filter-bank spaced uniformly on the mel scale, which is a non-linear, warped frequency scale. We use 13 mfcc coefficients, per frame derived from a 40 channel filter-bank, [69] [79] .

In each of these cases, we use a frame length of 100ms.

4.3.1.3 *Scalar points*

Finally, we describe the scalar point data extracted in this work.

- **Spectral roll off point:** This is the 95th percentile of the power spectrum. It is useful in distinguishing voiced from unvoiced speech and is also indicative of the skewness of the spectral distribution [75] .

- **Zero crossing rate variance:** This is just the variance of the zero crossing rate for the entire window [74].

While the cochlear features were chosen since they model the cochlear response well, the rest of the features were chosen for their ability to distinguish between speech and music [74] [75] [30] [69]. We now discuss the scalar sequence signal model in detail.

4.3.2 The scalar sequence signal model

The scalar sequence of feature values are described by an additive model comprised of three parts: a trend, a set of periodic components and noise. Thus,

$$F(k) = t(k) + p(k) + n(k), \quad (4.2)$$

where, $F(k)$ represents the feature value at the k^{th} time instant, $t(k)$, $p(k)$ and $n(k)$ representing the value of the trend, the periodic component and the noise respectively, at time k .

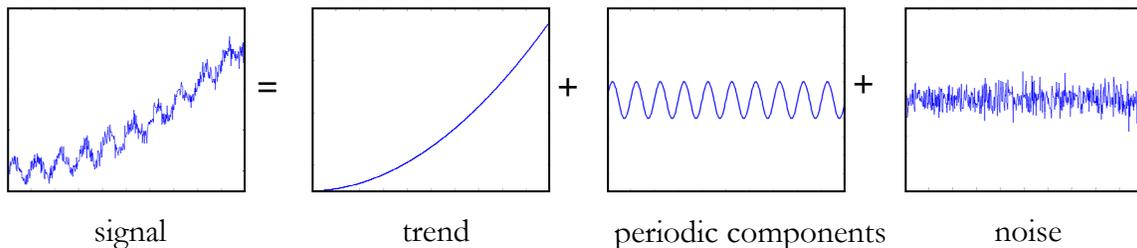


Figure 4.2: We model the scalar sequence as a sum of three components — trend, periodic components and noise.

The decomposition model of the feature sequence is intuitive. The trend is represented using a low order polynomial of maximum degree two, the periodic components are represented via a conjugate symmetric basis ([97], section 4.3.5) and the noise is not explicitly modeled. The low-order polynomial trend is a design constraint that implements Bregman's observation that source characteristics change slowly over time (ref. section 2.4.3). The noise is not explicitly modeled since we are primarily interested in computing a dissimilarity measure and not interested in reconstructing the signal.

The decomposition in equation (4.2), bears similarities to the *sines+transients+noise* model found in [97]. However, there are differences:

1. In [97], the author works on the problem of lossy signal representation of raw audio data, with a view towards scalable¹³ audio transmission and reconstruction. In our case we work on feature sequences instead of raw audio data. This changes the framework used to model the data.
2. We do not model feature transients; instead we model their graceful change. This is a direct consequence of Bregman's observations on the nature of change of source characteristics.

¹³ In terms of bit rate.

3. We focus on computing dissimilarities rather than reconstruction. As a consequence, the features that we compute (e.g. the zero crossing rate) do not allow us to perfectly reconstruct the original audio data.

4.3.3 Estimating components of the scalar sequence model

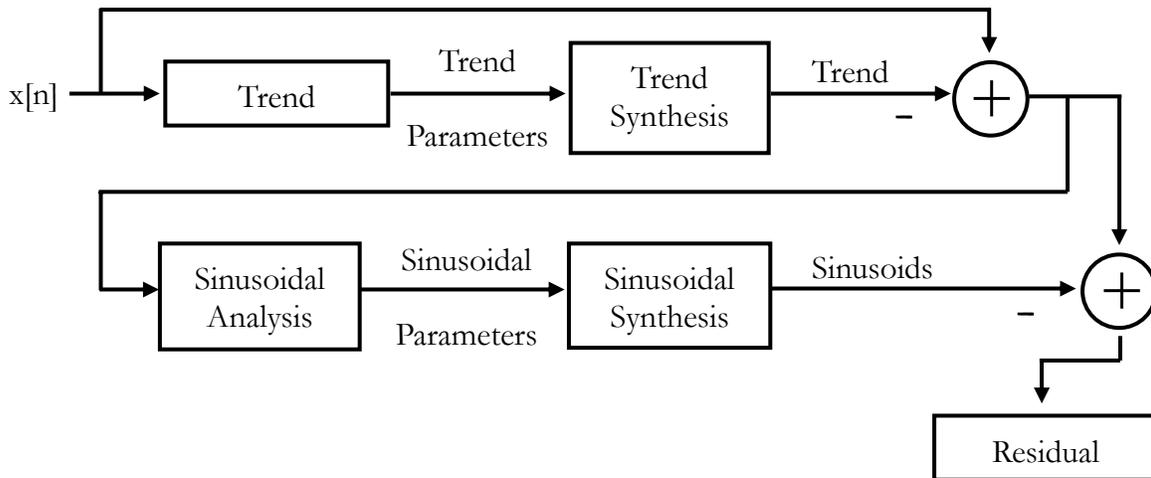


Figure 4.3: The overall computational architecture for analyzing an input scalar sequence $x[n]$. Trends are represented using low order polynomials, periodic elements are represented using a conjugate symmetric basis. The noise is not explicitly modeled.

In this section, we present an algorithm to decompose a scalar sequence signal into three components: the trend, periodic components and the residual noise. We estimate the trend using a minimum mean-square error polynomial fit, and we determine the order of the polynomial by using the minimum description length principle (MDL) [5] [36] [71]. Then, after removing this trend, we compute the sinusoidal components in the residue, using the matching pursuit algorithm [97]. The final residue is the noise. **Figure 4.3** summarizes this computational architecture.

4.3.4 Trends

The trend is a low order polynomial fit of at most degree two. The trend is obtained using a least square fit to the data and we determine the order of the polynomial using MDL. Briefly, MDL uses a two part coding scheme on the data to determine the correct model. The first part of the coding scheme, represents the model, while the second part represents the data encoded with this model. MDL attempts to minimize the total number of bits used to represent the data with a particular coding scheme¹⁴. Reference [36] is an excellent tutorial, while [71] is the classic paper on MDL.

The trend $t(n)$ has two possible representations:

$$\begin{aligned} p(n) &= a_0 + a_1n + a_2n^2 \\ e(n) &= \alpha \exp(-\beta n) \end{aligned} \tag{4.3}$$

where, $p(n)$ is the polynomial fit, n is the time index and the set $\{a_0, a_1, a_2\}$ represents the least square polynomial fit parameters. $e(n)$ represents the exponential fit to the data with parameters α and β . In order to determine the optimal model for the trend, we first compute four fits: one exponential fit and polynomial fits of degree 0, 1 and 2 respectively. In order to pick the optimal fit, we use gMDL, a mixture form of MDL [36]. This is defined as follows:

¹⁴ i.e. total number of bits = number of bits to represent the model + number of bits to represent the data encoded with the model.

$$\begin{aligned}
S &= \frac{R_{ss}}{(n-k)}, & F &= \frac{(\mathbf{y}^t \mathbf{y} - R_{ss})}{kS}, \\
gMDL &= \begin{cases} \frac{n}{2} \log S + \frac{k}{2} \log F, & F > 1, \\ \frac{n}{2} \log \left(\frac{\mathbf{y}^t \mathbf{y}}{n} \right), & \text{otherwise,} \end{cases} \quad (4.4)
\end{aligned}$$

where, R_{ss} is the residual sum of squares for the fit (i.e. the squared error when using the trend to fit the data), n is the length of sequence of data, k is the number of parameters in the fit and \mathbf{y} is the data sequence vector. The minimum value of $gMDL$ over the different models is used to choose the optimal model.

We chose to use $gMDL$ over other MDL formulations such as Bayesian Information Criterion (BIC) and Akaike's Information Criterion (AIC) since extensive experiments [36] indicate that $gMDL$ selects models better than BIC or AIC. In the next section we shall describe our approach to modeling the periodic components.

4.3.5 Periodic components

We use the idea of matching pursuits [54] to model the periodic components. This section uses analysis similar to the work in [97], where the author uses matching pursuits to model *sines* in his *sines+transients+noise* model of raw audio data. We first summarize the idea of matching pursuits and then discuss our particular algorithm.

4.3.5.1 Matching pursuits

We now present a brief overview of the matching pursuits algorithm, an iterative scheme to decompose signals using a highly redundant dictionary of vectors. Let there be M elements in a dictionary $D = \{\mathbf{e}_k\}; k \in \{0,1,..M-1\}$, where \mathbf{e}_k represent the basis vectors. All the elements are restricted to be unit norm. Let us assume that we need to model a signal \mathbf{x} . Then, with the initial residual as \mathbf{r}_0 , residual set to \mathbf{x} , we have the following at the k^{th} iteration:

$$\begin{aligned}\beta_m &= \frac{\langle e_m, r_k \rangle}{\langle e_m, e_m \rangle}, \\ \alpha_k &\triangleq \beta_i, \quad i = \arg \max_m |\beta_m|, \\ r_{k+1} &= r_k - \alpha_k e_{i_k}\end{aligned}\tag{4.5}$$

Where e_{i_k} is the dictionary element chosen in the k^{th} iteration, \langle, \rangle is the inner product operator, and where $||$ is the absolute value operator.

In equation (4.5), the definition of β_m ensures that the residual $r_k - \beta_m e_m$ is orthogonal to e_m . Clearly, the magnitude of r_{k+1} is minimized by maximizing the magnitude of α_i . Hence, at each iteration, we remove the maximum energy from the residual. It can be shown if we use this procedure, the energy of the residual converges to zero.

4.3.5.2 Subspace pursuits

Subspace pursuits are generalizations of the matching pursuits algorithm. Here, the residual is projected onto a subspace rather than a single vector. This is highly computationally intensive unless the subspaces are highly structured. The family of complex exponentials is an example of a subspace where each subspace has the vector and its conjugate. Then, if we use this subspace, the subspace matching pursuit problem resembles the analysis by synthesis sinusoidal modeling problem [97].

The dictionary element is defined as follows:

$$e_m[n] = \frac{1}{N} \exp\left(\frac{j2\pi mn}{M}\right), n = 0, 1, \dots, N-1, m = 0, 1, \dots, M, \quad (4.6)$$

where, N is the length of data sequence and M is the dictionary size. In our current implementation, we set the size of the dictionary is to be $4N$.

Then, with $\alpha_k = b_k \exp(j\phi)$ as the largest correlation coefficient, the last line of equation (4.5) is now modified as:

$$\begin{aligned} r_{k+1}[n] &= r_k[n] - b_k e_{i_k} - b_k^* e_{i_k}^*, \\ &= r_k[n] + \frac{2b_k}{N} \cos\left(2\pi \frac{i_k}{M} n + \varphi_k\right), \end{aligned} \quad (4.7)$$

where the asterix operator $*$ represents a complex conjugate. There are *two* terms at the end of the first line of equation (4.7), since we are projecting onto a subspace with two vectors — the complex exponential and its complex conjugate.

Now, the signal can be reconstructed as:

$$\begin{aligned}\hat{x}[n] &= \sum_{k=0}^{K-1} b_k e_{i_k}[n] + b_k^* e_{i_k}^*[n], \\ &= \sum_{k=0}^{K-1} \frac{2b_k}{N} \cos\left(2\pi \frac{i_k}{M} n + \varphi_k\right),\end{aligned}\tag{4.8}$$

Where, K is the number of components in our periodic model. Hence, the parameter set of amplitude, frequency and phase triples (we denote each triple as a phasor — see appendix 13.2) can be stored and used for discriminating between two sets of periodic components:

$$\left\{ a_k = \frac{2b_k}{N}, f_k = \frac{i_k}{M}, \theta_k = \varphi_k \right\}, k \in \{0, 1, \dots, K-1\}\tag{4.9}$$

We need a stopping criteria to determine K , the number of components in our periodic model. Selecting K on using an energy criteria (e.g. stop after residual energy is less than 1% of the original signal energy) has two implications: (a) the number K is variable and changes for each segment (b) the variance of the noise is now fixed. Instead we fix the number K , thus selecting the K components with the largest energies. In [97], the author derives a fast implementation of the subspace matching pursuit problem using the DFT. We have used that implementation in our work.

4.3.6 Noise

The noise is not explicitly modeled since we are not interested in transmitting and subsequently reconstructing the original feature sequence. Here, we are only interested in computing the dissimilarity between two noise vectors.

The order in which we compute the elements of the scalar signal sequence, makes a difference since the sinusoidal basis is an over-complete basis. In particular, if we first decompose the data into sinusoids, we will not be able to detect the trend since it will have been represented by the sinusoids.

4.4 Computing the feature dissimilarities

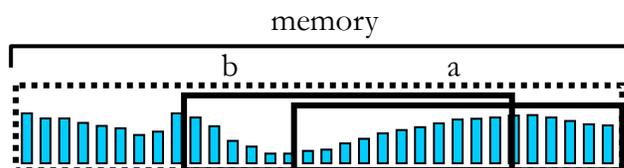


Figure 4.4: The features are extracted from two different sections of memory, a and b and then compared.

The features to be compared are extracted from different sections of the memory and then compared. **Figure 4.4** shows two sections a and b , from which we extract the features.

Each feature used in this work falls into one of three data types: scalar, vector and points. In this section, we shall derive the dissimilarities for the each of the three data types. Note that the scalar signals are further decomposed into three sub-types: trends,

periodic components and noise. Each subtype has its own dissimilarity measure associated with it. We begin our discussion with the scalar sequence.

4.4.1 The scalar case

In order to compute the distance between two scalar sequences, we decompose the sequence into three components: the trend, periodic components and the noise data, and then compute the dissimilarity for each subcomponent.

4.4.1.1 Trends

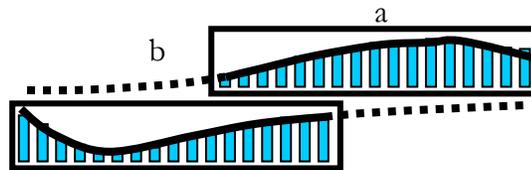


Figure 4.5: The two trends from sections a and b are extrapolated and then compared.

The trend is a low-order polynomial of maximum degree two. The polynomial degree of the trend is determined using MDL (ref. Section 4.3.4) and is hence variable. Let us compute the distance between two trends a and b , where a is defined over time-interval $[0, t_a]$ and b is defined over time-interval $[-\delta, t_b]$ and where $\delta > 0$ and $t_b < t_a$. Note, $|t_b + \delta| = t_a = T_{as}$, where T_{as} is the length of the attention span. The distance between two trends in memory is computed using a predictor based model. Hence, we extrapolate both trends such that both a and b are defined over $[-\delta, t_a]$. Then, the dissimilarity is defined as:

$$d_i(a,b) = \sum_{i=1}^N (a[i] - b[i])^2 \quad (4.10)$$

where, $a[i]$ and $b[i]$ are the discrete time samples¹⁵ of the two trends and where N is the number of time samples in the extrapolated trends.

An alternative formulation would be to compare the extrapolated trends to the original data, instead of comparing the two trends, as we do now. This is not a good idea since the original data contains all three elements of the signal model (trends + periodic components and noise).

4.4.1.2 Periodic components

The periodic components of the signal model have been derived using the *sines* model in [97]. As mentioned in section 4.4.1.2, we represent the signal using a set of k phasors. Each phasor is an ordered triple consisting of the amplitude, frequency and initial phase of the phasor. Now we wish to determine the distance between the corresponding periodic representations of two scalar signals, a and b . Let the two corresponding phasor sets be A and B . Then:

¹⁵ A feature value is generated every 100ms.

$$\begin{aligned}
 d(A_k, B) &= \min_i \left\{ D(A_k, B_i) \cdot \left(1 + \frac{|\omega_1 - \omega_2|}{\max(\omega_1, \omega_2)} \right) \right\}, \\
 d_b(A, B) &= \operatorname{median}_k \{ d(A_k, B) \}, \\
 d_p(A, B) &\triangleq \frac{(d_b(A, B) + d_b(B, A))}{2},
 \end{aligned} \tag{4.11}$$

where, A_k and B_i refer to the k^{th} and i^{th} phasors of sets A and B respectively. $D(A_k, B_i)$ is the phasor distance (ref. appendix 13.2), d_p is the distance between the two phasor sets and d_b is the modified Hausdorff distance¹⁶. Note that d_b is an asymmetric distance.

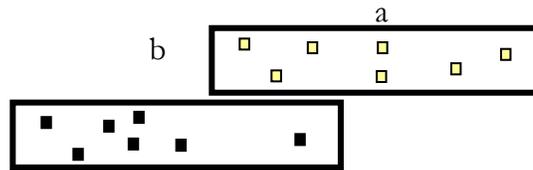


Figure 4.6: The solid blocks in the two sections represent phasors. Each phasor is an ordered triple: $\{a, \omega, \theta\}$ representing the amplitude, frequency and the initial phase of the periodic component.

4.4.1.3 Noise

Given two sets of data a and b with noise-like distributions, we adopt the following procedure to compute the distance between the two distributions.

1. Compute the standard deviations for each set: σ_a and σ_b .

¹⁶ The original Hausdorff distance is defined using the max function. However, this is sensitive to outliers.

2. Determine histogram centers for each set: $C_a = [-3\sigma_a + \Delta_a, \dots, 3\sigma_a - \Delta_a]$ and $C_b = [-3\sigma_b + \Delta_b, \dots, 3\sigma_b - \Delta_b]$, where $\Delta_a = 3\sigma_a/N$ and $\Delta_b = 3\sigma_b/N$. and where the centers are spaced by Δ_a and Δ_b apart respectively. This gives us $2N-1$ centers. Using 3σ for determining the extent (i.e. the domain) of the histogram since for an arbitrary zero mean distribution: $P(|x| \geq 3\sigma) \leq 1/9$ (from Chebyshev's inequality).
3. Quantize sets a and b using **both** centers C_a and C_b . This way we get four histograms: $H_{C_a}(a), H_{C_b}(a), H_{C_a}(b), H_{C_b}(b)$. The notation $H_{C_b}(a)$ refers to the distribution obtained by quantizing the set a with centers from C_b . By using the centers of one distribution to quantize the other, we can easily determine the extent to which the two distributions are “matched.” The histograms are then normalized to generate probability distributions.
4. We now use the Kullback-Liebler divergence measure [21] which is a measure of relative entropy between two distributions.

$$D_{KL}(p \parallel q) \triangleq \sum_{x \in X} p(x) \log \frac{p(x)}{q(x)} \quad (4.12)$$

where, p and q are two distributions defined over the set \mathbf{X} . This function is readily shown to be non-negative and is zero if and only if the two distributions are identical. One way to look at equation (4.12) is the following — it measures

the penalty accrued in bits, when attempting to encode a random variable with true prior pdf $p(x)$, with a different pdf $q(x)$.

Now, the distance between the two sets a and b is readily defined as:

$$\begin{aligned} d_1 &= D_{\text{KL}}(H_{C_a}(a) \| H_{C_a}(b)), \\ d_2 &= D_{\text{KL}}(H_{C_b}(b) \| H_{C_b}(a)), \\ d_n(a, b) &\triangleq \left(\frac{d_1 + d_2}{2} \right). \end{aligned} \quad (4.13)$$

The distances d_1 and d_2 are computing the mismatch between distributions. The advantages of using this distance lies in the fact that it makes no assumptions on the model generating the data. Relative entropy is also used as the standard divergence measure between distributions in Information Theory [21] . At this point we have discussed the computation of the scalar sequence dissimilarity, by computing the distances amongst the components of the scalar sequence.

4.4.2 Vectors

The distance between two vector sequences is computed as follows. Let us assume that we have two sequences of vector valued data \mathbf{X} and \mathbf{Y} with M elements each and where each component is n dimensional. (ref. **Figure 4.7**). Then, instead of computing an element by element Euclidean distance, we compute a novel circular distance. This is done in the following way:

$$d_k \triangleq \frac{1}{M} \sum_{i=0}^{M-1} d_f(x_i, y_{\text{mod}(i+k, M)}),$$

$$d_v = \min_k \{d_k\},$$
(4.14)

where, d_k is the k^{th} circularly shifted distance ($k \in [0, \dots, M-1]$), d_f is the feature distance¹⁷ and between the elements of the two vectors and d_v is the distance between the two sequences. This circular computation has several advantages:

- If the two vectors have a common sub-sequence, then, by using the circular shift,

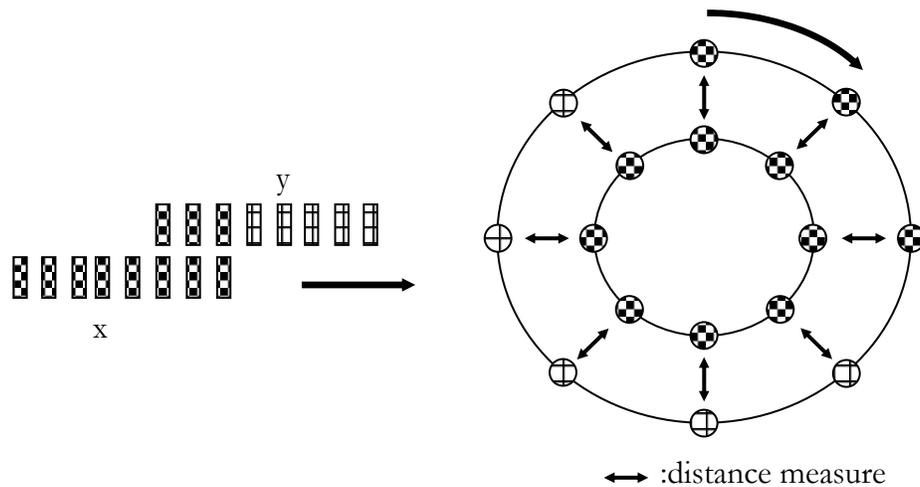


Figure 4.7: We compute a circular distance between two equal length vector sequences. The distance is computed by (a) arranging both sequences in a circle (b) rotating the outward sequence by one element each time and computing the average pair-wise distance between the corresponding elements. (c) the minimum of these rotated distances is the circular distance.

¹⁷ The distance function is different for different features.

we'd have found that sub-sequence in this formulation, thereby decreasing the distance.

- Even if two sub-sequences have been aligned using the circular shift, there is a penalty for misalignment in the rest of the sequence. Had we performed a linear shift instead of a circular shift, then we'd have to assign arbitrary penalty over the sections of the sequence that do not overlap.
- There is relative time ordering amongst the elements of the vector sequence. This would not be possible had we treated the vector time sequence as a point set and then used the Hausdorff metric to determine the distance as the distance between two point sets.

4.4.3 Point data

The point data (e.g. variance of the zero crossing rate) is a single value per segment (i.e. attention span). This is in contrast to the scalar and the vector variables generate a time sequence of values, per section. Hence the distance between two point values a and b is simply:

$$d_p(a, b) = |a - b| \quad (4.15)$$

Over the previous three sections, we have discussed the three types of features used in our framework, component estimation in the case of scalar sequences and the overall technique for computing dissimilarities for all three types.

4.5 The segmentation algorithm

We now present our segmentation algorithm. We begin with a discussion of the idea of using correlation of the present data with the past, to determine a distance measure between two segments. Then in section 4.5.2, we shall estimate the rate of increase of this correlation distance to determine the presence of a segment boundary. In section 4.5.3, we introduce the idea of a principal sequence that uses eigenvalue analysis to determine scene change locations. We conclude with a section on combining the results of the eigenvalue analysis.

4.5.1 Determining correlations from memory

The intuition behind using correlation between segments for audio scene segmentation is

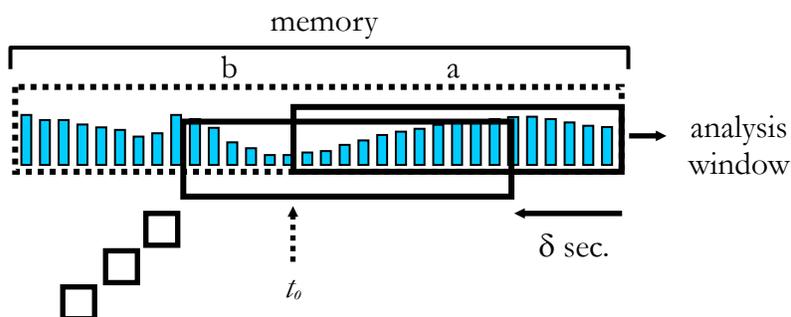


Figure 4.8: We compute each feature within the attention span. Then, we shift back the analysis window by $\delta \text{ sec.}$ and re-compute all the features. We proceed this way till all the memory has been covered. Then we compute the distances between the feature values in the attention span with the rest of the segments.

as follows: if there is a true segment boundary to the left of t_0 , we would expect a measure of correlation between segments to drop rapidly to the left of the segment

boundary. This is because we would expect that adjacent dissimilar scenes to have low correlation.

We determine correlations within memory anchored at t_0 , for each type of feature, using the following procedure (we set the analysis window size to be the same as that of the attention span):

1. Place the analysis window over the attention span and determine the feature values (see **Figure 4.8**).
2. Shift the analysis window back by δ sec. and re-compute the features.
3. Now, compute the distance between these two feature sequences using the distance measure corresponding to each feature type (ref. section 4.4)
4. Repeat steps 2-3 till we have covered the entire memory.

Then, we move the memory from t_0 to $t_0 + \delta$ and repeat steps 1-4. This is repeated until we have exhausted all of the data.

At the end of this procedure, we have a sequence of distance values for each feature, at discrete time intervals of δ i.e. at $t \in \{t_0 + p \delta\}$, where p is an integer. If a scene change was located at t_0 , to the immediate left of the attention span (ref. **Figure 4.8**), we would intuitively expect the distance values to increase rapidly (i.e. the correlation values to drop rapidly) as the data ought to be dissimilar across scenes.

4.5.2 Estimating the distance increase rate

We estimate the rate of increase if the distance to be a measure of the decay in the correlation across segments. Now from the previous section we have the distance sequence d_i , for every feature. We model the distance sequence as follows:

$$y(t) = \alpha + \beta t + \gamma t^2; t \in [0, -1, \dots, -N + 1],$$

$$\beta = \left. \frac{dy}{dt} \right|_{t=0}, \quad (4.16)$$

where $y(t)$ is the polynomial modeling the data sequence. $\therefore \beta$ is the rate of *increase* of the distance at time $t = 0$ and we associate this value at each discrete time instant $t \in \{t_0 + p\delta\}$. Hence, at the end of this analysis, we would have generated a time sequence of distance increase rate estimates β_i , for *each* feature i ¹⁸. Then, the local maxima of the distance increase rate sequence is the location of the segmentation boundary. This is because β can only reach a local maxima, when the correlation between the adjacent segments is low.

¹⁸ Note that each scalar sequence has three components: trend, periodic components and noise. A distance increase rate estimate is generated for each subcomponent.

4.5.3 β sequence analysis

We now present the algorithm to analyze the time sequence of the distance increase rate (i.e. β) estimates. Assume that we have M distance increase rate time sequences, one from each feature. Each time sequence is N points long¹⁹. Then we adopt the following procedure:

1. Define \mathbf{d}_n to be a row vector containing the n^{th} time sequence; $n \in \{0, \dots, M-1\}$.
2. Construct an M by N matrix \mathbf{P} such that the k^{th} row of \mathbf{P} is \mathbf{d}_k .
3. Then, using Singular Value Decomposition (SVD) [83], we can decompose \mathbf{P} in the following way:

$$\mathbf{P} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^t \quad (4.17)$$

where, t represents the transpose operator, the columns of \mathbf{U} (M by M matrix) are the eigenvectors of $\mathbf{P}\mathbf{P}^t$, the columns of \mathbf{V} (N by N) are the eigenvectors of $\mathbf{P}^t\mathbf{P}$, the r singular values on the diagonal of $\mathbf{\Lambda}$ (M by N) are the square roots of the eigenvalues of both $\mathbf{P}\mathbf{P}^t$ and $\mathbf{P}^t\mathbf{P}$.

4. Since the SVD automatically orders the eigenvalue according to magnitude, we determine the first L eigenvalues such that:

¹⁹ Since the distance increase rate for each feature is evaluated at the same set of discrete time instants $t \in \{t_0 + p \delta\}$, each sequence has the same number of points.

$$L = \arg \max_k \left(\frac{\sum_{m=0}^k \lambda_m}{\sum_{l=0}^r \lambda_l} < \alpha \right), \quad (4.18)$$

where, r is the number of singular values, and α is an energy threshold. Note, λ_0 is the largest eigenvalue.

5. Then, we compute $\hat{\mathbf{P}}$, an approximation to \mathbf{P} :

$$\hat{\mathbf{P}} = \mathbf{U}\hat{\mathbf{\Lambda}}\mathbf{V}^t, \quad (4.19)$$

where, $\hat{\mathbf{\Lambda}}$ approximates $\mathbf{\Lambda}$ by having only the top L singular values of $\mathbf{\Lambda}$, with the remaining $L-r$ singular values set to 0. By this procedure, we have projected the row vectors of \mathbf{P} onto the sub-space spanned by the L principal eigenvectors of $\mathbf{P}^t\mathbf{P}$, corresponding to the L chosen eigenvalues.

We now discuss a technique to combine the sequences within $\hat{\mathbf{P}}$.

4.5.4 Combining the β sequences

The sequences in $\hat{\mathbf{P}}$ contain the distance increase rate (β) values. Intuitively, we should expect β to take on large values at the location of the audio scene change. We will use a simple voting procedure to determine the degree of agreement amongst the sequences. Hence, for each sequence:

1. Determine the local maxima locations using $2w + 1$ as the window size.

Denote the maxima locations for the k^{th} sequence as $M_k = \{m_0, m_1, \dots, m_P\}$, where P is the number of time sequences.

2. Define O_k from the M_k as:

$$O_k(n) \triangleq \sum_{i=0}^{l_k} \delta(n - m_{i,k}) * h_p(n) \quad (4.20)$$

where, $m_{i,k}$ is the i^{th} maxima of the k^{th} sequence and h_p is the Hanning window of size p and l_k is the number of local maxima for the k^{th} sequence \mathbf{D}_k . M_k represents the audio scene change locations when only using the k^{th} subsequence. O_k represents a smoothed version of M_k ; this is intuitive and useful since there is ambiguity about the exact audio scene change location. Now, we adopt the following procedure:

3. Define $O \triangleq \sum_{k=0}^{|\mathbf{D}|-1} O_k$ where $|\mathbf{D}|$ is the cardinality of \mathbf{D} . O will now contain a

series of “mountains” corresponding to cases where the maxima of two sequences align and small “hills” when they don’t.

4. Determine the weighted²⁰ local maxima locations of O : \mathbf{M}_o . If the magnitude of O at these locations exceeds a threshold η , they are labeled as audio scene change locations.

Our decision formulation has the following advantages:

- Projecting onto the subspace of eigenvectors corresponding to dominant eigenvalues, has the effect of project of projecting onto the dominant subsequence.
- Each sequence in $\hat{\mathbf{P}}$ generates its own sequence of audio scene change locations. By smoothing (i.e. convolving with the hanning window) and *then* adding the sequences O_k to obtain O , we compensate for slight misalignments that arise in practice. These misalignments arise due to the fact that different features are sensitive to different aspects of the audio signal and it is unlikely that all features will change at exactly the same time instant.

We now describe our silence detection algorithm, an essential part of our multi-modal fusion strategy described in chapter 5.

²⁰ We just compute the weighted centroid of the “mountain.”

4.6 Silence detection

In this section we describe our approach to determine silence in the audio data. Silences become particularly useful in detecting c-scene boundaries where v-scene boundary occurs in a relatively silent section. There are two forms of silence in speech [82] : within phrase silences and between phrase silences. The within phrase silences are due to weak unvoiced sounds like /f/ or /th/ or weak voiced sounds such as /v/ or /m/. However, such silences are short usually 20~150 ms long. In [82] , the author uses a two class classifier using Gaussian models for each pause class, to come up with a threshold of 165ms. However, others have used a threshold of 647 ms [33] , for distinguishing significant pauses. In our approach, we shall be interested in silences greater than 500ms duration.

We shall use an adaptive threshold on the energy in the segment, to determine the presence of silence. This results in a simple but robust algorithm. We proceed as follows:

- Compute the mean energy in each frame, where each frame is 100ms.
- Compute a histogram of the energy values. Determine the median energy.
- Construct a new energy histogram of the frames that are less than the median energy.
- Determine the global maxima of this new histogram. The silence threshold is set to be at 3dB above this global maxima.

- Classify the frames below the threshold, as silence. Then post process the labels using a median filter.
- Since we have a duration threshold of 500ms. for significant silences, collect only those silences that meet or exceed this duration threshold.

Why do we adopt this two histogram strategy to determine the silence threshold? This is because from an empirical analysis of the silence histograms, it became apparent that this was a multi-modal distribution, with peaks at the median energy and another peak for frames with very low energy.

4.7 Experiments

In this section we describe our experiments on determining the computable audio scene. We begin by describing the labeling process, needed in order to establish the ground truth. Then we shall describe our experimental results followed by a section that compares the results against other algorithms.

4.7.1 Labeling the data

The data comprises one hour segments from three English films — *Sense and sensibility*, *Four weddings and a funeral*, and *Pulp Fiction*. We attempt to label the data into coherent segments (i.e. audio segments that exhibit long term consistency). Similar to the labeling procedure adopted for labeling the video data into computable video scenes (ref. section 3.6.1), we also set the minimum duration of an a-scene to be 8 sec.

Table 4.1: Ground truth a-scene data obtained from labeling the video data separately from the audio data.

Film	A-Scenes
Sense and Sensibility	69
Four weddings and a funeral	72
Pulp Fiction	50

Then, the labeling criteria were as follows:

- For silences greater than 8 sec. label the beginning and ends of the silence.
- 1. When encountering speech in the presence of music, label the beginning and the end of the music segment.
- Do not mark speaker changes as scene change boundaries.

The audio data was labeled independently of the video, since it became clear that the placement of the audio scene change boundary was affected by the act of watching the video. The result of using this labeling criteria is shown in **Table 4.1**.

4.7.2 Results

In this section, we discuss the a-scene change detector results. First, we discuss the parameters that we need to set. For detecting audio coherence, video we set the attention span to be 16 sec. and the size of the memory is set to 31 sec. The size of the ambiguity window is set to 5 sec. The parameter δ , which specifies the granularity at which the data

is analyzed is set to 1 sec. (i.e. we change 1 sec. of data from the memory, every time we wish to determine if a scene change has occurred.). In general, increasing the memory size reduces false alarms, but increases misses.

As with the video scene segmentation case, it is important to note that we are dealing with an asymmetric two-class problem (scene change vs. non-scene change) with few the number of ground truth scene change locations (50 ~ 70) in an hour of audio data. However, unlike the video scene case, where each shot boundary was a potential scene change boundary, there is no such “natural” potential audio scene change location.

We now need to define precision and recall, two standard measures of performance, that are primarily used in information retrieval. Precision: $\text{hits} / (\text{hits} + \text{false alarms})$, Recall: $\text{hits} / (\text{hits} + \text{misses})$, where hits, misses and false alarms refer to the detector performance. The results of our experiments are shown in **Table 4.2**. The results of the experiments indicates that the algorithm operated at low precision ($\sim 20\%$) and with high recall ($\sim 85\%$).

Table 4.2: Audio scene detection results. In order: hits, misses, false alarms, precision and recall.

Film	H	M	FA	Precision	Recall
Sense and Sensibility	59	10	198	0.22	0.91
Pulp Fiction	43	7	204	0.18	0.86
Four Weddings and a Funeral	55	17	198	0.22	0.80

Why did we focus on the high-recall and low-precision case? Clearly, we could have adjusted the parameters such that the precision was higher at the expense of recall.

There were several reasons:

- Computable a-scene false alarms do *not* impact the c-scene results drastically (see **Table 5.3**). There are several reasons:
 - Computable scenes have end synchronization requirements with the on the audio and the video computable scenes. A c-scene false alarm can occur *only if* we have a-scene false alarm *synchronized* with a detected v-scene.
 - Since the precision of the v-scene detector is much higher (see **Table 3.2**) than the a-scene precision, the c-scene precision is essentially limited by the precision of the v-scene detector. Hence, in order to decrease the number of c-scene misses, we need to have high a-scene recall.
 - Additionally, many a-scene false alarms will get eliminated due to the higher grouping rules for fusion (ref. chapter 5).
 - Even though we seemed to have signal level “coherence” requirement of the labeler, it is clear that the labeler seemed to have grouped some of the data at a semantic level. Many of the false-alarms make sense at the signal level, but were overlooked by the labeler.

Finally, we note that the precision and recall metrics, focus on the performance of the algorithm with respect to the scene change locations only. These measures are not necessarily indicative of the performance of the algorithm. For example, if we assume

the potential scene change location to occur at every point where we evaluated β (the distance increase rate), then the results are interesting. Note that the local maxima of the β sequence is the a-scene change location. The number of potential scene change points is then 3569 (i.e. 3600 - memory size). Then, correct rejection is $\sim 94\%$ for all three films.

4.7.3 Comparison with related work

We shall briefly review the current results against other auditory analysis systems. A more extensive review of the issues in prior work has already been presented in section 2.4. There has been much prior work on *classification* of audio into generic audio classes such as speech / music and environmental sounds [53] [74] [75] [107] [84] .

While the classification accuracies are very impressive — typically 95% for the best systems, the segmentation problem that we are dealing with here is a different problem. In our worldview, a coherent segment may actually contain multiple classes. They can be overlapping as in the case of a person speaking with street sounds in the background, they can repeat over short time scales, as in the case of the sound of footsteps. The segmentation problem is important because it results in coherent segments, a key criterion for creating audio-visual skims (chapter 7).

4.8 Summary

In this chapter, we have presented our approach towards detecting computable audio scenes. We began with a discussion of the features used in our system. The features

belonged to three types: (a) vector sequences (b) scalar sequences and (c) point features. Then, we presented a new scalar sequence signal model that decomposed the sequence as a sum of three components — trend, sinusoids and noise. We presented an architecture for the signal decomposition, and described in some detail the MDL based polynomial estimation for the trend and the matching pursuit based algorithm for determining the optimal sinusoids. The noise was not explicitly modeled, since in segmentation, we not interested in exact signal reconstruction, but in computing dissimilarities amongst segments. We also discussed metrics for each type of feature, including the sub-components for the scalar signal case.

The memory model was central to determining the existence of a scene boundary. We estimated the rate at which the features in the attention span change with respect to the rest of the memory. Then, the local maxima of the rate of change was used to identify a potential audio scene change location. These distance increase rates were computed per feature, and a voting was adopted to determine if a scene change exists.

The experimental results indicate that our algorithm has low precision but high recall. Having high-recall was important in order to avoid misses in the computable scene detection. The high false alarm rate did not cause problems because of two reasons: (a) the c-scene precision is essentially limited by the higher v-scene precision and (b) false alarms in computable scenes can only occur if we have the a-scene false alarm *synchronized* with a detected v-scene and finally, (c) many of the audio false alarms are eliminated by the imposition higher order grouping rules for multi-modal fusion.

5 Multi-modal fusion

5.1 Introduction

In this chapter, we show how to integrate information from the various detectors. In the previous two chapters, we showed how to detect the two elements that are central to the computable scene framework — (a) elementary audio and video computable scenes, and (b) silence. In a later chapter, we shall show how to detect the third key aspect of our multi-modal fusion framework — structure detection. Since we have investigated structure detection as a problem independent of the segmentation problem, we present the results of structure detection in chapter 6.

The use of silence and structure for c-scene segmentation, are a form of top-down constraints on the c-scene boundary detection algorithm. The reason why this imposition of higher forms of knowledge is necessary, is because the bottom-up approach in the computational scene model in section 2.5.4 can generate c-scenes that run counter to grouping rules that human beings routinely use. Note that we cannot infer the role that structure plays in visual comprehension (i.e. the structure level grouping that takes place with human beings), by merely detecting the existence of structure.

The rest of this chapter is organized as follows. In the next section we introduce the three basic rules that govern c-scene detection algorithm. Then, we present the

procedure for c-scene detection in section 5.3, and in section 5.4, we present our experimental results. There, we discuss our experimental results, and discuss the causes of our model breakdowns. Finally in section 5.5, we summarize the results in this chapter.

5.2 The three rules for detection

There are three principal rules for detecting c-scenes:

- We detect a c-scene whenever we can associate a v-scene with an a-scene that lies within a window of W_C sec.
- We declare a c-scene to be present when normal v-scenes (see section 3.5) intersect silent regions.
- We always associate a c-scene boundary with strong v-scene boundary locations.

The first rule is the synchronization rule for detecting c-scenes. The window W_C is necessary as film directors deliberately do not exactly align a-scene and v-scene boundaries. They do this since this causes a perceptually smoother transition between scenes. There are some exceptions to this rule, which we discuss later in the section.

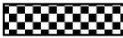
The second rule is important as many transitions between c-scenes are silent (e.g. the first scene ends in silence and then the second scene shows conversation, which also begins with silence). In such cases, audio scene boundaries may not exist within W_C sec. of the v-scene.

The third rule becomes necessary when there is no detectable a-scene boundary within W_C sec. of a strong v-scene boundary. Strong v-scene boundaries occur as transitions between two v-scenes that are long in duration, and which can differ greatly in chromatic composition. Consider the following example. Alice and Bob are shown having conversation. Towards the end of the scene, the director introduces background music to emphasize a certain emotion. Then, the music is held over to the next scene for more than a few seconds to establish thematic continuity. Note that in this case, there is no a-scene boundary near the v-scene boundary, nor is there a silent region near the v-scene boundary.

5.3 A procedure for detecting c-scenes

We now present the detailed algorithm for c-scene detection. The notation used in the figures in this section: gray box: silence, patterned box: structure, a-scene boundary: solid dot, v-scene: equilateral triangle, weak v-scene: solid right angled triangle (see **Table 5.1** for a list of the icons used). Now, given the locations of the basic a-scenes, v-scenes, structure and silence, we proceed as follows:

Table 5.1: The table shows a list of icons used in the figures in this section and their semantics.

Type	Icon
Silence	
Structure	
V-scene boundary	
A-scene boundary	
Weak V-scene	

Step 1: Remove v-scene or a-scene changes or silence within structured sequences (i.e. within dialogs and regular anchors) (**Figure 5.1**). This is intuitive since human beings recognize and group structured sequences into one semantic unit.

Step 2: Place c-scene boundaries at strong (see section 3.5) v-scene boundaries. Remove all strong v-scenes from the list of v-scenes.

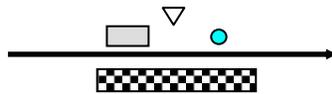


Figure 5.1: Remove a-scene and v-scene boundaries and detected silence, when detected in structured sequences.

Step 3: If an a-scene lies within W_C sec. of a v-scene, place a c-scene boundary at the v-scene location. However, there are three exceptions:

- Do not associate a weak v-scene with a weak a-scene.
- If the v-scene is weak, it must synchronize with a non-weak a-scene that is within $W_C/2$ sec. i.e. we have tighter synchronization requirements for weak v-scenes (**Figure 5.2**).

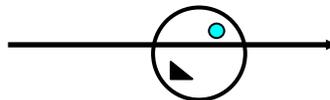


Figure 5.2: Tight synchronization is needed between weak v-scenes and non-weak a-scenes.

- Do not associate a normal v-scene with a weak a-scene marked as silent²¹ (ref.

Figure 5.3). The reason why this is done is because the a-scene is already weak, and additionally contains a significant amount of silence. The a-scene was probably a false alarm.

Step 4: Non-weak (see section 3.5) v-scene boundaries (i.e. normal boundaries. Note

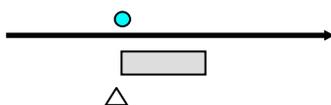


Figure 5.3: Do not associate normal v-scenes with a-scenes marked as silent.

that strong boundaries would have already been handled in step 2) that intersect silent regions²² are labeled as c-scene boundaries (**Figure 5.4**).

To determine whether a v-scene boundary intersects silence, we do the following:

²¹ An a-scene is denoted as “silent” if a significant portion of the local window around the a-scene boundary contains silence. Typically the window is 2 sec. long and 95% of the data in the window needs to contain silence, for this classification.

²² A region marked as “silence” is different from an a-scene marked as silent. The difference is that an a-scene boundary has been detected in the second case (no a-scene boundary exists in the case of silent regions), but contains a significant amount of silence around the scene boundary.

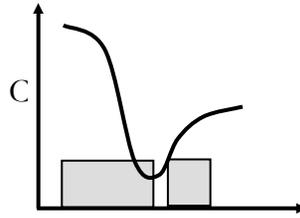


Figure 5.4: Non-weak video coherence minimum intersecting silences (gray boxes) causes a c-scene boundary.

- Compute the fraction of silence in a symmetric window ($2W_{VS}$ sec. long) around the v-scene boundary. Let L_{VS} and R_{VS} be the left and right silence fractions. i.e. the amount of data in the left and right windows that constitute silence.
- Then, declare a c-scene boundary if: $L_{VS} > 0.8 \vee R_{VS} > 0.8$

Now, we have a list of c-scenes, as well as lists of singleton video and audio scene boundaries. The c-scenes are then post-processed to check if additional structure is present.

5.3.1 Additional post-processing

Once we have detected all the c-scenes, we use a conservative post-processing rule to eliminate false alarms. An irregular anchor shot in a semantic scene is a shot that the director comes back to repeatedly, but not in a regular pattern, within the duration of the semantic scene. This is known in film-making, as the “familiar-image” [39].



Figure 5.5: The first and the last frames show the irregular anchor. This shot will appear sporadically over the whole scene.

In **Figure 5.5**, the first and the last frame shows the irregular anchor that will appear sporadically within this scene. We check if an anchor is present across adjacent scenes and merge them, if present. We make this rule transitive: i.e. if we have three c-scenes A, B, C, in succession, and if A and B have share a regular anchor and B and C share a (possibly different) regular anchor, then c-scenes A, B and C are merged into one c-scene.

5.4 Experimental results

In this section we shall discuss the experimental results of our algorithms. The data used to test our algorithms is complex: we have three one hour segments from three diverse films in English: (a) *Sense and Sensibility* (b) *Pulp Fiction* and (c) *Four Weddings and a Funeral*. We begin with a section that discusses the ground truth data. It is followed by sections on c-scene boundary detection and structure detection.

5.4.1 The ground truth

The audio and the video data were labeled separately (i.e. label audio without watching the video and label video without hearing the audio). This was because when we use *both* the audio and the video (i.e. normal viewing of the film) we tend to label scene boundaries based on the semantics of the scene. Only the author of this work labeled the data. Since we have already discussed the labeling procedure for the video and audio computable scenes in sections 3.6.1 and 4.7.1 respectively, we shall only summarize the

results of the labeling in **Table 5.2**. Note, that in the ground truth, an a-scene and v-scene are denoted to be synchronous if they less than 5 sec. apart.

Table 5.2: Ground truth data obtained from labeling the audio and the video data separately. The c-scenes are broken up into constituent units: P-pure, Ac-V (audio changes, visuals consistent), A-Vc: audio consistent, visuals change, and MM: mixed mode.

Film	A-scenes	V-Scenes	C-Scenes				Total
			P	Ac-V	A-Vc	MM	
Sense and Sensibility	69	57	33	11	5	2	51
Four weddings and a funeral	72	61	31	8	8	6	53
Pulp Fiction	50	40	25	4	8	1	38

The ground truth segmentation in **Table 5.2**, show in addition to the elementary audio and video computable scenes the breakdown of the types of computable scenes. Recalling the discussion in section 2.5.4.1, there are four types of computable scenes — (a) pure scenes (b) Ac-V (audio changes, visuals remain consistent), (c) A-Vc: audio remains consistent but visuals change, and (d) MM: mixed mode, where the scene has synchronized audio and video scene change beginning and ending, but the scene also contains unsynchronized audio and video scenes

Interestingly, pure c-scenes, i.e. those in which one audio scene synchronizes with one video scene, constitutes between 58-65% of the total number of c-scenes, thus validating the importance of computable scene model in section 2.5.4.1.

5.4.1.1 *Is it any easier to label c-scenes?*

One of the key issues that came up in this work on computable scene detection was that of labeling the audio and the video computable scenes. Note that the c-scene was easily determined using synchronization constraints on the elementary audio and video computable scenes.

We attempted to give the labeler (the author) signal level criteria for determining these scenes — e.g. the v-scene had to exhibit long term consistency in terms of chrominance and lighting, while the a-scene had to exhibit long term consistency in the ambient audio data. We had also ruled that these elementary audio and video computable scenes had to be more than 8 sec. long.

The evaluation of the ground truth labels indicates that despite the best efforts of the labeler, there were instances where the v-scenes had been grouped together based on a higher form of abstraction (e.g. all shots take place in the same house, and the small differences in the chrominance are not taken into account) rather than on the signal levels of the data. Often in the case of a-scenes, it was difficult to determine the precise a-scene boundary location in very quiet periods in the data. On other occasions, conversations that were based on the same topic were grouped together; clearly this uses knowledge that the feature-level segmentation algorithm cannot possess.

We believe that while c-scene labeling issue is problematic, it is not significant enough for us to consider going back to the idea of using semantic scenes. The semantics of scene

depend on so many non-measurable factors — the context of the scene, the level of understanding of the viewer, amongst others. Additionally, the semantics exist at multiple levels *simultaneously*, thus precluding a consistent labeling of the data.

The idea of the computable scene is important, since it forces to think in terms of task based segmentation — i.e. construction of segments that facilitate other tasks. The overall goal of our work in segmentation is to preserve the structure of the original video, while constructing these computable scenes. These syntactically correct computable scenes form the input to our summarization algorithm.

5.4.2 Segmentation results

In this section, we discuss the scene change detector results. First, we discuss the parameters that we need to set. The memory and attention span sizes for the audio and video scene detection algorithm, and the synchronization parameter W_C , which we set to 5 sec (i.e. c-scene boundary is marked when the audio and video scenes are within 5 sec. of each other). For detecting video coherence, video we set the attention span to be 8 sec. (in accordance with our labeling rule) and the size of the memory is set to 24sec. For detecting a-scenes the memory parameters are as follows: the size of the memory is set to 31 sec. while the attention span size is set to 16 sec. In general, increasing the memory size reduces false alarms, but increases misses. For the v-scene detection the duration of the shot-let is set to 1 sec. while for the a-scene detection, the duration of the shift is set to 1 sec.

In evaluating our results, we shall compare against c-scenes against the total number of shots in the film, since they are all candidate c-scene change points. Secondly, it is important to note that we are dealing with an asymmetric two-class problem (scene change vs. non-scene change) where the number of ground truth scene change locations is typically less than 10% of the total number of shots. Hence it is important to correctly reject non-scene change points in addition to correctly detecting the scene change points.

The information retrieval metrics precision and recall²³ are normally used to discuss detection results in the multimedia community. However, as discussed earlier (ref. sections 3.6.2, 4.7.2) we need better metrics for analyzing multimedia content analysis. This is because by focusing on the performance of the algorithm with respect to the scene change locations only, they do not illustrate the performance of an algorithm dealing with an asymmetric class distribution. Hence we present the entire confusion matrix (i.e. hits, misses, false alarms and correct rejection), in addition to presenting the precision and recall.

The **Table 5.3** shows the results for c-scene detection. These results are for the entire duration of the film (each film is one hour long) and for all types of transitions. We use the following notation: H: hits, M: misses, FA: false alarms, CR: correct rejection. Shots is just the number of shots detected by the shot detection algorithm. NCL: non-scene

²³ Note that this term is different from the similar term used in our memory model (see section 3.3), that is used to measure inter shot similarity.

change locations; this is just the number of shots less the number of ground truth scene change locations. Precision: $\text{hits} / (\text{hits} + \text{false alarms})$, Recall: $\text{hits} / (\text{hits} + \text{misses})$.

Table 5.3: C-scene detector results. In order: hits, misses, false alarms, correct rejection, number of shots, number of non-scene change locations, precision and recall.

Film	H	M	FA	CR	Shots	NCL	Precision	Recall
Sense and Sensibility	48	3	21	570	642	591	0.70	0.94
Pulp Fiction	36	6	22	414	478	436	0.62	0.85
Four Weddings and a Funeral	41	12	18	665	736	683	0.70	0.77

The result shows that the c-scene and the v-scene detectors work well. The recall for c-scene detectors varies between 77 ~ 94% while the precision varies between 62 ~ 70%. The recall for the v-scene detector varies between 80~91% while the precision varies between 55 ~ 70%. Note that the correct rejection is excellent — around 95% across all cases. We now discuss two relevant aspects relevant to our results — sources of error, the audio scene location uncertainty, and the relative importance of the low precision.

5.4.2.1 Shot detection errors

Shot detection errors cause errors in c-scene detection. This is because misses in the video shot boundary detection algorithm cause the wrong key-frame to be present in the buffer, thus causing an error in the minima location. In the film *Four Weddings and a Funeral*, there is a sequence of four c-scenes that is missed due to very low chrominance

difference between these scenes; it is likely that the labeler managed to distinguish them on the basis of a semantic grouping of the shots.

One possible solution to this problem is introduce a context dependency in our c-scene detection algorithm. For example, we could have a “low-chrominance” detector, and when this happens, bias c-scene detector towards the audio scene detector and if there is a strong a-scene change, we would then label it as a possible c-scene boundary.

5.4.2.2 A-scene location uncertainty

Uncertainty in the audio scene change location is additional source of error. Labeling the audio data is time consuming and often, there is genuine uncertainty about the a-scene change location. This can happen for example, when we have a long sequence of low amplitude sounds (e.g. background sounds, soft footsteps) that changes into silence. Thus, this can translate to c-scene misses. This uncertainty may be mitigated to a certain extent by using additional labelers, but is difficult to eliminate altogether.

5.4.2.3 The c-scene algorithm has low precision

It is clear that our algorithm apparently over-segments the data. A detailed look at the false alarms indicates that these scenes are correct from a computational standpoint (i.e. satisfied the requirements for a change), but were wrong semantically.

For example consider the following scene. The director shows two people engaged in a conversation, with the backdrop of a white wall. Then, there is a pause in the conversation and the couple are now shown before a large window with the backdrop of

a large green garden. Then, because of the pause in the conversation and the change in the background, we would have synchronized audio-visual scene change. The only way this can be prevented is to either incorporate motion continuity into the v-scene algorithm (in case the director used a pan shot), or to infer from auditory analysis that either the topic (or the speakers) have not changed.

This seems to imply that even though we had signal-level guidelines for labeling the ground truth, in many cases, the labeler ended up labeling the data on a semantic level.

Does the low-precision present a problem? The utility of our results clearly depends upon the task. One of the goals of our work is to generate video summaries by condensing computable scenes (see chapters 7-10). There, we condense each computable scene via an analysis of the visual complexity of the shots and by using film syntax. In such tasks, it is more important to generate c-scenes that preserve the syntactical elements of the original video (the structural elements are preserved), than to ensure that the semantics are consistent.

There are situations where it becomes important to have high precision. Consider the following example. Let us assume that an company that streams legacy videos overt the internet wants to insert advertisements periodically in the data stream. In such a situation, it would want to ensure that the precision of the automatic c-scene detection was close to 100%, since the users will get annoyed if the advertisement suddenly appeared in the middle of a scene (i.e. due to a c-scene false alarm).

5.4.3 Comparison with related work

We now briefly compare our results with prior work. But first, we present a high-level summary of our contributions to help place the comparison in a proper context. We have developed a conceptual framework for determining computable scenes. There are four types of such scenes based on the relationships between audio and video scene change boundaries. The framework is based on the following (a) the impact of film production rules on the data (b) the psychology of audition (c) high-level structural grouping constraints. Now, we compare the results against other algorithms for scene detection and structure detection. Note that these algorithms use different datasets, and also have different objectives in mind. Hence direct performance comparisons are difficult.

In [35], the authors segment film data on the basis of an adaptive shot clustering mechanism. They do not use auditory analysis, thereby ignoring the rich synergy between the audio and visual data. Their algorithm for clustering ignores the duration of the shot while segmenting the video. This is important since a semantically meaningful scene can also be a few shots (or even a single shot), but of a long duration. Hence the shot duration is an important consideration for segmentation. Additionally, they also do not consider the role of structure (especially dialogs) while grouping shots into a scene (ref. Chapter 6).

Prior work done in video scene segmentation used visual features alone [102], [44]. There, the authors focus on detecting scene boundaries for sitcoms (and other TV

shows) and do not consider films. However, since we expect the v-scenes in sitcoms to be mostly long, and coherent, we expect our combined audio visual detector to perform very well.

The key differences with the work in [37] are the following: (a) they use simplified audio visual model that looks for synchronized changes in the shot, audio and motion. (b) they do not investigate the long-term grouping and finally, (c) they do not analyze the effect of structure on the segmentation result.

5.4.4 C-scene detector breakdowns

In this section we shall discuss three situations that arise in different film-making situations. In each instance, the 180 degree rule (see section 2.5.3) is adhered to but where, our assumption of chromatic consistency across shots is no longer valid, thus causing errors in our c-scene detection algorithm.



Figure 5.6: There is a sudden change of scale that is not accounted by the model.

5.4.4.1 Sudden change of scale

A sudden change of scale accompanied by a change in audio cannot be accounted for in our algorithm. This can happen in the following case: a long shot²⁴ shows two people with low amplitude ambient sound; then, there is a sudden close up of one person as he starts to speak.

Detecting these breaks, requires understanding the semantics of the scene. While labeling, these types of scenes get overlooked by the labeler due to semantic grouping and hence are not labeled as change points.

5.4.4.2 Widely differing backgrounds

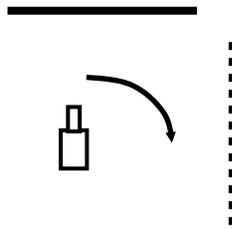


Figure 5.7: a right angled camera pan, between two very different backgrounds.

²⁴ The size (long/medium/close-up/extreme close-up) refers to the size of the objects in the scene relative to the size of the image.

Widely differing backgrounds can exist across shots causing false alarms in our c-scene detection. This can happen in two circumstances:

- A right angled camera pan and
- A set up involving two cameras.

In the first case (**Figure 5.7**), the coherence model will show a false alarm for v-scene, and if accompanied by an a-scene change, this will be labeled as a c-scene break. In the second case we have two opposing cameras having no overlap in their field-of-view causing an apparent change in the background. This can happen for example, when the film shows one character inside the house, talking through a window to another character who is standing outside.

5.4.4.3 *Change in the axis of action*

The axis of action (i.e. the line of interest, ref. section 2.5.3) can change in several ways. Let us assume that we have a scene which shows a couple engaged in conversation. The

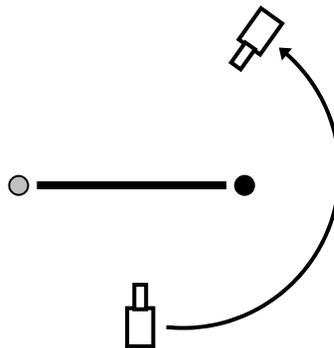


Figure 5.8: A circular tracking shot will establish a new axis.

director can change the axis of action within a scene [8] by:

- moving the one of two people across the room or
- by using a circular tracking shot around the couple, thereby establishing a new axis in both cases. (see **Figure 5.8**). The motion continuity alerts the viewer about this change.

These examples clearly indicate the simplicity of our computational model. These situations are problematic (incorrect boundary placement) only when they take place over long time scales (i.e. camera pans and stays there); Short term changes will be handled by our algorithm. Also, if these changes exhibit structure, (i.e. in a dialog or in a regular anchor), these false alarms will be eliminated. One way to overcome the use of pans in films is to incorporate motion information into our decision framework. Motion continuity will be of help in detecting the change of axis scenario.

5.5 Summary

In this chapter, we have presented a framework for integrating the three key components of our computational framework — (a) computable audio and video scenes (b) structural elements (e.g. dialogs) and (c) silence. We showed why this integration was necessary since the silence and structure information impose top-down constraints on the audio and video scene boundaries. This help resolve ambiguities that cannot be determined with using just the a-scene and the v-scene detection models.

The scene segmentation algorithms were tested on a difficult test data set: three hours from commercial films. They work well, giving a best c-scene detection result of 94%. We discussed three sources of error in our algorithm — (a) errors in shot detection (b) ground-truth uncertainty in the a-scenes and (c) computational scene model breakdowns. We discussed model breakdowns related to (a) sudden changes of scale, (b) widely different backgrounds and (c) changes to the axis of action.

5.5.1 Improvements

There are several clear improvements possible to this work

- The computational model for the detecting the video scene boundaries is limited, and needs to be tightened in view of the model breakdowns discussed. One possible improvement is to do motion analysis on the video and prevent video scene breaks under smooth camera motion.
- Since shot misses can cause errors, we are also looking into using entropy-based irregular sampling of the video data in addition to the key-frames extracted from our shot-segmentation algorithm.

6 Detecting structure

6.1 Introduction

In this chapter, we shall present our approach towards detecting visual structure. While detecting structure is an important problem in its own right, with specific semantics associated with a particular element of structure, it also plays an important role in segmentation. This is because human beings tend to group elements of structure into a single entity, and this “top-down” rule complements the “bottom-up” approach of segmentation algorithms, thereby preventing over-segmentation.

We define a discrete, deterministic label sequence to be structured, if that sequence is compressible i.e. the number of bits required to represent the sequence in a *lossless* manner, is less than the number of bits required to represent the original sequence. We shall be specifically looking at structures in video data, that have a generative mechanism (e.g. rules) associated with them. Additionally, we shall assume that these sequences have an associated metric, thereby defining the structural elements in terms of the topological relationships between the elements of the structure. In this work, we shall be examining structure only in the context of a deterministic framework.

Central to our structure detection algorithms is the idea of the topological graph. This structure has video shots at the nodes and the edge strengths are simply the distances

between the nodes. Associated with each topological graph is a topological matrix constructed using the edge strengths of the graph. In this chapter, we shall show two general mechanisms to detect structure:

- Using exploiting the regularities within the topological matrix.
- Randomizing the topology of the structure to be detected.

We will discuss the generative mechanisms associated with two specific visual structures that we are interested in — the dialog and the regular anchor, and we shall also show how we can use the general approaches outlined above, for detecting these two structures. It will be seen that the topological graph provides an extensible framework for detecting arbitrary generative structures.

The rest of this chapter is organized as follows: In the next section, we shall define the notion of structure for a discrete sequence, in terms of its compressibility. Then in sections 6.3 we shall look at the topology of video sequences, and discuss a specific framework, the topological graph, in section 6.4. In section 6.5, we shall discuss the two structures that are the focus of this work — the dialog and the regular anchor. In sections 6.6 and 6.7, we shall discuss our approach towards detecting dialogs and regular anchors respectively. In section 6.8, we present our experimental results and in section 6.9, we discuss related work. Finally, we summarize the chapter in section 6.10.

6.2 What is structure?

Let us assume that we are given a deterministic sequence \mathbf{S} , containing discrete labels. Let there be n elements in the sequence. Then, the sequence is said to be *structured* (or said to have a property of “structured-ness”) if the following relationship is true:

$$\frac{K(\mathcal{S} | n)}{n} < 1, \quad (6.1)$$

where $K(\mathcal{S} | n)$ is the Kolmogorov complexity [21] [51] (see section 13.1) of the label sequence. Since the Kolmogorov complexity of the sequence is shorter than the length of the original sequence, equation (6.1) implies that the sequence \mathbf{S} is compressible. Note that we are defining structured-ness of deterministic sequences only.

The definition of structure in terms of Kolmogorov complexity, may appear to make the problem of *discovering* structure intractable, since Kolmogorov complexity cannot be computed, but only estimated ([21] [51] , also see section 8.2.2.1). The importance of the definition lies in the fact that it forces us to think of structure in along of the following two dimensions:

- **Generative mechanisms:** Structure in deterministic sequences can occur as a result of a generative set of rules, or in the more general case, the output of a deterministic program.
- **Structure detection as a compression problem:** By constructing an equivalence between compression and structure, we can think of structure

detection as the problem of constructing efficient coding schemes for a label sequence.

Structures present in video sequences, or in discrete real sequences are characterized by their topological properties i.e. the metric relationship between the elements of the structure. However, this is not true in the case of non-ordinal sequences such as DNA sequences, where only a logical relationship exists between the elements of the sequence.

In this work, we shall be looking at detecting visual structure in discrete sequences that have associated metric space, via robust statistical tests. We shall assume that these structures have been generated via specific deterministic, generative mechanisms.

6.3 The topology of video shots

Structures (e.g. dialogs) contain important semantic information, and also provide complimentary information necessary to resolve v-scene boundaries. For example, in a dialog that contains very long shots (say 25 sec. each) showing very different backgrounds, the v-scene detection algorithm presented in section 3.4, will generate v-scene boundaries after each shot. Computationally, this situation is no different from two long shots from completely chromatically different (but adjacent) v-scenes. Human beings easily resolve this problem by not only inferring the semantics from the dialogue, but also by recognizing the dialog structure and grouping the shots contained in it into one semantic unit.

Structures in video shot sequences, have an important property that the structure is independent of the individual shot lengths. It is the topology (i.e. the metric relationships between shots, independent of the duration of the shots) of the shots that uniquely characterizes the structure (see **Figure 6.1**).

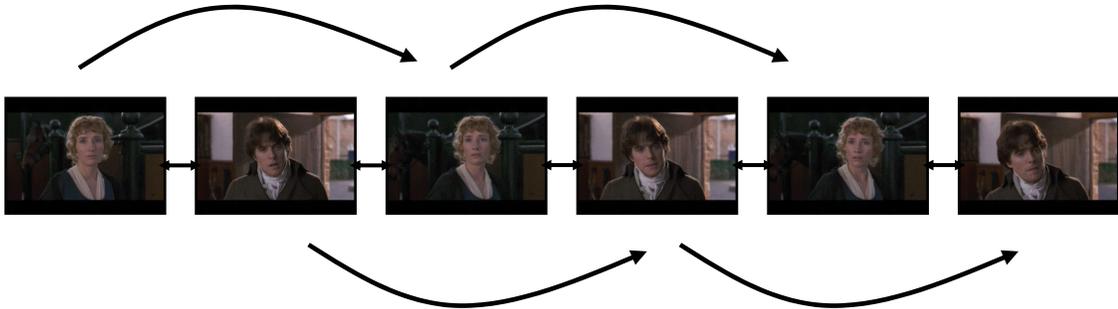


Figure 6.1: In the familiar dialog sequence, the sequence has interesting topological properties — (a) adjacent shots differ (b) alternate shots are alike and (c) this is independent of the duration of each shot.

For example, in a dialog sequence (A-B-A-B-..), the lengths of each shot will vary over the course of the dialog, and in general are related to: (a) the semantics of the dialog, (b) the presence of one speaker to who may dominate and (c) relationship dynamics between the speakers i.e. the dominant speaker may change over the course of the conversation. We now introduce the idea of the topological graph, central to all of our structure detection algorithms.

6.4 The topological graph

Let \mathbf{S}_d be the metric space induced on the set of all images \mathbf{I} in the video sequence, by the distance function \mathbf{d} . Then, the topological graph $T_G = \{V, E\}$ of a sequence of k images, is a fully connected graph, with the images at the vertices (V) and where the edges (E) specify the metric relationship between the images. The graph has associated with it, the topological matrix T_{MAT} , which is the k by k matrix where the entry $T_{MAT}(i, j)$ contains the strength (i.e. the distance between the two images corresponding to the nodes) of the edge connecting node i to node j in the graph.

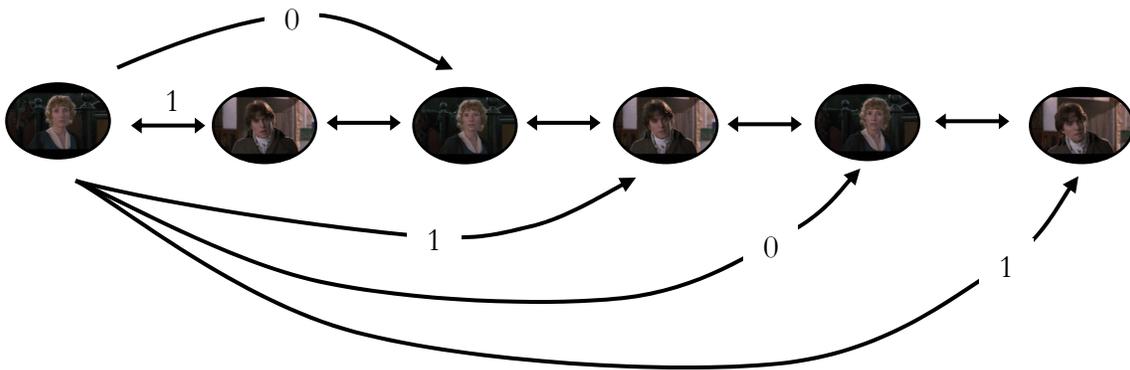


Figure 6.2: The idealized topological graph for a dialog subsequence, that is induced by associating a metric d on the on the entire video sequence \mathbf{I} .

The distance between two N by N fully connected graphs is defined to be the distance between the corresponding topological matrices. Given two topological matrices A and B , the distance between them is defined as follows:

$$d(A, B) \triangleq \sum_{i=1}^N \sum_{j=1}^N |A_{i,j} - B_{i,j}| \quad (6.2)$$

i.e. the distance is the sum of the absolute difference between the corresponding entries of the two matrices. Note that the distance is only defined for two graphs having the same number of nodes.

The idea of the topological graph is distinct from the scene transition graph [102] [103] (see section 2.3.1). There, the authors cluster shots and construct a graph whose vertices represent the clusters and whose edges represent the shot transitions between the clusters. Then, they examine temporal relationships between these clusters to determine scene change points as well as dialogs.

In this work, we are strictly interested in the topological property of a sequence of images and not in determining scene transitions. Each node in our fully connected graph represents a single shot and the edges represent the dissimilarity between the shots in each node.

6.5 Topological structures

We now investigate two generative structures in this work: the dialog and the regular anchor.

6.5.1 Dialogs

Dialogs in films have an interesting rule associated with them: showing a meaningful conversation between m people requires at least $3m$ shots [77]. Hence in a dialog that shows two participants, this implies that we must have a minimum of six shots. A six

image length dialog A-B-A-B-A-B, is completely specified with the following idealized topological relationship: $d(A, B) = 1$, $d(A, A) = 0$, $d(B, B) = 0$. Formally, the generative rule for a dialog sequence \mathbf{D} with n elements is as follows:

$$\begin{aligned} d(i, i+2) &= 0, & i &= 1, 2, \dots, n-2 \\ d(i, i+1) &= 1, & i &= 1, 2, \dots, n-1 \\ n &\geq 6, \end{aligned} \tag{6.3}$$

where, i represents the i^{th} element of the structure, d is the metric associated with the elements and n is the number of elements in the sequence. In **Figure 6.2**, we see the idealized topological graph corresponding to a dialog sequence. Then, for an idealized dialog sequence of 6 images (A-B-A-B-A-B) we would get the following topological matrix:

$$\begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \tag{6.4}$$

6.5.2 Regular anchors



Figure 6.3: A three image regular anchor.

A regular anchor is characterized by an anchor image, that repeats as every other image. The regular anchor topology is defined to begin and end with the anchor image, and hence regular anchor sequences are always odd length sequences. A three image regular anchor is shown in **Figure 6.3**.

The sequence A-I₁-A-I₂-A is an example of a five image regular anchor. The image A is the anchor and I₁ and I₂ can be arbitrary images. The idealized topological relationship is then: $d(A, I_1) = d(A, I_2) = 1$, $d(A, A) = 0$. The relationship between I₁ and I₂, can be arbitrary and we specify the edge (and the corresponding entry in the topological matrix) between the nodes representing I₁ and I₂ in T_G as -1. The extension to general (2n+1) sequence now immediately follows. Formally, the generative rule for a regular anchor of length 2n+1 is as follows:

$$d(i, j) = \begin{cases} 0 & i = j, \quad i, j: \text{anchor, non-anchor,} \\ 1 & i \neq j, \quad i: \text{anchor, } j: \text{non-anchor} \\ -1 & i \neq j, \quad i, j: \text{non-anchor} \end{cases} \quad (6.5)$$

$$i = \begin{cases} \text{anchor image,} & i = 1, 3, 5, \dots, 2n + 1, \\ \text{non-anchor image,} & i = 2, 4, 6, \dots, 2n, \end{cases}$$

$$n \geq 1,$$

where, the index i represents the i^{th} element, and d represents the distance function, and the number of elements in the anchor are $2n + 1$. Note that the definition states that the odd indexed elements (2n+1) are anchor images, while the rest are non-anchor images.

Topological matrices can contain a “don’t care” condition. For example, the topological matrix (equation (6.6)) for the five image regular anchor definition contains a “don’t-care” metric condition:

$$\begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & -1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & -1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix} \quad (6.6)$$

where, 0 specifies that the images are identical, 1 specifies that the distance is very large, and -1 implies that we do not care about the metric relationship between these pair of nodes. Why does the don’t care situation occur? This is because the definition of the regular anchor does not depend on the metric relationship between the two non-anchor images — the two images (in the case of the five image regular anchor) could be identical or be completely dissimilar to *each* other. What is most important is that they be *dissimilar* to the anchor images.

6.5.3 A note on the end conditions



Figure 6.4: A three element anchor surrounded by two end nodes.

When we are constructing topological matrices for any pattern, we need to take into account the end conditions i.e. the relationship between the two ends of the topological

sequence that we are trying to detect and the surrounding elements. This results in an augmented topological graph and matrix. For example, consider the three element regular anchor (**Figure 6.4**).

Now, what should the topological condition be? While this in general will depend on the topological sequence that we are trying to detect, in the case of the regular anchor the conditions are easy. The metric relationship between the two nodes can be arbitrary (i.e. a “don’t care” condition), but the two end nodes must be at a maximum distance from the topological sequence itself. Hence the augmented topological matrix is then:

$$\begin{bmatrix} 0 & 1 & 1 & 1 & -1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ -1 & 1 & 1 & 1 & 0 \end{bmatrix} \quad (6.7)$$

where, 0 implies that the two nodes are at zero distance, 1 implies that they are at maximum distance and -1 is the “don’t care” condition.

6.6 Detecting dialogs

A dialog has a specific local topological property: every 2nd frame is alike while adjacent frames differ (**Figure 6.1**). In the idealized topological matrix for the dialog (equation (6.4)), this appears as the 1st off-diagonal being all ones, the 2nd off-diagonal being all zeros and the 3rd off-diagonal being all ones. We detect dialogs by exploiting the regularities in the topological matrix for dialogs.

We proceed by defining a periodic analysis transform, that maps a N by N real matrix to N numbers in the real number line, to estimate existence of this pattern in a sequence of N shot key-frames. Let o_i where $i \in \{0, N-1\}$ be a time ordered sequence of images. Then, the transform is defined as follows:

$$\Delta(n) \triangleq 1 - \frac{1}{N} \sum_{i=0}^{N-1} d(o_i, o_{\text{mod}(i+n, N)}) \quad (6.8)$$

where, $\Delta(n)$ is the transform, d is the L^1 color-histogram based distance function, mod is the usual modulus function. The modulus function simply creates a periodic extension of the original input sequence.

We shall make use of two statistical tests: the student's t-test and the F-test. The student's t-test and the F-test are used to compare two series of numbers and respectively determine if the two means and the variances differ significantly. There are two student-t test's depending upon whether the variances differ significantly or not, and hence we use the F-test for variances to determine the appropriate Student's t-test.

6.6.1 The dialog presence condition

We now present our test for detecting dialogs. Let us assume that we have a time-ordered sequence of N key-frames representing different shots in a scene. Then we do the following:

- Compute the series $\Delta(n)$.
- Check if $\Delta(2) > \Delta(1)$ and $\Delta(2) > \Delta(3)$ (see **Figure 6.5**).

A dialogue is postulated to exist if one of two conditions in step 2 is at least significant at $\alpha = 0.05$ and the other one is at least significant at $\alpha = 0.1$. We are trying to reject the two null hypothesis' being tested, $\Delta(2) = \Delta(1)$ and $\Delta(2) = \Delta(3)$, at the significance level α . We reject the null hypothesis if we believe that the observed difference between the means occurred by chance with a probability less than α . The reason why we can use the student's t-test for the means, to determine whether the two means are different in a statistically significant sense, is because $\Delta(n)$ for each n is just the mean of N numbers.

6.6.2 The sliding window algorithm

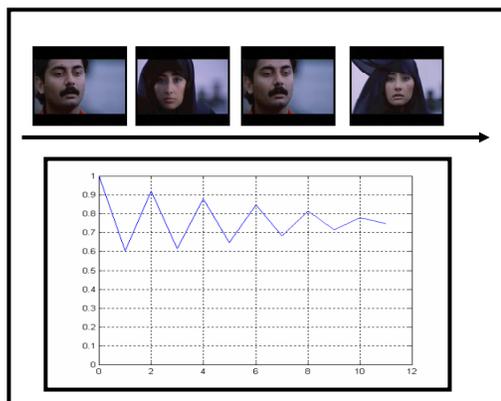


Figure 6.5: A dialogue scene and its corresponding periodic analysis transform.

Note the distinct peaks at $n = 2, 4 \dots$

We use a sliding window algorithm to detect the presence of arbitrary length dialogs (but they at least have six frames) in the entire shot sequence for the video. The window size is set to six frames, since that is the minimum size of the dialog [77]. The algorithm is as follows:

- Run the dialog detector on the current window.
 - If no dialog is detected, keep shifting the window to the right by one key-frame to the immediate right until either a dialog has been detected or we have reached the end of the video sequence.
 - If a dialog has been detected, keep shifting the starting point of the window to the right, by two key-frames, until we no longer have a statistically significant dialog or if we reached the end of the video sequence.
1. Merge all the overlapping dialog sequences just detected.
 - Move the starting point of the window to be the first frame after the last frame of the last successful dialog.

The sliding window algorithm can sometimes “overshoot” and “undershoot.” i.e. it can include a frame before (or after) as being part of the dialog. These errors are eliminated by simply checking if the local dialog topological property (i.e. the topological relationship A-B-A) holds at the boundaries. If not, we simply drop those frames. This results in an algorithm that generates statistically significant dialogs, with precise begin and end locations.

6.7 Detecting regular anchors

In this section, we discuss our approach towards detecting regular anchors. We need a new technique distinct from the one used for detecting dialogs, in order to detect regular anchors. Why does this become necessary? Note that the key point in our algorithm to

detect dialogs was the use of the student's t-test to make the detection robust.

However, the statistical test requires at least 6-7 points in the topological pattern, for it to give statistically significant results. An elementary regular anchor pattern has just three elements, too few for us to test for statistical significance using the student's t-test.

The result of this section is organized as follows: We begin by discussing the intuition guiding our solution in the next section, and then discuss regular anchor detection in section 6.7.2.

6.7.1 An intuition



Figure 6.6: The top sequence shows a three element regular anchor surrounded by the boundary elements. The lower sequence is the result of randomly permuting the first sequence. The intuition behind this permutation is that it will destroy the original topological structure, if such a structure was indeed present.

The intuition in solving the small topological sequence problem lies in following observation: if we randomly permute a sequence that has a specific topological structure (i.e. the topological relationships are well defined), then randomly permuting this sequence will destroy the topological structure of the sequence. However, if the

sequence did not have a strong topological structure to begin with, then such a permutation would not seem to alter the topological relationships dramatically.

In other words — permuting a sequence of random numbers, will more likely result in another random sequence, while permuting a structured sequence will more likely result in destroying the structure (thus making the sequence appear random).

6.7.2 The regular anchor detection algorithm

We now present the detailed algorithm. But before we jump into it, we need to introduce the idea of indistinguishable permutations. Then, we shall present our algorithm towards detecting three element regular anchors in section 6.7.2.2 and then conclude by presenting the algorithm for generalized regular anchor in section 6.7.2.3.

6.7.2.1 Indistinguishable permutations



Figure 6.7: This is a three element regular anchor, shown with the two boundary images. Permuting the first and the last image in the sequence does not alter the regular anchor pattern.

A random permutation of a three element regular anchor along with the two end point elements, will result in some permutations that are indistinguishable from each other (ref. **Figure 6.7**). This is because, by definition, the regular anchor images are indistinguishable, and importantly permuting all the non-anchor images (the internal non-anchor images and the two end points) does *not* affect the regular anchor pattern.

Now let us assume that we have a $2N-1$ regular anchor pattern r_a , with N anchor images and $N-1$ non-anchor images. Let us also assume that we have two end point images, thus making the total number of images to be $2N+1$. Now,

$$\begin{aligned} |I_r| &= 2!N!(N-1)!, \\ P(I_r) &= \frac{2!N!(N-1)!}{(2N+1)!}, \end{aligned} \tag{6.9}$$

where $|I_r|$ is the number of indistinguishable permutations, and $P(I_r)$ is the probability of a particular permutation being a member of the set of indistinguishable permutations. Note that the factor $2!$ appears because there are two end point images.

6.7.2.2 *Detecting the three element regular anchor*

We now discuss the case of detecting the basic three element regular anchor (see **Figure 6.4**). Then, the number of anchor images N , is 2. The corresponding idealized topological matrix T_{3R} corresponding to a three element regular anchor is given by equation (6.7). Note that the matrix definition includes the two end point elements. Given a five element sequence, we wish to determine if it is a regular anchor. Hence, we do the following:

- Compute the distance d_l (see equation (6.2)) between the idealized topological matrix and the topological matrix of the input sequence.
- Determine if it lies in the indistinguishable set. This is done in the following way:
- Compute $(2N+1)!$ permutations of the input sequence and compute the corresponding distance between each of the resulting topological matrices and the idealized matrix.
- A necessary condition for the test sequence being part of the indistinguishable set is the following:

$$l = \{d_i \mid d_i > d_1, i = 1, 2, \dots, (2N + 1)!\},$$

$$|l| \geq (2N + 1)!(1 - P(I_r)), \quad (6.10)$$

where, l is the set of distances that are greater than d_l , N is the number of regular anchors and $P(I_r)$ is the probability that a sequence is a member of the indistinguishable set. Equation (6.10) says that if the cardinality of l is greater or equal to the number of distinguishable elements. Now that we've established a threshold using the cardinality of the distinguishable elements, we also need a metric threshold on d_l . The test is useful, since if a test sequence does not belong to the indistinguishable set, it cannot be a regular anchor sequence.

- It becomes a very difficult task to set an absolute threshold on d_l since the dataset is very diverse. As a consequence, the dynamic range of the distances for

sequences from different films to be very different. Hence we decided to use a dynamic threshold on d_l is computed in the following way.

- Construct a large number of random five element image sequences, where the images are taken from a large pool of images, and compute the distance of the corresponding topological matrices with the idealized matrix.
- We then sort the distances and the threshold on d_l is as follows:

$$d_1 \leq \beta, \quad (6.11)$$

where β is a distance threshold such that a fraction α of the distances are greater than d_1 .

This way, we have two conditions for regular anchor detection: (a) the condition that the current sequence lie in the indistinguishable set, and (b) that the distance of the topological matrix of the input sequence with respect to the idealized topological matrix be smaller than a threshold. The threshold is data dependent and obtained dynamically using a pool of images.

6.7.2.3 *Detecting arbitrary sized regular anchors*

In order to detect regular anchors of arbitrary size, we need four five-element templates. This becomes clear in the following five element example.

In **Figure 6.8**, we see three of the four five element templates — initial template (*i*), the extension template (*e*) and the end template (*e*). The fourth template is just the simple regular anchor template (*s*) seen in **Figure 6.4**.

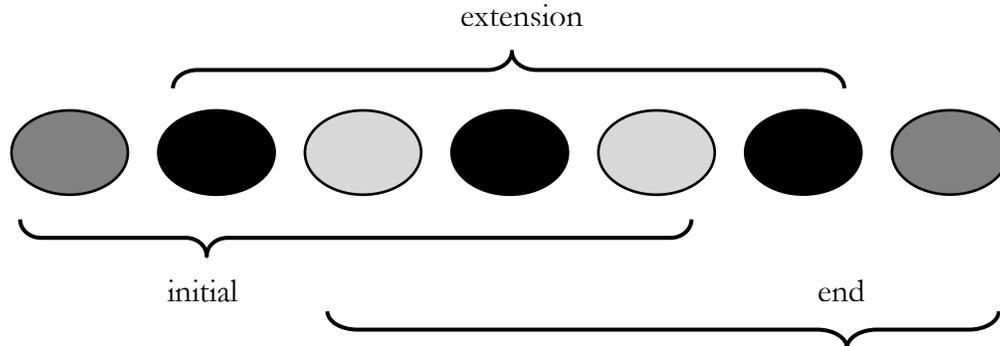


Figure 6.8: A five element regular anchor. The anchor images are black, the internal non-anchor images are light gray while the end images are dark gray. The figure shows three five element templates: initial (*i*), extension (*c*), and end (*e*).

These basic templates become necessary since the length of the regular anchor sequence is a priori unknown. These templates will allow us to detect any regular anchor sequence of length $2N-1$. Now the algorithm for detecting the arbitrary length regular anchor pattern of length $2N-1$, is as follows.

The algorithm has two pointers: a start pointer is that is at image k and a moving pointer at image m . Initially, we set $m = k$. In the following steps, we need to perform template matching. The procedure is identical to the one described in section 6.7.2.2 except that we use the template described in the step rather than the simple template (*s*) used in that section.

- Set $m = k$. Use the initial template (i) , to check if we are at the beginning of a regular anchor sequence. If there is a match set $m = m + 1$ and go to step 3 else proceed to the next step.
- Check if there is a match for the simple template (s) (section 6.7.2.2). If there is a match, store the location of the match, and then move the start pointer from k to $k + 4$, else increment k to $k + 1$. Go to step 1.
- Now that we have a match for an initial template, there must be a match for an extension template for this to be a regular anchor. Check for the match for the extension template (c) . If there is a match set $m = m + 2$ and repeat step 3 until there is no match. If there is no match, proceed to step 4.
- If there is no match, and we did not have any matches to the extension template, we don't have a regular anchor structure. Move the initial pointer k to $k + 1$ and go to step 1. However, if there was at least one extension template match, then set $m = m - 1$ (we moved two steps forward in step 3) and check for a match with the end template (e) .
- If there is a match to the end template (e) , we have successfully detected the regular anchor pattern. Set $m = m + 4$, $k = m$ and go to step 1.

In this manner, we can detect arbitrary length regular anchor patterns.

6.8 Experiments

In the sections that follow, we present our results on dialog detection and regular anchor detection.

6.8.1 Dialog detection



Figure 6.9: The figure shows an example dialog, where misses in the key-frame detection prevented the dialog from being detected. Here, misses occur after the 2nd and the 4th frames.

In this section, we present our dialog detection results. The statistical tests that are central to the dialog detection algorithm make it almost parameter free. These test are used at the standard levels of significance ($\alpha = 0.05$). The sliding window size T_w (6 frames).

Table 6.1: The table shows the dialogue detector results for the three films. The columns are: Hits, Misses, False Alarms, Recall and Precision.

Film	H	M	FA	Precision	Recall
Four Weddings and a Funeral	16	4	1	0.94	0.80
Pulp Fiction	11	2	2	0.84	0.84
Sense and Sensibility	28	3	0	1.00	0.91

The results of the dialog detector (ref. **Table 6.1**) show that it performs very well. The best result is a precision of 1.00 and recall of 0.91 for the film *Sense and Sensibility*. The

misses are primarily due to misses by the shot-detection algorithm. Missed key-frames will cause a periodic sequence to appear less structured.

6.8.2 Regular anchor detection

In this section we present our results on detecting the three element regular anchor. We choose to work with the same three films as before. The only parameter that we need to set is α , which we set to 0.75. (see equation (6.11)). The results for the regular anchor detection are shown in **Table 6.2**.

Table 6.2: The results for the regular anchor detection. The symbols are as follows: H: hits, M: misses, FA: false alarms, P: Precision and R: recall. The primed variables refer to the modified hits, and false alarms, after we take the results of the dialog detector into account.

Film	H	M	FA	P	R	H'	M'	FA'	P'	R'
Four Weddings and a Funeral	30	3	8	0.79	0.91	19	3	8	0.70	0.86
Pulp Fiction	10	4	2	0.83	0.71	5	4	1	0.83	0.55
Sense and Sensibility	34	4	4	0.89	0.89	14	4	3	0.82	0.78

In **Table 6.2**, we show two sets of results per film. The reason for that it as follows. When the detector is run, we get a lot of hits, and many of these hits are the boundaries of dialogs. This happens when the shot that we transition to from a dialog to the next shot has a large distance to the elements of the dialog, as well as the dialog element preceding the detected regular anchor element looks different from the other dialog elements (e.g. a different profile shot, or the shot may be at a slightly different scale). Note that all the detected structural elements satisfy the topological requirements of the

regular anchors, and are hence valid. So, the first set of results, are the “raw” results, without taking into account the results of the dialog detector. The second set eliminates all those detected regular anchors, that were part of dialogs.

The results of the regular anchor detection are good, with the best results for the film *Sense and Sensibility*. There were some interesting observations from the experiments.

- While examining the missed regular anchors, it became apparent that some of them were caused by shot-detector false alarms, which disrupted the topological structure by adding an extra image.
- Some of the false alarms, while accurate topologically, are deemed incorrect since either the first or the last element belongs to a different scene. The reason why they were deemed as regular anchors, is because there was a strong graphic match, and there was a very similar color distribution. i.e. the director uses a very similar spatial and color composition in the subsequent scene. This is because the similarity in composition and color will make the transition between scenes appear smooth.

We believe that the results of both detectors can be improved by changing the metric, since the distance between the images is computed in the present work using a simple color histogram distance. The metric plays an important role since the regular anchor (and the dialog) is really a graphic match i.e. the shapes *and* the colors in the anchor images are very similar. Hence a metric that incorporates the spatial information (e.g.

color coherence vectors, other shape features) should improve the regular anchor detector. However, while this may introduce a few extra false alarms, this scenario is preferable to having misses.

6.9 Related work

There has been some work done in the area of visual structure detection [101] [104] . This may be because much of the multimedia community has focused on the problem of detecting shots, scene segmentation or summarization.

In [104] , the authors detect dialogs in the following way. After detecting shots, they cluster the shots, and thus assign a label to each shot. Then, they look at a subsequence of these labels to see if there are two dominant labels. If these dominant labels exist, and satisfy a temporal ordering constraint, they declare the presence of a dialog. There are some additional constraints to eliminate “noise” labels. The papers do not present detection results in terms of precision and recall (they present the number of dialogs detected only), making it hard to judge the quality of their algorithm.

The important difference between the work in [101] [104] and topological analysis is that since that work relies on clustering before dialog detection, it would be difficult to detect structures where the metric relationships are arbitrarily specified (i.e. any number between 0 and 1; and not just 0 and 1). Such structures would be more amenable to detection by randomizing the topology. Secondly, more complicated structures may not

manifest themselves in a noticeable way in the STG; however regularities may be more visible in the topological matrix.

6.10 Summary

We now summarize the important aspects of this chapter, which focuses on detecting visual structure. We denoted a discrete sequence of labels to be structured, if the Kolmogorov complexity of the sequence was less than the sequence length. In this work we have focused on structures that satisfy the following two characteristics: (a) the structures formed by deterministic generative rules and (b) the structure is characterized by its topology, as a consequence of a metric associated with the sequence.

The idea of the topological graph, is central to our framework for detecting structure in a sequence. We showed that a topological matrix is associated with every such graph, and can serve to compare two topologies. Then, we discussed two structures — the dialog and the regular anchor and discussed the generative rules associated with each.

We detected dialogs in the following way. We constructed a transform that we called the periodic analysis transform, that exploited the regularities of the topological matrix associated with a dialog. Then, we used statistical tests to decide if the current sequence was a dialog. We described a sliding window algorithm that enabled us to detect dialogs of arbitrary length.

We detected regular anchors in the following way. Since the size of the three element regular anchor is small, we used the following intuition to detect regular anchor:

randomizing a structured sequence will destroy the structure in the sequence. Then, we showed how this randomization trick could be used to detect regular anchors of arbitrary size.

The experimental results indicated that both the dialog detection and the regular anchor detection worked well. However, the results in both cases can be improved, by taking into account new metrics that incorporate spatial information in addition to color.

Hence, there are two general approaches that one can use when detecting structure that is *a priori* known —

- Exploit the regularities in the topological matrix (like was done in the case of dialogs)
- Use the randomization of the topology, if there do not seem to be any regularities in the topological matrix.

The framework proposed in this chapter is novel, and is easily adapted to other problems that where the elements of the sequence have a metric associated with them. While the framework is extensible to other problems where the elements can be associated with a metric (e.g. detecting of footsteps in an audio sequence). However, the work presented in this chapter is the starting point of a theory of topological analysis of sequences; note that the at moment, we do not have a mechanism for detecting stochastic structures.

7 The summarization problem

7.1 Introduction

In this chapter, we introduce the problem of summarizing video. It builds on the prior work on computable scenes and structure detection, by generating a summary multimedia clip per computable scene. We repeat the definition of the summary as presented in the introduction. *The Oxford English Dictionary* defines the adjective form of the word summary²⁵ as:

Of a statement or account: Containing or comprising the chief points or the sum and substance of a matter; compendious (now usually with implication of brevity).

Note that the definition implies an understanding the semantics of the document (or video), and constructing a summary document (or video) based on this knowledge. Such a level of understanding for automatic analysis of the audio-visual data is difficult, since this requires access to a body of knowledge that is external to the data in the video.

²⁵ i.e. the word is to be used as in a “summary video” or a “summary text,” which in colloquial English is used as “video summary” and as “text summary” respectively.

The goals of a video summary include the construction of an audio-visual representational / visualization scheme that captures the underlying semantics of the scene (in the sense as presented above). Examples of such schemes include — image storyboards, slide shows and skims. An image storyboard is just a collection of key-frames laid out on a window; a slide show is a time-sequence of key-frames, accompanied with the audio. A skim is just a short audio-visual segment, that contains the important semantic elements of the original. Issues common to prior work include:

- There is no framework of for adaptation, in response to either changes in the user needs, or computational resources (device constraints, network conditions etc.)
- Fixed set of semantics (e.g. in the prior work on skims) i.e. the semantics of the summary is limited by the set of pattern detectors used.
- They do not examine the effect of the grammar underlying the produced data on the summary.

In this work, we propose a new conceptual framework for skim generation. In our worldview, the form of the summary is dependent on many factors — (a) the nature of the task (search vs. browsing), as this affects the kind of the information that the user is looking for (b) the capabilities of user interface and (c) the computational resources of the device rendering the summary.

Central to our skim generation approach is the entity-utility framework. The skims that we generate shall attempt to preserve certain *entities* that help satisfy the users

information needs. An entity is a real world “thing” with independent existence.

Entities consist of *attributes*; entities are constructed from the set of attributes by using an appropriate set of *predicates*. Associated with each entity is a utility function, and the skims are generated in a constrained utility maximization framework.

7.1.1 Scope of the problem examined in this thesis

We now indicate some of the issues related to summarization, that we did *not* consider in this thesis:

- We summarize the entire video, by individually summarizing each computable scene — i.e. we assume that each scene is independent of the other scenes. We do *not* exploit the inter-scene relationships that exist amongst scenes, at level of the whole video. For example, the first and the last scenes in a film could take place in a soccer field; then, knowing that the two locations are identical, we could more efficiently condense the second scene.[47] [48]
- We do not consider the use of text for the purpose of summarization. We decided against using text, since it is frequently unavailable in legacy video as well as in many foreign language productions. However, when available, text is an important attribute and ought to be part of the summarization scheme.
- We do not consider scene-level syntax for summarization. An example of such syntax is parallel story development — where the director develops the plot over two locations in a rhythmic fashion, alternating between the two locations.

Determining and exploiting such syntax is important since it will enable us to condense beyond what is possible by treating each scene independently.

- We shall limit the scope of the problem in this thesis to show adaptability to changes in target skim duration. However, the framework that we adopt is easily extensible (ref. [17]) to handle other characteristics such as device constraints, network bandwidth etc.

The rest of this chapter is organized as follows. In the next section we shall review several strategies for video summarization — (a) image based storyboards, (b) slide shows and (c) video skims. Then, in section 7.3, we shall discuss related work on audio summarization, mainly focusing on techniques used in the computational linguistics community. In section 7.4, we present our new conceptual framework for skim generation — the entity utility framework. Then, in section 7.5, we conclude by summarizing the main ideas in this chapter.

7.2 Related work: video

In this section we shall present an overview of the related work on video summarization. We shall discuss three visualization techniques — (a) image based storyboards, (b) slide shows and (c) video skims, which are the focus of research in this thesis.

7.2.1 Image based storyboards

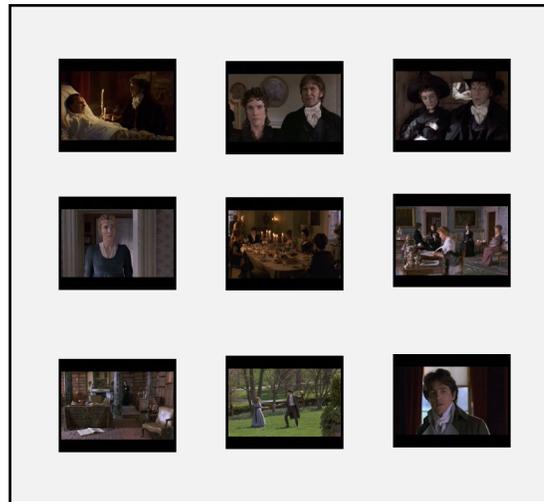


Figure 7.1: An image storyboard

Image based storyboards offer a key-frame based non-linear navigation of the video data [76] [92] [95] [102] . Briefly, all image based storyboard algorithms broadly share the following approach:

- First determine an appropriate feature space to represent the images (e.g. color histograms, edge orientation histograms)
- Select a clustering technique (e.g. k-means) to cluster the image sequence in the feature space.
- Compute a measure of importance to determine the appropriate key-frame to represent the cluster. For example, a common technique is to pick the key-frame closest to the centroid of the cluster. **Figure 7.1** shows an example of an image storyboard.

7.2.1.1 Video Manga

In this section, we review the Video Manga [95] approach to video scene visualization. The Manga project uses a measure of importance to select the important segments for display, and then introduces the novel idea of adaptive changing the key-frame size depending up the segment score.

In this approach the video data is first adaptively clustered (ref. section 2.3.2). Let us assume that there are k clusters, each cluster containing many video segments. In order to assign an importance measure to each segment, they proceed as follows. First they compute the weight of each cluster:

$$\omega_i = \frac{|C_i|}{\sum_{j=1}^k |C_j|}, \quad (7.1)$$

where C_i denotes the i^{th} cluster, the $||$ operator designates the duration of each cluster i.e. the sum of the durations of the constituent segments. Then, the importance of a segment is computed as follows:

$$I_j^k = L_j \log\left(\frac{1}{\omega_k}\right), \quad (7.2)$$

where, I is the importance measure assigned to segment j of the k^{th} cluster, L_j is the duration of the segment. Hence, a segment is deemed to be important if it is both long and rare. Segments with measures that are greater than one-eighth of the maximum

importance score are chosen to be part of the summary. The key-frame corresponding to each segment is chosen by selecting the frame corresponding to the center of the segment.

The authors set the *size* of the key-frames in proportion to the importance score. They use three discrete sizes corresponding to three score ranges for the key-frames. The largest size being three times the smallest size. The rationale for choosing variable key-frame sizes is that more important segments should be made more visible to the user, so that the user can access that portion of the video more quickly. The idea being that more prominent a key-frame, the viewer will notice it more quickly.

There are two criticisms of this approach:

- The importance based segment scoring can lead to problems in some cases. For example, consider the case of three clusters, with one segment per cluster. And the durations of the segments are $S_1 = 97$ sec. $S_2 = 2$ sec. and $S_3 = 1$ sec. In this case, equation (7.2) implies that S_2 is the most important segment, an unintuitive result given the dominance of segment S_1 . For example, S_1 could be a news anchor reading a story while S_2 and S_3 could be short clips showing a graphic.
- Making the size of the key-frame proportional to the importance of the segment, will have its disadvantages. What could easily happen is that the human eye while scanning the images fixates on the *smallest* image precisely because it appears difficult to comprehend because of its size. This criticism is just the visual analog to the familiar information theoretic argument that low probability events (in the

2D case, the size) have more information. Hence the viewer may decide to examine all the small key-frames before coming to a conclusion about the summary.

7.2.1.2 Scene transition graphs

In this section we review the scene transition graph as a visualization technique (in section 2.3.1, we discussed scene transition graphs as a method for video scene segmentation).

The scene transition graph (STG) [101] [102] [103] [104] (ref. **Figure 2.3**) offers a compact representation of the video. We can browse and navigate the video in a hierarchical, non-linear fashion. A STG shares similar characteristics to other image based storyboards in that it clusters frames in a feature space. However, by analyzing the shot transitions amongst the clusters, it provides for some of the temporal dynamics to be visualized in the storyboard. The analysis of the label transitions also allows the STG to detect elements of visual syntax, such as the dialog.

The problem with the scene transition graph is that like other image based storyboards, it offers a static snapshot of the entire video. While the transition edges do display temporal relations between the cluster, they do not provide an effective summary since they can be hard to interpret, unless the user is familiar with the semantics of the layout.

In the appendix section 13.3, we discuss methods to improve the static image based summaries.

7.2.2 Slide shows

A slide show is a video summary that is a time sequence of frames played back with audio. The sequence of frames, could just be the key-frames associated with each shot, or a subset of the set of the set of key-frames. Slideshows introduce an element of dynamism into the summary, but unlike a video skim, do not adequately recapture the experience of watching the original video. While user studies indicate that simple slideshows are not as preferable as other video summaries [38] , they may be useful over low-bandwidth networks.

7.2.3 Video skims

A video skim is a short audio-visual clip that summarizes the original video data. They are important because unlike the static, image-based video summaries, video skims preserve the dynamism of the original audio-visual data. Applications of audio-visual skims include:

- On demand summaries of the data stored in set-top boxes (interactive TV)
- Personalized summaries for mobile devices.
- News channels (e.g. CNN) that receive a tremendous amount of raw footage.
- Summarizing home video.

7.2.3.1 *Prior work*

In this work, we shall review work done in four projects — the Infromedia project at CMU [18] , the MoCA project [65] , and projects at Microsoft research [37] and at IBM Almaden [91] .

In the Infromedia skimming project[18] , important regions of the video were identified via a TF/IDF analysis of the transcript. They also used face detectors and performed motion analysis for additional cues. The user studies conducted in [18] gave mixed results, the “optimal” skim gave better results over the other skims being tested, but the differences were not statistically significant. The MoCA project [65] worked on automatic generation of film trailers. They used heuristics on the trailers, along with a set of rules to detect certain objects (e.g. faces) or events (e.g. explosions). It is difficult to evaluate the success of the MoCA project since they do mention the results of user studies. A conceptual issue with both the Infromedia and the MoCA projects is that they focus on detecting important patterns (e.g. faces etc.). Concatenating shots that just contain “important” patterns ignores the underlying grammar in the produced video, thus causing the resulting skim to appear incomprehensible.

Work at Microsoft Research [37] dealt with informational videos. They looked at slide changes, users statistics and pitch activity to detect important segments. In that work, they compared three algorithms: (a) skims based on slide transitions (b) a pitch activity based skim and (c) a skim derived from combination of usage statistics, slide transitions

and pitch activity. They found no statistically significant differences amongst the three methods. However, this could be a reflection of the particular domain that worked with.

Recent work [91] at IBM Almaden has dealt with the problem of preview generation by generating “interesting” regions based on viewer activity in conjunction with topical phrase detecting. The user activity was modeled by an HMM, where the output symbols represented the user actions, while the hidden nodes represented the user’s intent / satisfaction. When the summary of a new video was required, a few people need to watch it in order for the summary to be generated. Then, for each user, we decode the hidden states, and a majority vote is taken to decide the “interesting” segments.

7.3 Related work: audio

In this section, we shall discuss some of the related work in audio summarization. While there has been plenty of related work on audio scene segmentation and indexing, there has been little work done on summarizing generic auditory data. However, the problem of summarizing speech documents, has received much attention in the computational linguistics community [4] [40] [42] [33] [61] [81] [82] . Here we shall review *Speechskimmer*, [2] [4] Barry Arons’ work on summarizing speech, as well as review work on discourse segmentation.

7.3.1 Speechskimmer

In [2] [4] , the author describes *Speechskimmer*, a project that was geared towards summarizing spoken documents.

In [4] , the author uses pause information to skim the speech. Pauses are important indicators of semantic and syntactic cues within speech. In this work, pauses are used in two ways: pause shortening as well as the use of pauses to determine the start of syntactic boundaries. The key issue with pause based analysis is that pauses can be of two types — *hesitation* pauses as well as *grammatical* pauses. Grammatical pauses tend to be much longer than hesitation pauses, however the distributions of the two types of durations show considerable overlap [61] . Hence a threshold based on the pause duration alone, will often result in segments that correspond to mid sentence hesitation pauses. However, in [61] , the author points out that the pitch contour in the segment *prior* to the pre-segment pause is a good indicator of whether the pause is grammatical.

Skimming based on pitch analysis alone was discussed in [2] . There, the focus was on detecting emphasized portions of speech, since new topic introductions were often associated with an increase in the pitch of the speaker. The algorithm to detect the emphasized portions of speech was as follows. Pitch values were calculated over 10ms frames, were aggregated over one second windows. Then, once the pitch values were normalized for speaker variability, the top 1% frames were selected (i.e. the top 1% of the frames that have the highest pitch activity). Each one second window has a frame activity score — the number of frames above the pitch threshold. The activity scores are

aggregated over eight second windows. Then, these eight second windows are ranked in terms of their activity scores, thus resulting in an emphasized segment. While both [2] [4] have interesting approaches towards speech summarization, unfortunately neither of them presents user studies to evaluate the utility of the approach.

7.3.2 Discourse structure segmentation

In this section we shall review the structure of the spoken discourse, and the different approaches towards using the acoustic correlates of prosody [33] [40] [42] [81] [82] for discourse structure segmentation. Based on the work by [33], each of the intonational phrases are classified into five discourse categories [82]:

- Segment initial sister (SIS): The utterance beginning a new discourse segment that is introduced as the previous one is completed.
- Segment initial embedded (SIE): The utterance beginning a new discourse segment that is a subcomponent of the previous one.
- Segment medial (SM): An utterance in the middle of a discourse segment.
- Segment medial pop (SMP): The first utterance continuing a discourse segment after a subsegment has been completed.
- Segment final (SF): The last utterance in a discourse segment.

A single category of segment beginning utterances (SBEG's) is created by combining the SIE, SIS and SMP elements of the discourse [82]. Note that SBEG's represent the new topic introductions in the discourse.

The work done on acoustic correlates of prosody [33] [40] [81] [82] indicate that typically, SBEG's have a preceding pause that is significantly longer than for other phrases, higher initial pitch values (mean, variance), and smaller pauses that end the phrase than for other phrases. In [82], Lisa Stifelman uses a subset of these correlates of prosody to determine SBEG's in speech. It was used in the context of a note taking device *AudioNotes*, where it was important to determine topic boundaries. In her SBEG classification results, she obtains a precision of 44% and a recall of 59%.

Finally, it is important to note that there is an important difference between emphasis detection (e.g. [2]) and detection of SBEG's. Emphasis in speech can occur anywhere within the discourse, including SBEG's. Detecting SBEG's is important since they are an important aspect of the structure of the discourse.

7.4 An entity-utility framework

In this section we present a new conceptual formulation for the problem of skim generation. Then we shall show the application of this framework for a specific type of audio-visual skim. This conceptual formulation is needed for several reasons. While there has been much prior work in both image and video based summarization schemes [18]

[22] [32] [37] [52] [65] [76] [92] [95] [100] [101] [102] [106] they all focus very narrowly on creating summaries that attempt to preserve the semantics of the original.

However, while semantic level summarization is hard, importantly, prior work on summarization is also static, in the sense that neither the user information needs nor are the abilities of the device are taken into account when generating the summary. Our framework considers the audio-visual skim generation in terms of the following:

- Entities defined on audio-visual data, at different levels of abstraction,
- Utilities that are assigned to each entity.
- The user information needs.
- The device on which the skim is to be rendered.
- A skim generation mechanism that maximizes the utility of the entities considered.

An *entity* is a real world “thing” with independent existence²⁶. Entities could be concrete such as a video key-frame, or abstract such as a video segment titled “beauty.” Each entity is associated with a set of *attributes* and a *predicate* (i.e. a particular condition that specifies a relationship) on the set of attributes.

²⁶ The ideas for the entity relationship model are based in part on the familiar entity-relationship data model found in database analysis [27] .

Predicates can be caused by static (physical events, content producer) and dynamic (prior knowledge and user expectations) factors. The predicates can be of different types: (a) temporal, (b) syntactical, (c) semantic and (d) due to the conventions of the domain. Each entity is associated with a different utility function, that is altered depending on whether the entity is dropped or transformed.

A skim is short video clip containing the entities that satisfy the user's information needs (or tasks) as well the capabilities of the device on which it is being rendered. We can readily construct a partial taxonomy of skims: (a) semantic, (b) affect, (c) events and (d) discourse centric. These skims differ on their computability and in their usefulness to the viewer. In this work, we have focused on discourse centric skims (i.e. skims that focus on preserving SBEG's) since they are computable.

The rest of this section is organized as follows. In the next section, we shall define the notion of an entity and discuss some of the causes as well as the predicate relationships. In section 7.4.2, we shall discuss the formulation of the skim in terms of the entities that the user seeks. In section 7.4.3, we shall present the skim generation goals.

7.4.1 What is an entity?

An *entity* is a real world "thing" with independent existence. Entities could be concrete such as a video key-frame, or abstract such as a video segment titled "beauty." Each entity is associated with a set of *attributes* and a *predicate* on the set of attributes. For example, a video frame is an entity with attributes of the set of pixels (each of assume a

certain *value*.), where the predicate specifies the relative positions of the constituent pixels. It is instructive to think of an entity to be associated with a “bag” of items; the predicates then help select items from this bag, to construct the entity.

We can also construct entities from other entities in a hierarchical fashion. For example, a video shot is a collection of key-frames with predicates of start time, end time, and ordering (i.e. the exact sequence of frames) requirements. A dialog is an entity that comprises video shots with predicates of selection, topology and time ordering.

Hence, an entity is associated with an attribute set, and the predicates construct a single subset that defines a particular entity instance. For example, let us assume that we have a video segment that has N shots. Then, the k^{th} shot can be defined as follows:

$$V_k = \{S \in P(f) \mid f_s \geq \text{start}(k), f_s \leq \text{end}(k), \text{continuity}(S, k) = 1, \} \quad (7.3)$$

The k^{th} shot is the subset S of the powerset²⁷ P of the video frames f , such that each frame f_s belonging to the set S satisfies the start and the end requirements of the shot. The frames f_s must also satisfy continuity requirements — for every frame other than the last frame, the successor frame must also be present. Depending upon the predicate, entities can seem to exist at many different levels — shots, syntactical elements, semantic

²⁷ This is just the set of all subsets of a given set. For example, if the set $S = \{a, b\}$, then the powerset of S is $\{\{a, b\}, \{a\}, \{b\}, \{\emptyset\}\}$.

elements etc. We now discuss the causes of relationships between modalities and also discuss a few of the different types of relations.

7.4.1.1 *Causes of predicates*

Entities defined via relations on elements across different modalities can occur due to at least four different factors:

- The relationship is caused by a physical event (e.g. the firing of a gun).
- 1. The relationship is due to the creator of the content — the director shows a couple with romantic music on the audio, thus setting the “mood.”
- The prior knowledge of the viewer of the physical events shown as well as the conventions of the domain.
- The expectations of the viewer. Often the *interpretation* of the audio-visual data will be predicated on what the viewer wants; as a consequence, the viewer will construct relationships in an attempt to fulfill his expectations from the data.

We term the first two predicates to be *static* and the last two as *dynamic*. This is because the relations defined in the first two points deal with relationships that have been inserted statically by the content producer. However, in the last two cases, regardless of the static relations in the video, the viewer’s prior knowledge and expectations change the relations (or create new ones) amongst the entities.

7.4.1.2 *Types of predicates*

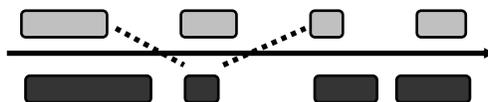


Figure 7.2: video and audio entities. The dotted line indicates a relation amongst the low-level entities.

In this section, we discuss some of the different types of relations that exist within and across modalities.

Temporal: These relations (which can occur within and across modalities) can either be (a) synchronous (e.g. lip movement accompanied by speech) or (b) causal (e.g. sounds of a doorbell followed by a visual of the door opening).

Syntactic: A topological relationship between elements — this can be amongst and within modalities. Examples of topological relations include the dialog, and the footsteps (i.e. visuals followed by a sound).

Semantic: These are high-level relations that exist due to the content producer and due to the prior knowledge of the viewer.

Film conventions: There are relations that exist in film that arise out of the director's desire to create a certain affect (i.e. emotional response). We list a few of them below:

- **Graphic:** In this form, the director will ensure that objects appear in certain locations of the video, in a periodic manner (e.g. location of the principals in a dialog).
- **Rhythmic:** The director can induce a certain rhythm to the shot sequence by carefully selecting the durations of the constituent shots.
- **Spatial:** In the absence of an establishing shot, the audience will spatially connect the shots in the sequence (i.e. the audience will assume that the shots are in the same physical location; this is also known as the Kuleshov effect [8]).

Note that different domains of production will contain different rules of syntax. For example, in soccer, the content producer will return to a zoom-out view (thus showing a global game state), after showing close ups and medium-shots.

7.4.2 Skims and entities

In this section, we first discuss factors affecting skims and then we shall formulate the skim in terms of the entities found in the data, the user needs and the device on which the skim is to be rendered.

7.4.2.1 *Factors that affect skims*

There are at least two factors that affect the skim generation algorithm — the task of the user and the device constraints. We divide up tasks into two broad categories: *active* and *passive* tasks. A task is defined to be active when the user requires certain information to

be present in the final summary (e.g. “find me all videos that contain Colin Powell.”).

In a passive task, the user does not have anything specific in mind, and is more interested in consuming the information. Examples include previews in a set-top box environment, browsing in a video digital library.

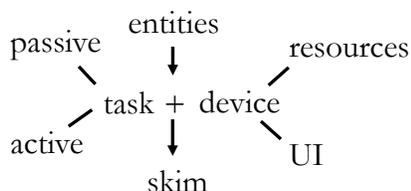


Figure 7.3: a skim depends on the user’s information needs and the device.

The device on which the skim is to be rendered affects the skim in at least two ways: the nature of the user interface and the device constraints. The UI can be complex (e.g. the PC), medium (e.g. a palm pilot) and simple (e.g. a cell phone). The UI affects the resolution of the skim, and also influences the kinds of tasks that the user has in mind (e.g. it is difficult to input a query on a cell phone).

The computational resources available on the device — cpu speed, memory, bandwidth, availability of a video rendering device, all affect the skim in the following ways: the resolution of the skim, as well as the decision to include video in the skim.

7.4.2.2 *Entities, tasks and the device*

A skim can be viewed as a time compressed audio-visual clip that contains those entities (from the original video) that best matches the information needs (or tasks) of the user and the capabilities of the device on which it is being rendered (see **Figure 7.3**). For

example, consider the problem of generating a summary for a baseball game that contains MPEG-7 metadata. Then, independent of the viewer, the video data contains entities that arise due to physical events, the semantics of the game as well as due to the content producer.

Now, let us assume that we have a viewer who is interested in watching a summary of a Yankees game highlighting the performance of the short-stop Derek Jeter. Then, the summary must contain all the entities relevant to the user's information needs — (a) events that change the game score (b) the entities that include this player (e.g. hitting, running around bases, fielding the ball and throwing someone out). Now, if the device in question cannot render video (but only audio and still images), then many of the entities that the user has requested, cannot be fulfilled (e.g. causal entities such as hit followed by speech).

7.4.2.3 *Taxonomy: skims come in different flavors*

In this section we attempt to identify some of the different skim forms and for each skim type, we also discuss its computability and usefulness to the viewer.

Semantic: Here, we attempt to preserve the key semantics in the data. These could be specified by the user (in the form of a query — “What did George Bush do today?”), or the content producer, who may specify (via MPEG-7 metatags) the content to be retained in the skim. This skim type is the closest to what the user wants, but in general,

it is very difficult to compute the entities required to satisfy the user needs. However, if the content producer inserts MPEG-7 metatags, generating the skim may be feasible.

Affect based: In this form, the user is interested in a skim that retains the “mood” or the affect [1] generated by the content producer. As an example, the heightened pace (arising from the sequence of fast cuts) during an action sequence can be maintained by preserving the film rhythm. This is useful for example, in creating movie previews. The viewer may be interested in knowing if the film is exciting, sad etc. While there has been some work done in computing affects [1], this is a challenging area of research.

Events: An event refers to the change of state (or the change in property) of an entity. This skim will contain a subset of all the events in the domain. For example, the user could specify that in the skim of a particular soccer game, only those events pertaining to a particular soccer player be retained. This is particularly useful in constrained domains (e.g. baseball videos) where it is possible to construct high-performance event detectors. Note that while events can have specific semantics associated with them, a semantic skim attempts to answer higher level questions. The answers will comprise as events well as non-events.

Discourse centric: This skim will attempt to parse the discourse structure of the speech in the video, and determine the most significant audio segments using prosody analysis. While this skim type can be computed automatically, detecting significant phrases in noisy environments is still a very challenging task. The assumption that speech conveys semantic information that is most easily understood, may not be true — if we drop

related speech segments, then the segment chosen for the summary may not be entirely clear.

7.4.2.4 *Entities and utilities*

In this thesis, we shall preserve five entities:

- Elements of syntax, as they are critical in keeping the skim intelligible.
- Speech segments, since they are the most direct form of the high-level entities present in the film (as opposed the visuals, for example.)
- Synchronous entities: these entities ensure that the audio and the video data are synchronous, thereby increasing the coherence of the skim.
- Segment coherence: we must ensure that each atomic audio or video segment is coherent.
- Film rhythm: The relationship between the durations of the shots in the sequence; this is a form of affect [1] .

In our framework, we associate a utility function with each entity. The utility is a measure of the entity's contribution towards the information that the user desires. In our particular implementation, while we shall associate explicit utility functions with the atomic audio and the video entities to ensure coherence, as well as the film rhythm entity, we shall see in chapter 10, that it is not necessary to construct explicit utility functions

with the other entities. We shall maximize their presence, in the process of searching for the optimal solution.

In general, the utilities that shall be associated with the entities need to be determined by an expert, particularly if the effects of the variation of the parameters of an entity have perceptual effects.

7.4.3 Skim generation goals

The goal of this work is the automatic generation of audio-visual skims for *passive* tasks, that summarize the video. This is to be achieved via a constrained utility maximization of the entities that satisfy the users information needs. While constructing the skim, we make the following assumptions:

- We do *not* know the semantics of the original.
- The data is not a raw stream (e.g. home videos), but is the result of an editing process (e.g. films, news).
- The time that the user has to watch the skim is known.

The assumption of the data stream being produced is an important one, since we shall attempt to preserve the grammar of the underlying produced video, so as to preserve meaning. The research issues that emerge in the entity-utility framework are as follows:

- How do we automatically determine a correspondence between the user information needs and the entities that can be found in the data?

- How to determine the “correct” form of the utility function? i.e. the formulation of the correct relationship between the parameters of the entity, to reflect our goals.
- How do entities across that exist in different media, affect each other’s utility function, when they appear at the same time?

Since we work on passive tasks, the user is not dynamically changing the entities that ought to be present in the skim. Based on our analysis of the desirable characteristics of passive skims, we can determine a set of entities that ought to be present in such skims.

7.5 Summary

In this chapter, we introduced the problem of video summarization, presented related work on video and audio summarizations and also presented a new conceptual formulation of the problem.

We presented related work on video summaries. We discussed image based story boards. There we discussed scene transition graphs and the Video Manga project that used adaptive key-frame sizes. In the context of video skims, we discussed work done at the Informedia and the MoCA projects, and showed some conceptual difficulties with their approach.

We presented a discussion on the audio summarization techniques. We focused on the work done in the computational linguistics community on discourse based segmentation. A particular element of the discourse SBEG, is important as these phrases often

represent topic changes, both in read and in spontaneous speech. We discussed that these elements of structure show strong correlates to the pitch and the pause durations associated with the segment, leading a way to detect them.

In this chapter we formulated the problem of video summarization as one of a constrained utility maximization of the entities that satisfy the user information needs. We defined entities to comprise attributes, that are selected using predicates to define a particular instance of an entity. Then we discussed the causes of the different kinds of predicates and also discussed the different types of predicates that lead to the formation of different entity types.

We discussed the different factors affecting skim generation — the task, the user interface and the device constraints. We discussed the different skim types — (a) semantic, (b) events and (c) discourse centric. This work focuses on generating discourse centric skims. Finally, we discussed the skim generation goals.

While we have chosen to focus on a particular subset of relationships amongst the entities, the nature the task (passive) and made assumptions about the device, the utility maximization framework is very general and is easily extended [17] . For example, if we are to render the audio visual data on which the audio device is malfunctioning, then we just have to set the utility of the audio segments to zero in the maximization. i.e. the utility functions of the entities that we are trying to preserve in the skim are easily modulated by device and network characteristics.

8 Skims: Visual analysis

8.1 Introduction

In this chapter, we present our approach to the problem of condensing video data. The solution uses two key intuitions — (a) we can reduce the duration of each shot, by analyzing the visual complexity of that shot and (b) the visual coherence of produced film data arises from the use of film grammar in constructing these video sequences.

The motivation for the use of visual complexity of a shot arises from empirical observations in film making as well as from experimental psychology. We define visual complexity using the idea of Kolmogorov complexity and show that it can be estimated using the Lempel-Ziv algorithm. A psychological experiment, helps relate the minimum amount of time required for comprehending a shot to the visual complexity of the shot.

Why is the underlying film grammar important? The shots comprising a scene in a film are highly fragmented in the sense that each shot shows only a portion of the entire scene. Yet the viewers manage to effortlessly “interpolate” these shots to construct an internal representation of the scene. The reason why this is possible is due to the use of grammatical rules in film construction. An important goal in this work is to reduce these syntactical elements in a manner such that their essential meaning is preserved. This is

done by analyzing the techniques for film production, and determining the minimum number of shots required for that element to be interpreted correctly by the viewer.

The rest of this chapter is organized as follows. In the next section we discuss the idea of visual complexity. We present motivating examples, a definition and sections that show how we can estimate the visual complexity of a shot. Then, in section 8.3, we present a detailed discussion on the importance of syntax in visual data, and how we can reduce these syntactical elements while retaining their essential semantics. In section 8.4, we present a discussion on the issues raised in this chapter. Finally, in section 8.5 we summarize the main results in the chapter.

8.2 Visual complexity

In this section, we shall discuss the relationship between visual complexity of an image and its time for comprehension. We begin with insights that stem from film-making and experimental psychology. Then, we shall define a measure of visual complexity and show how it can be estimated using standard compression algorithms. Then, we shall relate the comprehension time for a shot to its visual complexity via a psychological experiment and then obtain time-complexity bounds.

8.2.1 Insights from film-making and experimental psychology

In this section we shall discuss empirical observations in film-making followed by a brief review of recent experiments in psychology that indicate a relationship between concept complexity and concept learnability.

other hand, is usually filled with detailed information which requires eye-scanning over the entire tableau. The latter takes time to do, thus robbing it of screen time.

This phenomena is exploited by film-makers, allowing them to dilate and condense time based on visual effects alone. For example, in action or chase sequences, directors use shots with a lot of motion, and visual detail, that only lasts for a short while. This causes the audience to be always “behind” in understanding the sequence, thus heightening anxiety.

8.2.1.2 Concept complexity and learnability

In [28] , Jacob Feldman conducted a series of fascinating experiments on the relationship between the subjective difficulty in learning a Boolean concept and its Boolean complexity. For example, consider a concept C with three features and three positive examples:

$$C = a'b'c' + a'b'c + a'bc' \quad (8.1)$$

where, the literals a , b and c could represent a : round / not-round, b : moving / not-moving, and c : red / not-red. This formula can be easily reduced to its irreducible form by using the rules of Boolean logic :

$$C = a'(bc)' \quad (8.2)$$

The *Boolean complexity* of this concept C is defined to be three, since it has three literals in its irreducible form. More generally, a concept having n literals in its irreducible form has

Boolean complexity n . Concept C is from the 3[3] family, where each concept is defined as disjunction of three positive examples, and where each example is represented using three literals. More generally, a concept belongs to a D[P] family, where concepts are defined using P positive examples, using D features.

We also need to introduce the concept of parity. A concept belonging to the D[P] family could be defined using P positive examples and $2^D - P$ negative examples, or vice-versa; note that in both cases, the concept has the same Boolean complexity. Feldman defines up-parity to be the case when $P < 2^{D-1}$, and down-parity to be the case when $P \geq 2^{D-1}$.

Feldman conducted a series of experiments on human subjects, for different D[P] concept families. His main conclusion was that the subjective difficulty in learning a concept is well predicted by its Boolean complexity of the concept i.e. to its logical incompressibility. For concepts with the same Boolean complexity, the subjective difficulty was ordered by parity — i.e. concepts with up-parity were learnt more easily than concepts with down-parity. While parity and Boolean complexity accounted for most of the observations in the experiments, there was residual variance to suggest the presence of other factors.

Clearly, there is empirical evidence from film-making as well as experimental evidence from psychology to indicate a link between the visual complexity of a shot and its comprehensibility.

8.2.2 Defining visual complexity

We define the visual complexity of a shot to be its Kolmogorov complexity. Let \mathbf{x} be a finite length binary string of length n . Let $\mathbf{U}(p)$ denote the output of an universal Turing machine \mathbf{U} when input with program p . Note, a universal Turing machine \mathbf{U} is a Turing machine that can imitate the behavior of any other Turing machine \mathbf{T} . It is a fundamental result that such machines exist and can be constructed effectively [51]. Then, the Kolmogorov complexity of the string \mathbf{x} is defined in the following way:

$$K_{\mathbf{U}}(\mathbf{x} | n) \triangleq \min_{p: \mathbf{U}(p)=\mathbf{x}} l(p), \quad (8.3)$$

where, $l(p)$ is the length of the program p , and n is the length of the string \mathbf{x} and where $K_{\mathbf{U}}(\mathbf{x} | n)$ is the Kolmogorov complexity of \mathbf{x} given n . Hence, the Kolmogorov complexity of \mathbf{x} , with respect to an universal Turing machine \mathbf{U} is the length of the shortest program that generates \mathbf{x} . The Kolmogorov complexity of an arbitrary string \mathbf{x} is non-computable due to the non-existence of an algorithm to solve the halting problem [21] [28].

8.2.2.1 Bounds on Kolmogorov complexity

In this section we show how to generate an asymptotically efficient bound on Kolmogorov complexity using Lempel-Ziv encoding [21]. Lempel-Ziv encoding is a form of universal data coding that doesn't depend on the distribution of the source. We now prove a simple lemma, to show how Kolmogorov complexity can be asymptotically

upper-bounded by the Lempel-Ziv codeword. The proof involves the use of two theorems, which we shall state without proof, since they can be found in [21].

Theorem 8.1 *Let the stochastic process $\{X_i\}$ be drawn i.i.d according to the probability mass function $f(x)$, $x \in \mathfrak{N}$, where \mathfrak{N} is a finite alphabet. Let $f(x^n) = \prod_{i=1}^n f(x_i)$. Then there exists a constant c such that*

$$H(X) \leq \frac{1}{n} \sum_{x^n} f(x^n) K(x^n | n) \leq H(X) + \frac{|\mathfrak{N}| \log n}{n} + \frac{c}{n} \quad (8.4)$$

for all n . Thus

$$E \frac{1}{n} K(X^n | n) \rightarrow H(X). \quad (8.5)$$

where, $|\mathfrak{N}|$ is the size of the alphabet, $H(X)$ is the entropy of the stochastic source.

Theorem 8.2 *Let $\{X_i\}_{-\infty}^{\infty}$ be a stationary ergodic process. Let $l(X_1, X_2, \dots, X_n)$ be the Lempel-Ziv codeword length associated with X_1, X_2, \dots, X_n . Then*

$$\limsup_{n \rightarrow \infty} \frac{1}{n} l(X_1, X_2, \dots, X_n) \leq H(X) \quad \text{with probability 1.} \quad (8.6)$$

where $H(X)$ is the entropy rate of the process.

Now we can easily show the following lemma.

Lemma 8.1 Let $\{X_i\}_{-\infty}^{\infty}$ be a stationary, ergodic process over a finite discrete sized alphabet \mathfrak{S} .

Let $l_{LZ}(\mathbf{X})$ be the Lempel-Ziv codeword length of a string \mathbf{X} , where $\mathbf{X} = \{X_1, X_2, \dots, X_n\}$. Then,

$$\lim_{n \rightarrow \infty} \frac{1}{n} l_{LZ}(\mathbf{X}) \rightarrow \frac{1}{n} K_U(\mathbf{X} | n). \quad (8.7)$$

Proof: By the definition of Kolmogorov complexity, the following inequality is true:

$$K_U(\mathbf{X} | n) \leq l_{LZ}(\mathbf{X}) + c. \text{ The result then follows immediately from theorems 8.1 and 8.2.}$$

Hence, we can use the Lempel-Ziv compression algorithm to upper bound the visual complexity of a shot. Note however, that the bound is only asymptotic, hence for any finite sequence the Lempel-Ziv universal coding scheme may not be efficient.

8.2.2.2 Estimating visual complexity

In this work, we tested two lossless compressors — *gzip* [110] based on the Lempel-Ziv (LZ77) algorithm and *bzip2* [111] based on the recent Burrows-Wheeler transform (BWT) [14]. The algorithms based on the BWT have exhibited greater compression rates on files than those based on the well known Lempel-Ziv algorithm. In our experiments, the size of the *bzip2* codeword was typically 5 ~ 10 % smaller than the corresponding *gzip* codeword. *Bzip2* also produced a smaller codeword than *gzip* for every image in our corpus. Hence we estimated Kolmogorov complexity using *bzip2*.

The output of a compressor operating on an image I is a *program*. This program, when input to a universal Turing machine emulating the decompressor will decompress the

program to output the original image I . Hence the length of the BWT codeword for the image I is just the size of the bzip2-ped file in bytes. We normalize the complexity by dividing it by the image size.

The images were preprocessed to generate a header-less file. Each image in the corpus (3600 images) was converted to a gray-scale image. This is because the specific order in which the color channels are stored in the file will affect the compressibility of the file. For example, we could store the color information of each pixel as a RGB triple, or we could store the entire red channel first, then the green channel, finally followed by the blue channel. Other storage schemes are also possible. Since these effects are not central to the thesis that is being tested, we converted the images to gray-scale.

The *raster-order* in which the gray-scale image is stored in the file also affects its compressibility. This is because *bzip2* (and *gzip*) operates on one dimensional sequences. In this paper, we compute the compressibility of images stored in row-major and column-major form and chose the order that minimized the size of the resulting codeword.

8.2.3 Complexity and comprehension

In this section we discuss an experiment that helps determine the relationship between complexity and comprehension time. We conducted our experiments over a corpus of over 3600 shots from six films. A shot was chosen at random (with replacement) and then its key-frame presented to the subject (the author). We choose the 5th frame after

the beginning of the shot, to be its key-frame. We acknowledge that there exists a considerable debate in the community on the choice of the representative key-frame. Representing each shot by its key-frame is reasonable since our shot detection algorithm [109], is sensitive to changes in color and motion.

Then, we measured the time to answer the following four questions (in randomized order), in an interactive session:

- who: [man / woman / couple / people]
- when: [morning / evening / afternoon]
- what: [any verb e.g. looking, walking]
- where: [inside / outside].



Figure 8.2: A typical image used in the experiment.

The subject was expected to answer the questions in minimum time *and* get all four answers right. This prevented the subject from responding immediately. Note that questions such as “How?” or “Why?” were not used in the experiment since they cannot be answered by viewing just one image. Such questions need an extended context (at least a few shots) for an answer. In the example image shown in **Figure 8.2**, the answers

to the four questions would be who: people, where: outside, when: can't say, what: jumping / standing.

We conducted ten sessions (to avoid fatigue), where the subject was questioned on 100 key-frames. In the end, we had the reaction times to 883 unique shots (we averaged the reaction times over duplicates). For each question, “none” or “ambiguous” was an acceptable answer. The response times in such cases are valid since they measure the time to reach a decision.

8.2.4 Time-complexity bounds

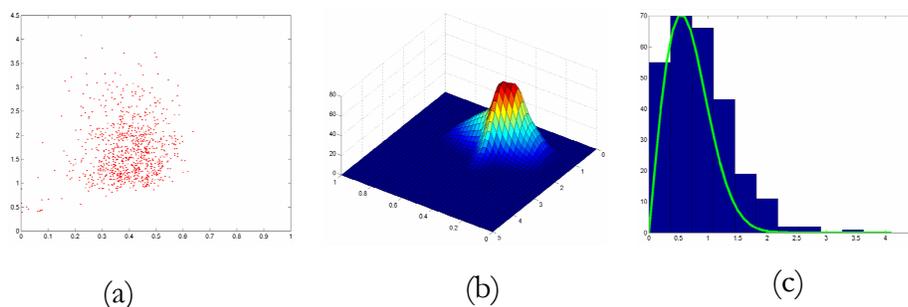


Figure 8.3: (a) avg. comprehension time vs. normalized complexity (b) histogram plot of avg. comprehension time across different complexity bins (c) one slice from plot (b) suggests a Rayleigh distribution. Plots (a) and (b) show time against visual complexity, while in plot (c), we plot a slice of the density for a specific value of complexity.

We now analyze the relationship between the average comprehension time (i.e. the average of the times to answer who? where? what? and when?) of an image \mathbf{X} and the normalized complexity of the key-frame (i.e.: $K(\mathbf{X})/N$, where N is the image size).

Figure 8.3 (a) shows a plot of the average comprehension time versus visual complexity. If we generate histograms of the average comprehension time by discretizing the

complexity axis, then each histogram slice is well modeled by a Rayleigh distribution (**Figure 8.3** (b)-(c)). By using the 95th percentile cut-off for each histogram we get an estimate of the *upper-bound* on the comprehension time. The *lower-bound* on the comprehension time is generated by determining a least squares fit to the minimum time in each histogram. The resulting bounds are shown in **Figure 8.4**. The equations for the lines are as follows:

$$\begin{aligned} U_b(c) &= 2.40c + 1.11, \\ L_b(c) &= 0.61c + 0.68, \end{aligned} \tag{8.8}$$

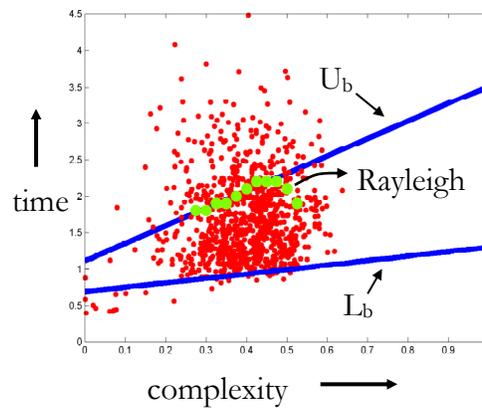


Figure 8.4: Avg. comprehension time (sec.) vs. normalized complexity (x-axis) showing comprehension (upper/lower) bounds. The dots in the figure represent the Rayleigh (95th percentile) bounds.

where c is the normalized complexity and U_b and L_b are the upper and lower bounds respectively, in sec. The lines were estimated for $c \in [0.25, 0.55]$ (since most of the data lies in this range) and then extrapolated.

Let us assume that we are given a shot of duration t_0 and normalized complexity c . Then, if condense it to at most $U_b(c)$ sec by removing the last $t_0 - U_b(c)$ sec., we can be sure

that the resulting shot would be comprehensible 95% of the time. Note that if the target time of the shot is less than the upper bound U_b for that shot, we use the upper bound. However, if the original length of the shot is less than the upper bound it is *not* reduced any further.

8.2.4.1 Notes on the time complexity relationship

There are several interesting observations to be made from **Figure 8.3** and equation (8.8).

- Both the lower and upper bounds *increase* with complexity, keeping in line with intuition.
- The key-frames used in the experiments are from six films with widely varying content. It is interesting to note that the average reaction time is *not* uniformly distributed per complexity bin.
- The existence of the lower bound indicates that regardless of the complexity of the shot, a minimum reaction time is required for comprehension.

The upper bound comprehension time is actually a conservative bound. This is because of two reasons:

- There is an implicit assumption in our method that the shots as viewed by the viewer are i.i.d. In this work, we do not incorporate into our analysis, the effect of structure (e.g. dialogs) or temporal correlation on comprehension time. Hence, we should expect that after the first shot in a scene, time to comprehend the

remainder of the shots will be less than the amount of time required to decode those shots independently. For example in the case of the familiar dialog (A-B-A-B-A-B), once we've seen the first image, it should be easier to comprehend the third image in the sequence.

- In the experiments, the subject *consciously* attempts to answer the four questions. However, while watching an film, we do not do so. Hence, we should expect these experimental times to reflect an additional “conscious processing” time.

8.3 Syntax analysis

In this section we shall give a brief overview of what constitutes “film syntax.” Then, we shall discuss why it becomes useful when analyzing films. Then we shall give algorithms for time condensation with two specific syntactic elements.

8.3.1 What are the elements of syntax?

An examination of the sequence of shots in a scene reveals that at any given moment, the shots represent fragments of the whole scene. i.e. they only show a small portion of entire setting at any one time. This is to be contrasted with viewing actors on the stage in a play. There, we see whole set, all the time. How do viewers, who watch the scene, effortlessly interpolate between these shots? The answer lies in understanding film syntax.

Film syntax refers to the specific arrangement of shots so as bring out their mutual relationship [77] . In practice, this takes on many forms (for more examples see chapter 2, [77]):

- Varying the shot duration, to direct attention.
- Changing the scale of the shot (there are “golden ratios” concerning the distribution of scale).
- The specific ordering of the shots (this influences the meaning).

These syntactical “rules” may appear *ad-hoc* but it is important to keep in mind that they have been developed by trial and error by film-makers over the past hundred years. It is the syntax of the film that lets us piece together the information contained in the shots.

8.3.2 Understanding the role of syntax

In order to appreciate the importance of syntax, we need to contrast shots with words in a written document. Words have more or less fixed meanings and their position in a sentence is driven by the grammar of that language. In films however, it is the *phrase* (a sequence of shots) that is the fundamental semantic unit. Each shot can have a multitude of meanings, that gets clarified only in relation to other shots. It is the film syntax that provides meaning to the filmic phrase.

The Informedia [18] and the MoCA [65] projects use object detectors (e.g. face detectors etc.) over the entire video, for selecting important segments. In the Informedia project the audio was selected by first performing a TF-IDF analysis of the transcript

and then selecting the complete phrase surrounding the highly ranked words. The resulting “best” skim did better than other types of skims but fared significantly worse than the full video. In the MoCA project, faces (particularly close-ups) were considered important and preserved in the skim; they also used specific audio event detectors (e.g. explosions etc.) to locate audio segments that they considered important.

An object detector based approach (e.g. Informedia, MoCA) to skims, for films, at a conceptual level, makes the analogy “shots as words.” However, this is in contrast to the way film-makers create a scene, where the syntax provides (or changes) the meaning of the shot sequence. Hence, deciphering the film syntax ought to give us a mechanism to reduce data while preserving the essential semantics. Secondly, since films vary widely in their content, we face the question of the number and the kinds of detectors to implement. Using a limited number of object detectors will imply that every summary ought to contain a subset of those patterns.

The Informedia and the MoCA projects analyze data over the *entire* video. However, they do not perform scene level analysis for skim generation. In this thesis, we analyze the data *within* one scene. In future work, we plan on utilizing the interesting syntactical relationships amongst scenes that exist in the video [77], for condensation.

8.3.3 Rules for syntax based removal

In this section, we develop rules for syntax based reduction for two elements — (a) the progressive phrase and (b) the dialog.

We define a phrase to be a sequence of shots designed to convey a particular semantic. A progressive phrase is a linear progression of visuals without any repetitive structure. For example, consider the following scene: Alice enters the room looking for a book. We see the following sequence of shots (a) she enters the room (b) she examines her book-shelf (c) looks under the bed (d) locates the book and the camera follows her as she sits on the sofa to read. A dialog is the familiar structure discussed in depth in section 6.5.1, is just a repeating A-B-A-B-A-B pattern of canonical images.

Table 8.1: Three types of syntax reductions that depend on the element (dialog/progressive) and the number of shots k .

Element	Min. phrase length	Breakpoints for each type		
		I	II	III
Dialog	6	$k \leq 15$	$15 < k < 30$	$k \geq 30$
Progressive	3	$k \leq 6$	$6 < k < 15$	$k \geq 15$

According to the rules of cinematic syntax, a phrase must have at least three shots:

Two well chosen shots will create expectations of the development of narrative; the third well-chosen shot will resolve those expectations. [77] .

Sharff also notes that depicting a meaningful conversation between m people requires at least $3m$ shots. Hence in a dialogue that shows two participants, this rule implies that we must have a minimum of six shots.

Let us assume that we have a scene that has k shots. Then, we perform three types of syntax reductions (break points based on heuristics) based on the number of shots k

(**Table 8.1**). The number and the location of the dropped shots depend on k and the syntax element (i.e. dialog or progressive). It is reasonable to expect that the number of phrases in a scene, increase with the number of shots. In the following discussion, we examine three types of scenes using a fictional film with a character called Alice.

For short scenes (type I reduction) we assume that there is a single phrase, containing one principal idea, in the scene. For example, the director could show Alice, walking back to her apartment, in a short scene.

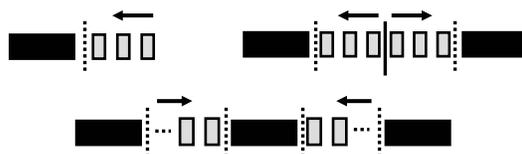


Figure 8.5: Three syntax reduction mechanisms. The black boxes are the minimal phrases and will not be dropped, while the gray shots can be dropped.

In scenes of medium duration (type II reduction) we assume that there are at most two phrases. For example, < 1st phrase >: Alice could be shown entering her room, switching on the lights, and be shown thinking. < 2nd phrase >: then, she is shown walking to the shelves looking for a book, and is then shown with the book. We assume that scenes of long duration, (type III reduction) contain at most three phrases. Modifying the previous example — < 1st phrase >: Alice is shown entering the room, < 2nd phrase >: she is shown searching for the book, < 3rd phrase >: she walks with the book to her desk and makes a phone call. Hence, the reduction attempts to capture the

phrase in the middle and the two end phrases. Unlike written text, there are no obvious visual²⁹ “punctuation marks” in the shots to indicate a “phrase change.” Hence our syntax reduction strategy, which will capture the phrases in scenes of short and medium duration, may cause error in scenes of long duration.

We now explain the shot dropping strategy. In type I reduction, (**Figure 8.5** (I)–(III)) we drop shots from the right, since the director sets up the context of the scene using the initial shots. In type II, we expect an initial context, followed by a conclusion. Here, we start dropping shots from the middle, towards the ends. In type III, the scene is divided into three equal segments, and shots are dropped from the two interior segment boundaries. This strategy thus preserves the important phrases that begin, conclude the segment as well as preserving important portions of the middle phrase.

8.3.4 Dealing with shot detector uncertainty

In this section we show how the errors in shot detection algorithms affect the rules of syntax. Practical shot-detection algorithms have misses and false alarms, each of which has a different effect on skim generation:

- Shot misses can cause our shot condensation algorithm (ref. Section 8.2.4) to remove a missed shot. For example, consider a progressive phrase that has five

²⁹ However, we may be able to detect a “phrase change” within a scene, by analyzing the audio and the transcript.

shots, however the last shot was not detected, leaving us with only four shots.

Then, if we use the shot condensation algorithm discussed in section 8.2.4, in the case of maximum condensation, we would entirely remove shot five from the skim, since the key-frame shot four would dictate the length of the condensed shot.

- False alarms will cause our syntax based reduction algorithm to remove more shots than permitted by the minimum requirements. Again let us consider the case where we have five shots in the progressive phrase. Let us assume that there is one false alarm, between shots one and two, increasing the number of shots to six. Then from **Table 8.1**, the minimum number of shots that must be kept when doing syntax based reduction, is three. Now if we attempt to remove the last three shots, we would have removed the shot number three from the correct sequence, one that should not have been removed.

In this work, we only focus on false alarms. This is because false alarms manifest themselves as an increase in the number of shots detected, with the shot boundaries known (unlike shot misses, that can occur anywhere in the segment), hence we can compensate for them.

We conducted a test with nine variations of our shot detection algorithm [109]. The algorithm had two adjustable thresholds for motion and color, and each parameter was set to three possible values: {low, medium, high}. **Table 8.2** shows the results of the tests. We used 112 shots from four scenes, each from a different film. The test set had 54

dialog shots and 58 non-dialog (i.e. progressive) shots. The shot detectors had no false alarms in the section which had dialogs. Hence the parameter of interest is the probability of false alarm given a non-dialog scene: $P(F_a | N_d)$.

Table 8.2: Shot detector variations. Each row show the in order: the motion and color parameter thresholds, recall, precision, prob. of false alarm $P(F_a)$, $P(F_a | N_d)$: prob. of false alarm give a non-dialog scene. The 97.5% confidence upper bound for $P(F_a | N_d)$.

Motion	Color	Recall	Precision	$P(F_a)$	$P(F_a N_d)$	97.5%
L	L	0.97	0.72	0.28	0.55	0.64
L	M	0.96	0.71	0.29	0.55	0.65
L	H	0.96	0.71	0.29	0.55	0.65
M	L	0.92	0.93	0.07	0.14	0.23
M	M	0.91	0.94	0.06	0.12	0.21
M	H	0.91	0.94	0.06	0.12	0.21
H	L	0.90	0.94	0.06	0.11	0.19
H	M	0.86	0.95	0.05	0.10	0.18
H	H	0.84	0.95	0.05	0.10	0.18

We use standard statistical techniques [57], to compute the 97.5% confidence upper bound for $P(F_a | N_d)$. In this current work we use the following shot detector: {motion: M, color: L}. The table implies that for this shot detector, $P(F_a | N_d) \leq 0.23$, with 97.5% confidence, over unseen data sets.

The upper bound on $P(F_a | N_d)$ is then used to modify the minimum number of shots that must remain in a progressive scene. If the progressive scene contains N shots, with m minimum number of shots (from the rules in section 8.3.3), then m is modified as:

$$m' = m + \lfloor N \cdot P(F_a | N_d) + 0.5 \rfloor \quad (8.9)$$

where m' is the new minimum $\lfloor \rfloor$ is the floor function. The second part of equation (8.9), is just the expected number of false alarms. For example, in a scene with type II reduction, assume that $N = 12$, then, $m = 6$ and $m' = 9$ (from equation (8.9)) Hence the increase $\Delta m = 3$, and the minimum number of shots to be retained is 9. Modifying the minimum number of shots to drop ensures that we do not violate the minimum shot requirements of the syntactical elements of the scene.

8.4 Discussion

In this section we analyze some of issues raised in this chapter in some more detail. We begin with a discussion of the implicit assumptions in our comprehension time experiment. In section 8.4.2, we present alternatives to the video segment selection heuristic adopted in this work.

8.4.1 The comprehension time experiment

We now discuss two aspects of the comprehension time experiment that are important:

(a) use of a single still image and (b) the number of subjects in the experiment.

Implicit in the use of a single image in our experiments, there are two simplifying assumptions here:

- The semantics of the shot (in terms of the four questions) are adequately captured by our key-frame.
- The semantics do *not* change during the course of the shot. (i.e. the answers to who / where / when / what do not change during the course of the shot).

Clearly, in complex shots, both of these assumptions may be violated. We also tried to conduct the experiment with videos (i.e. by watching the shot) rather than with key-frames. However, watching the video proves to be too distracting to measure the response time to the four specific questions.

Only the author of this work participated in the experiment. This is problematic and clearly, a more comprehensive study is needed. However, in order to compensate for this deficiency, the subject was tested on 1000 images, picked at random from a very diverse corpus, so as to generate reliable statistics. The user study results from the experiments (appendix 13.3, 13.5 and the results in section 10.7) are encouraging. They indicate that the viewers find the skims resulting from the shots condensed to their upper bounds to be highly coherent, thus validating our comprehension experiment.

8.4.2 Video segment selection

In this work, we reduce the duration of a shot by throwing away frames towards the end of the shot. For the sake of definiteness, we discuss the case of reducing a 10 sec. shot to 2.5 sec.

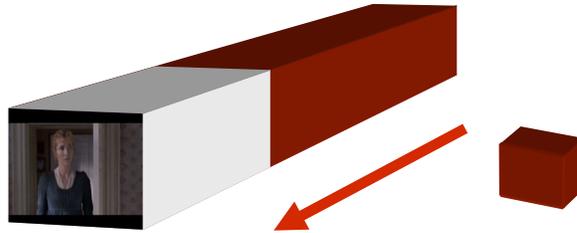


Figure 8.6: we drop the frames in a shot from the end.

We choose to keep the initial segment for the following reason — the director “sets-up” the shot (i.e. provides the context in which the action takes place) in the first few frames. Hence, by preserving the initial segment, we hoped to preserve the context. In a shot of long duration, the semantics of the shot can change considerably, and in such cases, our strategy will not capture the important information within the shot.

We did consider several alternate strategies. One is to sub-sample the shot (in the current example, by factor of 4). Since the skim is to be rendered at 30 fps, this will cause several problems: (a) the video will be sped up causing the data will appear too quickly, at four times the information rate of the original shot and (b) since we wanted to include audio in future work, lip-synchronization will be lost and finally, (c) the aesthetics (that exist at 30 fps) of the original shot will also be lost.

Another strategy is to sub-divide the original shot using a heuristic. In the current example, we could choose to keep the first second, the last second and the middle 0.5

second of the current shot. This will create two new shot boundaries where none existed and we conjecture that this will prove to be disconcerting to the viewers (particularly at high condensation rates). Note that since film content is completely unconstrained, creating a universal measure to mark a segment within a shot to be important, proves difficult.

8.5 Summary

In this chapter, we have presented a novel framework for condensing computable visual scenes. The solution has two parts: (a) analysis of the visual complexity of the shot, (b) film syntax analysis. We define the visual complexity of a shot to be its Kolmogorov complexity. Then, we proved a simple lemma that helped us bound this measure via the Lempel-Ziv algorithm. Then we conducted an experiment that allowed us to map visual complexity of a shot to its comprehension time. This resulted in a conservative time-complexity bound, that allows us to compute the minimum duration of the shot for it to remain comprehensible.

The syntax of the shot sequence influences the semantics of a scene. We devised algorithms based on film-making techniques to come up with heuristics to reduce the length of the syntactical element. We showed syntax reduction strategies for two elements of syntax — the progressive phrase and the dialog. We showed how we could guard against the errors in practical shot-detectors in our syntax reduction rules. Finally, we discussed the important issue of syntax based summarization schemes against traditional object detector based summarization schemes. While we showed that syntax

based summarization schemes are powerful in the case of arbitrary structured domains when we potentially have a large number of “objects.” However, we also concluded that it may be useful to combine the two schemes in certain in the case of in specialized domains, where efficient object detectors may be available and where the semantics of the object is well known and assumed fixed.

9 Skims: Auditory analysis

9.1 Introduction

In this chapter, we discuss our approach for auditory analysis for generating the skim. The issues presented here are a dual to the visual analysis presented in the previous chapter, and we shall use the results of both chapters to create the skim.

In prior work on audio-visual skims, audio was either used with the transcript [18] or to serve as an object detector [52] i.e. determination of interesting segments by detecting events such as gunshots etc. Since the insistence of a transcript is limiting, as is the idea of using a few object detectors, we need a principled approach, that will work well on generic audio data. In our approach to summarization, we do two things:

- Classify segments into different classes such as speech, noisy speech, silence and environmental sounds. The skim generation mechanism is sensitive to the class context.
- Determine significant topic boundaries. These are phrases of speech that serve to introduce new topics, in structured as well as in spontaneous speech.

Then, after post processing, we have a ranked list of audio segments, where we have placed maximum emphasis on the significant phrases.

The rest of this chapter is organized as follows. In the next section, we shall briefly discuss why simple approaches to audio summarization do not work well. Then in sections 9.3 and 9.4, we shall present our approach to classifying audio segments and detecting significant phrases. Then, in section 9.5, we shall present our results. Finally we shall conclude by presenting a summary of this chapter.

9.2 Why is summarizing audio a hard problem?

A summary of the audio data must have at least these three characteristics: (a) the individual audio segments must be coherent (i.e. intelligible), (b) must satisfy certain minimum duration constraints depending upon the audio class and (c) the speech segments must contain complete speech phrases i.e. they must not contain phrases that are cut mid-sentence. The speech segments are particularly important in any audio summary, since their semantics are immediately clear.

Let us now examine three approaches where we condense the duration of the audio without regard to the context (i.e. there is no classification of the data into different classes). Let us assume that we wish to condense an audio track that is 100 sec. long, by 90%.

- **Downsampling:** Downsampling the audio data by 90% will leave the audio to be severely degraded. This is because from elementary signal analysis, the frequency spectrum of the audio signal will expand by a factor of 10, thus causing the pitch of the speech segments to increase dramatically. A secondary

consequence of subsampling is that in order to maintain lip-synchronization between the audio and the video data, we must now subsample the video data by a factor of 10. This will cause the video to appear incoherent as well.

- **Pause-Removal Synchronous Overlap Add (PR-SOLA):** PR-SOLA [37] is a non-linear time compression technique that eliminates long pauses, and attempts to preserve the original pitch in the output. This is done in the following way:
 - Detect silences, and condense them to a minimum of 150ms.
 - Overlap adjacent frames of audio data, and determine the overlap extent that causes maximum cross-correlation between the two frames. The second frame is then added to the first at the maximal overlap point.

PR-SOLA is hard to use with such speedups since user studies [39] indicate that users do not prefer to have the speech sped up beyond 1.6x (i.e. ~40% compression).

- **Synchronous segment selection:** What happens if we first condense all the video shots, as discussed in chapter 8, and then select only those audio segments that are synchronous with the condensed video shots? Then, user studies indicate that the resulting audio stream is choppy and difficult to comprehend [18] .

9.2.1 Our approach to summarization

We use a context sensitive (i.e. audio class dependent) approach to generating coherent audio summaries. There are two key aspects to our analysis of audio:

- Segment classification:** Construct robust classifiers on the audio data using Support Vector Machines (SVM) [19] robust classifiers on the audio data. There are four classes of interest: (a) clean speech, (b) noisy speech (c) environmental / music sounds and (d) silence. Then, we use a duration dependent Viterbi decoding [69] algorithm to impose a class dependent minimum segment duration.

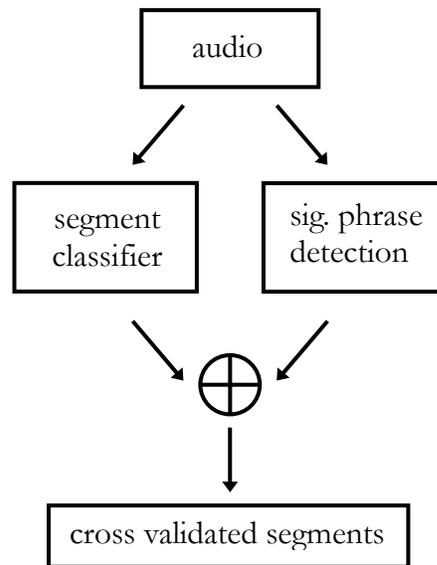


Figure 9.1: The audio data is processed in parallel, by the audio classifier and the significant phrase detection algorithm.

- Significant phrase detection:** We detect significant phrases in speech by constructing SVM classifiers on pitch and pause information. These phrases represent important discourse boundaries (i.e. new topic boundaries), in both structured and spontaneous speech [33] [40] .

The significant phrase classifier is run in parallel and cross-validated against the audio segment classifier result. and combine them with the classification result. In the end, we

are left with long coherent segments of audio data, that include segments which have significant phrases. **Figure 9.1** shows the system diagram for the auditory analysis. In the next two sections we shall discuss the audio segment classification and the significant phrase detection algorithms in detail.

9.3 Audio segment classification

In this section we discuss the procedure for audio segment classification, we begin with the features used, followed by a description of the algorithm used.

9.3.1 Features used

We use 16 features in our approach ([30] , [53] , [69] , [72] , [75]): (1) loudness, (2) low-band energy (3) high-band energy (4) low energy ratio (5) spectral roll off (6) spectral flux (7) spectral centroid (8) spectral entropy (9) MFCC (10) delta MFCC (11) RASTA, (12) PLP and four variants of the zero crossing rate (13) ZCR, (14) mean ZCR, (15) variance of the ZCR [74] and (16) high ZCR [53] .

The cepstral features, RASTA and PLP were chosen since they are well known to be good speech discriminators [30] [69] . All other features were chosen for their ability to discriminate between music and speech. Also see section 4.3.1 for a more detailed description of the features. We now discuss the classification procedure.

9.3.2 The approach to segmentation

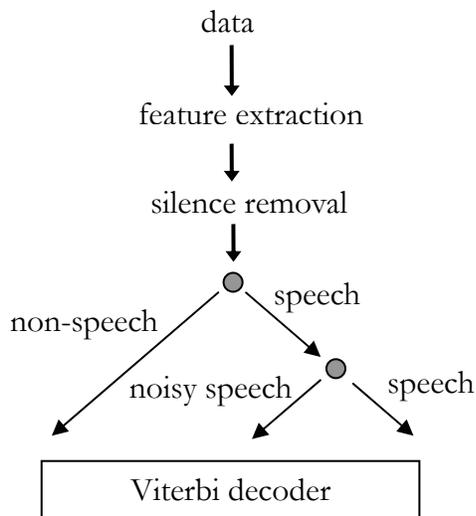


Figure 9.2: The tree structure for the audio segment classifier. Each gray dot represents a two-class SVM classifier

We use a tree structured classification procedure (ref. **Figure 9.2**) to classify each frame (100 ms. duration) into four classes: speech, noisy speech, music / environmental sounds and silence. The features used for the classification are extracted per frame. After removing all frames classified as silent (ref. section 4.6 for details on silence detection) the remaining frames are then classified using a tree structured SVM classifier (ref. **Figure 9.2**) into three classes (non-speech (i.e. environmental sounds / music), speech, noisy speech).

Why do we adopt a classification strategy here instead of modifying the segmentation algorithm proposed in chapter 4? Firstly, since the segmentation algorithm uses long term coherence to determine boundaries, the segments from the segmentation algorithm

may encompass more than one class. This can happen if the class is repeated quickly.

Since we are interested in the class of the segment as this affects the skim algorithm, we just use the classifiers.

Each classifier used in our approach is a C-SVM classifier [16] [19] [112], used with the radial basis kernel. The C-SVM classifier has two parameters that must be set: C and γ . The optimal parameters are found using a grid search on C and γ , by performing five-fold cross-validation on the data, for each candidate pair. The optimal parameters are determined to be $C=5.0$ and $\gamma=0.025$.

The classification results are then fed into a modified duration dependent Viterbi decoding algorithm, to smooth the classifier output. We discuss the duration dependent decoding in more detail in the next section.

9.3.2.1 The duration dependent Viterbi decoder

The modified Viterbi decoder makes use of the class transition probabilities, classifier error and a duration utility function to come up with the maximum likelihood class path. The class transition probabilities are determined from the ground truth. The classifier confusion matrix is determined by running the classifier over the training data.

The duration utility function is used to penalize short and long segments. The form of the function is class dependent since each class has a different minimum duration requirement. Formally:

$$U(d) = \begin{cases} \frac{\exp(d)-1}{\exp(D_{\min})-1} & 0 \leq d < D_{\min}, \\ 1 & D_{\min} \leq d < D_{\max}, \\ \frac{D_f - d}{D_f - D_{\max}} & D_{\max} \leq d < D_f, \\ 0 & d \geq D_f, \end{cases} \quad (9.1)$$

where d is the duration of the segment, $U(d)$ is the utility of the segment of duration d , D_{\min} and D_{\max} are the minimum and the maximum duration of the class respectively. D_f is a hard upper bound beyond which the utility of the segment drops to zero. Note that $D_f \neq D_{\max}$.

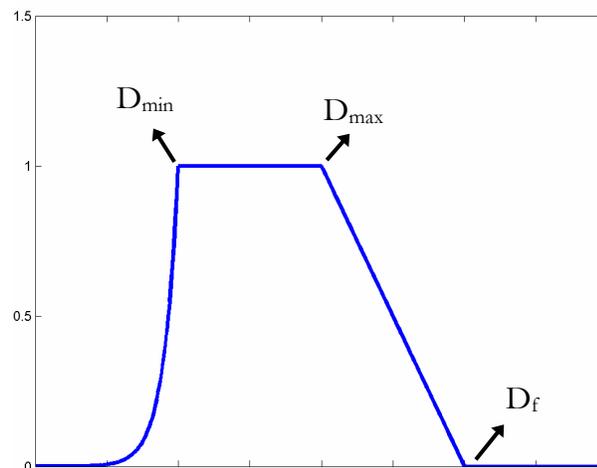


Figure 9.3: The duration utility plotted as a function of the segment duration.

The first part of the utility function rises exponentially to 1 since we would like the utility to drop exponentially, when the duration is *less* than the lower bound for that class. The utility is a constant within the bounds since we do not discriminate between the different durations. Beyond the preferred upper bound, the utility drops linearly to zero at D_f . The reason why this drop is linear rather than exponential is that longer duration segments

will sound coherent to the user, so the rate at which the utility drops, cannot be as great as for the case when the duration is less than the lower bound. The time D_t behaves like an hard upper limit on the segment duration.

The update equations for the Viterbi decoder are slightly different from the standard Viterbi decoder, and we present them here for the sake of completeness.

$$\begin{aligned}
 \delta_1(i) &= \pi_i + C_i(o_1) & 1 \leq i \leq N \\
 \psi_1(i) &= 0 \\
 \psi_t(j) &= \arg \max_i \left\{ \alpha \delta_t(i) + \beta a_{ij} + \gamma \hat{U}_{t-1}(i) \right\} & \begin{array}{l} 2 \leq t \leq T \\ 1 \leq j \leq N \end{array} \\
 \delta_t(j) &= \delta_{t-1}(i) + a_{ij} + C_j(o_t) & \begin{array}{l} 2 \leq t \leq T \\ 1 \leq j \leq N \end{array} \\
 \delta_T^* &= \max_i \left\{ \delta_T(i) \right\}
 \end{aligned} \tag{9.2}$$

Where, δ_t is the likelihood score at time instant t , a_{ij} is the class transition probability, $C_i(o)$ is the probability of the true class being i when the classifier outputs class o as the result of the classification. \hat{U}_t is the product of the segment utilities, prior to time t . N is the number of classes, T is the total number of observations, α , β and γ are weights in the update equation, used to modulate the influence of the duration utility function. Note, in the implementation of the algorithm, we use the logarithm of all the variables.

The result of the classification procedure can result in some short segments (since the utility is non-zero for small durations). Then, we run a median filter to merge adjacent segments, and additionally merge adjacent segments marked as silent. We now present our computational approach towards detecting significant phrases.

9.4 Detecting significant phrases

In this section, we shall summarize our work on detecting phrases in speech that introduce a new segment in the discourse [41] ,[80] . These segment beginnings (SBEG's) are important as they serve as the introduction of new topic in the discourse (ref. section 7.3).

There has been much work in the computational linguistics community [40] [41] [61] [80] to determine the acoustic correlates of the prosody in speech. Typically, SBEG's have a preceding pause that is significantly longer than for other phrases, higher initial pitch values (mean, variance), and smaller pauses that end the phrase than for other phrases [40] [41] . Note that pauses can be either silent, or filled (e.g. “umm..”). In this work, we shall concern ourselves with silent pauses only, since filled pauses do not reflect topic changes.

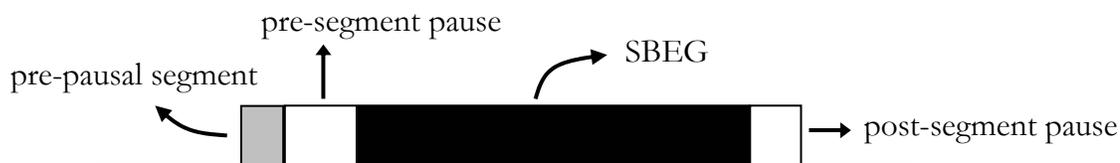


Figure 9.4: When classifying a candidate segment, we extract features from different parts of the candidate segment, as well as extract features from the pre-pausal segment. The duration of the preceding and the succeeding pauses are also used for classification.

In our algorithm, we extract the pitch and the energy values at for different parts of a segment to be classified, in addition to extracting the pause durations. Given a candidate segment, we shall extract pitch and energy values from the beginning, the end and from

the entire segment. We shall also extract pitch and energy features from a small segment *prior* to the pause that precedes the candidate segment. Work in [61] demonstrated the utility of the pre-pausal segment in disambiguating between grammatical and ungrammatical pauses. This is illustrated in **Figure 9.4**. Prior work [2] [4] [37] that implements speech based summarization, indicates that users prefer relatively long segments of speech. In this work we restrict our attention to phrases that last between five to fifteen seconds.

We now present our approach towards detecting SBEG's:

- Perform silence detection, and determine significant pauses. In prior work on detecting SBEG's [41] [80] using the pre-segment pause, the significant pause threshold (i.e. the threshold above which a pause was deemed significant) ranged

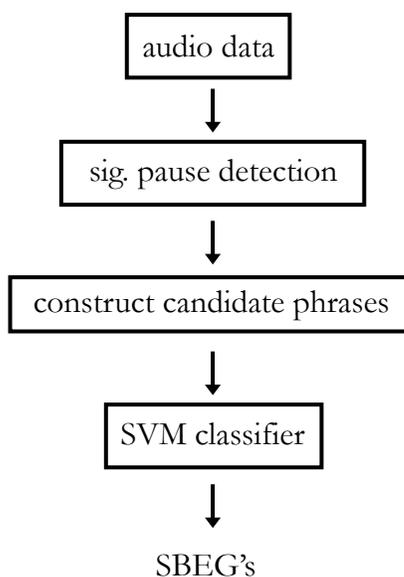


Figure 9.5: The system for detecting segment beginnings (SBEG), which mark new topic phrases.

between 160ms and 576ms. We use a threshold of 150ms; i.e. in order for a segment to be a candidate, the pause duration must exceed 150ms.

- A candidate segment lies between two significant pauses, and in addition satisfies the segment duration requirements (i.e. is between 5~15 sec. long) mentioned earlier.
- Extract the pitch and energy values for different parts (beginning, end and the entire segment) of the candidate segment, as well as the pre-pausal segment. The beginning and the end segment durations are 2 sec. long, while the length of the pre-pausal segment is 500ms. Concatenate the extracted feature values into one single feature vector.
- The feature vector extracted from the candidate segment is then classified using a C-SVM classifier, that uses a radial basis function kernel. This approach is summarized in the system diagram, **Figure 9.5**

9.5 Results

In this section, we present results of the audio segment classification as well as the significant phrase detection.

9.5.1 Segment classification

We used 45 minutes of audio data from two films (*Bladerunner*, *Four Weddings and a Funeral*) to train our SVM classifier. The data is complex and contains speech overlaid with background sounds, music and other environmental sounds. The data was labeled

by the two people — the author and another PhD student using the following labeling criteria: (a) segments classified as “music” were western music; speech was labeled as “noisy” or “clean” depending upon the level of the background sound; all other segments were labeled as “environmental” sounds. Weak speech segments embedded in environmental sounds (e.g. sounds from the street) were labeled as environmental sounds. The confusion matrixes for the two SVM classifiers (after a five-fold cross validation) is are shown in **Table 9.1** and **Table 9.2**.

Table 9.1: the confusion matrix result of the speech (S) vs. Non-speech classifier (\neg S). The rows indicate the classification result for each ground truth class.

T \ C	S	\negS
S	0.76	0.24
\neg S	0.13	0.87

Table 9.2: the confusion matrix result of the speech (S) vs. noisy-speech classifier (N_s). The rows indicate the classification result for each ground truth class.

T \ C	S	N_s
S	0.84	0.16
N_s	0.06	0.94

Where the labels in the tables are as follows: T: true label, C: classifier result, S: Speech, \neg S: non-speech (i.e. music / environmental sounds), N_s : noisy speech.

In work related to speech/music classification both [74] and [75] report excellent results (98.4% and 95% accuracy respectively). However, both of them test on commercial FM

music stations, where the speech (the disc jockey's voice) tends to be very clean and the music segments are restricted to the songs. The results in [53] are over a more varied dataset (MPEG-7 test CD), and the results are excellent — 96% accuracy in discriminating speech versus non-speech. While our results are not quite as good, we believe that increasing the size of the training set, and adding more discriminating features will improve our performance.

9.5.2 Significant phrase detection

We used data from three films *Blade Runner*, *Sense and sensibility*, *Pulp fiction* to label 324 phrases as “significant” or as “non-significant.” We labeled only those phrases that were complete grammatical phrases as significant. Examples of non-significant phrases include — phrases that begin or end mid-sentence, and the list of cue phrases (e.g. “now, what do you want to eat?”) [40] .

The phrases used for training, were detected automatically, but were labeled manually, by the author of this research. For the purposes of training, every segment that satisfied the following criteria was labeled by the author as a significant phrase or not a significant phrase:

- It was a speech segment.
- The segment lasted between 5 and 15 sec. long
- The segment began and ended with a significant pause.

A simple program was then used to generate all segments that satisfied these criteria, and the labels were assigned by listening to the audio data.

The ground truth had 48 significant phrases and 276 non-significant phrases. The results of five fold cross-validation on the 324 phrases, using an SVM (radial basis kernel, $\gamma = 0.04$, $C = 100$, 117 support vectors) gave 100% precision and 100% recall. We believe that this result is perhaps due to two factors: (a) labeling by one person only, and (b) the data was very consistent. We expect the performance to be lower in a more diverse test set.

In [81], the author uses pitch analysis for discourse segmentation, and she compares her result to manually segmentation of the discourse. The discourse is segmented to five levels. She achieves a precision and recall of 50% and 53% respectively, for SBEG's that include level 3 and lower. Level 0 refers to the outermost level of the discourse. Note that in [2] [4], the author focuses on pitch based detection of emphasized portions of speech; however, emphasized portions of speech can occur anywhere within a discourse, and do *not* necessarily mark the beginning of a new segment boundary.

The results of the audio segment classifier output and the significant phrases are merged as follows. Whenever a segment is marked as a significant phrase, the corresponding audio classifier labels are marked as clean speech. We do this as our significant phrase classifier is more reliable than the audio segment classifier. This will cause some segments to become fragmented, and they are merged by taking a majority vote of the adjacent segments. We run the two classifiers in parallel, since using the classifiers

sequentially, would have the undesirable result of having the results of one classifier affect the results of the other.

The segments are then ranked with the following priority (a) clean speech (b) environmental sounds (c) noisy speech. These priorities are important when we start to drop audio segments, as will be discussed in the next chapter.

9.6 Summary

In this chapter, we have discussed our approach to auditory analysis for skims. We began by discussing three simple approaches to summarization that do not use the class context: (a) downsampling (b) PR-SOLA and (c) synchronous segment selection. Our approach to generating audio summaries has two principal components: (a) segment classification and (b) significant phrase detection.

The audio segments were detected as follows. After removing the silences from the data, we classify each frame of data using robust SVM classifiers. The result of the classifier is then fed into a duration dependent Viterbi decoder that smooths the output labels using a class dependent utility function and the prior classifier error.

The significant phrases were detected as follows. We first detect significant pauses, as they often precede significant phrases. Once this is done, we select candidate segments that lie between two consecutive significant pauses, and extract features from different parts of the segment. Then, we use a SVM classifier to classify the feature vector thus extracted, as a significant phrase. After some additional post-processing, we are left with

a ranked list of audio segments, some of which have been marked as significant.

While the results of the segmentation and the significant phrase detection are good, we believe that they can be improved.

10 Utility maximization

10.1 Introduction

In this chapter, we formulate the problem of passive skim generation as a constrained optimization problem. We attempt to maximize the utility of the entities (video shots, audio segments, syntactical elements etc.) that we seek to preserve in the skim. Such a conceptual framework is important for several reasons:

- The optimization formulation causes us to think of a skim as a collection of entities, and forces us to come up with *measures* of each entity (i.e. its utility) and its relevance to the overall skim.
- The entity-utility optimization framework is a very general, extensible framework [17]. It allows us to model arbitrary interactions amongst entities as constraints on the optimization, thus enabling us to determine the solution in a principled manner, as opposed to resorting to *ad-hoc* methods whenever we are confronted with preserving a specific constraint.

We begin by identifying the different entities that we seek to preserve in the final skim.

The utility functions for both video and the audio segments have certain constraints

imposed on them, that arise principally from boundary conditions as well from asymptotic behavior. Other functional constraints include symmetry, differentiability and concavity. We shall additionally derive functions that govern their mutual interaction.

The skims have three types of constraints: (a) audio visual synchronization constraints (b) duration bounds on the segments and (c) constraints due to visual syntax. The optimization procedure is biased towards preserving the target entities. In this work, we derive sufficient conditions for the optimization solution to exist, and the constraints on the audio and video segments are relaxed till these requirements are met.

The user study evaluates the optimal skim against two other competing skim generation algorithms. The results indicate the optimal algorithm results in statistically significant improvements over the other two algorithms, at high condensation rates.

The rest of the chapter is organized as follows. In the next section, we review the notation to be used in the rest of the chapter. Then, in section 10.2, we shall discuss the specific entities that we shall preserve in the course of generating the skim. In section 10.3, we shall discuss the derivation of the audio and visual sequence utility functions and in the following section, we shall discuss functions that govern the mutual interaction amongst video and audio segments. In section 10.5, we shall introduce the idea of tied multimedia constraints and in section 10.6 we shall present the detailed constrained minimization procedure. Finally, we shall conclude the chapter with sections on experiments and a summary.

10.1.1 Notation

In the sections that follow we shall use the following notation. The letters a and v when used as subscripts, are used to denote the audio, video segments respectively, while the subscript o is used to denote the original segment. N_v and N_a represent the total number of video shots and audio segments in the original sequence; T_o is the original duration of the sequence, the target skim duration is T_f , and $t_{o,n,v}$ and $t_{o,k,a}$ represent the original duration of shot n and the k^{th} audio segment in the scene respectively. Define indicator function $\phi_v(n) = 1$ iff. n^{th} video shot is present in the condensed scene. Define $N_{\phi,v} = \sum \phi_v(n)$, the number of shots in the condensed scene. $\phi_a(n) = 1$ iff. n^{th} audio segment is *not* silent, and hence $N_{\phi,a} = \sum \phi_a(n)$ is the number of non-silent audio segments.

10.2 What entities will we preserve?

The focus of our work is in the creation of passive, discourse centric skims (ref. section 7.4.3), that are also maximally coherent, in the sense of intelligibility. In order to construct such skims, we seek to preserve the following entities.

- **Syntactical elements:** The syntax of the video shots has a dual role: (a) its presence is essential to communicate meaning and (b) the specific elements of film grammar used, alter the meaning of the video sequence. Hence the

preservation of the syntactical entities will preserve the original intent of the director.

- **Speech segments:** This entity is the most *direct* form of communication by the director (or, in other words the least ambiguous of the semantic entities used by the director.), when compared to visuals in the data or the specific sounds, that are designed to *evoke* a specific feeling (fear, repulsion etc.). Hence, the preservation of the speech entities should increase the intelligibility of the sequence³⁰.
- **Synchronous entities:** Many entities that are to be found in produced data, have synchronous audio-visual events — the shot of a person speaking (the lips will be synchronized to the speech) etc. The preservation of such entities will again increase the overall coherence of the skim.
- **Segment coherence:** We must ensure that each of the “atomic” entities such as video shots and the audio segments, are individually coherent — this implies that we keep their original durations as far as possible, and reduce their duration with care, in a class dependent manner.

³⁰ Of course, this is not always true. For example, an actor may speak in response to a visual entity; hence preserving the speech segment alone will not be sufficient. Indeed, it may engender confusion; the only way to prevent this is to *recognize* the visual entity and its relationship to the speech segment. This is beyond the scope of this thesis.

- **Film rhythm:** This is *affect* entity i.e. the relationship between the durations of the individual shots have been carefully chosen to convey a specific emotion (or affect) (ref. section 7.4.1.2). Its preservation is also important to preserve meaning.

Now that we have discussed the entities of interest, over the next few sections, we shall describe the utility functions associated with some of these entities and the optimization procedure for generating the optimal skim. While we shall describe utility functions associated with segment coherence and film rhythm, we shall preserve other entities without associating an explicit utility value to with them. The syntactical elements are preserved by following a specific procedure for syntactical element reduction (ref. section 8.3.3). The synchronous entities as well as the speech entities are preserved *implicitly* by biasing the optimization search procedure.

10.3 Utility functions

In order to preserve the entities that were discussed in the previous section, we need to associate utility functions with video shots and audio segments as well as preserve some of the affect entities.

We model the utility of each of the entities separately, and in particular, we do not model *joint* interactions between audio and video segments. While such a model will be more sophisticated than the ones that we shall present in the following sections, we do not do so for two reasons:

- We do not have experimental evidence on the effects of this coupling (i.e. how the presence of audio affects the visual comprehension and vice versa) to justify using a heuristic. This coupling has a strong effect on comprehension, as anecdotally evidenced by the effect of music on Mtv montage video sequences. There, we do not seem to be unduly bothered by the fast transitions between images and the montage style editing of the video; this is because of the coherence of the music.
- The separable model is simple and computationally tractable.

We make the additional assumption that the sequence of video shots (and the audio segments) in a scene are assumed to be i.i.d., thus allowing us to model the utility of a video shot (audio segment) independently of other shots (segments).

10.3.1 Video

The video shot utility function, models the comprehensibility of a shot as a continuous function of its duration and its visual complexity. Why is this new formulation necessary given the complexity bounds in section 8.2.4 ? The bounds in that section delineate the

bounds of comprehension³¹ — importantly, they do *not* tell us how the comprehensibility of a shot *changes* when we decrease its duration.

The non-negative utility function of a video shot $S(t, c)$, where t is the duration of the shot and c is its complexity, must satisfy the following constraints:

1. For fixed c , $S(t, c)$ must be a non-decreasing function of the duration t :

$$S(t_1, c) \leq S(t_2, c), \quad t_1 \leq t_2, \quad (10.1)$$

This is because decreasing the shot duration by dropping frames from the end of the shot (section 8.2.4, that discusses shot condensation), cannot increase its comprehensibility.

2. Boundary constraints:

$$\begin{aligned} S(t, 0) &= 0, \\ S(t, 1) &= 0, \quad \forall t, \end{aligned} \quad (10.2)$$

This constraint arises due to the following observation: complexity $c = 0$ implies the complete absence of any information, while $c = 1$ implies that the shot is purely random. Hence the utility of such shots must necessarily be set to zero.

³¹ For example, the upper bound (ref. equation (8.8)) tells us that 95% of shots when reduced to their upper bounds will remain comprehensible.

3. Constraints on the function form: Additionally, we would like the shot utility function to be bounded, differentiable, separable and concave. While boundedness prevents singularities, differentiability allows us to use faster gradient descent optimization algorithms. The effect of the utility function being concave, is that we can readily make use of convex optimization techniques to ensure a single solution. We make the function form separable only for tractability.

Thus, the form of the shot utility function is as follows:

$$S(t, c) = \beta c(1 - c) \cdot (1 - \exp(-\alpha t)), \quad (10.3)$$

The values for α and β were determined from the result of the first user study test (see Appendix 13.4). Note that the function is concave in both t and c . The use of the exponential makes the function asymptotically bounded, and it also agrees with an intuitive observation — the *rate of change* of the shot utility should decrease with increase in the shot duration. For example, let us assume that we have two shots: one that is 2 sec. long, while the other is 200 sec. long. Now, in both cases if we remove the last one sec. the decrease in utility in the first case should be much *larger* than the decrease in utility in the second case.

Given the symmetry in the boundary constraints, symmetry with respect to complexity is reasonable. The functional form stems from differentiability and the concavity constraints. It is easy to come up with a more general separable model (e.g. higher order

polynomials), however, we chose not to do so, to avoid the possibility of over fitting.

The shot utility function is shown in **Figure 10.1**.

10.3.1.1 What happens when we drop a shot?

Let us assume that we have 10 shots in a sequence, and we need to drop the 6th shot. What then is the loss in utility as a consequence of dropping a shot? Since a shot is just barely comprehensible when it is condensed to its lower bound, the amount of utility

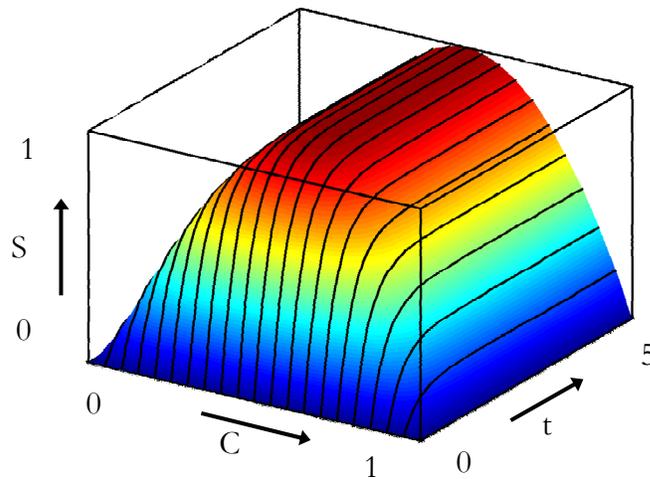


Figure 10.1: The shot utility function plotted as a function of the complexity (x-axis) and time (y-axis). The z-axis shows the function value.

that is *irretrievably* lost when that shot is dropped is proportional to $S(t_{L,b},c)$ i.e. the utility evaluated at its lower bound. Then, when a shot i is dropped, we assign a utility loss $P(t_{p,i})$ to the shot as follows:

$$P(t_{p,i}) = \lambda(t_{p,i}) \cdot S(t_{Lb,i}, c_i),$$

$$\lambda(t_{p,i}) = \begin{cases} 1 & t_{p,i} \geq t_{Ub,i} \\ \frac{t_{p,i} - t_{Lb,i}}{t_{Ub,i} - t_{Lb,i}} & t_{Lb,i} \leq t_{p,i} < t_{Ub,i} \\ 0 & t_{p,i} < t_{Lb,i} \end{cases} \quad (10.4)$$

where, λ modulates the shot utility, $t_{p,i}$ is the proportional time for shot i i.e. $t_{p,i} = t_{o,i} \cdot T_f / T_o$, and where $t_{Lb,i}$ and $t_{Ub,i}$ are the lower and upper time bounds for shot i , and S is the shot utility function (see **Figure 10.2**).

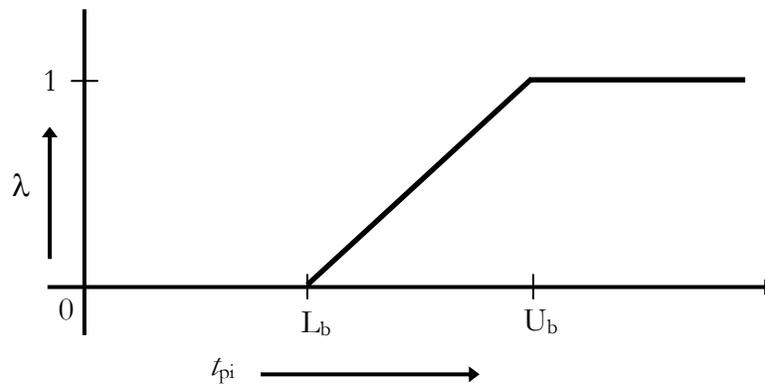


Figure 10.2: A graph of the modulation function λ plotted against the proportional duration.

The modulation function λ has the following effect — when the proportional duration is greater than the upper bound, the loss in utility is large. However, when the proportional duration is less than the upper bound, the λ decreases linearly to zero at the lower bound. i.e. the loss in utility at durations close to the lower bound or below the lower bound is negligible. There are two reasons why we modulate the shot utility loss $S(t_{Lb}, c)$:

- The shot utility loss function is an *estimate* of the loss in utility. Certain high compression rates cannot be achieved even if we condense each shot to its lower bound. Hence, in such cases, the utility loss associated with dropping a shot must be *less* than the utility of the shot evaluated at the lower bound. An estimate of the utility loss in such cases, can be achieved using the proportional solution to generating the skim.
- Let us return to the example with 10 shots. Let us assume that the original duration of the sequence was 100 sec. and we are required to meet a target time of 10 sec. Then, one solution is the proportional solution — reduce the duration of each shot by 90%. However, in this case, some of the shot durations would fall below the lower bound for each of those shots, and thus become incomprehensible. Hence we need to associate zero utility loss with such shots.
- The motivation for the form of the utility loss, also comes from an observation in one of the early user studies on visual skims (**Table 13.2**): users preferred as far as possible to see the complete sequence, over the possibility of dropping shots (see the “best” and “worst” ratings on the skims in **Table 13.2**).

Hence the modulation function λ , biases the optimization towards creating sequences that have no dropped shots and in which each shot remains above the upper bound for that shot. Note the following:

- A shot cannot be arbitrarily dropped; shots can only be removed as dictated by the rules of syntax.

- Two shots that have the same complexity, but different original durations, will have different drop penalties, since their lambda term will be different. This is understandable since we *should* impose a greater penalty term for dropping a shot of longer duration (the proportional durations will be in the ratio of the original durations.).

10.3.1.2 The sequence utility function

The sequence utility function is then readily defined as follows:

$$U_v(\vec{t}_v, \vec{c}, \phi_v) = \frac{1}{N_{\phi_v}} \left(\sum_{i:\phi_v(i)=1} S(t_{i,v}, c_i) - \sum_{j:\phi_v(j)=0} P(t_{p,j}) \right) \quad (10.5)$$

where, $\vec{t}_v : t_0, t_1 \dots t_N$ and $\vec{c} : c_0, c_1 \dots c_N$ represent the durations and complexities of the shot sequence, S and P represent the shot utility and the loss function respectively. Equation (10.5) says that the net utility of a shot sequence is the sum of the utilities of the shots in the sequence less the loss due to the shots that were dropped. In the next section, we discuss the utility function for an audio segment.

10.3.2 Audio

We do not have any experimental results indicating a complexity-time relationship for audio similar to the experiments of section 8.2.4, however, it seems fairly reasonable to conjecture its existence. Hence, the form of our audio utility function will be similar to the video shot utility function presented in the previous section.

Table 10.1: The variation in λ and β with the class type.

Type	λ	β
Silence	6.67	1
Environmental sounds	0.33	3
Clean Speech	0.2	5
Noisy Speech	0.2	2

We define the utility function for a non-silent audio segment of duration t belonging to a class³² k as follows:

$$A(t, k) = \beta_k (1 - \exp(-\lambda_k t)) \quad (10.6)$$

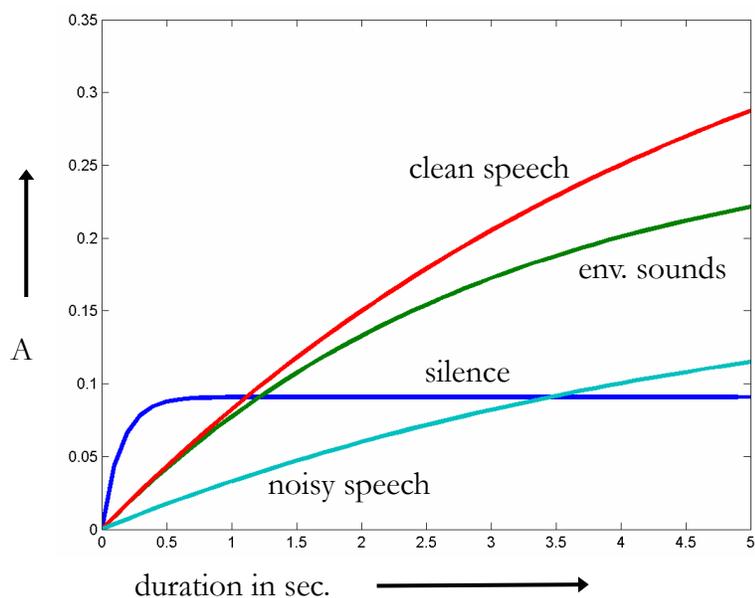


Figure 10.3: The audio utility function for the different classes, plotted as a function of segment duration. The plot has the y-axis scaled.

where, A is the utility function and where β and λ are class dependent parameters.

The class dependent parameters change the rate of change of the utility function with the duration of the segment **Table 10.1** shows the variation in β and λ with the class type.

The β and λ values in **Table 10.1** were generated using a heuristic, but the relative ordering of the λ values is important. The β and λ values were chosen so as to bias the utility functions in favor of the clean speech. Since noisy speech was deemed to be less “useful” than environmental sounds / music, we placed a greater utility on the environmental sounds over noisy speech. Note that since λ affects the rate at which the utility function reaches its asymptotic value (i.e. β), it was important that the utility of silence be insensitive to changes in duration, above the minimum silence duration. This is the reason for the large value of λ , for the silence utility function.

10.3.2.1 The audio segment utility loss function

When an audio segment is dropped, we assign a utility loss as follows:

$$L(t_i) = ((t_i - t_o) / \theta)^2 \quad (10.7)$$

³² Recall from section 9.3, that we have four classes in our system: clean speech, noisy speech, environmental sounds and silence.

where, t_i is the duration of the i^{th} silence, and where t_0 and θ are normalizing constants. Note that a “dropped” audio segment is actually *silent*. The equation seems to indicate that we penalize large as well as very small silences. This is intuitively correct since an extremely short silence (say duration $< 50\text{ms}$) will sound as harsh break. **Figure 10.4** shows the plot of the utility loss function as a function of the segment duration.

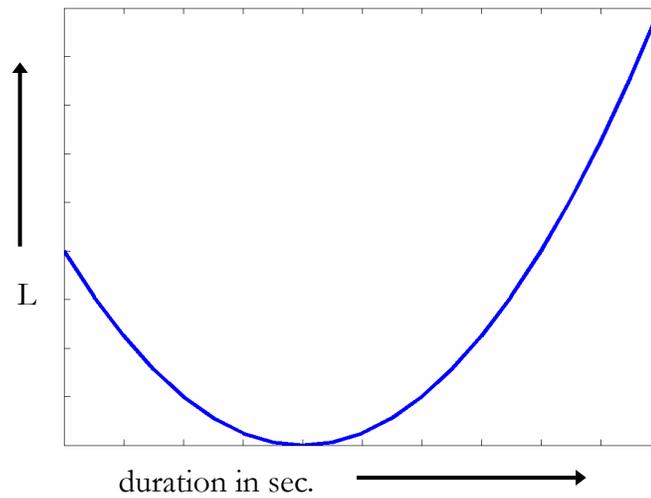


Figure 10.4: The plot shows the audio utility loss function as a function of the segment duration.

Then, similar to equation (10.5), we define the audio utility to be the sum of the utilities of the constituent segments.

$$U_a(\vec{t}_a, \vec{k}_a, \phi_a) = \frac{1}{N_{\phi_a}} \left(\sum_{i:\phi_a(i)=1} A(t_{i,a}, k_i) - \sum_{i:\phi_a(i)=0} L(t_{i,a}) \right) \quad (10.8)$$

where, $\vec{t}_a : t_0, t_1 \dots t_N$ and $\vec{k} : k_0, k_1 \dots k_N$ represent the durations and the class labels of the audio segments in the skim, and where the functions \mathcal{A} and L represent the audio segment utility and the loss function respectively.

10.3.3 A note on dropped segments

Why do we convert dropped segments to silence? After all, we do not insert blank (i.e. black) frames, when we condense video shots. We do so following reasons:

- Unlike video shots that typically last 2~4 sec. in a film, speech phrases can last up to 20 sec. Hence, dropping a significant speech phrase³³ has a very different impact as far as meeting the target skim duration is concerned. Hence, it may be possible to *undershoot* the time budget, by removing the speech segment in its entirety. This implies that we cannot meet the time budget at all. We avoid this situation by inserting a silence.
- Silence, unlike black video frames, is perceptually acceptable.
- The silence utility function is flat beyond the minimum duration, thus allowing us to condense silence effectively.

³³ Note that only speech segments (either significant or noisy) are dropped in their entirety. This is because we cannot trim speech, as this will create harsh breaks. Non-speech sounds are trimmed to their lower bound, before they are dropped.

10.4 Penalty functions

In this section we introduce the penalty functions for both audio and video segments. These penalty functions circumscribe the interactions amongst groups of video and audio segments.

10.4.1 Film rhythm

In this section we show how we can preserve the film rhythm entity discussed in section 7.4.1.2 (introduced there in the form of a predicate). The original sequence of shots have their duration arranged in a specific proportion according to the aesthetic wishes of the director of the film. Clearly, while condensing a scene, it is desirable to maintain this “film rhythm.” For example, consider a scene with three shots of durations 5 sec. 10 sec. and 5 sec. maintaining the scene rhythm would imply that we preserve the ratios of the duration (i.e. 1:2:1) of the shots. We define the rhythm penalty function as follows:

$$R(\vec{t}, \vec{t}_o, \phi) = \sum_{i:\phi(i)=1} f_{o,i} \ln \left(\frac{f_{o,i}}{f_i} \right), \quad (10.9)$$

$$f_{o,i} = \frac{t_{o,i}}{\sum_{i:\phi(i)=1} t_{o,i}}, f_i = \frac{t_i}{\sum_{i:\phi(i)=1} t_i}.$$

where R is the penalty function, and where, t_i is the duration of the i^{th} shot in the current sequence, while $t_{o,i}$ is the duration of the i^{th} shot in the original sequence. The ratios are recalculated with respect to only those shots that are not dropped, since the rhythm will

change when we drop the shots. The function as described in equation (10.9), penalizes the shot sequence only if the original film rhythm is *not* maintained.

10.4.2 Audio slack

In film previews, one common method of tightly packing audio segments within a limited time, is to make them overlap by a slight duration. This not only condenses the audio segment, but also makes for a smooth transition amongst audio segments. We use a simple quadratic function to implement the cross-fading of the overlapping segments.

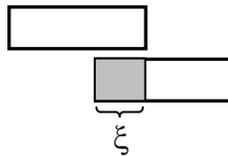


Figure 10.5: The figure shows two audio segments overlapping by ξ .

We associate a slack variable ξ , with each audio segment that allows it to overlap the previous segment by ξ sec (see **Figure 10.5**). This variable is bounded as $-2 \leq \xi \leq 0$, for all segments except the first segment, for which it is necessarily zero. This allows us to condense audio data a little more without losing too much comprehensibility. We need to penalize excessive slack, and hence we have a slack penalty function.

$$E(\vec{\xi}, \vec{k}) = \frac{1}{E_o} \sum_{i=2}^{N_a} \eta(k_{i-1}, k_i) \xi_i^2 \quad (10.10)$$

where, ξ_i is the slack variable for the i^{th} segment, E_o is a constant that normalizes the sum to 1, k_i is the class label for the i^{th} segment, and η is a class dependent coupling function that weights the interaction between adjacent classes, in a class dependent manner.

Table 10.2: The table for determining the coupling factor η . The columns indicate the class of the preceding segment, while the rows indicate the class of the current segment.

Type	Speech	Silence	Environmental sounds / music
Speech	∞	∞	1
Silence	∞	∞	∞
Environmental sounds / music	1	∞	0.5

The coupling table for η , is shown in **Table 10.2**. We have three classes, with noisy speech and clean speech being treated alike. Similarly, we do not make a distinction between music and environmental sounds. There, the rows indicate the class of the previous segment, while the columns indicate the class of the current segment. For example, the table indicates that the $\eta(\text{speech}, \text{speech})=\infty$. This is because we do not want two adjacent speech segments to overlap, since the overlap will cause both speech segments to lose coherence. Here, the coupling factor $\eta=\infty$ forces the overlap ξ to be zero. Note also, that the coupling factor related to silence is always ∞ . This is because we never want a segment marked as silence to overlap with any other segment. Such an overlap only reduces the *duration* of the silence. It is preferable to do this reduction using

the utility associated with the silence entity. We now discuss the principal constraints on our optimization.

10.5 Constraints

There are three principal constraints in our algorithm: (a) audio-visual synchronization requirements (b) minimum and maximum duration bounds on the video shots and the audio segments and (c) the visual syntactical constraints. We shall only discuss the first two constraints since we've have extensively covered the syntactical constraints in section 8.3.3.

10.5.1 Tied multimedia constraints

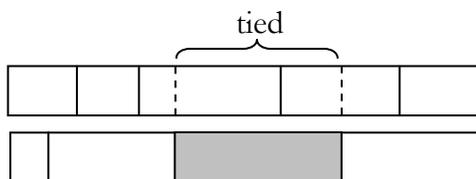


Figure 10.6: the gray box indicates a speech segment and the dotted lines show the corresponding tied video segment.

A multimedia segment is said to be fully *tied* if the corresponding audio and video segments begin and end synchronously, and in addition are *not condensed* (i.e. no part of either the video shot or the audio shot has been removed). Note also, that video shots / audio segments that are tied cannot be dropped from the skim. The multimedia segments can also be partially tied only on the left or on the right, but in this case the

corresponding segments are only synchronous at one end, and the video (audio) can be compressed.

In **Figure 10.6**, we show a fully tied segment corresponding to the section of audio marked as a significant phrase. Since the beginning (and ending) of a significant phrase will *not* in general coincide with a shot boundary, we shall split the shot intersected by the corresponding audio boundary into two fragments. To each fragment, we associate the complexity of the parent shot.

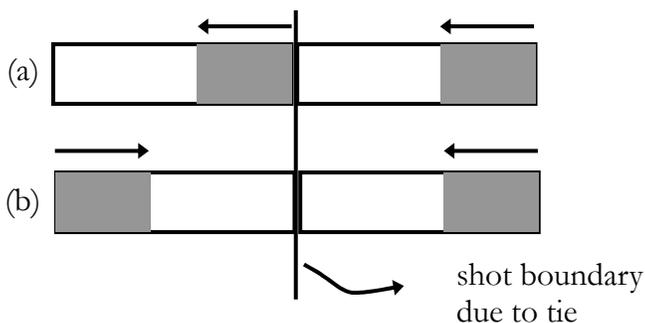


Figure 10.7: Both figures show an artificial shot boundary, in a single shot, induced by a tie constraint. The gray boxes refer to the portions of the shot fragments to be dropped, while the arrows indicate the direction of the condensation. In figure (a) if we condense the two shots from the ends, we will create a new, visible shot boundary that didn't exist before. If we adopt the condensation scheme shown in figure (b), no new shot boundaries would be created.

We need to modify the shot condensation procedure detailed in section 8.4.2. Since we have introduced a new shot boundary by artificially splitting a shot into these fragments, blindly adopting the condensation procedure in section 8.4.2 will create visible artifacts (i.e. the viewer will see new shot breaks, that were absent in the original film). Hence, in order to prevent new shot boundaries from appearing in the skim, we condense the shot

fragment to the left of the new boundary to the *right* while the shot fragment to the right of the new boundary is reduced according to 8.4.2, to the *left*. **Figure 10.7** illustrates this procedure.

Each tie boundary induces a synchronization constraint:

$$\sum_{i=1}^{N_1} t_{v,i} = \sum_{j=1}^{N_2} t_{a,j} + \xi_j \quad (10.11)$$

where N_1 , N_2 are the number of video and audio segments to the left of the boundary respectively, $t_{v,i}$ is the duration of the i^{th} video segment, $t_{a,j}$ is the duration of the j^{th} audio segment and ξ_j is the slack variable associated with each audio segment. In equation (10.11), the left side is just the sum of the duration of all the video shots to the left of the synchronization boundary. Similarly, the right side is the sum of the duration of all the audio segments and their corresponding slack variables. Note, a fully tied segment will induce two synchronization constraints, while a partial tie will induce one synchronization constraint.

A skim represents a highly condensed sequence of audio and video, with a high information rate. Hence, a tied segment by virtue of being uncondensed and synchronous allows the viewer to “catch-up.”

10.5.2 Duration bounds

Each video shot and audio segment in the skim satisfies minimum and maximum duration constraints. For both the audio and the video segments, the upper bounds are

just the original segment durations. For the video shot fragments, the lower bounds are determined from the complexity lower bound (equation (8.8)).

For the audio segments, we have heuristic lower bounds, per audio class: silences 150ms, music / environmental sounds: 3 sec. Speech segments (i.e. complete phrases that are bounded by significant silences) are kept in their entirety and hence lower and upper bounds are made equal to the original duration. These heuristics will ensure that we have long audio segments in the skims, thus increasing the coherence of the skim.

We reduce the duration of the music / environmental sounds by trimming the end of the segment (except for the last segment, which is trimmed from the beginning). Speech segments are either kept in their entirety or dropped completely if the target time cannot be met. The reason why this is done is because trimming speech segments will make them sound incoherent since we may then end up cutting a sentence in the middle.

10.6 Minimization via constraint relaxation

In this section we shall describe the procedure for obtaining the audio and video segment duration using a principled minimization procedure.

10.6.1 Algorithm bias

We focus on the generation of passive discourse centric summaries that have maximal audio visual coherence. We ensure skim coherence by biasing the algorithm in the following manner:

- **Speech segments:** We deem the speech segments to contain the maximum information, and we shall seek to maximize the presence of speech segments in the skim in two ways:
 - Biasing the audio utility functions in favor of the clean speech class.
 - Biasing the audio-visual skim solution search to favor the presence of clean speech segments. In case the audio segments do not contain clean speech, the skim generation algorithm will work just as well, except that in this case we would not be biased with respect to any one class.
- **Visual syntax:** We ensure that the principles of visual syntax are not violated
- **Ties:** We shall attempt to ensure that we have as many tie constraints as possible. This is because these constraints ensure synchrony between the audio and the video segments.

10.6.2 Constructing tied multimedia segments

In this implementation we construct fully tied multimedia segments the following way. We first assume that the audio has been segmented into classes with all the significant phrases marked. All the speech phrases marked as significant (and additionally labeled as “clean speech”) are ranked depending on the degree of significance of the segment. Then, we associate multimedia tied segments with these ranked speech segments only. All other segments (including “noisy speech”) have the same low rank and are not tied to the video.

In the sections that follow, we assume the following: we are given T_j , the target duration of the skim, the audio has been segmented and the clean speech segments ranked, and that the video has been segmented into shots. We now discuss our algorithm in two stages — first by presenting an overview of the solution, followed by the details of the algorithm.

10.6.3 Solution overview

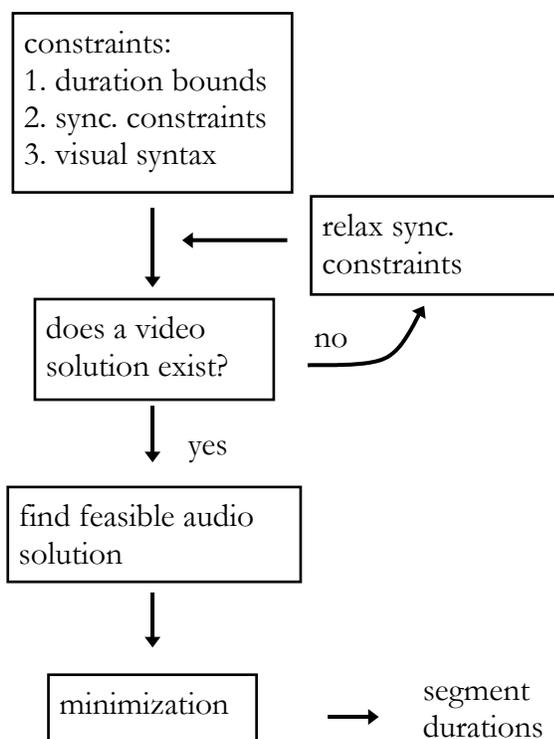


Figure 10.8: searching for a feasible solution by successively relaxing the synchronization constraints.

We first present the intuition behind our strategy for ensuring that a feasible solution region exists for the optimization algorithm. The approach has the following key ideas:

1. A fully tied multimedia segment ensures the following: (a) the corresponding audio and video segments are uncondensed and (b) none of these segments can be dropped. Hence, removing one synchronization constraint from a fully tied segment allows us to condense the audio and video segments and if necessary, drop them from the final skim.
2. Only clean speech segments (i.e. significant phrases marked as clean) are fully tied. Since speech segments have been ranked according their significance, we remove one constraint starting from the lowest ranked speech segment.
3. Once a feasible video solution region has been found, a feasible audio solution is guaranteed to exist, since we can always drop audio segments that are not in fully tied segments, by converting them to silence Note: (a) the silence class has a very small lower bound: 150ms. Hence we shall always be able to meet our audio skim budget (b) we pick the audio segment to drop in this order: pick noisy speech segments first. Then, if none exist, pick the segment that minimizes the deficit.

Briefly then, we do the following: (a) start with all clean speech segments tied. (b) drop shots and relax constraints starting with the least significant tied speech segment, till the video budget is met. (c) Once the video budget is met, meet the audio budget by dropping audio segments in rank order. This is summarized in **Figure 10.8**.

The visual syntax constraints require that a minimum number of shots be present in the skim. Hence, there will some condensation rates that cannot be met even after removing all the synchronization constraints. In that case, we create a “best” effort skim. This is

done by first removing as many shots as allowed by the rules of syntax, and then setting the duration of each shot to its lower bound. The skim target duration is then modified to be the sum of the duration of these shots.

We now present the details of the strategy to ensure feasible solutions for our optimization procedure. We are interested in ensuring solution feasibility, because we wish to avoid an iterative backtracking procedure, in the case where there is no solution possible (i.e. the target skim time can be achieved. This can happen for example when there are too many multimedia tie constraints.).

10.6.4 Ensuring a feasible video solution

In order to ensure that a video solution exists before performing the optimization, we adopt a two stage check strategy. We first make sure that the video shots can meet the constraints and once this is ensured, we ensure that the audio segments can meet the target skim time.

The algorithm to ensure that the solution exists is as follows. We assume that we have N_v video shots and that N_{\min} is the minimum number of shots due to the syntactical constraints (ref. section 8.3.3).

1. Begin with all the clean speech segments marked as fully tied. Then, the corresponding video shots will also be marked as tied and will be uncondensed.
2. While $N_v \geq N_{\min}$ check if a video skim will satisfy the constraint equation:

$$\sum_{i=1}^{N_v} (l_{v,i} + (u_{v,i} - l_{v,i})\chi(i))\phi_v(i) \leq T_f \quad (10.12)$$

where, $l_{v,i}$ and $u_{v,i}$ are the lower and upper bounds for the i^{th} shot, $\chi(i) = 1$ if the i^{th} shot is not condensed and $\phi_v(i) = 1$ if the shot is not dropped. T_f is the target duration. In equation (10.12), note the following: (a) if a shot is not compressed, then the contribution of this shot is just the original duration. (b) If the shot *can* be condensed, then it can be condensed up to its lower bound. The equation simply says that the sum of the durations of all the shots that are *not* dropped, must be less than the target duration T_f .

3. If eq. (10.12) is not satisfied then (if $N_v > N_{min}$ drop one shot (ref. section 8.3.3) and repeat step 2 until the constraint is satisfied, else go to step 3).
4. If the video feasibility constraint eq. (10.12), is not satisfied, *untie* one constraint from the lowest ranked speech segment. Note, untying a constraint does two things: (a) allows the video shots to be condensed and (b) if necessary, allows them to be dropped from the skim.
5. If no solution is feasible even after untying all the constraints, the best effort video solution is generated. This is done by first removing as many shots as allowed by the rules of syntax, and then setting the duration of each shot to its lower bound. The skim target duration is then modified to be the sum of the duration of these shots.

At the end of this procedure, have a feasible video solution region and we know the following variables: the number of constraints in the final skim, and the number of video shots to be retained in the skim.

10.6.5 Ensuring a feasible audio solution

We now present our algorithm to ensure that we have feasible solution region for audio.

1. A feasible audio solution region exists if for every two consecutive tie boundaries, the following inequality holds:

$$\sum_{i:g_a(i,\kappa)=1} (l_{a,i} + \xi_i) \leq V_{\min} \quad (10.13)$$

$$V_{\min} = \sum_{j:g_v(j,\kappa)=1} (l_{v,j} + (u_{v,j} - l_{v,j})\chi(j))\phi_v(j)$$

where, we are examining the audio segments between tie boundaries $\kappa-1$ and κ , $g_a(i,\kappa) = 1$ if the i^{th} audio segment is present between the two tie boundaries, $l_{a,i}$ is the *minimum* duration of the audio segment, and ξ_i is the corresponding slack variable. Similarly, $g_v(j,\kappa) = 1$ if the j^{th} video shot is present between the two tie boundaries. Note that the minimum duration of an audio segment depends upon the class.

Why is this inequality sufficient to ensure that an audio solution is feasible? First note that V_{\min} is the minimum possible duration of the corresponding video shots between the two tie boundaries. The equation for V_{\min} is simply the sum of the minimum durations of the shots that are not dropped, between the two tie

boundaries. Then, the first part of equation (10.13), simply states that the sum of the *minimum* durations of the audio segments plus the corresponding slack variable, between the two tie boundaries must be less than V_{\min} . If this is true then for every possible video solution, we will have an audio solution.

2. If the inequality is not satisfied, we remove one audio segment, by converting it to silence. To do this pick the segment that minimizes the deficit in the inequality (10.13). We look to remove noisy speech segments first, followed by music and environmental sounds and then speech (lowest ranked first). Note that converting the segment to silence causes the minimum duration of the segment to drop to 150ms. Go to step 1.

At the end of this procedure, we shall have a feasible audio solution region.

10.6.6 The mathematical formulation

We now define the objective function O_f that gets minimized as a consequence of our minimization procedure. Then, given the target duration T_f :

$$O_f(\vec{t}_a, \vec{t}_v, \vec{\xi}, n_c) = \omega_1 O_A(\vec{t}_a, \vec{\xi}) + \omega_2 O_V(\vec{t}_v) \quad (10.14)$$

where, ω_1, ω_2 are constant weights. O_A, O_V , represent the audio, and video objective functions respectively and are defined as follows:

$$\begin{aligned} O_A(\vec{t}_a, \vec{\xi}) &= 1 - U_A(\vec{t}_a, \vec{k}, \phi_a) + \lambda_1 E(\vec{\xi}) \\ O_V(\vec{t}_v) &= 1 - U_V(\vec{t}_v, \vec{k}, \phi_v) + \lambda_2 R(\vec{t}_v) \end{aligned} \quad (10.15)$$

where, λ_1, λ_2 are weighting factors. Note that once we have feasible solution regions, \bar{k}, ϕ_a, ϕ_v are constants. Note that the functions E and R , refer to the audio slack and the film rhythm penalty functions. The individual shot durations are determined as follows:

$$\begin{aligned}
 (\vec{t}_a^*, \vec{t}_v^*, \vec{\xi}^*, n_c^*) &= \arg \min_{\vec{t}_a, \vec{t}_v, \vec{\xi}, n_c} O_f(\vec{t}_a, \vec{t}_v, \vec{\xi}, n_c) \\
 &\text{subject to:} \\
 &t_{L_v, i, v} \leq t_{i, v} \leq t_{O, i, v}, \quad i: \phi_v(i) = 1, \\
 &T(k_i) \leq t_{i, a} \leq t_{O, i, a}, \quad N_{\phi, v} \geq N_{\min}, \\
 &\sum_{i: \phi_v(i)=1} t_{i, v} = T_f, \\
 &\sum_j t_{i, a} + \xi_i = T_f, \\
 &\sum_{i=1}^{N_{l, v}} t_{v, i} = \sum_{j=1}^{N_{l, a}} t_{a, j} + \xi_j, \quad l: 1 \dots n_c
 \end{aligned} \tag{10.16}$$

where n_c is the number of final synchronization constraints, $T(k)$ is the class dependent audio lower bound. The first two constraints in equation (10.16) are duration constraints, the next two are total time budget constraints, while the last equation refers to the synchronization constraints. Note that N_{\min} is the minimum number of shots to be retained in the scene, and this arises from the syntactical rules discussed earlier. Also, the shots can be dropped only in a constrained manner using the rules in section 8.3.3.

10.7 Experiments

In this section we shall detail the results of our user study that was used to test the efficacy of our approach for audio-visual skims (Two other user studies that deal with

visual skims i.e. skims that do not contain audio, are discussed in the appendix (13.4 and 13.5)).

10.7.1 Introduction

The scenes used for creating the skims were from three films: *Blade Runner (bla)*, *Bombay (bom)*, *Farewell my Concubine (far)*. The films were chosen for their diversity in film-making styles. While we arbitrarily picked one computable scene from each film, we ensured that each scene had a phrase as well as a dialog.



Figure 10.9: The application for generating audio-visual skims.

All the audio-visual analysis, as well as the utility optimization was done in MATLAB, while the actual skim generation was done using the windows media video SDK (see **Figure 10.9**).

10.7.2 Three algorithms compared

We conducted experiments with three different skim generation mechanisms:

- The optimal audio-visual skim
- A proportional skim
- A semi-optimal skim

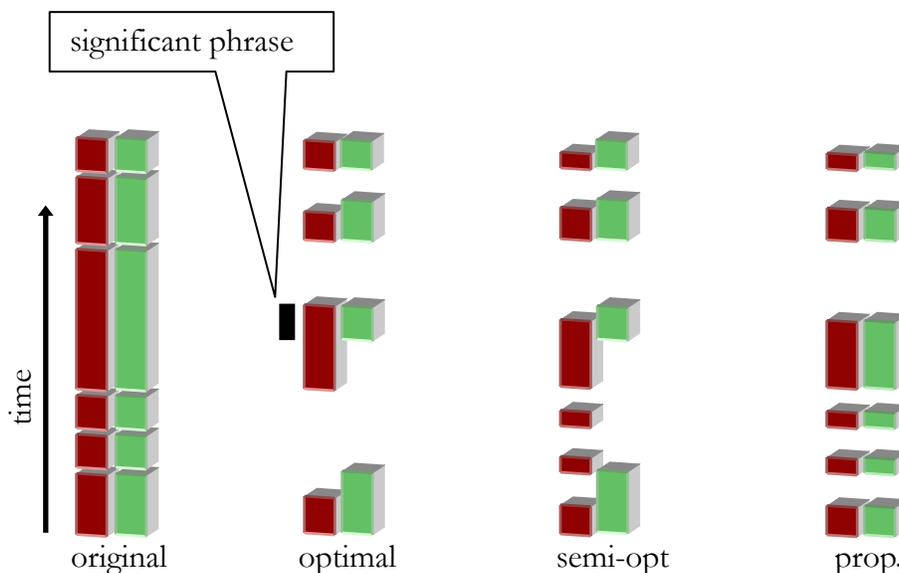


Figure 10.10: The original video and the four skims: optimal, semi-optimal and proportional.

segment types —red: video, green: audio; black: discourse segment. . Note that the semi-optimal and the proportional skims have the same proportional video data, and the optimal and the semi-optimal skims have the same optimal audio data.

Where, the optimal audio-visual skim is the one generated by our utility based optimization. The proportional skim is generated as follows: let us assume that we wish to condense the segment by 75%; then in a proportional skim, each video shot and each audio segment would be compressed by 75%; in both cases, we trim the data from the right of each video shot (audio segment)). A semi-optimal skim has proportional video (as described in the preceding few lines) and the optimal audio segments from our audio

segment analysis (ref. chapter 9) algorithm. . In **Figure 10.10**, the red segments (appears as dark gray in print) represent the video data, while the green (light gray) segments represent the audio segments; the black rectangle shows the significant phrase. Note that the semi-optimal and the proportional skims have the same proportional video data, and the optimal and the semi-optimal skims have the same optimal audio data.

In the earlier user study results (see appendix 13.4 and 13.5) with visual skims (that contained no audio) showed that a utility based visual skims were perceptually better (in a statistically significant sense) than proportionally reduced visual skims. Since presence of the optimal audio segments will make the third skim more coherent than the proportional skim, we deem it semi-optimal.

10.7.3 The user study

In our user study, we compared the algorithms at three condensation rates: 90%, 80% and 50%. Ideally, we would have liked to create one skim per condensation rate, per film. However, this would have meant that each user would have had to rate 27 skims (because there are three films, three scenes and three condensation rates), an exhausting task. Instead, we used a single scene at each condensation rate, thus creating three skims for each rate (one from each algorithm), and since there are three condensation rates, (90%, 80%, and 50%), we thus created nine skims. We compressed the film *Bombay* at 90%, *Blade Runner* at 80% and *Farewell my Concubine* at 50%.

We conducted a pilot user study with twelve graduate students. Each film was on the average, familiar to 2.33 students. The testers were expected to evaluate each skim, on a scale of 1-7 (strongly disagree – strongly agree), on the following metric: Is the sequence coherent? They were additionally asked to indicate their agreement in answering the four generic questions of who? where? when? what? (ref. Section 8.2.3) for each skim. The setup was double-blind and each user watched the skims in random order.

10.7.4 Results

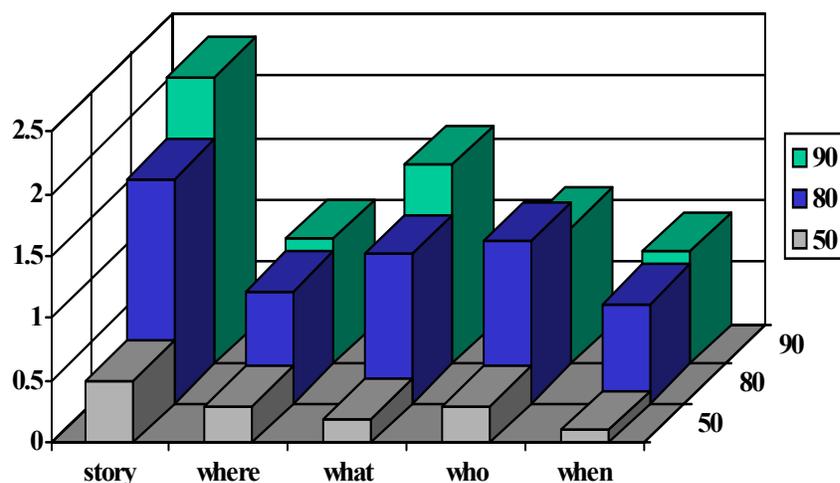


Figure 10.11: The difference between the raw optimal score and the minimum of the other two scores. The differences are significant at 80% and 90% compression rates

The results are shown in **Table 10.3**. The rows in **Table 10.3** show the averaged scores across users for the optimal algorithm and the differences from the optimal for the other two algorithm. Positive numbers imply that the raw score of the optimal algorithm is greater than the corresponding non-optimal skim. The numbers in bold indicate statistically significant differences from the optimal. We computed the statistical

significance using the standard student's t-test. The test scores indicates that the optimal skim is better than the other two skims, at a confidence level of 95% at the high condensation rates (90% and 80%). Interestingly, the optimal skim was *not* significantly better than the other two skims at the 50% condensation rate. **Figure 10.11** shows a graphical illustration.

Table 10.3: User test scores. The columns: algorithms (optimal, semi-optimal, proportional), the film, condensation rate, and the five questions. The table shows values of the optimal and the difference of the scores of the (semi / pr) skims from the optimal. Bold numbers indicate statistically significant differences.

Algo.	Film	Rate	Coh?	who?	when?	where?	what?
opt	bom	90	5.25	5.92	5.50	6.00	5.58
semi / pr			2.3 / 0.9	1.1 / 0.75	0.9 / 0.5	1 / 0.75	1.6 / 0.6
opt	bla	80	4.83	6.00	5.50	6.00	5.58
semi / pr			1.8 / 1.7	0.5 / 1.3	0.3 / 0.8	0.3 / 0.9	1.1 / 1.2
opt	far	50	4.75	5.92	5.75	6.00	5.25
semi / pr			-0.2 / 0.5	0.3 / 0.2	0.1 / 0.1	0.3 / 0.3	0.0 / 0.2

10.7.5 Discussion

Why do the significance tests yield different results at the high and low rates? At low condensation rates, our optimal skim, does have proportionately reduced video shots.

This is because of two reasons:

- The rhythm penalty function (see section 10.4.1) ensures that the shots are proportionately condensed.

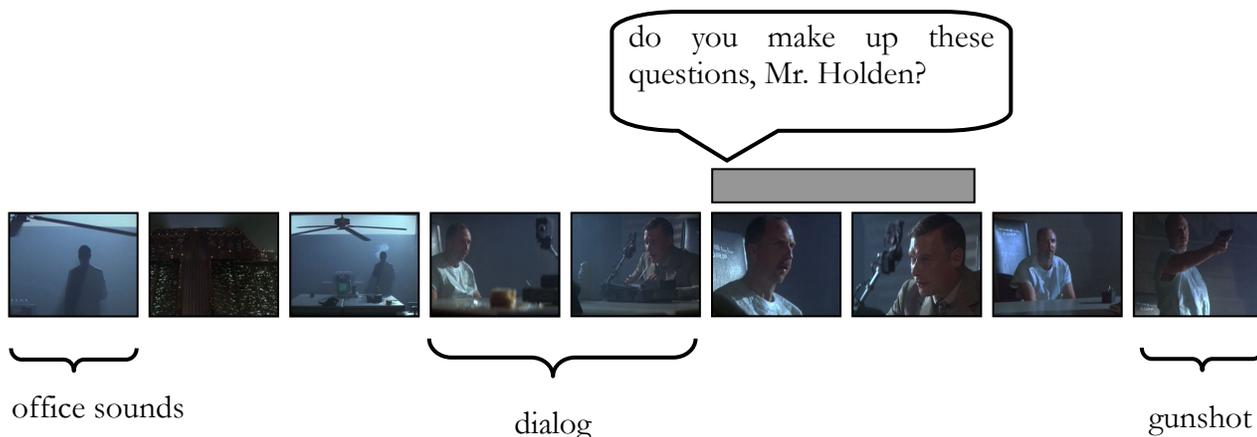


Figure 10.12: The *blade runner* optimal skim showing the important elements captured by our algorithm. The gray box shows the location of the sig. phrase.

- The utility function is exponential (see section 10.1.1) — flattening out at long durations.

Hence the optimal result *will* be pretty similar to the other two skims. At the high condensation rates, proportionately reducing the skim is not possible as this approach will severely decrease the skim utility (it decays exponentially). This is because as some shots will fall below the lower bound for comprehension.

In **Figure 10.12**, we show the parts of the optimal skim at 80% condensation rate. Note that the skim has captured the dialog element, the significant phrases (the shots are not condensed), and preserves the synchronized beginnings and endings. We do not have an gunshot detector, and it appears in the skim because the end is synchronized. The other two skims will not contain significant phrases, the audio and video will be completely unsynchronized.

10.7.5.1 *What about other skim forms?*

This work focuses on creating discourse centric skims, by ensuring that as many speech segments get included and are synchronized as far as possible. The approach also makes sure that the skims are coherent by preserving the underlying film syntax. The user study results are very positive indicating that the skim is coherent at high condensation rates.

However, we can improve on this approach for some constrained domains (e.g. medical videos). When the relationship between the data and the semantics is clear, using object detectors followed by a skim generation mechanism that preserves syntax, will generate meaningful skims, that capture the essential semantics. Affect and event driven skims are easily created within our framework by adding additional constraints on the audio and video segments. However, it is much harder to detect affect, except for very specific forms. Event based skims make sense in specific domains (e.g. baseball), but event discovery in an unconstrained domain is a difficult problem.

10.8 Summary

In this chapter, we have formulated the problem of skim generation as a constrained optimization problem that attempts to preserve certain entities within the skim. We are interested in five different entities: (a) elements of syntax (b) speech segments (c) synchronous multimedia entities (d) coherent audio and video segments and (e) film rhythm. While some of them are explicitly associated with utility functions, others are implicitly preserved by biasing the solution search.

We derived the utility functions of the audio and video segments, by employing three kinds of constraints: (a) boundary constraints, (b) asymptotic constraints and (c) constraints on functional form. We also derived utility loss functions that are used to penalize dropped audio visual segments. This results in the formulation of an audio, and video sequence utility function. Our approach uses a separable models, making the problem computationally tractable.

We introduce the idea of tied multimedia segments that ensures synchronization for the speech entities. Given a time budget, we first ensure solution feasibility by ensuring that the video and the audio time budget can be met separately. This is done in the following way. Since all the audio segments are classified into four different classes — (a) clean speech, (b) noisy speech (c) environmental sounds / music and (d) silence, we start the solution search by ensuring that all the speech segments are fully tied. Then, we relax the constraints on the tied segments till we have solution for the video segments. The audio solution existence is guaranteed by ensuring that certain duration bounds are satisfied. Once we are guaranteed of the existence of the audio and the video solutions optimization routine generates the optimal solution.

We conducted a pilot user study on three scenes, generating skims using three skim generation algorithms, at three different condensation rates. The results of the user study shows that while the optimal skims are all regarded as coherent, the differences are statistically significant only at the high rates (i.e. 80% and 90%).

11 Conclusions and future work

11.1 Introduction

In conclusion, we shall first summarize the work presented in this thesis, along the lines of the three sub-problems that have been tackled here — segmentation, structure detection and audio visual summarization. Then, in section 11.3, we shall present some possible improvements to the models used in this work. Then in section 11.4, we shall present a few potential areas of future research.

11.2 Research summary

We now present a summary of the work done in this thesis. We begin by summarizing our approach to segmentation.

11.2.1 Segmentation

This thesis developed a novel framework for segmenting audio-visual data. We focused on the detection of *computable scenes*, that arise out of interactions between elementary computable audio and video scenes. These elementary scenes are segments of data that are consistent with respect to a certain set of features. There were four key ideas in our approach on segmentation:

- The idea of a finite, causal memory model to segment both audio and video data.
- Making use of production rules in film-making.
- Using constraints from computational auditory scene analysis.
- Incorporating higher forms of knowledge such as structure and silence.

In chapter 3, we presented our approach to determining computable video scenes. The scene detection is predicated on the use of a memory model. We showed how to compute two important measures on the data stored in the memory — *recall* and *coherence*. Recall computes the similarity between shots in the memory buffer, and which incorporates a dissimilarity measure, the duration of the two shots and the time separation between the shots. Coherence, a measure of inter-segment similarity, was defined using the measure for recall. Computable scenes were deemed to exist at the local coherence minima.

In this chapter, we also presented three new conceptual memory models that improved on the basic model used for experiments in this thesis. While the first measure introduced the idea of segment *self-coherence* to help improve the segmentation results, the other two models made use of the of Kolmogorov complexity to determine the relationships between the data in the memory.

In chapter 4, we presented our work on detecting the computable audio scene. As with the problem of detecting computable video scenes, the computable audio scene

detection made use of a model for memory. We computed three types of features from the data in the attention span — (a) vector sequences, (b) scalar sequences and (c) point data. Then, we presented a new *sines+trends+noise* signal model for scalar sequences that additionally incorporated Bregman's observation on audio source continuity. Then we presented dissimilarity measures for each generic type of feature.

The a-scenes were detected in two stages. First, we computed the correlation of the *feature* values computed in the attention-span with the rest of the data. Then, for each feature, we computed a scene change location by detecting the local maxima of the rate of increase of the feature distance. Then, these local maxima computed from each feature are merged in a simple voting scheme. In this chapter, we also presented our silence detection algorithm.

Finally in chapter 5, we presented our approach to detecting computable scenes via a multi-modal fusion framework. Here, we presented rules that incorporated higher forms of knowledge such as structure and silence information to detect computable scenes. The higher forms of knowledge (e.g. the fact that viewer group the shots in a dialog sequence to be part of the same scene) are extremely important, since these groupings (rules) *cannot* be inferred from an analysis of the feature level data. We also presented several cases where the assumptions underlying the computational model, break down.

11.2.2 Structure detection

This thesis presented a novel topological framework for detecting a priori known structures. We defined structure in terms of the compressibility of the data, using the framework of Kolmogorov complexity. In this work, we have focused on detecting discrete time, temporal structures that have known generative mechanisms. We also assumed that the elements of the sequence have a metric associated with them.

In chapter 6, we presented our approach towards structure detection. Central to our framework on structure detection is the idea of the topological graph that is constructed from the sequence to be analyzed. This is a fully connected graph that contains shots at the nodes and whose edge strengths are defined using the dissimilarities between the shots in the nodes. We also introduced a topological matrix, associated with each topological graph.

We presented two general techniques for structure detection:

- Exploiting the structure of the topological matrix for detection. This was used for detecting dialogs.
- Using random permutations of the topology to determine the presence of structure. This was used in regular anchor detection.

While we have applied the theory to detecting visual patterns, the framework that we present is applicable to other problems that have deterministic generative mechanisms.

However, a lot of work still needs to be done — note for example, that we do not handle the case of stochastic generative models.

11.2.3 Summarization

In this thesis, we presented a novel approach toward summarizing audio-visual computable scenes. These are the key ideas in our framework:

- An entity-utility framework for skim generation. In this approach, we attempt to preserve those entities that satisfy the user information needs. The skim is generated by maximizing the utilities associated with those entities that we are trying to preserve in the skim.
- The idea that comprehensibility of a shot is related to its visual complexity. We define visual complexity using the idea of Kolmogorov complexity.
- The idea that preservation of video syntax is essential to keeping the skim coherent. We show reduction mechanisms for specific syntactical elements, that preserve the semantics associated with the syntactical elements.
- The use of prosody for detecting new topic beginnings in the speech data.

In chapter 8, we presented our visual analysis for skim generation. There were two key ideas in this chapter — (a) the notion of visual complexity and its relation to comprehension time and (b) the idea that preservation of visual syntax is essential to making the resulting skims coherent.

In chapter 9, we presented our auditory analysis for skim generation. There, we did two things in parallel — (a) we classified segments using a robust SVM classifier, into three classes and (b) we detected significant phrases (i.e. SBEG's) by using the acoustic correlates of prosody. We merged the results from the two components and in the end, we were left with a sequence of labeled audio segments that were ranked according to their significance.

In chapter 10, we presented our utility based framework for skim generation. The goal of this framework was to maximize the utility of those entities that satisfy the users information needs. In this work we were interested in generating *passive* skims, and hence the entities to be retained in the skim are a priori known. There were three broad issues that were raised in this chapter — (a) identification of the specific entities that we need to preserve, (b) the utilities that we need to associate with these entities and (c) the search strategy for ensuring that the existence of a feasible solution to the optimization problem.

We conducted user studies that compared the output of the optimal skim generation mechanism against two other algorithms. The user studies indicate that the skims were received well by the users, and all were regarded as coherent. However, the improvements of the optimal skim over the other two algorithms were significant in a statistical sense, only at the highest condensation rates — 80% and 90% condensation.

11.3 Improvements to the framework

The task of building a system that *automatically* summarizes the entire video is a challenging one. While the thesis began as an effort to present a complete solution to the problem, there are necessarily gaps to be found in the framework that need to be addressed in the future.

11.3.1 Complex memory models

The memory models presented in sections 2.5.2.2, 3.7.1-3.7.2 are simplistic in their approach to memory. They do not take in account the structure of the shot sequence (or the structure in the audio data), or the context in which the shot (or the audio segment) appears in the video. The context in which the shots appear and the order in which they appear are important, since they (e.g. shot repetition) can serve to reinforce certain emotional responses to the data.

It would also be interesting to create a model of memory that is able to incorporate top-down rules (e.g. directorial styles.). For example, often when we are watching the films of a particular director, we have become “trained” to that director’s visual style (specific color compositions, the pace of the film, and any non-linear story mechanisms.). Hence incorporating such top-down rules will adapt the memory models to new situations.

11.3.2 Film-making constraints

In this thesis we have incorporated several important ideas from film-making — the importance of visual syntax for comprehension, camera placement rules that were used for segmentation and making the sound flow over the cut (i.e. allowing sound segments to overlap while creating skims (ref. section 10.4.2)). However, there were other constraints that were not included.

Continuity editing refers to a set of principles that attempt to make the shot cuts invisible (or in other words, continuous) to the viewer. The camera arrangement rule (ref. section 2.5.3) is an example of the principle. We now give more examples [8] :

- **Match on action:** Motion continuity is an important aspect of continuity editing. Here, the director will attempt to preserve the direction of motion across shots. For example, if in a shot an actor moves from left to right, then in the next shot (even if it is part of another scene), the director will have camera movement / an object / actor move in a left to right fashion.
- **The 30° rule:** Directors ensure that the successive positions of the camera at least differ by thirty degrees, from shot to shot. The reason that they do this is because if the difference in the viewing angle is too small, the two shots will look very similar and the cut between shots will seem abrupt.
- **Match on location:** Often, when directors are showing a sequence of faces (or objects), they will ensure that the location of the faces / objects are “matched” i.e. they occur in the same location on the screen. For example, in a dialog scene,

the director will ensure that the locations of the actors with respect to the frame, remains the same, over the course of the dialog.

11.3.3 Joint audio-visual experiments

In this thesis, we have approached the problem of generating an audio-visual skim, by condensing the audio and the video segments independently. We conducted experiments on the relationship of visual complexity to visual comprehension time and determined duration bounds on visual comprehension.

However, this assumption that the audio and the video data do not interact when we comprehend multimedia data is simplistic. This interaction can clearly affect the time for comprehension — for example, the viewer may not mind watching a video that has plenty of short duration video shots, as long as the audio is long and coherent (e.g. MTV videos). A slideshow is another example where the user doesn't mind seeing still key-frames of the shot, as long as the audio data is long and coherent.

The existence of a joint model may also allow us to perform a tradeoff in two dimensions simultaneously — time (i.e. the duration of the video shot / audio segment) and spatio-temporal quality (i.e. spatial resolution of the video over the duration of the shot, the quality of the audio over the duration of the audio segment.). Hence, it is important that we conduct psychological experiments that investigate the joint coupling of the audio-visual data on comprehension.

11.3.4 Fine-grained estimates of complexity

In this thesis, we use a single measure of complexity, per shot. Implicit in this approach is the assumption that the visual data does not change much over the course of the shot. However, it is clear that visual complexity is really a continuous function over the course of the shot. By using a single number for the shot, we have essentially performed a zero-order hold on the original curve.

Why is the continuous curve useful? When we reducing the duration of the shot in the current framework, the change in the utility for this shot will lie on the intersection of the $c=constant$ plane and the original utility function (see **Figure 10.1**). This implies that given a specific shot, the *rate of change* of the utility, only depends upon the duration of the shot. This will change, once the complexity measure is continuous.

The consequence for this will be felt when we are trying to decrease the duration of the entire sequence. Now, since we are using a continuous measure of complexity, the rate of change of will depend on both the current complexity value and the duration. Thus, for example, for a particular shot, the loss in utility may suddenly decrease if the current portion of the shot is of low complexity. The main advantage of such a solution is that it enables us to better adapt to changes in the shot's visual complexity in the process of condensing the shot.

11.4 Future directions

We now highlight a few possible future research directions.

11.4.1 Summaries: anytime, anywhere

One of the key contributions of this thesis is the idea of the entity-utility optimization framework. In this thesis, we have focused on the problem of automatic generation of passive summaries. While the thesis shows a framework for utility maximization of a small set of entities, there were implicit assumptions in this work, on the bandwidth, the capabilities of the device and the nature of the user interface.

It would be an important extension of this framework to come up with a single framework that given the following: (a) entities requested by the user (b) network resources (c) device capabilities (CPU speed, type of user interface) (d) utilities associated with the entities and (e) user preferences, *generates* a convex optimization problem with constraints. While some of the constraints will arise due to the properties of the individual entities, other constraints that need to be inferred from how entities interact with each other.

Clearly, the nature of the utility function will play an important role in this framework. These functions will often have to be specified by a domain expert (e.g. relationship between bandwidth and perceptual video quality); however, it is a non-trivial task even for the expert to determine the form of the utility function i.e. the relationship between the parameters that affect the utility of the entity. These ideas will follow up on recent work [17] , where we discuss several open issues in this direction, and also introduce a formal modeling of ARU (adaptation-resource-utility) spaces and their relationships.

11.4.2 Extensions to current work on summarization

The work on video summarization presented here, can be extended in at least two directions:

- Video skims that incorporate scene level structure analysis, transcript analysis and user preferences.
- Developing algorithms for other forms of summaries such as image summaries that capture the dynamism of the video (i.e., that include audio, transcripts and animations). Appendix 13.3, has more details on how image storyboards could be enhanced.

We are also interested in a particular video summarization problem of creating *dynamic summaries* for meetings. The idea involves creating summaries incrementally, allowing a participant who is late for a meeting to *catch-up* quickly.

11.4.3 Working with raw data

This thesis has focused on analyzing *produced* data from commercial films. The central idea in our work, that we can use the syntactical elements in the domain for summarization, can also be used in other produced domains, once their essential syntactical elements have been identified.

What then, of raw non-produced data? There is a tremendous amount of data that is shot by ordinary consumers in the form of home videos, as well plenty of raw footage

received by news organizations, from reporters working in the field. For example, CNN receives over 300 hours of raw footage every day.

For unstructured content, manually editing and annotating raw footage is time-consuming. Editing tools by themselves, do not ensure interesting home videos. It would be intriguing to apply the rules of cinematic syntax investigated in this thesis as well as the principles of continuity editing, onto this raw data and thereby *impose* structure on this data, thus automatically synthesize new content.

We plan to adapt the current work on audio-video segmentation and summarization to impose structure on raw video. However, a fully automatic solution to the problem is not feasible, given the diversity of user preferences. Hence, a solution would necessarily involve some user interaction.

11.4.4 Structure discovery

The work presented in this thesis deals with structure *detection*, the problem of detecting a priori known patterns in data. One of the important challenges in multimedia analysis is to be able to automatically discover the presence of structure in the data. The important questions include:

- When is a sequence said to be structured?
- What are the forms (i.e. stochastic, deterministic, specific generative mechanisms etc.) of structure?

- At what time scales does the structure manifest itself? Can we discover them automatically?

We believe that our definition of structure using Kolmogorov complexity is a useful starting point for addressing this problem, since it makes the problem of structure discovery equivalent to the problem of determining efficient compression mechanisms. One can conceive of an optimization framework that incorporates MDL (the minimum description length principle) for determination of the segment boundaries and the different time scales, as an initial guess at the solution.

11.4.5 Multimedia information visualization

An interesting problem in multimedia content analysis is to be able to visualize information in large collections of data. Examples of where this can happen include digital libraries, where the user's query may yield many valid results; thus a mechanism that allows the user to quickly drill down to the relevant video is needed.

Work at the Infromedia project at CMU has focused on the issue of visualizing the results of the video query using semantic concept maps, and additionally specific visualizations such as geographical maps and time scales, to represent the results. However, the relationship between the data and the method of visualization was project dependent, and the relationship was set by the algorithm designer.

The work at the Infromedia project is really about the *search* for the right visualization in the space of all possible visualizations (including different forms of auditory feedback). We would like to contribute to this work in two ways:

- A framework that given the specific data set, uses (a) a library of visualizations (b) device capabilities and (c) user preferences, to *automatically* determine the correct form of the visualization.
- Focus on visualization of a *single* video as opposed to a collection. This is useful in visualizing data such as surveillance videos.

11.4.6 Environment attentive algorithms

In any content analysis algorithm, there are trade-offs (with some qualifications) between representational complexity and the resulting distortion and error rate. We are interested in the problem of how real-time content analysis / classification algorithms adapt to changes in computational resources (CPU, memory, device constraints) and still give real-time performance perhaps at the expense of additional error. This idea is useful in mobile devices that have limited power and device capabilities.

Newer chips from Intel and Transmeta, dynamically change the operational speed of the CPU, in response to battery conditions. In such cases, static content analysis algorithms (e.g. video decoders, handwriting / voice recognition etc.) will no longer operate in real-time, unless they change their representational schemes. This will increase the error, but that may be tolerable in many situations.

12 References

- [1] B. Adams et. al. *Automated Film Rhythm Extraction for Scene Analysis*, Proc. ICME 2001, Aug. 2001, Japan.
- [2] L. Agnihotri, K. Devara, et. al., *Summarization of Video Programs Based on Closed Captioning*, SPIE Conf. on Storage and Retrieval in Media Databases, pp. 599-607, San Jose, CA, January 2001.
- [3] B. Arons *Pitch-Based Emphasis Detection For Segmenting Speech Recordings*, Proc. ICSLP 1994, Sep. 1994, vol. 4, pp. 1931-1934, Yokohama, Japan, 1994.
- [4] B. Arons *SpeechSkimmer: Interactively Skimming Recorded Speech*, Proc. of ACM UIST '93, pp. 187-196, Atlanta, 1993.
- [5] A. Barron, J. Rissanen, B. Yu *The Minimum Description Length Principle in Coding and Modeling*, IEEE Trans. On Information Theory, Vol. 44, No. 6, pp. 2743-2760, Oct. 1998.
- [6] A.B. Benitez, S.F. Chang, J.R. Smith *IMKA: A Multimedia Organization System Combining Perceptual and Semantic Knowledge*, Proc. ACM MM 2001, Nov. 2001, Ottawa Canada.

- [7] R.M. Bolle et. al. *Video Query: Beyond the keywords*, IBM Tech Report # RC 20586, Oct. 1996.
- [8] D. Bordwell and K. Thompson, *Film Art: An Introduction*, 6th ed, McGraw Hill, NY, NY, 2001.
- [9] A.S. Bregman *Auditory Scene Analysis: The Perceptual Organization of Sound*, MIT Press, 1990.
- [10] A.S. Bregman *Auditory scene analysis: hearing in complex environments*, appears in *Thinking in Sound: the cognitive psychology of human audition*, eds. S. McAdams and E. Bigand, Oxford University Press, pp. 10-36, 1993.
- [11] K.M. Brooks *Do story agents use rocking chairs? The theory and implementation of one model of computational narrative*, Proc. ACM Multimedia 1996, pp. 317-328, Nov. 1996, Boston MA.
- [12] G.J. Brown *Computational auditory scene analysis: A representational approach*, Ph.D. thesis CS-92-22, CS dept., Univ. of Sheffield, 1992.
- [13] B. Burke and F. Shook, *Sports photography and reporting*, Chapter 12, in *Television field production and reporting*, 2nd Ed, Longman Publisher USA, 1996
- [14] M. Burrows, D.J. Wheeler *A Block-sorting Lossless Data Compression Algorithm*, Digital Systems Research Center Research Report #124, 1994.

- [15] N. Carver, V. Lesser, *Blackboard systems for knowledge-based signal understanding*, appears in *Symbolic and Knowledge-Based Signal Processing*, eds. A. Oppenheim and S. Nawab, New York: Prentice Hall, 1992.
- [16] C.-C. Chang, C.-J. Lin, *LIBSVM: a library for support vector machines*, <http://www.csie.ntu.edu.tw/~cjlin/libsvm>
- [17] S.-F. Chang, *Optimal Video Adaptation and Skimming Using a Utility-Based Framework*, Tyrrenian International Workshop on Digital Communications (IWDC-2002), Capri, Italy, Sep. 2002.
- [18] M.G. Christel et. al *Evolving Video Skims into Useful Multimedia Abstractions*, ACM CHI '98, pp. 171-78, Los Angeles, CA, Apr. 1998.
- [19] N. Christianini, J. Shawe-Taylor *Support Vector Machines and other kernel-based learning methods*, 2000, Cambridge University Press, New York.
- [20] M.P. Cooke, *Modeling auditory processing and organisation*, Ph.D. thesis, CS dept., Univ. of Sheffield, 1991.
- [21] T.M. Cover, J.A. Thomas *Elements of Information Theory*, 1991, John Wiley and Sons.
- [22] D. DeMenthon, V. Kobla and D. Doermann, *Video summarization by curve simplification*, Proc. ACM Multimedia 1998, pp. 211-218, Sep. 1998, Bristol, UK.
- [23] N. Dimitrova, J. Martino, L. Agnihotri, H. Elenbaas, *Superhistograms for video representation*, IEEE ICIP 1999, Kobe, Japan.

- [24] R.O. Duda, Hart and D. Stork, *Pattern Recognition*, 2nd ed.
- [25] S. Ebadollahi, S.F. Chang, H. Wu, *Echocardiogram Videos: Summarization, Temporal Segmentation And Browsing*, to appear in ICIP 2002, Sep. 2002, Rochester NY.
- [26] D.P.W. Ellis *Prediction-Driven Computational Auditory Scene Analysis*, Ph.D. thesis, Dept. of EECS, MIT, 1996.
- [27] R. Elmasri, S.B. Navathe *Fundamentals of database systems*, 2nd ed. Addison-Wesley, Menlo Park CA, 1994
- [28] J. Feldman *Minimization of Boolean complexity in human concept learning*, Nature, pp. 630-633, vol. 407, Oct. 2000.
- [29] Bob Foss *Filmmaking: Narrative and Structural techniques* Silman James Press LA, 1992.
- [30] B. Gold, N. Morgan *Speech and Audio Signal Processing*, 2000, John Wiley and Sons, New York.
- [31] Y. Gong, L.T. Sin, C. Chuan, H. Zhang and M. Sakauchi, *Automatic parsing of TV soccer programs*, Proc. ICMCS'95, Washington D.C, May, 1995.
- [32] Y. Gong, Xin Liu *Generating optimal summaries*, Proc. IEEE Conf. on Multimedia and Expo (ICME) 2000, Aug. 2000, NY, USA.

- [33] B. Grosz J. Hirshberg *Some Intonational Characteristics of Discourse Structure*, Proc. Int. Conf. on Spoken Lang. Processing, pp. 429-432, 1992.
- [34] F.R. Hampel et. al. *Robust Statistics: The Approach Based on Influence Functions*, John Wiley and Sons, 1986.
- [35] A. Hanjalic, R.L. Lagendijk, J. Biemond *Automated high-level movie segmentation for advanced video-retrieval systems*, IEEE Trans. on CSVT, Vol. 9 No. 4, pp. 580-88, Jun. 1999.
- [36] M. Hansen, B. Yu *Model Selection and the Principle of Minimum Description Length*, Technical Memorandum, Bell Labs, Murray Hill, N.J., 1998.
- [37] L. He et. al. *Auto-Summarization of Audio-Video Presentations*, ACM MM '99, Orlando FL, Nov. 1999.
- [38] L. He et. al. *Comparing Presentation Summaries: Slides vs. Reading vs. Listening*, Proc. ACM CHI 2000.
- [39] L. He, A. Gupta *User Benefits of Non-Linear Time Compression*, MSR-TR-2000-96, Microsoft Corp.
- [40] J. Hirschberg D. Litman *Empirical Studies on the Disambiguation of Cue Phrases*, Computational Linguistics, 1992.
- [41] J. Hirschberg, B. Grosz *Some Intonational Characteristics of Discourse Structure*, Proc. ICSLP 1992.

- [42] J. Hirschberg, C.H. Nakatani *A Prosodic Analysis of Discourse Segments in Direction-Giving Monologues*, Association of Computational Linguistics, June 1996, Santa Cruz, CA.
- [43] J. Huang, Z. Liu, Y. Wang, *Joint video scene segmentation and classification based on hidden Markov model*, Proc. ICME 2000, P 1551 -1554 vol.3, New York, NY, July 30-Aug3, 2000
- [44] J. Huang, Z. Liu, Y. Wang, *Integration of Audio and Visual Information for Content-Based Video Segmentation*, Proc. ICIP 98. pp. 526-30, Chicago IL. Oct. 1998.
- [45] G. Iyengar, A.B. Lippman *Content-based browsing and editing of unstructured video*, Proc. ICME 2000, Aug. 2000, New York.
- [46] A. Jaimes and S.F. Chang, *Concepts and Techniques for Indexing Visual Semantics*, book chapter in Image Databases, Search and Retrieval of Digital Imagery, edited by V. Castelli and L. Bergman. Wiley & Sons, New York, 2002
- [47] R.S. Jasinschi, N. Dimitrova, T. McGee, L. Agnihotri, J. Zimmerman, and D. Li, *Integrated Multimedia Processing for Topic Segmentation and Classification*, Proc. of IEEE ICIP 2001, Thessaloniki, Greece, Oct. 2001.
- [48] R.S. Jasinschi, N. Dimitrova, T. McGee, L. Agnihotri, J. Zimmerman, D. Li, and J. Louie *A probabilistic layered framework for integrating multimedia content and context information* IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP) May 2002, Orlando, Florida, USA.

- [49] J.R. Kender B.L. Yeo, *Video Scene Segmentation Via Continuous Video Coherence*, CVPR '98, Santa Barbara CA, Jun. 1998.
- [50] D. Li, I.K. Sethi, N. Dimitrova, T. McGhee, *Classification of General Audio Data for Content-Based Retrieval*, Pattern Recognition Letters, Vol. 22, No. 5, pp. 533-544, Apr. 2001.
- [51] M. Li, P. Vitányi *An Introduction to Kolmogorov Complexity and its Applications*, 2nd ed 1997, Springer Verlag New York.
- [52] R. Lienhart et. al. *Automatic Movie Abstracting*, Technical Report TR-97-003, Praktische Informatik IV, University of Mannheim, Jul. 1997.
- [53] L. Lu et. al. *A robust audio classification and segmentation method*, ACM Multimedia 2001, pp. 203-211, Ottawa, Canada, Oct. 2001.
- [54] S. Mallat Z. Zhang *Matching Pursuit With Time-Frequency Dictionaries*, IEEE Trans. On Signal Processing, Vol. 41, No. 12, pp. 3397-3415, Dec. 1993.
- [55] D. Marr *Vision*, W. H. Freeman and Company, San Francisco, CA, 1982.
- [56] MPEG MDS Group, *Text of ISO/IEC 15938-5 FDIS Information Technology — Multimedia Content Description Interface — Part 5 Multimedia Description Schemes*, ISO/IEC JTC1/SC29/WG11 MPEG01/N4242, Sydney, July 2001.
- [57] T. Mitchell *Machine Learning*, 1997, McGraw Hill, New York.

- [58] J. Nam, A.H. Tewfik *Combined audio and visual streams analysis for video sequence segmentation*, Proc. ICASSP 97, pp. 2665–2668, Munich, Germany, Apr. 1997.
- [59] M. Naphade et. al. *Probabilistic Multimedia Objects Multijets: A novel Approach to Indexing and Retrieval in Multimedia Systems*, Proc. I.E.E.E. International Conference on Image Processing, Volume 3, pages 536-540, Chicago, IL, Oct 1998.
- [60] M. Naphade et. al *A Factor Graph Framework for Semantic Indexing and Retrieval in Video, Content-Based Access of Image and Video Library* 2000 June 12, 2000 held in conjunction with the IEEE Computer Vision and Pattern Recognition 2000.
- [61] D. O’Shaughnessy *Recognition of Hesitations in Spontaneous Speech*, Proc. ICASSP, 1992.
- [62] S. Paek and S.-F. Chang, *A Knowledge Engineering Approach for Image Classification Based on Probabilistic Reasoning Systems* , IEEE International Conference on Multimedia and Expo. (ICME-2000), New York City, NY, USA, Jul 30-Aug 2, 2000.
- [63] R. Patterson et. al. *Complex Sounds and Auditory Images, in Auditory Physiology and Perception* eds. Y Cazals et. al. pp. 429-46, Oxford, 1992.
- [64] R. Patterson et. al. *Complex Sounds and Auditory Images*, in Auditory Physiology and Perception eds. Y Cazals et. al. pp. 429-46, Oxford, 1992.

- [65] S. Pfeiffer et. al. *Abstracting Digital Movies Automatically*, J. of Visual Communication and Image Representation, pp. 345-53, vol. 7, No. 4, Dec. 1996.
- [66] S. Pfeiffer et. al. *Automatic Audio Content Analysis*, Proc. ACM Multimedia '96, pp. 21-30. Boston, MA, Nov. 1996,
- [67] D. Ponceleon et. al. *Key to effective video retrieval: effective cataloging and browsing*, Proc. ACM Multimedia 1998, pp. 99-107, Sep. 1998, Bristol UK.
- [68] W.H. Press et. al *Numerical recipes in C*, 2nd ed. Cambridge University Press, 1992.
- [69] L. R. Rabiner B.H. Huang *Fundamentals of Speech Recognition*, Prentice-Hall 1993.
- [70] K. Reisz, G. Millar, *The Technique of Film Editing*, 2nd ed. 1968, Focal Press.
- [71] J. Rissanen *Modeling by Shortest Data Description*, Automatica, Vol. 14, pp. 465-471, 1978.
- [72] Y. Rui et. al. *Automatically extracting highlights for TV Baseball programs*, ACM Multimedia 2000, pp. 105-115, Los Angeles CA, Nov. 2000.
- [73] C. Saraceno, R. Leonardi *Identification of story units in audio-visual sequences by joint audio and video processing*, Proc. ICIP 98. pp. 363-67, Chicago IL. Oct. 1998.
- [74] J. Saunders *Real Time Discrimination of Broadcast Speech/Music*, Proc. ICASSP '96, pp. 993-6, Atlanata GA May 1996.

- [75] E. Scheirer M.Slaney *Construction and Evaluation of a Robust Multifeature Speech/Music Discriminator* Proc. ICASSP '97, Munich, Germany Apr. 1997.
- [76] B. Shahraray, D.C. Gibbon *Automated Authoring of Hypermedia Documents of Video Programs*, in Proc. ACM Multimedia '95, pp. 401-409, 1995.
- [77] S. Sharff *The Elements of Cinema: Towards a Theory of Cinesthetic Impact*, 1982, Columbia University Press.
- [78] M. Slaney *A Critique of Pure Audition*, Computational Auditory Scene Analysis. Dave Rosenthal and Hiroshi Okuno (Eds.).Mahwah, NJ : Lawrence Erlbaum Associates, 1997.
- [79] M. Slaney *The AuditoryToolbox v. 2.0*, Tech. Rep. #1998-010, Interval Research Corp.
- [80] L.J. Stifelman *A Discourse Analysis Approach to Structured Speech*, AAAI 1995 Spring Symposium Series: Empirical Methods in Discourse Interpretation and Generation. March 27-29, 1995, Stanford University.
- [81] L.J. Stifelman *A Discourse Analysis Approach to Structured Speech*, appears in the AAAI 1995 Spring Symposium Series: Empirical Methods in Discourse Interpretation and Generation. Mar. 1995, Stanford, CA.

- [82] L.J. Stifelman *The Audio Notebook: Pen and Paper Interaction with Structured Speech*, PhD Thesis, Program in Media Arts and Sciences, School of Architecture and Planning, MIT, Sep. 1997.
- [83] G. Strang *Linear Algebra and its Applications*, 3rd ed. Harcourt Brace Jovanovich, 1988.
- [84] S. Subramaniam et. al. *Towards Robust Features for Classifying Audio in the CueVideo System*, Proc. ACM Multimedia '99, pp. 393-400, Orlando FL, Nov. 1999.
- [85] H. Sundaram S.F. Chang *Audio Scene Segmentation Using Multiple Features, Models And Time Scales*, ICASSP 2000, International Conference in Acoustics, Speech and Signal Processing, Istanbul Turkey, Jun. 2000.
- [86] H. Sundaram L. Xie Shih-Fu Chang *A framework work audio-visual skim generation*. Tech. Rep. # 2002-14, Columbia University, April 2002.
- [87] H. Sundaram, S.F. Chang *Determining Computable Scenes in Films and their Structures using Audio-Visual Memory Models*, Proc. Of ACM Multimedia 2000, pp. 95-104, Nov. 2000, Los Angeles, CA.
- [88] H. Sundaram, S.F. Chang, *Condensing Computable Scenes using Visual Complexity and Film Syntax Analysis*, IEEE Proc. ICME 2001, Tokyo, Japan, Aug 22-25, 2001.
- [89] H. Sundaram, S.F. Chang *Computable Scenes and structures in Films*, IEEE Trans. on Multimedia, Vol. 4, No. 2, June 2002.

- [90] H. Sundaram, Shih-Fu Chang, *Constrained Utility Maximization for generating Visual Skims*, IEEE Workshop on Content-based Access of Image and Video Libraries (CBAIVL-2001) Dec. 2001 Kauai, HI USA.
- [91] T. Syeda-Mahmood, D. Ponceleon, *Learning video browsing behavior and its application in the generation of video previews*, Proc. ACM Multimedia 2001, pp. 119 - 128, Ottawa, Canada, Oct. 2001.
- [92] Y. Taniguchi et. al. *PanoramiaExcerpts: Extracting and Packing Panoramas for Video Browsing*, in Proc. ACM MM 97, pp. 427-436, Seattle WA, Nov. 1997.
- [93] R. Tansley. *The Multimedia Thesaurus: Adding A Semantic Layer to Multimedia Information*. Ph.D. Thesis, Computer Science, University of Southampton, Southampton UK, August 2000.
- [94] V. Tovinkere , R. J. Qian, *Detecting Semantic Events in Soccer Games: Towards A Complete Solution*, Proc. ICME 2001, Tokyo, Japan, Aug 22-25, 2001
- [95] S. Uchihashi et. al. *Video Manga: Generating Semantically Meaningful Video Summaries* Proc. ACM Multimedia '99, pp. 383-92, Orlando FL, Nov. 1999.
- [96] H. Ueda et. al. *Automatic Structure Visualization for Video Editing*, Proc. ACM conf. on human factors and computing systems (INTERCHI 1993), pp. 137-141, Apr. 1993, Amsterdam, The Netherlands.

- [97] T. Verma *A Perceptually Based Audio Signal Model with application to Scalable Audio Compression*, PhD thesis, Dept. Of Electrical Eng. Stanford University, Oct. 1999.
- [98] L. Xie, S.F. Chang, A. Divakaran and H. Sun, *Structure Analysis Of Soccer Video With Hidden Markov Models*, Proc. ICASSP 2002, Orlando, FL, May 2002.
- [99] P. Xu, L. Xie, S.F. Chang, A. Divakaran, A. Vetro, and H. Sun, *Algorithms and system for segmentation and structure analysis in soccer video*, Proc. ICME 2001, Tokyo, Japan, Aug 2001
- [100] B.L. Yeo, B. Liu *Rapid Scene Analysis on Compressed Video*, IEEE Trans. On Circuits and Systems for Video Technology, pp. 533-544, Vol. 5, No. 6, Dec. 1996.
- [101] B.L. Yeo, M. Yeung *Classification, Simplification and Dynamic Visualization of Scene Transition Graphs for Video Browsing*, Proc. SPIE '98, Storage and Retrieval of Image and Video Databases VI, San Jose CA, Feb. 1998.
- [102] M. Yeung B.L. Yeo *Time-Constrained Clustering for Segmentation of Video into Story Units*, Proc. Int. Conf. on Pattern Recognition, ICPR '96, Vol. C pp. 375-380, Vienna Austria, Aug. 1996.
- [103] M. Yeung, B.L. Yeo and B. Liu, *Segmentation of Video by Clustering and Graph Analysis*, Computer Vision and Image Understanding, V. 71, No. 1, July 1998.

- [104] M. Yeung, B.L. Yeo *Video Content Characterization and Compaction for Digital Library Applications*, SPIE Electronic Imaging '97, Storage and Retrieval of Image and Video Databases, San Jose 1997.
- [105] D. Yow, B.L. Yeo, M. Yeung, and G. Liu, *Analysis and Presentation of Soccer Highlights from Digital Video* Proc. ACCV, 1995, Singapore, Dec. 1995.
- [106] H.J. Zhang et. al. *Video parsing, retrieval and browsing: an integrated and content-based solution*, Proc. ACM Multimedia 1995, pp. 15-24, Nov. 1995, San Francisco, CA.
- [107] T. Zhang C.C Jay Kuo *Heuristic Approach for Generic Audio Segmentation and Annotation*, Proc. ACM Multimedia '99, pp. 67-76, Orlando FL, Nov. 1999.
- [108] D. Zhong and S.F. Chang, *Structure Analysis of Sports Video Using Domain Models*, Proc. ICME 2001, Tokyo, Japan, Aug. 2001
- [109] D. Zhong *Segmentation, Indexing and Summarization of Digital Video Content* PhD Thesis, Dept. Of Electrical Eng. Columbia University, NY, Jan. 2001.
- [110] <http://www.gzip.org> (*gzip* version. 1.2.4)
- [111] <http://sources.redhat.com/bzip2/index.html>
- [112] http://eewww.eng.ohio-state.edu/~maj/osu_svm/

13 Appendix

13.1 Kolmogorov complexity

In this section we briefly review the idea of Kolmogorov complexity [21] [51]. Let \mathbf{x} be a finite length binary string of length n . Let $\mathbf{U}(p)$ denote the output of an universal Turing machine \mathbf{U} when input with program p . Note, a universal Turing machine \mathbf{U} is a Turing machine that can imitate the behavior of any other Turing machine \mathbf{T} . It is a fundamental result that such machines exist and can be constructed effectively [21] [51]. Then, the Kolmogorov complexity of the string \mathbf{x} is defined in the following way:

$$K_U(\mathbf{x} | n) \triangleq \min_{p: \mathbf{U}(p)=\mathbf{x}} l(p), \quad (13.1)$$

where, $l(p)$ is the length of the program p , and n is the length of the string \mathbf{x} and where $K_U(\mathbf{x} | n)$ is the Kolmogorov complexity of \mathbf{x} given n . Hence, the Kolmogorov complexity of \mathbf{x} , with respect to an universal Turing machine \mathbf{U} is the length of the shortest program that generates \mathbf{x} . The Kolmogorov complexity of an arbitrary string \mathbf{x} is non-computable due to the non-existence of an algorithm to solve the halting problem [21] [51]. In section 8.2.2, we shall show efficient asymptotic upper-bounds on Kolmogorov complexity, as well techniques to estimate complexity.

13.1.1 Estimating $K(x | y)$

In this section we show how to estimate the conditional Kolmogorov complexity of an image x given another image y . Let us assume that the size of image x is N by N pixels. We also assume that the image x can be divided up into l non-overlapping blocks of size $k \times k$ ³⁴. Then for each block B in image x we do the following:

- Find the corresponding $k \times k$ region in image y that minimizes the L^2 distortion. Note the start coordinates of the block and also store error matrix (i.e. the pixel-by-pixel difference between the two regions) to allow for exact reconstruction of the block. The optimal start locations is computed in the following way:

$$(p_x, p_y) = \arg \min_{a,b} \sum_{i=1}^k \sum_{j=1}^k \left(I_y(a+i, b+j) - B(i, j) \right)^2 \quad (13.2)$$

where, a and b are the start coordinates and can lie anywhere in image y , B is the block in image x , for which we are trying to find the best match and where k the size of the block in image x .

³⁴ The fact the image and the block are both square is for clarity of explanation. The image and the blocks could be rectangular; additionally the blocks could also be of different sizes.

- Repeat step 1 for all the blocks. Now, we have the locations of the corresponding regions in image y for each block in image x as well as the error matrix.
- Now, construct one dimensional sequence of symbols in the following manner.. We process the blocks in image x in raster scan order. Then for each block in image x :
 - First store the pixel locations of the best match in image y . i.e. store the locations as (p_x, p_y) .
 - Store the error matrix in raster scan order.
 - Repeat the first two steps until we have processed all the blocks in image x .
 - At the end of the previous step, we would have constructed an one dimensional sequence of symbols. Now compress this one dimensional sequence using a standard compressor such as bzip2 [111] .

We have thus computed $K(x|y)$, the conditional Kolmogorov complexity of image x with respect to image y .

13.1.2 Estimating $K(x|A)$

In this section, we show how to compute $K(x|A)$, the conditional Kolmogorov complexity of an image x given a buffer of images A . Computing this measure is very similar to computing $K(x|y)$, the conditional Kolmogorov complexity of image x when given image y , presented in the previous section.

As in the previous section, let us assume that the size of image x is N by N pixels.

We also assume that the image x can be divided up into l non-overlapping blocks of size $k \times k$. Now, additionally assume that we have r images in buffer A . Then, the procedure to compute the Kolmogorov complexity remains identical to the previous section, with one important difference — the minimization in equation (13.2) is now to be carried out over the entire buffer. Thus:

$$(p_x, p_y, q) = \arg \min_{a,b,n} \sum_{i=1}^k \sum_{j=1}^k (I_n(a+i, b+j) - B(i, j))^2 \quad (13.3)$$

where as before, n is the image index in the buffer A and can be any one of the r images, a and b are the start coordinates and can lie anywhere in image I_n , B is the block in image x , for which we are trying to find the best match and where k the size of the block in image x . The rest of the procedure to calculate the conditional Kolmogorov complexity remains the same as the previous section.

13.2 The phasor distance

A phasor $p(n) = a \exp(-j(\omega n + \theta))$ is represented using an ordered triple: $\{a, \omega, \theta\}$.

Where, a is the amplitude of the phasor, ω is the frequency and θ is the initial phase. To determine the distance between two phasors, assume that we are given two phasors $p(n)$ and $q(n)$ with corresponding triples $\{a, \omega_1, \theta_1\}$ and $\{b, \omega_2, \theta_2\}$. We further assume that they are defined over $n \in [0, ..N-1]$. From figure A.1 it is easy to see that the instantaneous distance is:

$$d(p, q, n) = \sqrt{a^2 + b^2 - 2ab \cos((\omega_1 - \omega_2)n - (\theta_1 - \theta_2))}. \quad (13.4)$$

Now, the squared distance averaged over N points is:

$$D(p, q) = a^2 + b^2 - 2ab \frac{1}{N} \operatorname{Re} \left\{ \exp(j\theta) \left(\frac{1 - \exp(j(\omega_1 - \omega_2)N)}{1 - \exp(j(\omega_1 - \omega_2))} \right) \right\}. \quad (13.5)$$

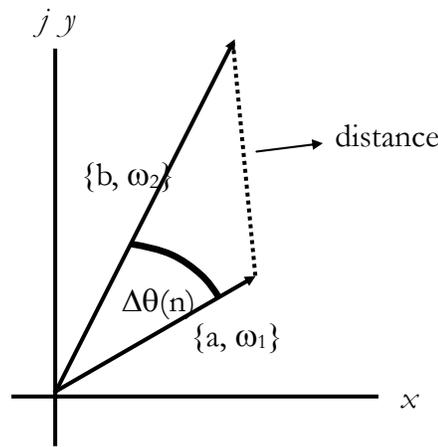


Figure 13.1: Instantaneous distance in the complex plane, between two phasors $\{a, \omega_1, \theta_1(n)\}$ and $\{b, \omega_2, \theta_2(n)\}$ at time n .

Such a distance formulation has many advantages.

- The distance should be a function of all components of each phasor $\{a, \omega, \theta\}$ as well as the number of data points, N .
- The squared distance between two phasors will be uniformly distributed in the range $[a^2 + b^2 - 2ab, a^2 + b^2 + 2ab]$. Hence, for large values of N , we should expect D to converge to $a^2 + b^2$ (weak law of large numbers).

1. For small N , we should expect the distance to be affected by the initial phase and this should gradually be less significant.

13.3 Enhancing image storyboards

Conventional image based storyboards do not capture the dynamism of the underlying audio-visual data. Hence there has been some effort to improve the interactivity of these schemes [95] . There, the image summary was enhanced with text (either from manual transcripts or OCR) and presented in a *manga*³⁵ like fashion. We outline four possible ways of enhancing current image summarization schemes.

- **Text balloons:** If we have text aligned transcripts, then it may be possible to extract the important sentences corresponding to the cluster and then displaying them when the user moves the mouse over the relevant image.
- **Audio segments:** If we perform an acoustic analysis of prosody [40] , then we can identify important boundaries in the discourse (both for spontaneous and structured speech). Then, we could associate each key-frame with this audio segment. In the usage scenario, the user would click on the key-frame to hear the corresponding audio segment.
- **Animations:** At present image based storyboards are static i.e. the image representing each cluster does not change over time. An interesting variation

³⁵ Manga is the Japanese for a comic book.

would be to represent each image by an animated GIF, which cycles through other images in the cluster when the user moves the cursor over the storyboard key-frame. Another attempt at infusing dynamism is the *dynamic STG* [101] in which the shots comprising each cluster are rendered slowly over time.

- **Structure and syntactical highlights:** Many domains possess very specific rules of syntax and characteristic structural elements that are meaningful in that domain. Examples of structure in films include the dialog and the regular anchor (ref. section 6.5). Specialized domains such as baseball, echocardiogram videos have a very specific syntactical description. These domains would greatly benefit from higher-order domain grouping rules that arranges the key-frames of the cluster in a manner highlighting the rules of syntax and the domain specific structures.

13.4 Skims: The first user study

This was the first user study done for the evaluation of visual skims [88] . We now note the important differences with the audio-visual skim generation discussed in chapter 10:

- The skims were generated using visual analysis alone. They contain no audio, i.e. they are silent.
- There is no utility framework for modeling user comprehensibility, thereby precluding a principled generation of the skim.

- Without the user model, it is, for example, hard to quantify the effect of reducing a shot by say 10% on comprehensibility.
- Lack of a utility model also has another effect. Within the framework of this experiment, it is not possible to come up with a unique way to achieve a certain condensation rate. Say we want to condense a sequence by 50%. Then, this condensation could be achieved either by condensing all the shots by 50%, or by dropping half the shots, or perhaps a combination of the two. What is optimal? This problem is addressed with the utility optimization framework of chapter 10.
- The syntax based shot dropping strategy in chapter 10 is more sophisticated, since it incorporates the shot detection false-alarm rate when making a decision. Also, for a dialog, in this experiment, the shots were only dropped from the right.

The scenes used for creating the skims were from four films: *Blade Runner* (*bla*), *Bombay* (*bom*), *Farewell my Concubine* (*far*), *Four Weddings and a Funeral* (*fou*). The films were chosen since they exhibit diversity in film-making styles.

13.4.1 Experimental set-up

For each video sequence, it is possible to condense the sequence to any condensation rate. Also note that since we are conducting a user study, it is only possible to coarsely sample the time complexity graph **Figure 8.4** (note that each shot is a single point on this graph), since each user can only see a small number of skims before getting fatigued.

Since this was to be our first experiment, we chose to test our theories at critical locations, where we felt that it may break down.

The skims were of the following types:

- Upper bound (U_b): each shot was reduced to its upper bound (ref. Section 8.2.4) after estimating its visual complexity. In this skim, no syntax reduction scheme is employed. i.e. all the shots in the original sequence are present.
- Lower bound (L_b): each shot was reduced to its lower bound (ref. Section 8.2.4) after estimating its visual complexity. In this skim, no syntax reduction scheme is employed. i.e. all the shots in the original sequence are present.
- Pure syntax (P_s): In this skim, we employ maximal syntax based reduction. However, all the shots that are present in this skim, are kept at their original lengths. i.e. none of the shots in this skim form undergo visual complexity based duration reduction.
- Syntax with upper bound (P_s-U_b): In this skim, we employ maximal syntax based reduction, and all the shots that remain in the skim are reduced to their upper bound.
- Syntax with lower bound (P_s-L_b): In this skim, we employ maximal syntax based reduction, and all the shots that remain in the skim are reduced to their lower bound.

Hence, each skim represents a maximally reduced skim for that type (**Table 13.1**). We conducted a pilot user study with five PhD students. The study used four films (one clip

from each), with five skims per clip. Each clip had progressive and a dialog scene.

The testers were largely unfamiliar with the films (each film on the average was familiar to 1.5 students) and were expected to evaluate the skims on two metrics:

- Coherence: do the sequence of shots tell a story?
- Skip original?: confidence that having seen the skim, there is no need to see the original.

The metrics were on a scale of 1-7 (strongly disagree = 1 and strongly agree = 7). None of the clips or the skims had audio. We additionally asked each tester to rate the “best” and the “worst” skim per clip. In case of ambiguity, they could name more than one “best/worst” skim.

Table 13.1: Skims lengths in seconds for each clip and skim type. The numbers in brackets represent the percentage reduction. The films in order: *Blade Runner*, *Bombay*, *Farewell my Concubine*, *Four Weddings and a Funeral*.

Film	Orig.	U_b	L_b	P_s	P_s-U_b	P_s-L_b
bla	184	44 (76)	21 (89)	114 (38)	35 (81)	16 (91)
bom	114	45 (60)	21 (82)	41 (64)	22 (81)	10 (91)
far	153	53 (65)	26 (83)	103 (33)	31 (80)	15 (90)
fou	165	31 (81)	14 (92)	58 (65)	17 (90)	8 (95)

13.4.2 Results

Table 13.2: Test scores from five users. C: coherence, S_o : skip original? The last two rows represent best/worst preferences.

Film	U_b		L_b		P_s		P_s-U_b		P_s-L_b	
	C	S_o	C	S_o	C	S_o	C	S_o	C	S_o
bla	5.8	4.6	5.0	4.0	6.8	5.6	5.2	4.2	5.4	4.2
bom	6.6	6.4	5.6	5.6	6.0	5.0	5.0	4.0	4.6	3.8
far	6.2	5.6	5.8	5.6	5.4	4.2	3.8	3.6	4.0	3.6
fou	6.0	4.6	5.4	4.4	5.6	3.8	5.4	3.0	4.8	3.0
all	6.15	5.3	5.45	4.9	5.95	4.65	4.85	3.7	4.7	3.65
best	9		5		4		2		1	
worst	1		3		3		6		9	

We showed the users a test clip and explained the two questions of coherence and “skip original?” We then showed the original clip, two skims, the original clip again and then followed by three more skims. The original was shown first to establish a context and then shown again in the middle of the test to refresh the user. This procedure was repeated for the remaining three clips. For each clip and each test taker, we randomized the order of the skims. The results are shown in **Table 13.2**.

13.4.3 Discussion

The raw test scores as well as the “best/worst” classification by the user (**Table 13.2**) clearly indicate that the upper bound (U_b) works well. The use of syntax has mixed results; while P_s and P_s-U_b get high coherence scores, they are not consistently judged to

be the best (only 6/21). P_s-L_b has the maximum data reduction, hence it is not surprising that it fares poorly.

The **Table 13.2**, has some interesting trends that are worthy of scrutiny:

- For each film, the coherence of the upper bound skim is greater than the coherence of the lower bound skim. This implies that for the same sequence complexity, decreasing the shot durations decreases intelligibility of the skim.
- When the skim is generated in the “pure syntax” mode, they are still perceived as highly coherent (average score: 5.95). However, when coupled with complexity based reduction (upper and lower bound based reduction), the perceived coherence falls off dramatically. This leads us to conclude that the information loss due to dropping a shot may not be perceptible when the shots are close to the original lengths (i.e. at low condensation) and become more rapidly perceptible at high condensation rates.

A note on the user study. One of the things that became apparent after the user study was that the participants were sometimes judging the quality of the skim based on comparisons with the original, using specific markers. For example, if a shot containing a flag that was present in the original, but was missing from the skim, the participants would label it as less coherent, whereas they ought to have judged the skim on its own merit. This “pattern-recognition” that seemed to have taken place in the experiment

prevents us from improving the skim quality in a meaningful way, since skims by definition entail a loss of information.

13.4.4 Conclusions

We conducted a pilot user study on four clips by using five different skim types, each generated at maximal condensation. The results of the user study indicate that while all skims are perceived as coherent ($C > 4.7$) the upper bound based skim (60~80% condensation) works the best with the syntax based summaries providing mixed results.

13.5 Skims: The second user study

This section will detail the results of the second user study done at Columbia. These are the main differences with respect to the optimal skim generation algorithm in chapter 10.

- The skims used in this experiment have no audio i.e. they are silent.
- The utility based skim generation formulation used in this experiment is less sophisticated than the one described in chapter 10. This is because of the following reasons:
 - There is no utility formulation for audio data.
 - The incorporation of audio necessitates the development of coupled constraints on both audio and video, making the optimization more difficult.

The scenes used for creating the skims were from four films: *Blade Runner (bla)*, *Bombay (bom)*, *Farewell my Concubine (far)*, *Four Weddings and a Funeral (fou)*. The films were chosen

for their diversity in film-making styles. While we arbitrarily picked one scene from each film, we ensured that each scene had one dialog segment. We detected shots using the algorithm to be found in chapter 2, [109] , and with the shot-detector parameters {motion: M, color: L} (ref. **Table 8.2**).

13.5.1 Experimental set-up

We used this study to compare two algorithms — the one described in section 13.1, [88] and the one described in [90] . Briefly, in the first algorithm, we attempted reduction to the target time by proportionately reducing the duration of each shot. If the target time could not be met in this manner, we would then drop shots according to rules of syntax. The second algorithm uses a utility based skim generation formulation, that also incorporates shot detector uncertainty in its formulation.

We created two skims (one from each algorithm) at each of the four different condensation rates (90%, 80%, 70% and 50%). Ideally, we would liked to have created one skim per condensation rate, per film. However, this would have meant that the user would have to rate 32 skims (two algorithms, four rates, four scenes), an exhausting task. We ordered the scenes according the number of shots, and the scene with the maximum number of shots was compressed at 90%; the scene with the next highest number of shots at 80% and so on. A little thought indicates that this is a difficult test set for testing our algorithm.

We conducted a pilot user study with five graduate students. Each film was on the average, familiar to 2.25 students. The testers were expected to evaluate (**Table 13.3**) each skim, on a scale of 1 - 7 (strongly disagree – strongly agree), on the following metric: Does the sequence tell a story? They were additionally asked to rate their confidence in answering the four generic questions of who? where? when? what? for the four skims (ref. Section 8.2.3). Each user watched the skims in random order. After the viewers had evaluated all the eight skims, we also asked the users to evaluate the two skims at each condensation rate. For each pair, they were asked indicate (**Table 13.4**) degree of agreement (scale 1 - 7) with the statement: skim A is better than skim B.

13.5.2 Results

The test scores indicate that the new improved algorithm outperforms the old one at almost every condensation rate and at every question. This improvement is statistically significant. We computed the students t-test on the result of the direct skim comparison. The null hypothesis that the two skim algorithms were identical was rejected at confidence level better than 99.99%.

Table 13.3: Test scores from five users. The columns: the algorithm used, the film, condensation rate, and the five questions.

Algo.	Film	Rate	Story?	where?	what?	who?	when?
new / old	bla	90	4.8 / 4.0	6.8 / 6.6	5.8 / 5.6	6.6 / 6.8	5.2 / 5.0
change			+ 0.8	+ 0.2	+ 0.2	- 0.2	+ 0.2
new / old	far	80	5.8 / 5.2	7.0 / 6.8	6.4 / 6.2	7.0 / 6.8	6.4 / 6.4
change			+ 0.6	+ 0.2	+ 0.2	+ 0.2	0.0

new / old	bom	70	6.4 / 5.6	7.0 / 6.8	6.4 / 6.2	7.0 / 7.0	6.2 / 6.2
change			+ 0.8	+ 0.2	+ 0.2	0.0	0.0
new / old	fou	50	6.4 / 5.8	7.0 / 7.0	6.4 / 6.0	7.0 / 6.8	5.8 / 5.4
change			+ 0.6	0.0	+ 0.4	+ 0.2	+ 0.4

Table 13.4: The result of the double-blind side-by-side comparison of the two skim algorithms. The new algorithm is preferred over the old one for all condensation rates. This improvement is statistically significant.

Rate	New > Old?
90 %	5.0
80 %	5.2
70 %	6.2
50 %	5.6
<i>All</i>	5.50

13.5.3 Discussion

From the **Table 13.3**, it seems that the excellent user feedback tapers off at the 80% condensation rate (the story? column in **Table 13.3**), indicating that perhaps the syntax based reduction is too restrictive; at 90% condensation rate it may be better to drop more shots thus increasing the average shot duration of the remaining shots.

13.5.4 Conclusions

We conducted a pilot user study on four scenes, generating skims using two skim generation algorithms, at four different condensation rates. The results of the user study shows that while all skims are perceived as coherent, the skims generated at less than

80% condensation work well. We also showed that the improvement in skim comprehension, due to the new algorithm over the old one is statistically significant.