Computing Department
Lancaster University

Automating Video Annotation

Edward Hartley

Submitted for the degree
of
Doctor of Philosophy
July 15, 2004

Abstract

Video annotation enables novel applications in moving picture production, archiving and other domains. Annotation associates descriptive material with moving picture sequences. Here, a bi-directional approach is adopted that combines bottom-up image feature extraction with a top-down content representation model. This model contains a generic framework for time-based visual content representation, from the narrative and event levels, to visual objects and their attributes. The model is developed from a combination of: interval graphs to represent temporal relationships, the application of scripts and conceptual dependency to represent events, and conceptual graphs to represent object interrelationships. Worked examples show that the model can be refined for specific domain applications. Continuous domain feature visual-object identification and location techniques are described. A transform domain colour and spatial frequency feature extraction and edge detection method is developed. It is shown that modifying a video decoder produces parallel feature and image streams. An end-user tool architecture, its implementation status, a methodology to support annotation through incremental knowledge base instantiation and overt "binding" of features to perceived objects using search and inference is described. Extensions to the semiotic model and their application to the development of the top-down content model are described. The basis of these developments in an analysis of video content representation techniques and their relationship to artificial intelligence, narrative, film theory, semiotics, continuous domain content-based image retrieval and still and moving picture compression technologies is described. An overall qualitative evaluation is performed. An assessment of the method's psychological plausibility is made. Outlines of quantitative evaluation and further research are given.

How to train the mind in the profound view of the middle way

I seek your blessings to realize the meaning of Nagarjuna's intention,
That there is no contradiction but only harmony
Between the absence of even an atom of inherent existence in samsara and nirvana
And the non-deceptive dependent relationship of cause and effect.
(Offering to the Spiritual Guide, p24, compiled by Losang Chokyi Gyaltsan,
reproduced with permission, Tharpa Publications, Ulverston, England Glen Spey, New York [115])

Acknowledgements

The financial support of the EPSRC is gratefully acknowledged under grants GR L81/604 Multimedia Information Modelling (MIM) GR M81/755 Automating Video Annotation AVA. Participation in MPEG standardisation activies was supported by grants GR S42 1871/01 and GR S421871/01.

Financial support has also been provided by Hartley family resources.

Contents

Abstract **i**

ii

iii

Acknowledgements **iv**

Contents **v**

List of Figures **xii**

List of Tables **xvi**

1 Introduction **1**

1.1 Video annotation: an enabling technology 1

1.2 Scope 1

1.3 Goals, objectives and claims 5

1.3.1 Research goals 6

1.3.2 Research objectives 7

1.3.3 Requirements 11

1.3.4 Claims 15

1.4 Approach 17

1.4.1 Bi-directional aspects 17

1.4.2 Software design 18

1.4.3 Software implementation 18

1.5 Structural and content overview 19
1.5.1 Chapter 2 Video theory and content modelling 20
1.5.2 Chapter 3 spatio-temporal segmentation. 21
1.5.3 Chapter 4 AVA architecture and methodology. 22
1.5.4 Chapter 5 evaluation future work and conclusions 23
1.5.5 Appendices 23

2 Video theory and content modelling 25

2.1 Introduction 25

2.2 Semiotics 26
2.2.1 Development of Semiotics 26
2.2.2 Types of sign 27
2.2.3 Arbitrariness of signs 28
2.2.4 Content and expression 29
2.2.5 Denotation and connotation 29
2.2.6 Paradigmatic and syntagmatic axes 30
2.2.7 Application of semiotics 30

2.3 Narrative theory 30
2.3.1 Semiotics of narrative and film 31
2.3.2 Story and discourse 32
2.3.3 Process or stasis 33
2.3.4 Story and plot 34
2.3.5 Representing events in scenes 36
2.3.6 Explicit and implicit scenes 37

2.4 Film theory 38
2.4.1 Film language and Christian Metz 39
2.4.2 Pasolini and the cineme 40
2.4.3 Eco: iconic signs, semes and cinematic articulation 41
2.4.4 The practitioners perspective; Arijon's grammar 44
2.4.5 Elements of Arijon's lanaguage 45
2.4.6 Additional rules in the grammar of the film language 49

2.5 Artificial Intelligence 49
2.5.1 Knowledge representation 50
2.6 Content modelling approaches 52
2.6.1 Bloch 54
2.6.2 Parkes and CLORIS 55
2.6.3 Nack and Auteur 57
2.7 Knowledge representation and feature extraction 58
2.7.1 Representing colour 59
2.7.2 Representing texture 60
2.7.3 Representing edges 61
2.7.4 Representing objects 61
2.7.5 Class membership 62
2.7.6 Representing which time? 64
2.8 AVA content model development 67
2.8.1 Revising semiotic content models 68
2.8.2 Applying the semiotic model 70
2.8.3 The extended semiotic model 70
2.8.4 The R2 semiotic model 71
2.8.5 The classical content model 73
2.8.6 Description of existents 77
2.9 Structure of the AVA content model 79
2.9.1 Description to image feature and region binding 80
2.9.2 Existent relationships 81
2.9.3 The existent context 82
2.9.4 Spatial and temporal decomposition 83
2.9.5 Existent context set 84
2.9.6 A replacement for the setting 84
2.9.7 Representing actions in scenes 86
2.10Applying Schank's conceptual dependency theory 87
2.10.1Notation 88
2.10.2Conceptual categories in conceptual dependency 89

2.11Completing the AVA model 99

2.12Conclusions 99

3 Image and video analysis 104

3.1 Introduction 104

3.2 Preliminaries 105

3.2.1 Colour spaces 105

3.3 Image and video compression 109

3.3.1 Overview 109

3.3.2 Still picture compression 111

3.3.3 Moving picture compression 111

3.3.4 MPEG-1 visual 112

3.3.5 MPEG-2 visual 115

3.3.6 MPEG-4 visual 116

3.3.7 MPEG-4 advanced video coding 116

3.4 Visual object recognition and location 118

3.4.1 Visual object recognition 118

3.4.2 Image histograms 119

3.4.3 Visual object location 120

3.5 Temporal segmentation 121

3.5.1 Temporal segmentation of uncompressed video: cut detection 121

3.5.2 Temporal segmentation of uncompressed video: other transitions 124

3.5.3 Histogram comparison methods 124

3.5.4 Temporal segmentation of MPEG video 127

3.5.5 DC term and macro-block coding mode methods 130

3.6 MPEG decoder modifications 131

3.7 Novel transform domain image analysis 135

3.7.1 Preliminaries 135

3.7.2 Edge detection 138

3.7.3 DCT edge filter and steerable gaussians 140

3.7.4 Spatial frequency texture mapping 141

3.7.5 Transform domain colour histogram 144
3.7.6 Grey level contribution 144
3.7.7 Transform domain edge histogram 145
3.7.8 DCT domain image analysis conclusions 145
3.7.9 DCT domain temporal analysis 145

3.8 Conclusions 146

4 AVA architecture and methodology 148

4.1 Overview 148

4.2 AVA User Interface 150
4.2.1 The video player 150
4.2.2 The still image drawing area 151
4.2.3 Information pane 154
4.2.4 Schema definition 154
4.2.5 Knowledge base 155
4.2.6 The forms interface 156

4.3 Still and moving picture codecs 157

4.4 Numeric processing 158

4.5 Feature extraction 159

4.6 Video content representation object factories 159

4.7 Knowledge and feature databases 160

4.8 Inference system(s) 162

4.9 AVA system controller 162

4.10 XML schema serialisation 163

4.11 Annotation methodology 164

4.12 Annotation procedure 164

4.13 A childrens' TV example 171

4.14 Conclusion 177

5 Evaluation, Future Work and Conclusions 179

5.1 Overview 179

5.2 Evaluation against requirements 180

5.2.1 Content model 180

5.2.2 Feature extraction 182

5.2.3 Temporal segmentation 183

5.2.4 Video decoder modifications 183

5.2.5 Transform domain feature extraction 184

5.2.6 Architecture and annotation methodology 184

5.2.7 Development process 189

5.2.8 Implementation approach 190

5.2.9 Software engineering considerations 192

5.3 Quantitative evaluation proposals 192

5.3.1 Transform domain video feature extraction 192

5.3.2 Transform domain temporal segmentation 193

5.4 Future work 193

5.4.1 Database extensions 193

5.4.2 Further integration 194

5.4.3 Existent, existent context and context set identification 195

5.4.4 Hardware implementation 196

5.5 Psychological plausibility of the model and fNMRI 197

5.6 Standards for content description exchange and markup 198

5.7 Conclusions 200

A MPEG input documents 202

A.1Technology Input Documents 202

A.2Procedure and Process Input Documents 203

B Formal definition of the CLORIS model 205

C Story Grammars 210

D Teletubbies Representative Frames 213

E Python source code listings **217**

E.1 Video Decoding 217

E.1.1 Semantic and visual factory classes 224

F Interface and C source code listings **239**

F.1 SWIG interface 239

F.2 Helper functions 240

F.3 Video Decoding 243

G Matlab Source **277**

References **280**

Filmography **300**

List of Figures

1.1 Media creation cycle 17

2.1 Pierce's tripartite sign 27
2.2 Content and expression 29
2.3 Story and discourse 33
2.4 Content and expression of story and discourse 34
2.5 Implicit scene in Teletubby land 38
2.6 Kernel and other events 39
2.7 The grand syntagmatique of Metz 40
2.8 Iconic signs, semes, kines and kinemorphs 44
2.9 Close up Teletubby 46
2.10 Close shot Teletubby 47
2.11 Medium shot Teletubby 47
2.12 Full shot Teletubby 47
2.13 Full shot Teletubby 48
2.14 Content modelling concerns 53
2.15 Assumed structure of films considered by Parkes 56
2.16 The CLORIS content model 57
2.17 Content representation in Auteur 58
2.18 An example of retrieval in Auteur 59
2.19 Allen's [147] binary temporal relationships 65
2.20 Revised Semiotic Model 70
2.21 Application of the semiotic model 71
2.22 Extended semiotic model 72

2.23 R2 semiotic model 73
2.24 Classical content model in UML 74
2.25 Substance of the content and data of narrative, ... 75
2.26 Visual to semantic class binding points 77
2.27 Content and description of story and plot 78
2.28 Substance of content, data, description and dxpression of story and plot 78
2.29 Substance of content, data, description and expression of existents 79
2.30 AVA description of existents 80
2.31 AVA existent relationships 82
2.32 The existent context in UML 83
2.33 The existent context set in UML 85
2.34 The narrative context in UML 86
2.35 Narrative bridging by description revised in UML 101
2.36 Narrative content and data bridged by description in UML 102
2.37 Narrative data and content bridging by description 103

3.1 Image colour planes and space 107
3.2 Representation of RGB to HSV conversion 109
3.3 Taxonomy of moving picture technologies 110
3.4 JPEG image compression 110
3.5 Zig-zag DCT coefficient scanning 111
3.6 MPEG-1 video decomposition hierarchy 115
3.7 MPEG-1 picture types 115
3.8 Camera movement motion field patterns 128
3.9 MPEG dual player showing I frames only 134
3.10 $\nabla_2 G$ curves for varying σ 139
3.11 $FFT(\nabla_2 G)$ curves for varying σ 139
3.12 $FFT\Delta_2 G$ Curves for varying σ 141
3.13 Approximation to $FFT\nabla_2 G$ 142
3.14 Results of applying filter $\sigma = 0.1$ 142
3.15 Regions in DCT block for texture mapping 143

3.16 Region 6 binary mask results 143
3.17 Texture space 144

4.1 AVA architecture overview 149
4.2 The AVA user interface 151
4.3 The video player component 152
4.4 The image draw component screen shot 153
4.5 The Image Draw Component Screen Shot 153
4.6 AVA interface showing object inspection 155
4.7 A Portion of a description hierarchy 157
4.8 AVA user interface graph pane 158
4.9 Form invoked from image annotation 159
4.10 Naive content model 160
4.11 Early overview of the annotation process 161
4.12 AVA database 162
4.13 Spatio-temporal decomposition 166
4.14 Interruption process flow diagram 167
4.15 Model of visual semantic object binding 168
4.16 Existent feature to description binding points 169
4.17 Description object creation cycle 170

5.1 ASIC core for MPEG-2 decoding 196
5.2 ASIC core for MPEG-2 decoding and feature extraction 197

B.1 Abstract event description 207

D.1 213
D.2 213
D.3 213
D.4 213
D.5 214
D.6 214
D.7 214
D.8 214

D.9 214
D.10 214
D.11 214
D.12 214
D.13 215
D.14 215
D.15 215
D.16 215
D.17 215
D.18 215
D.19 215
D.20 215
D.21 216
D.22 216
D.23 216
D.24 216
D.25 216
D.26 216
D.27 216
D.28 216

List of Tables

2.1 Substance and form in expression and content 32
2.2 Substance and Form of Narrative and Story 33
2.3 Comparison of story terminology 36

3.1 MPEG-1 Visual Layers Syntax 112
3.2 Valid slice types in MPEG 4 pictures 117

Chapter 1

Introduction

1.1 Video annotation: an enabling technology

Video annotation is the process of associating extra information with video sequences. Annotation is carried out with the goal of achieving comparable capabilities in machine search, retrieval, management and understanding of video content to those achievable with text. Video annotation is needed because these tasks have until recently required humans to accomplish them, automation is needed as manual annotation is also a time consuming and potentially expensive task. Annotation provides mechanisms to improve retrieval, search, storage, and workflow operations. These improvements benefit video content creation, manipulation and archiving applications irrespective of the underlying storage media.

1.2 Scope

The scope of this work can be defined by identifying four of the main themes that have emerged in video research. At its outset content description research focused on two main areas of investigation, these will be characterised as: the *top-down* approach, which derives from interest in video content semantic representation applications and the *bottom-up* approach, which derives from interest in image processing based video analysis applications. As these two domains matured interest in content description *interchange* developed to further facilitate novel applications. The work on the previous three themes was, in turn, supported by advances in video *compression* technology. These four themes are now described in outline here, to provide the basis for the definition of the goals and objectives of this research, and reviewed in more detail in the later chapters.

1. Top down knowledge representation approaches
   The top-down approach was pioneered by Bloch [35] [36] and Parkes' work [214] [213] [212] into intelligent videodisc based tutoring systems and his students Nack [201] [200] [202] and Butler who worked on automated editing and structural representation [48] [47]. The top down approach has been further developed particularly in work led by Nack with his emphasis on the application of semiotics with his collaborators Putz [204], Lindley [203] and others [218]. From this early work it became apparent that in addition to facilitating basic search and retrieval operations, that many novel applications could be developed; provided meaningful or semantic descriptions of video could be easily created, recent work continues to reinforce this view. The annotations used in the early demonstrator projects contained information about the *semantic* content of the video material and *cinematographic* information easily expressed in human readable form as opposed to any quantitative information about the *image characteristics.* Davis [189] is now (in 2004) pointing to the need for the integration of low level features with knowledge representation (k-r) structures.

2. Bottom up image feature analysis approaches
   In the bottom-up approach image processing derived features or clusters of features are extracted from image pixels and their statistical variation over time in video frames. These may then be mapped onto sets of semantic categories. In the image domain this has been driven by interest in visual object identification and retrieval and in the video domain by interest in segmenting video into scenes or shots and production effect identification. The bottom-up approach is here considered to span two domains: image object location and retrieval, and video analysis. Colour based image object identification and location was considered by Swain and Ballard [278] and

has been developed further by Smith [256] and Schiele [245]. Video analysis has received intense interest following the pioneering work of Aigrain and Joly [4] from the temporal segmentation viewpoint and Fischer *et al.* [175] from the genre identification viewpoint; more recently Li and Luo [172], and Adams [2] give detailed accounts of model based statistical approaches to automated temporal indexing. The latter two authors can be considered to be tending to a bi-directional approach in that they are mapping feature clusters onto content models.

3. Content description interchange
   As these two approaches started to mature and a more widespread interest in what became known as meta-data developed, international bodies such as ISO/IEC JTC1 SC29 better known as the Moving Picture Experts Group (MPEG) took an interest in content description interchange in 1996 and the MPEG-7 standardisation activity took shape resulting in the MPEG-7 standard [167]. Standardisation work was also undertaken by professional bodies such as the European Broadcasting Union (EBU) and the Society for Motion Picture and Television Engineers (SMPTE) which resulted in the SMPTE standards [258] [259]. This work is driven by the commercial benefits to be obtained by having standardised approaches to content description interchange.

4. Compression technologies
   The three themes outlined above are necessarily underpinned by the advances in digital video compression technologies and in particular the coding and decoding (codec) technologies developed by MPEG are of particular importance. The video compression standards defined in the MPEG-1 [144] MPEG-2 [145] and MPEG-4 [141] [143] standards and complimentary audio and systems standards underpin the content distribution applications such as DVD, MP3 and digital terrestrial broadcasting that have emerged recently.

These four themes provide the context for the definition of the scope of this research since content descriptions can be created using either of the *top-down* or *bottom-up* approaches and *interchanged* in standardised format to facilitate the search, retrieval and re-use of content stored and distributed in standardised *compression* formats. However, there are problems with this vision that derive from limitations in the current state of the top-down approach and in the details and underlying assumptions of the bottom-up approach namely that:

It has been shown that novel applications can be developed using the top-down approach when rich semantic descriptions of content are available [202]. These demonstrator applications have necessarily been limited in scale since the descriptions they have used have been created completely manually. Although in some approaches such as that adopted by Solway [261] automated *text* analysis of ancillary information has been used to create instances of knowledge representation structures the issue of how such instances could be instantiated using image and video analysis has received little attention. In the *image* object location and retrieval domain of the bottom-up analysis theme the emphasis has been on similarity-based retrieval from larger image databases that have typically not had semantic information associated with them. In the temporal segmentation domain the main goal has been segmentation of digital *video* into shots or the identification of production effects such as zooms, fades and dissolves. This work has derived from an often implicit, underlying assumption that these boundaries correspond to minimal temporal units of meaning. This overlooks much about the way that meaning is conveyed in film and video content that is considered in the later analysis considering narrative and film theoretic issues. The main goal of this work derives from the limitations that have been identified with the top-down approach and bottom-up approach: namely to investigate how top-down knowledge representation structures can be instantiated using bottom-up image and video image-processing based techniques to create rich semantic descriptions of video content.

Participants in the content description interchange standardisation activities also recognised the distinction between top-down approaches and bottom-up approaches and they have gone on to create a large number of descriptive terms to describe content. These efforts have deliberately set out not to address how such descriptions would be created, as MPEG in particular has traditionally considered encoders to be out of scope and concerned itself with interchange formats and decoder compliance. In the context of this research these approaches represent the *external* representation of description or representation whereas this work is primarily concerned with an application that can create an *internal* description or representation of the content. These efforts to develop content description interchange formats largely proceeded in parallel with this work, and the development of MPEG-7 both informed and was informed by this author's contributions to the MPEG-7 process, see appendix A and [123]. Thus the approach adopted here is contrasted with that of MPEG in later chapters and the internal representation is shown to be conceptually capable of representation in terms of these formats.

During the course of this research, digital video has emerged from being a laboratory novelty to become a mainstream method for content distribution, transmission and storage. All of the major non-proprietary and most of the proprietary digital video standards available at the outset of this work and those developed whilst this work was in progress, rely on the Discrete Cosine Transform (DCT) [228] which makes a major contribution to their compression efficiency. Preliminary investigations into the work of Aghagolzadeh and Ersoyon on transform domain edge detection [3], led to the recognition that, quantitative image features useful in creating content descriptions, could be extracted more efficiently from *partially* decoded MPEG streams, than by using methods where images are completely reconstructed from compressed streams and related transforms then applied to accomplish feature extraction. As a result, DCT based compression technologies and research into DCT transform based feature extraction form part of the scope of this work. DCT compression technologies and feature extraction techniques therefore form part of the bottom-up image feature extraction aspects of the main goal of this work and are described in more detail in later chapters.

1.3 Goals, objectives and claims

Identifying the scope of this research through the description of the four video research themes given above enabled the primary goal of this research to be identified. The primary goal derives from the view that knowledge representation techniques could be combined with quantitative image processing technologies to give a bi-directional approach to creating semantic annotations of video. The primary goal of this research into automating video annotation (AVA) is therefore to identify a bi-directional bottom-up, top-down method combining both the computer vision approach *and* the semantic knowledge based approach. In the bi-directional approach instances of knowledge representation structures are created in response to quantitative analysis of image and video. From the primary goal further subsidiary goals have been derived which are now identified.

1.3.1 Research goals

1. The primary goal is investigation into and development of a methodology for automating video annotation using a bi-directional approach. This methodology is referred to as the AVA methodology.

2. The development of a coherent content model to represent time-based visual media from the narrative and event levels to visual objects, their attributes and relationships. The model should provide the basis for creating content description instances through use of the software architecture implementation. This is referred to as the AVA content model.

3. The design and implementation of a software architecture to support the methodology identified in goal 1 that implements the model identified in goal 2. This architecture is referred to as the AVA architecture.

4. The development of a user interface for the implementation of the architecture identified in goal 3 . This goal and the AVA architecture goal complete the definition of the prototype AVA application.

5. Identification of new and the refinement of existing, knowledge representation structures for video content representation to form the top-down knowledge representation structures for the AVA bi-directional approach. This contributes to the development of the upper layers of the model in goal 2. These are referred to as the AVA high-level model components.

6. Identification of techniques for image object identification and location, video temporal segmentation and feature extraction in the continuous and DCT

transform domains. These techniques provide the bottom-up image and video analysis portions of the AVA bi-directional approach and contribute to the development of the lower layers of the model in goal 2. These are referred to as the AVA low-level feature extraction techniques.

1.3.2 Research objectives

The primary goal of this research; investigation into and development of the AVA video content description methodology has already been described. It can be considered that the remaining secondary goals of this research are in fact the main objectives through which the primary goal is achieved. Alternatively, each of the secondary goals could be and often is, treated as a separate research goal in its own right. There is however much to be gained by considering the primary and secondary goals as a whole and proceeding to identify the objectives through which they can be realised. In addition, the 6 goals contribute to the requirements analysis, the partitioning of the rest of the research and the synthesis of the results. The objectives of this research related to each of the goals are now described. Following this the requirements that derive from the objectives are outlined.

1. AVA methodology.
   The AVA methodology can be investigated and identified by:
   - 1.1 Identifying the capabilities and limitations of existing high level content description approaches.
   - 1.2 Identifying the capabilities and limitations of existing low level content description approaches.
   - 1.3 Achieving a synthesis of the capabilities of the low and high level techniques to overcome the limitations of each.
   - 1.4 The objective immediately above is achieved in turn through the following sets of objectives related to the secondary goals of the research.
2. AVA content model.
   To achieve the synthesis of the capabilities of existing high and low level

approaches the following objectives related to the AVA content model need to be realised.

2.1 Review existing approaches to knowledge representation (k-r).

2.2 Identify k-r approaches relevant to video content.

2.3 Review existing approaches to video content representation.

2.4 Identify and investigate relevant background material from associated domains to inform the development of the content model including but not limited to the following:

2.4.1 Semiotics.

2.4.2 Cinematography.

2.4.3 Narrative Theory.

2.4.4 Film Theory.

2.4.5 Cognitive Psychology.

2.5 Develop a terminology for describing the components of the AVA content model.

2.6 Review existing approaches to image-processing based video and image content analysis.

2.7 Define a temporal hierarchy of k-r schema capable of expressing the decomposition of video content in a coherent way from:

2.7.1 content item

to

2.7.2 pixel.

2.8 Define techniques for video and image analysis.

2.9 Identify a method for dynamically combining the results of the previous 2 objectives.

2.10 Identify a methodology for temporal relationship representation.

2.11 Identify a methodology for object attribute relationship representation.

2.12 Identify a methodology for representing events in video content.

2.13 Identify a representation methodology that distinguishes between objects forming part of a visual scene and objects participating in events.

3. AVA architecture objectives.

   The AVA architecture should provide capabilities to support the following objectives:

   3.1 Investigation into the AVA methodology.

   3.2 Implementation of the AVA methodology.

   3.3 Investigation into implementation approaches for the AVA content model.

   3.4 The AVA high-level model knowledge representation objectives identified in objective 5.

   3.5 The AVA low-level image video analysis objectives identified in section 6 of the objectives.

   The architecture must also provide support for:

   3.6 Image compression.

   3.7 Image decompression.

   3.8 Video decompression.

   3.9 Video manipulation.

4. AVA application interface.

   The AVA application interface needs to support the investigations into the following other objectives of the research.

   4.1 The AVA annotation methodology.

   4.2 The AVA content model.

   4.3 High level content representation.

   4.4 Low level image and video analysis.

   In addition the interface should provide user interaction with and feedback on:

   4.5 Video playback.

  4.6 Image rendering.

  4.7 Knowledge representation schema storage.

  4.8 Knowledge representation schema instances.

  4.9 Image analysis.

  4.10 Video analysis.

5. AVA high-level model components objectives.

   The goal of integrating high level knowledge representation techniques with low level feature extraction techniques depends on research to investigate and identify mechanisms to support the following capabilities:

  5.1 Schema creation.

  5.2 Schema extension.

  5.3 Entity instance creation.

  5.4 Entity relationship definition.

  5.5 Entity relationship extension.

6. AVA low-level feature extraction techniques objectives.

   The goal of integrating low level feature extraction techniques with high level knowledge representation techniques depends on investigation into and identification of the following techniques.

  6.1 Visual object recognition in continuous domain.

  6.2 Visual object recognition in the DCT transform domain.

  6.3 Visual object location in the continuous domain.

  6.4 Visual object location in the DCT transform domain.

  6.5 Temporal segmentation in the continuous domain.

  6.6 Temporal segmentation in the DCT transform domain.

1.3.3 Requirements

The importance of the requirements process in software engineering has been emphasised by many authors notably Sommerville [262] and Pressman [224]. The requirements for this research overlap substantially with those of MPEG-7 so reference to the current version of that document [220] which is readily available is necessary. Members of MPEG and in particular, the MPEG-7 requirements ad-hoc group which included the present author as contributor, and at times as editor, expended considerable effort identifying the requirements for multimedia content description during the development of the standard. The MPEG-7 requirements have been updated on a continuing basis from the versions containing the contributions of this author outlined in appendix A of which Hartley *et al.* and MPEG input documents M2634 and M4879 are of particular relevance here. These input documents and other contributions of the present author are still evident in the MPEG-7 requirements document, even now at version 18 of the MPEG requirements document [220]. The model which provides much of the conceptual underpinning of MPEG-7, illustrated in it's figure 1 is taken from one of these original input contributions. The content model adopted by MPEG-7 from input document M2634 by MPEG-7 has been developed further in the course of this work into the AVA content model as described in full in chapter 2. The MPEG-7 requirements document was in turn developed through a detailed analysis of potential applications for multimedia content description to which this author also contributed. This analysis is in keeping with requirements best practice recommended by Sommerville and Sawyer [263] as at the time of the initial requirements no large scale bodies of accessible multimedia content related meta-data existed with the exception of archive catalogues in repositories such as INA [100] in France and at the British Broadcasting Corporation (BBC). Hence user studies of annotation systems were impractical as no systems existed, although user studies of media manipulation activities have been carried out by Nack [200] and latterly by Nack and Putz [204]. Space does not permit a complete critique of the MPEG-7 requirements document but it should be noted that MPEG-7 requirements are derived from considerations of content description *interchange* and not with the requirements for content description *creation*. The requirements outlined here derive from the content description *creation* viewpoint in keeping with the goals of the research. The difference that the different viewpoint primarily brings into focus is that the description is a dynamic entity until the description of a piece of content has been completed. In contrast, the MPEG-7 consensus viewpoint is that descriptions are static entities subject to relatively infrequent up-date. It should be understood therefore that the requirements developed here were substantially developed through participation in the MPEG-7 requirements process. This provided a substantial insight into the large number of potential application classes for content description identified by MPEG [142] provided that descriptions could be effectively created. So these requirements differ from those of MPEG-7 because their development has been moderated by the viewpoint of the description creation task. The requirements are structured to reflect the different goals and objectives of the AVA research and are now described in outline.

1. Annotation methodology requirements.
   A methodology for video content description should be developed that combines the following:
   - 1.1 High-level knowledge representation based decomposition of moving picture content.
   - 1.2 Low-level image processing based analysis of moving picture content.
2. Content model requirements.
   A model for video content should be developed capable of representing entities from the following content levels.
   - 2.1 The narrative level.
   - 2.2 The event level.
   - 2.3 The visual object level.
     The model should also support description of the following:
   - 2.4 Spatial relationships between entities in the model.
   - 2.5 Temporal relationships between entities in the model.

2.6 Attributes of entities in the model.

2.7 Typing of entities in the model (i.e. is-a relationships).

2.8 Compositional relationships between entities in the model (i.e. is part-of relationships).
These requirements derive from goal 2 and its associated objective 2. Instantiating these entities should be capable of being carried out by utilising techniques from the image-processing based analysis of images and video. So the model should also contain the following:

2.9 Video temporal feature references.

2.10 Video temporal feature cluster references.

2.11 Image features.

2.12 Image feature clusters.

These requirements derive from the AVA model goal 2, and are met through both goal 5 and associated set of objectives 5 and the low level goal 6 and associated objectives 6.

3. Software architecture requirements.
A software architecture should be developed capable of supporting:

3.1 Investigation into the bi-directional methodology for video annotation from requirement 1 above.
The architecture should also support the integration of the following capabilities defining the entities and relationships identified in requirement 2 for the AVA content model above.

3.2 Knowledge representation schema.
The knowledge representation schema should support the following capabilities:

3.2.1 Provision of generic schema elements for video content representation.

3.2.2 Storage of the knowledge representation schema.

   - 3.2.3 Extension of the knowledge representation schema for specialised content domains.
   - 3.2.4 Creation of generic schema entity instances.
   - 3.2.5 Creation of specialised schema entity instances for specialised content domains.
  - 3.3 Moving picture decoding, playback and display.
  - 3.4 Image processing based analysis of moving picture content.
  - 3.5 Storage of numeric features representing moving picture characteristics.
  - 3.6 Image processing based analysis of still picture content.
  - 3.7 Storage of numeric features representing still picture characteristics.
  - 3.8 Provide a user interface for interaction with content and descriptions.

4. User interface requirements.
  - 4.1 Provide video playback with capabilities and controls for:
    - 4.1.1 normal speed forward playback,
    - 4.1.2 variable speed forward playback,
    - 4.1.3 variable speed backward playback,
    - 4.1.4 single frame stepped playback.
  - 4.2 Provide frame reference display.
  - 4.3 Provide video still image display.
  - 4.4 Provide capabilities for video still image interaction.
  - 4.5 Provide knowledge representation (k-r) schema display and exploration.
  - 4.6 Provide k-r schema instance display and exploration.

5. Requirements for the semantic representation of video content.
   Develop a knowledge representation hierarchy for the semantic representation of moving picture content decomposition. This decomposition hierarchy will be known as the high level hierarchy.
  - 5.1 The high-level hierarchy should support "objective" descriptions.

5.2 The high-level hierarchy should support "subjective" descriptions.

5.3 The high-level hierarchy should support the decomposition video content according to the requirement 2 of the AVA model.

5.4 The high-level decomposition hierarchy should support decomposition of video material into semantically meaningful entities.

5.5 The high-level decomposition hierarchy should be capable of representing objects and their relationships.

5.6 The high-level hierarchy should be capable of representing temporal relationships between entities.

5.7 The high-level hierarchy should support the decomposition of moving picture content into events.

5.8 The high-level hierarchy should support the decomposition of moving pictures into semantically meaningful visual objects.

6. Requirements for image processing based analysis of moving picture content.

6.1 Develop a low-level hierarchy of image processing based techniques for the decomposition of moving picture content.
This hierarchy should be capable of temporal decomposition of video and spatial decomposition of images. This leads to the following requirements:

6.2 The temporal segmentation of moving picture content should be supported.

6.3 Visual object identification in moving picture content should be supported.

6.4 Visual object location in moving picture content should be supported.

1.3.4 Claims

These are challenging objectives and, whilst this work does not claim to represent a complete solution to the problem, it does claim to provide the following: an overview of the relevant background technologies, a survey of previous and known current approaches. In particular it describes the following: the Automating Video Annotation (AVA) content model for moving picture content description, the AVA approach to automating video annotation, the AVA software architecture and AVA demonstrator system. The latter is an implementation of the architecture incorporating the model to illustrate the feasibility of the approach and architecture.

The AVA model of video content defines a comprehensive and coherent hierarchical spatio-temporal decomposition of video from narrative to image segment. The AVA annotation approach is to support the video annotator or *description author* with software tools combining a bi-directional combination of knowledge and video representation, processing, search and retrieval techniques. In the approach image feature extraction, search, and video analysis are combined with a knowledge base containing generic *a-priori* domain knowledge, instance specific *a-priori* knowledge and instance knowledge obtained from the low-level image features.

The AVA architecture defines the interrelationship between the components of the system. These components include: user support via a user interface incorporating several views on the video, including still images from the video, the *a-priori* knowledge bases, and the annotations.

The architecture relies on a number of technologies, including video and image codecs, extensible knowledge representation facilities, object factories, an object database, low-level feature extraction and image processing capabilities, together with an image database. On the basis of these developments the thesis asserts that automated tools supported by some limited human intervention can be used to provide *meaningful* descriptions of digital video content.

The approach has been applied to completed content; since access to a production environment has not been available, this is content that has completed the production cycle. The approach however, can be usefully applied to content anywhere in the media content creation cycle particularly during production. The media production cycle is outlined in figure 1.1

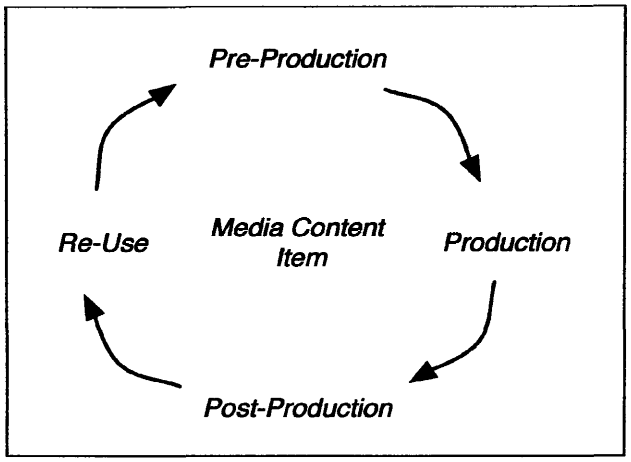


Figure 1.1: Media creation cycle outline

1.4 Approach

1.4.1 Bi-directional aspects

The developments outlined in section 1 are considered to be top-down in their approach to content representation, since they derive from consideration of the perceived content of sequences and build their knowledge based structures through a top-down decomposition of the content. Consideration of MPEG coding technology[144] [145], the Discrete Cosine Transform (DCT) [228] and work on transform domain edge detection [3] led to the view that, quantitative image features could be extracted efficiently from *partially* decoded MPEG streams. These approaches are considered in this context to be bottom-up deriving from consideration of pixels, frames and how their statistics vary over time. In turn this led to the view that knowledge representation techniques could be combined with quantitative image processing technologies to give a bi-directional approach to creating semantic annotations of video. The AVA approach is therefore a bi-directional bottom-up, top-down combination of both the computer vision approaches *and* the semantic knowledge based approaches. This does not mean to say that rigid adherence to forward-chaining or backward-chaining based reasoning should be adhered to when applied to either of these domains.

1.4.2 Software design

Throughout the development process use has been made of the Unified Modelling Language (UML) [235] from the development of the Extended Semiotic Model 2.22 through to the analysis of the object instantiation cycle presented in section 4.17. The approach of using UML representation was presented to the MPEG-7 community along with the model in M2634 and led to the decision that the MPEG-7 technology proposals should use UML as the recommended technique for defining the descriptors for evaluation. During the AVA research a large number of technologies were considered and there has been reliance on languages and libraries not directly supported by the UML development environments. Therefore the UML approach has provided a conceptual background to the analysis of specific areas of the work under consideration. This is in contrast to using UML to provide a complete iterative "round trip engineering" solution advocated by proponents of the UML such as Eriksson and Penker [86] and used to describe the cycle of modelling, code development, reverse engineering and further modelling. Latterly elements of the Patterns methodologies [101] [102] have been adopted where appropriate particularly with respect to the use of the Factory Creational Pattern and its relation to the knowledge base and associated classes discussed in detail in section 4.12.

1.4.3 Software implementation

The basic methodological approach adopted has been exploratory programming based on a consideration of the goals, objectives and requirements described above. The need for an end user description creation tool was seen to be of equal importance to the development of any of the other capabilities. To this end a number of approaches were initially considered which were subsequently rejected. A detailed discussion of the approaches investigated and the rationale for their investigation is given in section 5.2.7. In developing a project of this nature it was judged that as far as possible existing libraries or packages should be used to implement the core components of the system with a close coupling between the controller and the user interface. This leads to the view that an implementation language capable of supporting rapid prototyping and object technologies with a good capability for interfacing to existing C [154] or C++ [276] libraries is needed or one with extant powerful numerical capability. One such language is Python [24] and this was ultimately adopted as the implementation language. This has turned out to be a successful approach.

1.5 Structural and content overview

From the description of the research goals and the analysis of objectives and requirements, it will be seen that realisation of the primary goal is heavily dependent on the subsidiary goals; also there are multiple dependencies between the subsidiary goals. These interrelationships are best understood if they are considered from the perspective of knowledge representation structure instances being created in response to image processing based analysis of the video content. This requires a definition of the knowledge representation structures to exist before they can be instantiated. So the goals are treated in the following order:

1. Video content modelling research,
2. Knowledge representation structure description, and definition research,
3. Video and image feature extraction analysis and research,
4. Software architecture and user interface definition research and implementation,
5. Video annotation methodology research and definition.

It will also be appreciated from the discussions above that this research has a broad multidisciplinary scope. In consequence, the structure of this thesis does not follow the normal pattern of a single review chapter followed by a description of the contributions made by the author. In contrast, each chapter contains both review material and author contributions. The novel material contributed by the author is clearly identified. A short synopsis of each of the succeeding chapters is now given.

1.5.1 Chapter 2 Video theory and content modelling

This chapter addresses the high level modelling objectives of the research and contains an analysis of theoretical background material from several domains. These are semiotics, narrative, and film theories, artificial intelligence and previous work on content modelling. The role each of these contributed to the development of the AVA content model is first outlined and then the material is analysed in detail. The analysis begins with a review of the semiotic distinction between expression and content in the case of text. A review of approaches to narrative and film theory including narrative grammars and structural approaches to cinema theory is then carried out. This analysis supports the development of the content representation structures for top-down video content decomposition that form the upper parts of the content model developed later in the chapter. Related previous work on video content representation that has contributed to the development of the AVA content model in section 2.6 is described. Investigations into the requirements for representing: colour, texture, edges and objects are described. A novel analysis of the distinctions between the differing temporal dimensions in the content cycle are described and these in turn inform the development of the user interface. Novel revisions and extensions to the semiotic model to enable it's application to continuous and compressed domain multimedia content are described in sections 2.8.1 and 2.8.3. These extensions to the model distinguish between expression, data and content planes in the continuous and compressed domains. Further novel revisions to the model are then developed in section 2.8.4. These revisions introduce image features and the concept of feature to denotative description binding to the model. Consideration of these semiotic models plays a significant role in the development and description of the AVA content model. A critique of Parkes' [213] CLORIS content model to determine the feasibility of adopting it for an automated annotation system, in part by recasting it in UML is carried out. This leads to a reappraisal of Parkes' *op.cit.* setting construct as the bridge between content expression and description.

The reappraisal results in the development of several new constructs that are combined to form the AVA moving picture content model. At the level of objects in images the new constructs introduced in section 2.8.6 are the existent description which is decomposed into denotative and connotative descriptions and feature sets. It is shown that existent descriptions can be used to model object relations. The new mechanism of "binding" is introduced in section 2.9.1 to overtly associate one or more temporally separated image regions and associated features with a descriptive category. Another new construct, the existent context is introduced in section 2.9.3 and defined as the temporal extent over which an object in a descriptive category can be recognised by a given image object recognition algorithm. A further new construct, the existent context set is developed in section 2.9.5 and is defined as a number of frames in which a number of existents are expressed. The setting is then replaced in section 2.9.7 with another new construct, the scene description that bridges the expression, data and description planes derived from the extensions to the semiotic model. The scene description references existent context sets and spatio-temporally locates class definitions corresponding to Schank's [243] scripts which in turn decompose into relations from Schank's [238] conceptual dependency (CD) modelling. The introduction of Schank's scripts is based on an analysis of Schank's CD in section 2.10. In turn the conceptual relations also reference the existents and existent context sets.

1.5.2 Chapter 3 spatio-temporal segmentation.

This chapter meets the image and video analysis objectives in the continuous and transform domain of the AVA research. This is done by describing the reliance of still and moving image compression techniques on the discrete cosine transform (DCT). The role of the DCT is described with particular reference to the moving picture experts group's (MPEG) suite of standards. An outline of the encoding schemes used in this suite of compression standards is given. Methods for visual object identification and location based on colour and spatial frequency histogram, comparison and back projection in the continuous domain are described. Methods for the temporal segmentation of moving picture sequences in the continuous and transform domains are then reviewed. This provides a context for the description of a novel series of modifications made to an MPEG-1 decoder in section 3.6. These modifications facilitate the DCT domain, colour and spatial frequency feature extraction and edge detection. techniques developed during the AVA research. A novel computationally efficient method for visual object edge detection in the DCT domain utilising an approximation to the Laplacian of the Gaussian is described in section 3.7.2. A novel approach to extending the visual object identification and location methods into the DCT domain using a colour transform and simple masks is then described in section 3.7.3. This uses a methodology similar to the edge detector. It is shown how these techniques can together be combined with the modifications to the video decoder described, to provide a feature stream and edge image stream temporally parallel with the decoded image stream. Novel extensions of the transform domain image segmentation approach to facilitate temporal segmentation are also considered in section 3.7.9.

1.5.3 Chapter 4 AVA architecture and methodology.

Chapter 4 describes the current implementation of the novel AVA architecture for semi-automated video annotation in terms of UML class and process diagrams. The chapter shows how the individual interface components that make up the novel user interface of the architecture are combined. These provide the content describer with: a fully featured video player, a rudimentary still image drawing tool, views on the *a-priori* knowledge bases, the instantiated description elements and views on the temporal image features. It describes how the *a-priori* knowledge base exploits a novel use of object factories and multiple inheritance to provide user interaction with the description and persistent storage. A novel approach to the persistent storage of the instantiated content descriptions is also described based on the utilisation of an object orientated database. The numerical capabilities that the system is dependent on, and how these are combined with still and moving image codecs to provide still and moving picture image decompression and still image compression are also described. This chapter also outlines how the methodology of using automated wrapper technologies around existing libraries is exploited. The novel AVA annotation methodology is also described and a more formal definition of the algorithm provided. This methodology allows automated or user defined interruption points to be identified in the video stream, at these, instances of existent description classes are bound to image feature sets. A worked example of the application of the methodology to the narrative part of a children's television program is provided. This shows that within a simple restricted domain the binding of visual image features with descriptive class definitions can be achieved and annotations created automatically.

1.5.4 Chapter 5 evaluation future work and conclusions

This chapter carries out a qualitative analysis of the research results against the goals, objectives and requirements described above. A quantitative evaluation program for the transform domain image object recognition and location approaches is defined. Extensions to the AVA implementation demonstrator are proposed and several avenues for further research are discussed. Support for the psychological plausibility of the model is then sought from a review of functional nuclear magnetic resonance imaging (fNMRI) results.

1.5.5 Appendices

1.5.5.1 A: MPEG input documents

Provides a list of input documents to MPEG to illustrate the authors contribution to the development of the MPEG-7 requirements, applications and systems parts of the standard.

1.5.5.2 B: The formal CLORIS model

The formal definition of the CLORIS content model has been extracted from Parkes' thesis [213] and provided the basis for the UML representation of the model used in the critique in chapter 2.

1.5.5.3 C: Story grammar analysis

A more extended analysis of the story grammar background material is provided in this appendix.

1.5.5.4 D: teletubbies representative frames

Shows a number of representative frames from part of a children's television program used in the worked example in chapter 4.

1.5.5.5 E: Python Source code listings.

Python source code listings for the basic video player which provides the basis for the video decoding capability of the AVA implementation and a rudimentary driver program for the MPEG decoder is also provided. The Python semantic and visual factory classes described in chapter 4 that provide the basis for the rejection of this approach are also provided.

1.5.5.6 F: Interface and C source code listings

The SWIG interface file for the C++ MPEG decoder library is provided together with the helper function definitions, the SWIG generated Python interface file and a portion of the corresponding SWIG generated C wrapper file.

1.5.5.7 G: Matlab source code

The Matlab source code for the DCT based filtering and masking developed for the simulation work discussed in chapter 3 is provided.

Chapter 2

Video theory and content modelling

2.1 Introduction

Defining the AVA model for moving picture content representation is the research goal pursued in this chapter. Moving picture content model development has previously relied on concepts from a broad range of sources, though few if any have taken the full range of considerations presented here into account. The development of the AVA content model is based on a theoretical position derived from analysis of semiotic, narrative and film theories, artificial intelligence, a review of previous work and an analysis of representation concerns. Whilst there is considerable reliance on non computational theory in this chapter, the underlying task being addressed throughout is the development of a *computational* decomposition hierarchy for moving picture content representation. The non computational material therefore provides a theoretical position from which to develop the AVA content model. The contribution each domain makes to the repertoire of analytical approaches used in the development of the model is now shown; this is then followed by the development of the content model.

Semiotics provides a methodology for the analysis of sign systems. Moving picture or video content is taken to form a sign system. So adopting the analytical approach provided by semiotics enables the development of a systematic framework to identify and discriminate concerns in moving picture content description.

Narrative theory and its application to film content, provides terminology and an analytical approach that supports the temporal decomposition of moving picture content. This is needed because the way in which meaning is conveyed in moving pictures is inherently time based and implicitly contains statements about actors, objects and events in scenes.

Film theory provides insights into the way moving picture image sequences are constructed to convey meaning to the viewer and how the viewer infers meaning from moving picture sequences.

Artificial intelligence provides the techniques and tools to represent and structure, knowledge about content necessary for its description in machine-readable representations for processing and reasoning.

Previous work on moving picture content representation is also provided to place the AVA developments in context, due to the role the previous work has played in shaping the AVA work. AVA also seeks to provide tools for description creation capable of being used in successor projects to some of those described.

Representation concerns are described to support positions adopted during the development of the model.

The relevant parts of each of these areas of research is in turn described along with the role it plays in the development of the content model and then the AVA content model is developed.

2.2 Semiotics

2.2.1 Development of Semiotics

In the late 19th and early 20th centuries, Pierce [125] in the USA and Saussure [237] in France systematised the study of signs and sign systems, named semiotics by the former and semiology by the latter. Of the two, Pierces term *semiotics* has now achieved more widespread acceptance. These studies were taken up again in the mid 20th century by Hjelmslev [128] and Eco [80] who developed the subject further, while Barthes [20] is also notable for his semiotic consideration of still

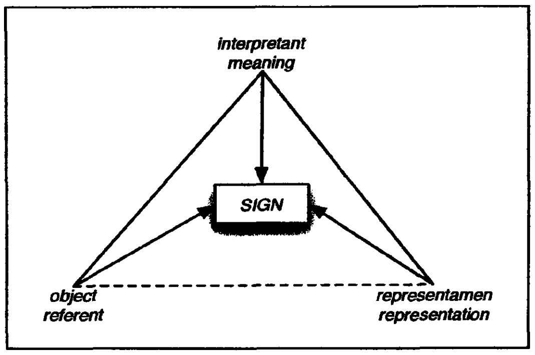


Figure 2.1: Tripartite sign structure after Pierce [125]

images and Metz [194] played a leading role in developing ideas of film as language, based on semiotic analysis. Eco in particular is of interest because he considers the information theoretic implications in his work and applies semiotic analysis to moving picture content [79] along with Metz [195] and Passolini [271] considered later. Semiotics has been used in moving picture content modelling by Parkes [213] and Nack [200] described in section 2.6. There is also growing interest in the application of semiotics in other branches of computing such as de Souza's application of semiotics to Human Computer Interfaces (HCI) [69]. Over the last 50 years semiotics as the study of signs and sign systems has developed from linguistics into a discipline in its own right. In doing so it has evolved a specialised language to distinguish, not only in the case of text, what is present on the page (expressed), the concepts conveyed by the text (denoted), and the concepts invoked in the reader's mind (denotated), but also distinguishes these for film. Of the two approaches identified above, that of Pierce is useful because of the tripartite treatment of the sign that he provides, as shown in figure 2.1.

2.2.2 Types of sign

In Pierce's analysis a sign stands *for* something *to* the idea which it produces or modifies [125]; this is valuable because he introduces a distinction between three types of sign namely:

Iconic, defined by Pierce as a a sign determined by its dynamic object by virtue of its own internal nature. In this case the sign conveys the meaning by providing a direct representation of the object as for example in a pictogram.

Indexical, defined by Pierce as a sign determined by its dynamic object by virtue of being in a real relation to it. In this case, there is a causal link between the sign and the meaning the sign conveys as in a thermometer or smoke signifying fire.

Symbolic signs represent their objects solely by linguistic convention. This is the type of sign system most normally encountered in natural language.

This distinction will be returned to in the analysis of film theory later in the chapter.

2.2.3 Arbitrariness of signs

Of particular interest to Saussure is the arbitrariness of signs and in particular textual marks on the page and the meanings they convey. He took the relationship between the signifier; that is the acoustic, visual or other signal used to invoke the meaning and the signified; the mental concept invoked by it to be of central importance in this relationship. This led to analysis of the minimal units of meaning in text; identified as *morphemes* which are turn decomposed into sub-entities; the *phonemes* devoid of meaning. This decomposition of sign systems into minimal units of meaning in turn decomposed into meaningless sub-units is referred to as double articulation. In Saussure's view it is the arbitrariness of the relationship between the signs and the ideas they represent that give rise to the flexibility and versatility of language. The issue of what might correspond to a set of minimal units of representation in film and moving pictures will be returned to later. Eco explores the constituents of iconic signs and shows that the general arbitrary nature of signs extends to iconic signs irrespective of their aspects of direct representation. He develops the view that an iconic sign is a seme which as a recognisable unit of meaning that

> "...doesn't correspond to a word in the verbal language but is still an utterance. The image of a horse does not mean horse but as a minimum 'a white horse stands here in profile' " Eco ([79] p.597).

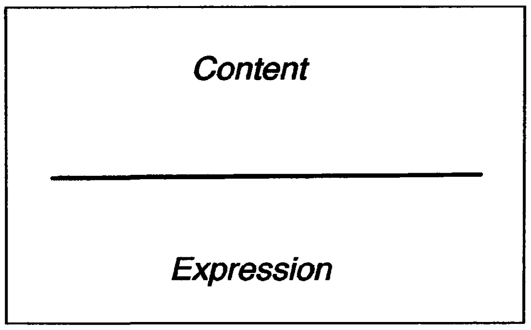


Figure 2.2: Content and expression after Eco [80] p.51

2.2.4 Content and expression

In the case of the textual analysis; Saussure [237], Pierce [125] and Eco [80] identify the marks on the page and the ideas invoked in the reader's mind as different entities. Eco [80] discusses Helmslev's [128] ideas that a sign function can be considered to be separated into it's expression and it's content i.e. the signifier and the signified, where content is the set of ideas conveyed by the expression of a series of signs through the constituents of a given medium. In the case of text, the medium is the marks on the page and the ideas immediately invoked or denoted by the words are the same as those printed. However, in the case of moving pictures, expression is the play of patterns of light projected or rendered on a screen from the frames of a film or video and the ideas invoked in the viewers mind form the content. This is illustrated in figure 2.2.4, where the diagram Eco [80] is intended to convey the view that the content and its expression occupy different planes. It is also interesting to note that this distinction can be traced at least as far back as Bishop Berkeley [177].

2.2.5 Denotation and connotation

Eco also emphasises the distinction between the denotation of a sign system and its constituent signs and their connotation. These two terms are used to distinguish the types of meaning invoked when signs are expressed. The meaning *denoted* by the expression of a sign is the idea that it immediately invokes and the meaning *connoted* by a sign is a meaning that can be inferred from the signs expression through the mental experience of the reader or viewer, which may be emotionally induced.

2.2.6 Paradigmatic and syntagmatic axes

Sausurre also developed the contrast between the syntagmatic axis of a sign system and its paradigmatic axes. These axes are typically represented diagrammatically as two orthogonal axes in a plane.

> "The identity of any linguistic sign is determined by the sum of paradigmatic and syntagmatic relations into which it enters with other linguistic signs in the same language system" (Stam *et al.* [272] p.9).

Typically the horizontal axis is the syntagmatic representing the expression of the signs (note the implicit linearity in this representation) and the vertical axes the paradigmatic, which represents the other signs that are compared and contrasted with the actual sign expressed.

2.2.7 Application of semiotics to still and moving pictures

The terminology introduced here provides part of the conceptual framework for the remaining theoretical sections on narrative and film and the development of the AVA content model. Still images are considered by Barthes [19] to denote and connote aspects of objects in scenes. In turn from a semiotic perspective, a still image may or may not convey an event. The task of identifying how meaning is conveyed in moving picture content is postponed to the film theory section. It is now assumed that moving pictures convey meaning through a sign system representing in some way entities in the real world; such as actors, objects in scenes and these are engaged in events.

2.3 Narrative theory

It follows from Barthe's view that still images represent objects in scenes, and that since moving pictures are made up of sequences of images many objects in many scenes will generally be present. The elements used in storytelling are typically objects in scenes engaged in events. In narrative theory the relationship between objects and actors in scenes engaged in events is developed. Applying narrative theory to moving pictures therefore provides further terminology and an analytical approach supporting the temporal decomposition of moving picture content. Analysis of these interrelationships has had a long history beginning with Aristotle's Poetics [15] in the 5th century BC which is critiqued further by Dolezel [75]. Since then several important authors have developed literary theories, and aspects of these are discussed briefly from a terminological perspective in section 2.3.4. Notably Lyons [182], Helmslev [128], Chatman[54], and Barthes [20] contribute to this analysis. Chatman takes the view that Aristotle opens as many questions as he answers and goes on to apply a semiotic perspective to narrative analysis which is considered in detail here. Also of interest is the work developed from Propp's analysis of folktales [225] resulting in the development of story grammar theories which are considered later and in Appendix C. As with the preceding discussions about the relationships between the expression and content, the distinction between what is directly observable and what the observer infers i.e. what is denoted and what is connoted by a story element will be retained.

2.3.1 Semiotics of narrative and film

Chatman *op.cit.* as noted by Nack [200] takes up Barthes' [20] analysis of cinema and narrative from a semiotic perspective. Chatman attributes the cross-cutting of the distinction between expression and content with substance and form to Saussure [237] and Hjelmslev [128]. Chatman uses Lyons' [182] definitions of the form and substance of expression, where the substance of expression is the material through which the expression occurs i.e., as above, in the case of text, the marks on the page. The form of expression therefore, is the set of basic constituents through which the expression occurs. In the case of speech, this would be the phonemes. The substance of content is *all* the emotions and ideas invoked in the observer through the content. The form of content is:

"...the abstract structure of relationships ... which a particular lan-

	Expression	Content
Substance		
Form		

Table 2.1: Substance and form in expression and content after Chatman [54]

guage imposes on the same underlying substance" Lyons [182] p.55

. This interrelationship can be tabulated as in table 2.1 but further discussion is needed to cross cut the table with figure 2.3 to populate it as shown in table 2.2 to develop the distinctions shown in figure 2.4.

There are some interesting consequences of this analysis; there is an underlying assumption that a medium which is capable of communicating ideas is necessarily a language, that there are some underlying primitives that constitute the language and that these are potentially capable of further decomposition. These are points that will be returned to later in section 2.4.1.

2.3.2 Story and discourse

The distinctions between the form and substance of the expression and content of a language may at first sight appear to be almost inconsequential yet they have profound implications when considering the formal description of narrative. This will now be developed, again after Chatman. A further distinction must first be introduced; namely that between story and discourse. Where the story is the sequence of events that occur together with the actors, objects and scenes, the discourse is the means by which story is communicated. Chatman [54] shows this diagrammatically in figure 2.3. He elaborates this representation further to obtain the substance and form of content and expression in figure 2.4.

The distinctions between substance and form in expression and content can now be used to assist the development of the distinction between story and discourse.

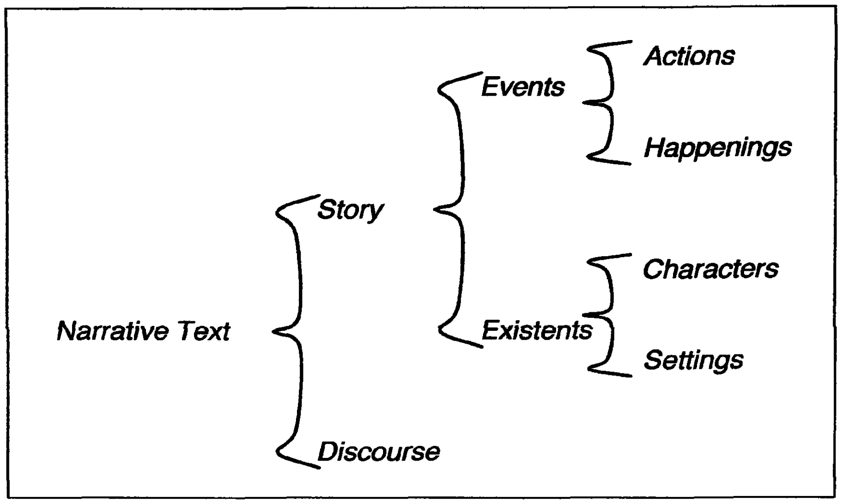


Figure 2.3: Story and discourse after Chatman [54]

	Expression	Content
Substance	Media in so far as they communicate stories. (Some media are semiotic systems in their own right.)	Representations of objects and actors in real and imagined worlds that can be imitated in a narrative medium, as filtered through the codes of the authors society.
Form	Narrative discourse (the structure of narrative transmission) consisting of elements shared by narratives in any medium whatsoever	Narrative Story components: events, existents, and their connections

Table 2.2: Substance and Form of Narrative and Story after Chatman [54]

2.3.3 Process or stasis

Discourse tells the story and relies on 2 classes of statement; process or stasis. Process statements enact or recount events and are either; acts or actions where the existent is the agent of the event, or happenings, where the existent is the patient. Chatman notes that "He stabbed himself" and mime plunging an imaginary knife into his heart; both enact the same process statement so effectively these are DO or HAPPEN statements. The other class of statements are stasis statements where IS provides the equivalent verb.

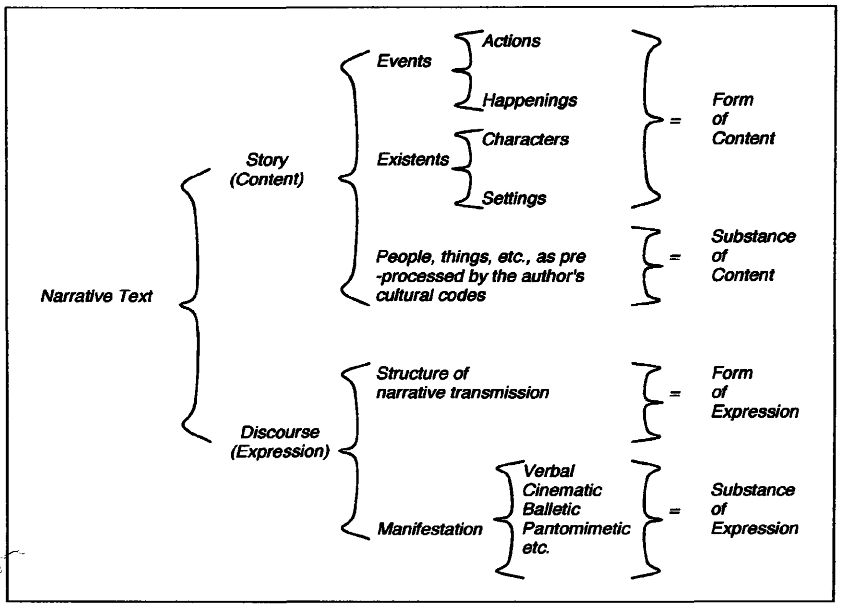


Figure 2.4: Content and expression of story and discourse after Chatman [54]

2.3.4 Story and plot

The distinction between story and plot provides terminology for the analysis of the distinction between a real or imagined sequence of events and the way these are narrated or described. The terms story and fabula are *normally* used interchangeably to denote the archetypical sequence of events realised through plot or syuzhet; the events in the order that they are portrayed. However, Hawthorne [126] observes in his glossary that the terminology used in the analysis of plot and story is confusing. Hawthorne illustrates this confusion by quoting from Onega and Landa [211], describing Robinson Crusoe by Defoe [70], where what would normally be considered to be the distinction between the story: the sequence of events *and* the plot: the order in which they are portrayed is described as,

> "... the *text* is the linguistic artefact that we can buy and read, written *de facto* by Defoe and supposedly told by Robinson. The *fabula* is whatever happened to Robinson in his travels and on his island. The *story* is the precise way in which that action is conveyed, the way the fabula is arranged into a specific cognitive structure of information ..." Onega *et al.* [211] p.6.

There are clearly further difficulties when the relationship between the account above and the *true(?)* story of Alexander Selkirk is considered. Daniel Defoe is said to have taken Selkirk's account of his life marooned on a desert island having run away to sea reputedly to escape indenture as a miner in Fife, as the basis for the novel. This serves to further emphasise some of the problems with the *story - plot: fabula - syuzhet*[1] terminology which will shortly be resolved.

From table 2.3 it could be considered that the use of any of these terms in a technical sense is in some way problematic. Fludernik [95] sees the relationship between story and discourse as one between levels rather than a planar one. This is in contrast to the view held by the current author: that story and plot are in separate space-time constructs. There is the space-time of the archetypical story or fabula and there is the discourse space-time within which the story is retold with temporal elision, retardation etc. to form the plot or syuzhet. Chatman distinguishes between the form and substance of narrative which is not accounted for in Fludernik's table. Chatman's interpretation is that discourse has the role of laying out the plot. This results in separate sets of causal relationship links; one corresponding to the archetypical story and the other to the story as portrayed. Thus, the plot is the story portrayed and there are *separate* sets of causal relationships in both of these domains if there is any story present at all. This introduces another distinction that must be considered, namely that between causal and non-causal event sequences. This causal non-causal dichotomy is taken up again in relation to Metz's [194] taxonomy reproduced in figure 2.7. To overcome any confusion in the AVA content model development the term *story* is taken to be a synonym of *fabula* and *plot* is taken to be a synonym of *syuzhet.* Subsequently story and plot will be used with their conventional meanings[2], where story is the archetypical sequence and plot the dramatic realisation in the medium of interest.

[1]syuzhet, in Russian сюжет is used in preference to plot by some writers on narrative. сюжет is defined as subject, topic, plot in Russian[7].

[2]This is not helped by the variant spellings for syuzhet found in the *English* literature i.e. syuzhet, sjuzhet, sjuzet, and sjužet have all been found as Anglicisations of the Russian.

	Events in chronological order	**Events causally connected**	**Events ordered artistically**	**Text on Page**	**Narration as enunciation**
Genette	*histoire*		*discours (recit)*		narration (voice + focalization)
Chatman	story	discourse			
Bal	*fabula*	story and focalization		narration (+ language, + voice)	
Rimmon-Kenan	story		text		narration
Prince	narrated			narrating	
Stenzel	-	story		mediation by teller or reflector	mediation by teller or reflector + enunciation if teller figure

Table 2.3: Comparison of story terminology based on Fludernik [95] p. 62

2.3.5 Representing events in scenes

Mandler [186] gives the following description of a possible event schema and usefully notes that events will naturally decompose into sequences of more primitive events.

> An event schema is a hierarchically organized set of units describing generalized knowledge about an event sequence. It includes knowledge about what will happen in a given situation and often the order in which the individual events will take place. It is organized like a categorical structure in that the knowledge is arranged in a hierarchy with more general classes of events containing more general classes of events within them. It differs from a categorical structure however, in that

> the hierarchy does not consist of class inclusion relations. A dog is an example of the class of mammals. But sitting down at table is not an *example* of eating dinner; it is *part* of eating dinner. Thus, a schematic hierarchy consists of part-whole relations, and hence is a type of collection, not a class inclusion structure. Mandler, [186] p.14.

Mandler also identifies the following aspects of scene structure that are of considerable importance to this discussion.

> As in the case of event schemas, the hierarchies organizing scene schemas are collections rather than class-inclusion hierarchies. A dining room contains walls and windows, tables and chairs, but only when these parts are put together into a particular organization does the overall scene emerge. The relations in a scene schema are spatial, not temporal. Mandler, [186] p.14.

Mandler then goes on to show the psychological *validity* if not the psychological *reality* of story grammar schema on the basis of experimental evidence showing better recall for story structured sentence groupings compared with non structured groupings. This evidence supports the validity of story constituents. The work of Haberlandt [118] [117] *et al.* is referred to in support of the psychological validity of the episodic structure. Story grammars are considered further in Appendix C.

2.3.6 Explicit and implicit scenes

The occurrence of events in some spatio-temporal location is often implied in natural language; as in sentences such as *"the man hit the ball"* quoted by Chomsky [58] (p.26). In moving pictures just as the image always depicts specific instances of classes there is always an explicit spatio-temporal location, as for example in figure 2.5. Chatman makes another useful contribution here as he distinguishes between *kernel* events i.e. those that move the plot forward, *satellite* events i.e. those that can be deleted without disturbing the logic of the plot and *spur* events i.e. those which represent possible narrative paths that are not followed. This is shown diagrammatically in figure 2.6 where, the circles are story blocks, the

Figure 2.5: Implicit scene in Teletubby land

squares kernel events, the dots satellite events the diagonals spur events and the dotted arrows are anticipatory or retrospective satellite events.

2.4 Film theory

Motion analysis had a long history from the birth of the motion picture industry in the late nineteenth century in the work of Muyerbridge [199] in the US and Marey [43] in Europe. This purely still image based work is important in technical terms for the role it plays in showing that motion can be both analysed and simulated through the use of multiple still images. A complete discussion of cinematography and film theory is not appropriate in this work but techniques and considerations from works in this field are relied on in the development of the AVA model. The development of early cinema is adequately described elsewhere by Elsaesser and Barker [83] and the role of persistence of vision in the practicalities of motion picture production is described in many texts on the human visual system and by Parkes [213]. Motion analysis can be considered for its aesthetic qualities which can be seen to inspire moving picture sequences in particular Riefenstahl [230] which recreate the multiple exposure techniques used in the early analytical work. Such images and film content are usefully considered because they provide representative instances of content to explore the boundaries of representation schemes.

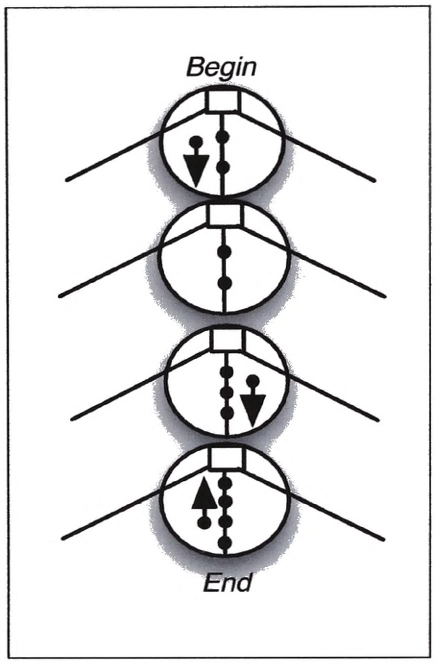


Figure 2.6: Kernel and other events after Chatman [54]

2.4.1 Film language and Christian Metz

There has been an ongoing debate amongst film theorists and others about whether the image track of film is a language and in consequence what makes up the constituents of such a language. Since the process of automating video annotation can be considered to be the process of automating the recognition of film language these issues should be examined. The debate is typified by Metz's [194] semiotic analysis of cinema which centres around the film language question (albeit ultimately inconclusively). In the course of his discussion Metz develops the use of the term **syntagma** according to Stam *et al.* [272]. Metz uses it

> "... as the general term to designate units of narrative autonomy, the pattern according to which individual shots can be grouped, reserving both 'sequence' and 'scene' to designate specific types of syntagmas." Metz [194] p. (40)

In addition Metz develops a taxonomy shown in figure 2.7 of content types that is oft-quoted in film-theoretic circles and is known as the grand syntagmatique. It will be useful to contrast this with other categorisations discussed later in section

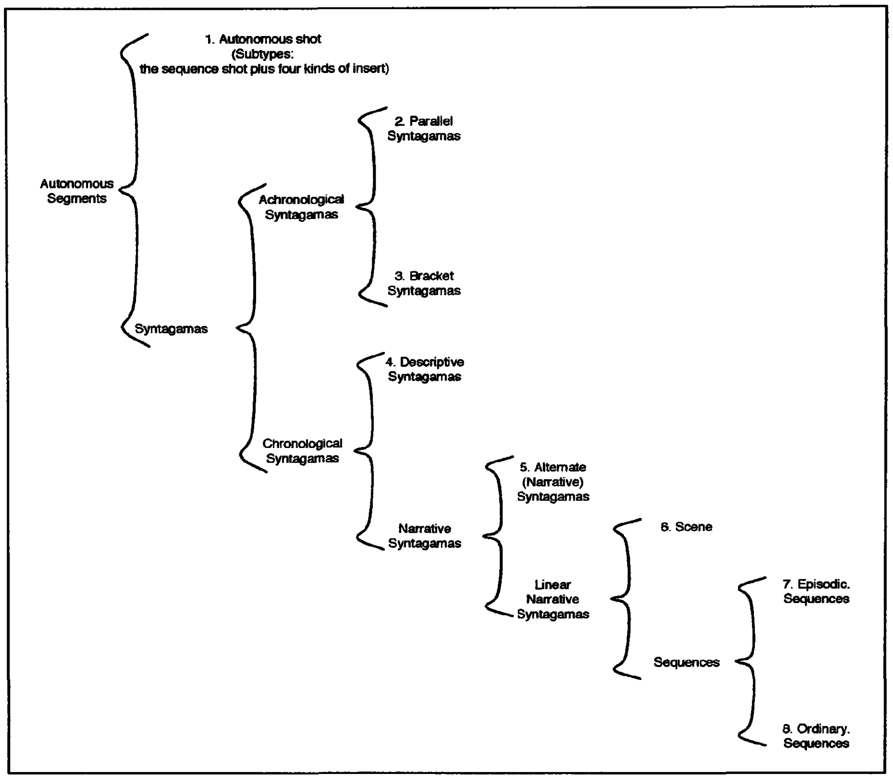

1. Autonomous shot (Subtypes: the sequence shot plus four kinds of insert)
Autonomous Segments
Syntagamas
Achronological Syntagamas
2. Parallel Syntagamas
3. Bracket Syntagamas
Chronological Syntagamas
4. Descriptive Syntagamas
Narrative Syntagamas
5. Alternate (Narrative) Syntagamas
Linear Narrative Syntagamas
6. Scene
Sequences
7. Episodic. Sequences
8. Ordinary. Sequences

Figure 2.7: The grand syntagmatique of Metz

2.4.4 in relation to Arijon's grammar of the film language [14]. The question of whether live action sports is narrative or not does not seem to have been addressed in this discussion. For the time being it is sufficient to note here that the taxonomy is inadequate in this respect and leave analysis for future work.

2.4.2 Pasolini and the cineme

Whilst Metz's ideas are useful and have contributed to the conceptual background of the AVA model, other theorists take a different approach in particular Pier Paolo Pasolini who quoted by Nichols in [208] and in conversation with Hawthorne [271] suggests that the cinema is a system of signs corresponding to a possible semiology of reality itself. This concept is particularly useful from the perspective of content description. Pasolini proposes that the smallest units in the cinema; equivalent to phonemes are unaltered by being reproduced on film. Nevertheless he contends that the language of the cinema has it's own version of double articulation i.e. decomposition of meaningful units into arbitrary non-meaningful units. The minimal units of cinematic language he argues are the various real objects that occupy the frame. He designates these minimal units **cinemes** by analogy with phonemes[271]. Pasolini identifies that there is a relationship between the objects potrayed in the frame and objects in the real world, and he then chooses to identify the mimimal units of articulation within the frame with phonemes rather than nouns. In his view the cinemes are then joined into a larger unit the frame, which corresponds to the morpheme of natural language. This attempt to find a minimal unit of meaning in the visual field is productive and was pursued further by Eco.

2.4.3 Eco: iconic signs, semes and cinematic articulation

Based on his analysis of iconic signs as semes, Eco criticised Pasolini's position and was influential in Metz changing his initial position on cinema as language. Eco's view [79] is that the cineme is deficient as a minimal unit of a cinematic language on the grounds that:

1. The various real units in the frame are in fact "iconic semes" (Eco *op.cit.* p.600) so:

   > " In giving the supposedly real object the function of signifier, Pasolini does not distinguish clearly between ... sign, signifier and signified and referent. " Eco *op.cit.* (p.600)

2. The minimal units of meaning cannot be defined to be equivalent to phonemes since in (verbal) languages phonemes are the *meaningless* units of the second articulation.

   > "Whereas the cinemes of Pasolini (images of various recognizable objects) still retain their own unit meaning;" Eco *op.cit.* (p.600)

3. The frame does not correspond to the morpheme (moneme in Eco *op.cit.* p.601) but rather to an utterance.

Eco prefers identifying the iconic seme as the minimal unit of a film language which in turn leads to his analysis of the articulations of cinema. The position Eco develops is the view that cinema has a third articulation.

> "Let's look again at a frame indicated by Pasolini - a teacher talking to students in a classroom. Consider it at level of one of its photograms, isolated synchronically from the diachronic flux of moving images. Thus we have a syntagm whose component parts we can identify as semes combined together synchronically - semes such as 'tall blonde man stands here wearing a light suit... etc.' They can be analysed further into iconic signs -'human nose', 'eye', 'square surface', etc., recognizable in the concept of the seme, and carrying either denotative or connotative weight. In relation to a perceptive code, these signs could be analysed further into visual figures: 'angles', 'light contrasts', 'curves', 'subject-background relationships'. " Eco *op.cit.* (p.601-602)

As Nack says:

> "Thus an image is based on the triple articulation of photograms, iconic signs, and iconic semes and receives its expression by convention." Nack [200] (p.55)

Eco himself says

> "If we were to make a diagram of this double articulation according to current linguistic conventions, we might make use of two axes at right angles to each other (paradigmatic and syntagmatic axes). But passing from the photogram to the frame, the characters accomplish certain gestures: the *icons* generate *kines*, via a diachronic movement and the kines are further arranged to compose *kinemorphs.* Except that the cinema is a little more complicated. As a matter of fact kinesics has raised the question of whether kines are meaningful gestural units (and thus if you like are equivalent to monemes, and defineable as kinesic signs) can be decomposed into kinesic figures i.e. a discrete kine fractions having no share in the kine meaning (in the sense that a large number of meaningless units of movement can compose various meaningful units of gesture).
>
> Now kinesics has difficulty in identifying discrete units of time in the gestural continuum. *But not so the camera.* The camera decomposes

> kines precisely into a number of discrete units which still on their own mean nothing, but which have differential value in respect to other discrete units. If I subdivide two typical head gestures into a number of photograms (e.g. the signs 'yes' and 'no') I find various positions which I can't identify as kines 'yes' or 'no'. In fact, if my head is turned to the right, this could be either the *figure* of the *kine* 'yes' combined with kine 'nodding to the person on the right' (and in which case the *kinemorph* would be: 'I'm saying yes to the to the person on the right'), or the figure of the kine 'no' combined with the figure of 'shaking the head' (which could have various connotations and in this case constitutes the kinemorph 'I'm saying no by shaking my head'). This the camera supplies us with with the meaningless kinesic figures which can be isolated within the synchronic field of the photogram, and can be combined with each other into kines (or kinesic signs) which in turn generate kinemorphs (or kinesic semes, all encompassing syntagms which can be added to one another without limit)." Eco *op.cit.* (p.602,603)

Eco supports this discussion with a pair of diagrams here reproduced in combined form in figure 2.4.3. In fact it should be noted that kinematics has developed more than one elaborate notational scheme for gesture and movement representation based on graphic symbols capable of capturing the temporal relationships of such gestures, of these Laban notation [163] [164] is the most complete. In this notation both fine grained and large scale movements in canon are represented on a musical stave rotated by 90 degrees, in turn the notational marks decompose into meaningless graphic marks.

Elaborating a little further this can be understood as meaning that the still image (photogram) is composed of recognisable units of meaning: iconic signs these are decomposable into iconic semes i.e semes contributing meaning to an iconic sign. The iconic semes may then be further decomposed into meaningless fragments of the overall image. This decomposition into meaningless units capable of, or being used for, *machine-recognition* is effectively the subject of chapter 3. From this analysis it is also apparent that due to the potential for ambiguity of meaning in a single frame discussed by Eco in the quote above that there is also a

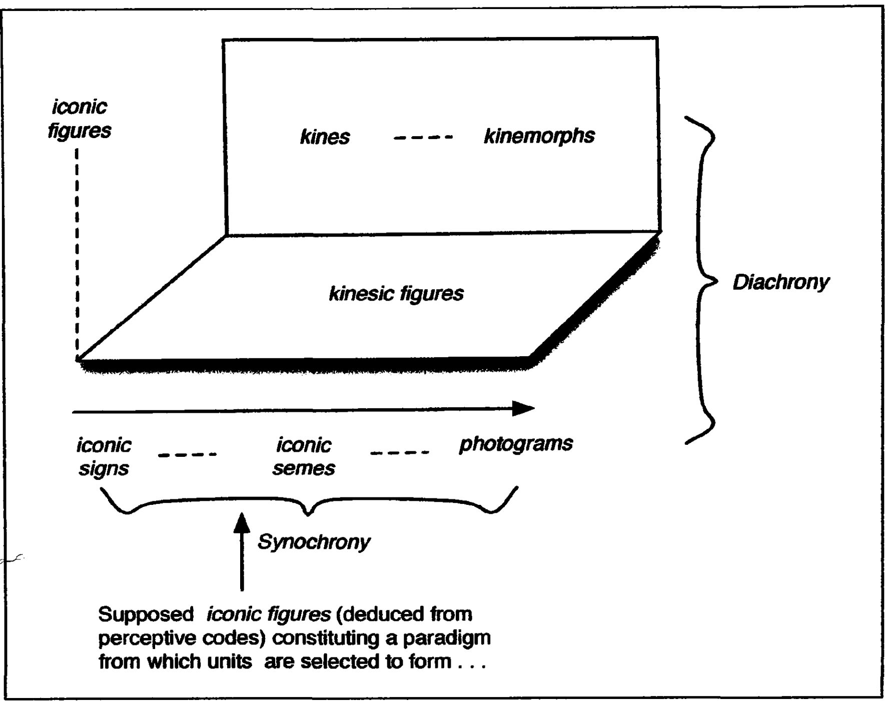

iconic figures
kines ---- kinemorphs
kinesic figures
Diachrony
iconic signs ---- iconic semes ---- photograms
Synochrony
Supposed iconic figures (deduced from perceptive codes) constituting a paradigm from which units are selected to form . . .

Figure 2.8: Iconic signs, semes, kines and kinemorphs after Eco [79] p. 603

contribution to the meaning of the seme from *the temporal axis* which Eco refers to as the kinemorph. The utility of this analysis will be demonstrated soon when the AVA content model is constructed.

2.4.4 The practitioners perspective; Arijon's grammar

In common with the other authors already discussed Arijon [14] treats film as a language in his analysis starting from a position comparable to that of natural language understanding. Paraphrasing Arijon's introduction, his position is that all moving picture productions are productions in the film language, since they use moving pictures to derive meaning through the combination of sequences. He then provides a thorough analysis of parallel editing to create two or more centres of interest which may conflict or be related. He asserts that the basic plot structure of a fiction film is composed of three main elements, summarised from the discussion of his grammar above as:

> "the statement of a situation, thorugh a development of a conflict, to a denoument that closes the play. All scenes fall within these three categories:
>
> 1. dialogues without action
> 2. dialogues with action
> 3. actions without dialogue
>
> these are of course simplified categories. " Arijon [14] (p. 14)

This categorisation can be usefully compared with Metz's taxonomy in section 2.7. The advent of the steadicam has relaxed these distinctions because the requirement that cameras be held in fixed positions for dialogue shots no longer holds, nonetheless they still provide a good basis for analysis. Arijon notes that translation of scenes from script to picture imposes rules that have the dual effect of maintaining continuity across edits and provide solutions for editorial problems as they arise in different situations. Two methods of control are proposed: the first is the distance at which the event is recorded, the second is through the motion of objects. Editing issues are treated extensively by Bloch *op.cit.* who in particular relies on these two criteria and Nack *op.cit.* who introduces additional features most notably colour.

2.4.5 Elements of Arijon's lanaguage

The 5 elements in Arijon's analysis of film language offer much from the perspective of clustering temporal segments. In this treatment he defines the elements the film language to be:

1. The Shot

   The shot is regarded by Arijon as the basic compositional entity in the construction of moving picture content. This is defined by Arijon to be a length of film limited only by the size of the reel, which may be used whole or inter-cut with other material [3].

[3]This definition is consistent with the rejection of the shot as a minimal temporal decomposition unit.

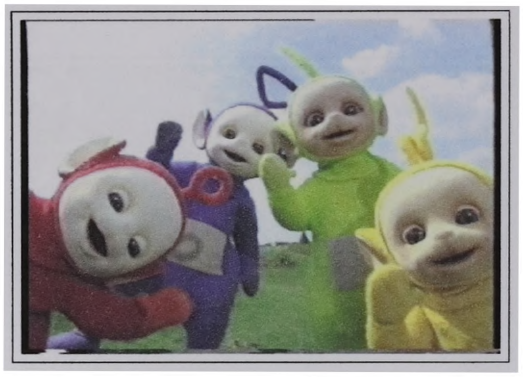


Figure 2.9: Close up of 1 Teletubby (and others) defined as below the armpits.

2. Movement

   Movement is divided into camera motion and object motion and is considered to be present at variable distances, obtained optically *or* physically. He observes that experience has taught cameramen and editors that there are 5 main distances at which to photograph the body and that these are best defined in terms of body parts as shown:

   2.1 Close up or big close up (CU)
   See figure 2.9

   2.2 Close shot (CS)
   See figure 2.10

   2.3 Medium shot (MS)
   See figure 2.11

   2.4 Full shot (FS)
   See figure 2.12

   2.5 Long Shot (LS)
   See figure 2.13

   These definitions of shot distances are very useful when considered alongside the existent-context construct developed in section 2.30 and the limits to the object recognition and location capabilities outlined in chapter 3. The criteria for defining the differences between the shot types is widely known and reasonably consistent across authors although some extensions are seen

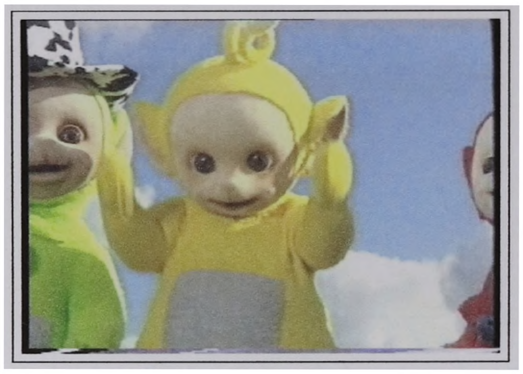


Figure 2.10: Close shot Teletubby defined as below the chest.

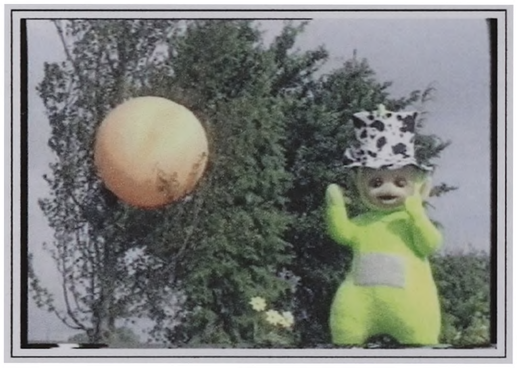


Figure 2.11: Medium shot Teletubby defined as below the waist.

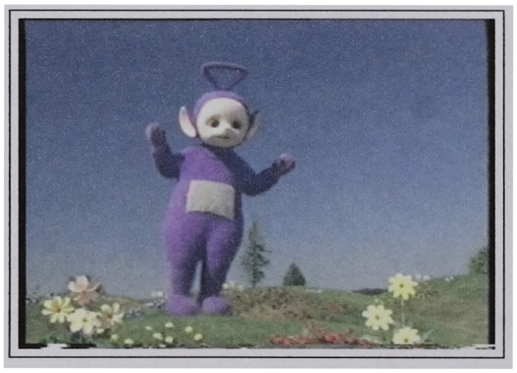


Figure 2.12: Full shot Teletubby defined as below the crotch.

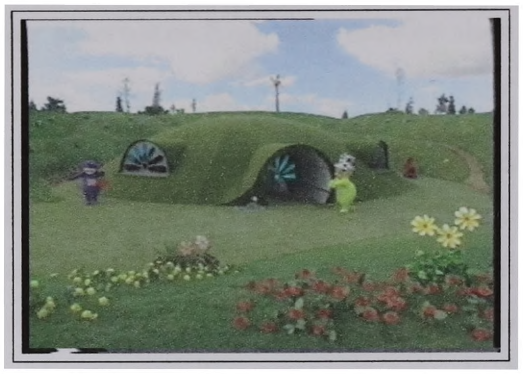


Figure 2.13: Full shot Teletubby defined as below the knees.

to the list at the limits in Katz. [64]. (The extended list undoubtedly reflects changes in the production technologies available in the 20 years separating the publication of the two works). Another observation that can be made from consideration of these definitions is that there is approximately a 2-1 ratio between the amount of actor body height shown in each shot class, it can be expected from this that the colour and texture recognition methods discussed in chapter 3 would fail across these boundaries. This will be discussed further in the conclusions.

3. Editing Types

   In Arijon *op.cit.* the analyses of editing types are restricted to the "classical" Hollywood types whose development is described elsewhere [83]. Other editing paradigms are described by Bordwell in [37]. These are only briefly defined here due to limitations of space as:

   3.1 Master shot scene registration

   Here a master shot is used to register the whole scene c.f. figure 2.5 showing the teletubby house.

   3.2 Intercut mastershot

   The master shot is inter-cut with shorter takes covering fragments of the scene from different distances or other subjects at other locations.

   3.3 Blended master shots

   Two or more master shots are blended in parallel so the point of view of the observer alternates from one master shot to the other.

4. Visual Punctuation
   Visual punctuation is defined as a straight cut or optical effect such as fade in or fade out (Effect recognition is discussed in depth in the next chapter). Arijon takes visual punctuation to be what occurs at shot boundaries,. i.e. cuts or other optical effects such as wipes and dissolves. In other words what is considered to be the temporal segmentation task at the image feature level is equivalent to the recognition of punctuation marks at the film language level. It will be appreciated however that these visual "punctuation marks" do *not* map onto natural language punctuation marks.

5. Scene Matching
   To match scenes Arijon identifies that three requirements must be met namely: postion, look and movement. This is discussed further below in the context of the analysis of Bloch's [35] work.

2.4.6 Additional rules in the grammar of the film language

Arijon then goes on to consider set of rules governing different aspects of shot creation in more than 20 categories. Potentially many of these can be adapted into the visual object recognition and temporal segmentation algorithms to aid these processes. This is also considered to be beyond the scope of the current work.

2.5 Artificial Intelligence

Artificial intelligence (AI) had its origins around 50 years ago in the work of von Neumann [289] amongst others and began to mature in the 1970's and was typically concerned with search, search representations, deduction, problem solving, planning, learning and their applications [294]. These approaches developed further in the 1980s with advances in what became known as expert [146] [107], blackboard [84] and machine learning [283] systems and their applications. Case base reasoning was developed in the late 1980s and early 1990s [158] for systems using cases or instances of knowledge in a domain. The term knowledge based systems also emerged to identify systems using reasoning and knowledge [273]. Whilst these remain core concerns of AI, latterly research has shifted towards multi agent [90] and distributed agent systems and more advanced application domains. These multi and distributed agent systems typically embody multiple instances of the techniques from the earlier phases of the development of AI and are concerned with communication and co-operation between the instances. This has renewed interest in languages for the representation, which originated with KRL in [41] and interchange of knowledge with the development of KQML [92] Since formal logic [284] and logic programming have played a core role in AI work specialized programming languages such as LISP [295] Prolog [59] [274] and POP [251] appeared. Prolog is of interest as it implements term resolution as its core processing activity [5] and has been integrated into image processing systems by Batchelor [21]. Whilst the AI specific languages are attractive because of the representational economy they provide, these languages often introduce as many problems as they solve. So hybrid languages and environments emerged such as Walthers Babylon [292] and the commercial Harlequin Lisp-Prolog product[178]. Mention must be made to the Hearsay-II blackboard system [234] in [84] as it provided some of the early conceptual background to the development of the AVA architecture. In addition Bayesian [169] an overview of which can be found in Duda *et al.* [231] and neural network techniques [39] are used in pattern recognition system approaches to video temporal segmentation.

2.5.1 Knowledge representation

Knowledge representation is a core theme in AI with a long history, see Brachman [40] for a collection of classic papers. Since it underpins expert blackboard and case based reasoning systems as well as the AVA model it will now be examined in more detail. The role of knowledge representation can best be summarized by reiterating Smith's [256] *Knowledge Representation Hypothesis* as follows:

> "Any mechanically embodied intelligent process will be comprised of structural ingredients that a) we as external observers naturally take

to represent a propositional account of the knowledge that the overall process exhibits, and b) independent of such external attribution, play a formal but causal role in engendering the behavior that manifests that knowledge.' Smith [254] (p.33 in [40])

Key pieces of work on knowledge representation that have contributed to the approach adopted in the AVA content model are Sowa's conceptual graph theory [267], [265] [266] , Schank's conceptual dependency [238] , scripts [243] and later story work [241]. Sowa and others have extensively treated semantic networks and knowledge representation. Since a combination of semantic networks and Schanks conceptual dependency theory is adopted here, more elaboration of semantic networks is necessary. There are many notations for semantic networks, Sowa considers that they generally exhibit the following themes in spite of divergent logic operators, meanings, quantifiers etc.:

1. The nodes in the net represent concepts of entities, attributes, events, and states.

2. Some conceptual relations represent linguistic cases, such as *agent, patient, recipient, or instrument.* Other conceptual relationships represent spatial, temporal, causal, and logical connectives. Still others specify the role that one entity plays with respect to another, such as *mother, owner,* or *residence* but the representation of roles such as relations is an area of divergence between different systems.

3. Concept types are organized in a hierarchy according to levels of generality, such as ENTITY, LIVING-THING, ANIMAL, CARNIVORE, FELINE, CAT. This hierarchy is often called a *type hierarchy* or *taxonomic hierarchy.* It is also called a *subsumption hierarchy* since the instances of a general type such as ANIMAL *subsume* the instances of a more specialized type such as CAT.

4. Relationships that hold for all concepts of a given type are *inherited* through the hierarchy by all subtypes. Since every animal requires oxygen, the property of requiring oxygen is inherited by every carnivore, feline and cat.

The AVA model adopts a UML representation of semantic networks, Schank's conceptual dependency and script theories, through an analysis that is provided in section 2.10 during the development of the AVA model. Minsky [196] considered a less formal approach and suggested that when a new situation like a childs birthday party is encountered a data structure for representing a stereotyped situation is selected. He proposes that a network of nodes and relations fulfills the role that he calls a frame. The top level of the frame is fixed: always true about these situations and the lower levels have slots that must be filled to define instances of situations, frames are linked in frame systems with different frames representing descriptions of the scene from different positions. He defines four levels of networks namely: surface syntactic, surface semantic, thematic and narrative frames.

2.6 Content modelling approaches

Existing work on video and image content modelling has typically been driven by interest in; novel, such as automated editing, archiving, browsing or in hypermedia presentation applications. This has resulted in differing emphases being placed on: semantic description, feature extraction, temporal decomposition and statistical modelling by researchers in the field. Most recently Davis has added location contextual information to this categorisation [189]. Although clearly still image description is not concerned with temporal decomposition, video content modelling has typically, to a varying extent involved two or more of these areas of concern. This hampers a taxonomic characterisation of previous work because of the intersections of these concerns as illustrated in figure 2.14. So an outline of key pieces of pieces of prior work is given before going on to describe in more detail those that have had a formative influence on the development of the AVA content model. The key pieces of prior work include Aguierre-Smith and Davenport's [252] Stratification System where textual annotations were manually associated with media "strata"; effectively overlapping hierarchical temporal decompositions of the video *c.f.* Lougher *et al.* [179]. Each strata represents a temporal extent over which the proposition implied by the annotation is applied. This was targeted primarily at browsing applications. Davis [68] in Media Streams developed

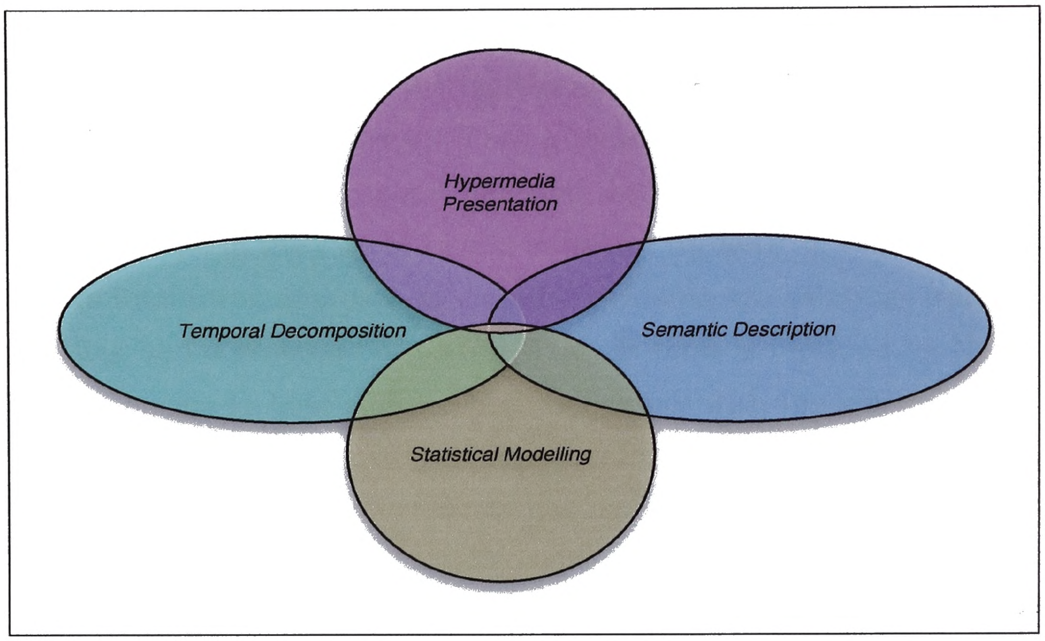


Figure 2.14: Content modelling concerns

a system for editing and retrieval based on manually assigning iconic depictions of content with a number of content sequences. Davis maintains [67] a position advocating automated editing and emphasises the need for a formalist[4] semiotic approach to content modelling. Solway [261] has manually annotated moving picture sequences using terminology automatically extracted from collateral texts in specialist domains; in particular an archive of dance related material. The work at Columbia by Benitez *et al.* [27] using an integrated application of WordNet [89] to textual annotations together with an extended set of visual descriptors is also worthy of consideration in this context. Several other browsing tools have been developed including the ellision and fold based approach of Lougher *et al.* and key frame approaches of which Komlodi and Marchionini's [159] is typical. Latterly Appan and Sundaram[12] describe a networked event driven collaborative browsing approach. There are also more recent tools based on MPEG-7 descriptors such as the motion activity descriptor based approach of Divakaran *et al.* [74].

The known examples of automated semantic annotation of images and video are found in the European Union (EU) sponsored ASSAVID project [156][17] and an

[4]Formalist here in the semiotic school not formal methods sense

earlier Photobook effort at MIT [219]. There have been efforts to automatically produce video indices of which the multimodal approaches by Li and Kuo [172] and Adams [2] are of most interest. They give accounts of model based statistical approaches to automated temporal indexing, based on mapping feature clusters onto; film theoretic in the case of Li and narrative theoretic in the case of Adams, models. An interesting approach to describing chiaroscuro paintings using semantic modelling and a grammar of feature analysis tools is reported by Nack *et al.* [218]. This appears to have suffered from difficulties with the inherent ambiguity of art criticism terminology. Content modelling from the perspective of integrating multimedia into hypermedia documents has received attention from Hardman *et al.* [181] [122] and Auffret *et al.* [100] owing to the inadequacies identified with applications of the Dexter [119] hypermedia model (modelled formally in Z [73]) to multimedia content. Nack and Putz [204] describe the A4SM project and the associated developments of hardware tools and software applications to automatically annotate video in a news media production context.

2.6.1 Bloch

Bloch's [35] [36] like Parkes' work was pioneering in the field of moving picture content representation and provides some of the conceptual framework for both Nack and the current author of this work. In common with Parkes he adopted the use of concepts from Schank that are discussed later in this chapter and given an in depth treatment in section 2.10. His system generated film sequences based on rules defining acceptable montage conditions controlled by three manually annotated parameters namely:

1. **LOOK** Individual people in suceeding shots should appear to look at one another. If the first is looking left the second should look right if context is to be maintained from shot to shot.

2. **POSITION** This means that the respective positions of characters in the first shot should be maintained in succeeding shots to maintain visual context.

3. **MOTION** Again this rule states that perceived speed of moving objects should be maintained across the shot boundaries.

Bloch uses the term **setting** for the spatio-temporal extent of the shot under consideration; this usage is at variance with Parkes' use of the term setting as discussed further below. It is clear that Bloch's work in part derives from an analysis of Arijon's film [14] grammar reviewed in section 2.4.4. It is also clear that he is in effect defining sets of propositions about the content and seeks to maintain context through propositional continuity c.f. Garnham's assertion from Appendix C.

2.6.2 Parkes and CLORIS

Parkes [213] developed the CLORIS prototype system for applications of Computer Controlled Videodiscs (CCV) into Computer Aided Instruction (CAI) systems. CLORIS was intended for application in a system defining knowledge about a restricted narrative world for CAI purposes. In the CLORIS prototype system Parkes' developed a model of film structure assumed true for a restricted set of syntagmas within Metz's [194] taxonomy to simplify his model development. The syntagmas were: 6 Scenes i.e. spatially and temporally integrated events, 8 ordinary sequences i.e. those where events having no relevance to the overall could be omitted. Parkes' [213] assumed structure of films is illustrated in figure 2.15.

In this diagram the setting formalism provides the basis for Parkes' spatio-temporal hierarchical decomposition of film, in preference to the shot, sequence or scene, largely because of the ambiguities inherent in the use of these terms. The setting definition is subsequently generalised to be a set of images sharing the same objectively describable visual state and the setting description as the objective description of the setting. Parkes contends that on this basis the shot disappears from view in terms of the description of events. As events map onto settings, settings are constituents of scenes and settings share location characteristics with scenes. This leads to the development of the structure shown in figure 2.15. Parkes relies on Carroll's [53] structural analysis of film in developing his model. Based on this model of film structure a set theoretic representation language for film content

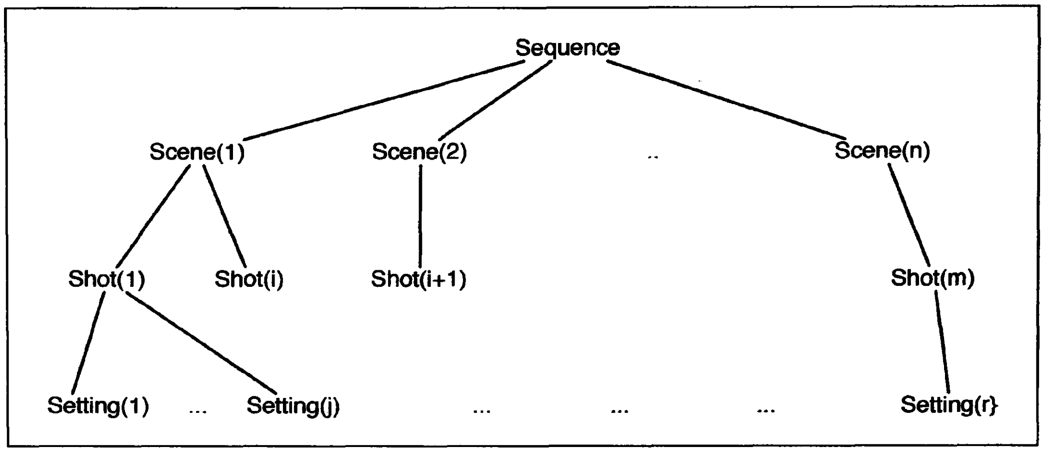


Figure 2.15: Assumed structure of films considered by Parkes [213]

adhering to the model was developed. The model and film language rely on formal definitions of a number of terms which are reproduced in Appendix B. To assist the analysis of CLORIS and in the AVA content model development the CLORIS model has been recast in UML in Figure 2.16. The UML representation of the CLORIS model is based on (i) Parkes' assumptions about the structure of films shown in figure 2.15, (ii) the abstract event description from figure B.1, (iii) the formal logic and set theory quoted above and (iv) a close reading of Parkes (1988). When the language and model are reviewed from the perspective of automating content description *creation* conceptual deficiencies are identified. In particular, it has been found that the CLORIS model does not fully account for the distinction between the expression, data and content planes. These distinctions have been found to invalidate the setting concept during the development of the AVA model. Whilst the CLORIS model is clearly influenced by object orientated modelling concepts they are not rigorously applied in the model. In fairness to Parkes, this is not surprising, since CLORIS pre-dates the analysis of content that led to the development of the revised and extended semiotic models. CLORIS also predates many of the advances in system and object orientated modelling techniques of the last 15 years. Not withstanding these criticisms Parkes attempt to identify a minimal unit of content decomposition and its identification with the setting provided a key conceptual foundation for the development of the AVA content

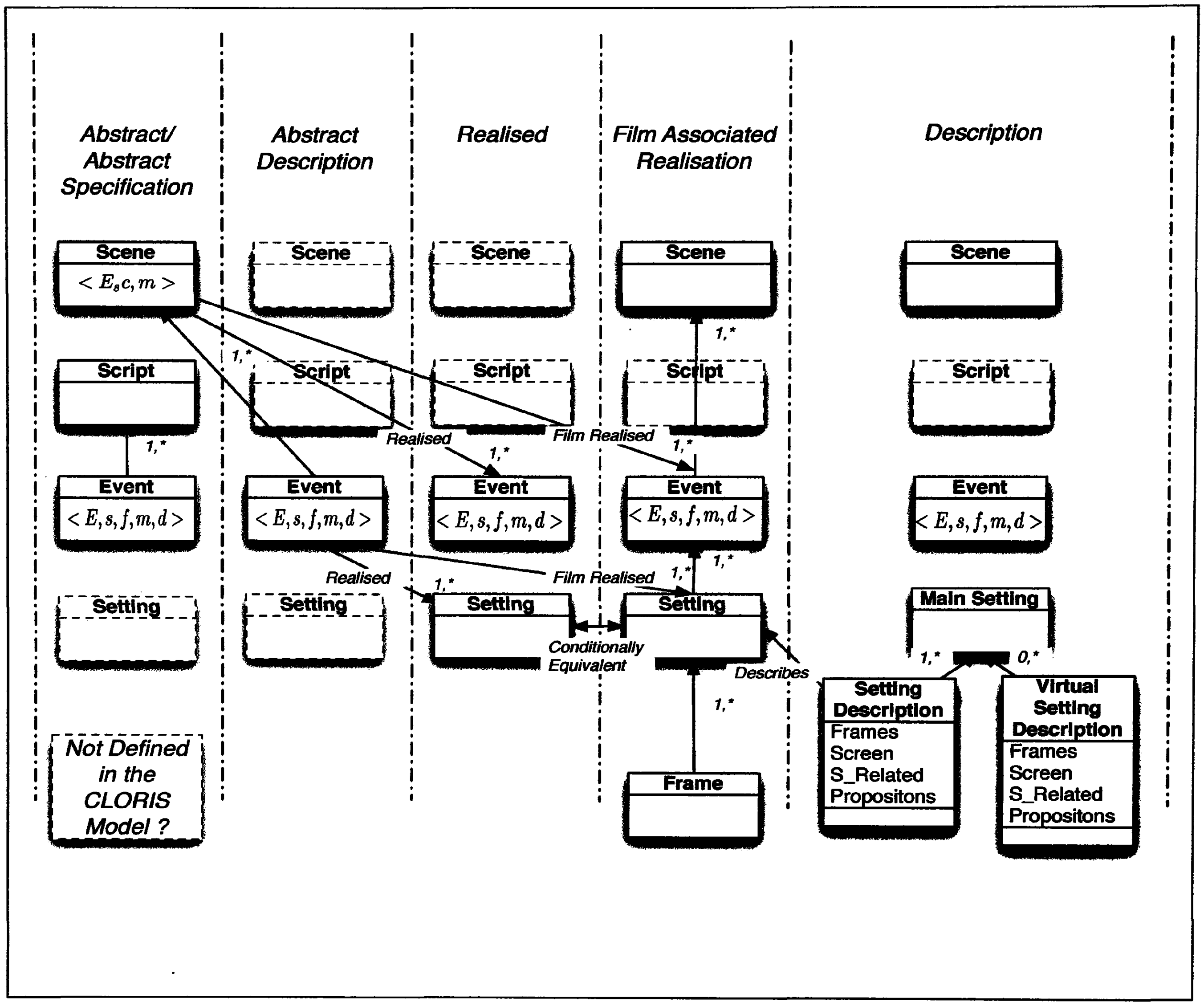
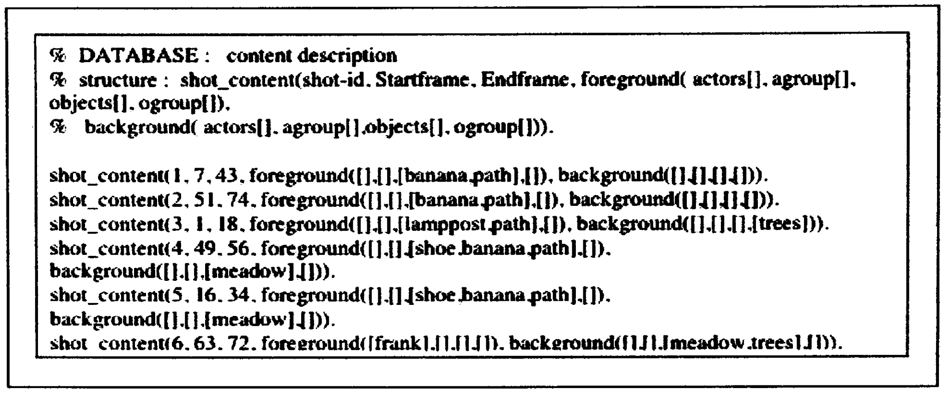


Figure 2.16: The CLORIS content model

model.

2.6.3 Nack and Auteur

Nack viewed film editing as a planning task based on his studies and analysis of the film editing process. He subdivided the image into foreground and background allowing each to contain objects and actors. Whilst his representation language was the Prolog [59] programming language, his analysis of content was by nature



```
% DATABASE : content description
% structure : shot_content(shot-id, Startframe, Endframe, foreground( actors[], agroup[],
objects[], ogroup[]),
%  background( actors[], agroup[],objects[], ogroup[])).

shot_content(1, 7, 43, foreground([],[],[banana,path],[]), background([],[],[],[])).
shot_content(2, 51, 74, foreground([],[],[banana,path],[]), background([],[],[],[])).
shot_content(3, 1, 18, foreground([],[],[lamppost,path],[]), background([],[],[],[trees])).
shot_content(4, 49, 56, foreground([],[],[shoe,banana,path],[]),
background([],[],[meadow],[])).
shot_content(5, 16, 34, foreground([],[],[shoe,banana,path],[]),
background([],[],[meadow],[])).
shot_content(6, 63, 72, foreground([frank],[],[],[]), background([],[],[meadow,trees],[])).
```

Figure 2.17: Formal content representation in Auteur [200]

semiotic in that it assigned Prolog terms to the main visual objects denoted by each sequence. This identification of the role of semiotics in content representation has played a crucial role in the AVA work. A fragment of his representation scheme is provided in figure 2.17. The majority of the work was concerned with analysis of narrative humour and the development of strategies for automatically generating humorous moving picture sequences. Sequence retrieval in Auteur used Prolog term resolution on propositions defining the content's appearance in a sequence, which is illustrated in figure 2.18

2.7 Knowledge representation and feature extraction

Techniques are identified later for extracting features from the continuous and compressed data planes to identify optical effects and, identifying and locating visual objects. To allow development of the content model it is now necessary to examine how these features can be related to descriptive categories from natural language and used in a formal manner in the model. This, in effect, begins the development of knowledge representation for the AVA model. This is continued by examining objects and their attributes in still images, temporal issues and an initial treatment of temporal segmentation from a descriptive point of view.

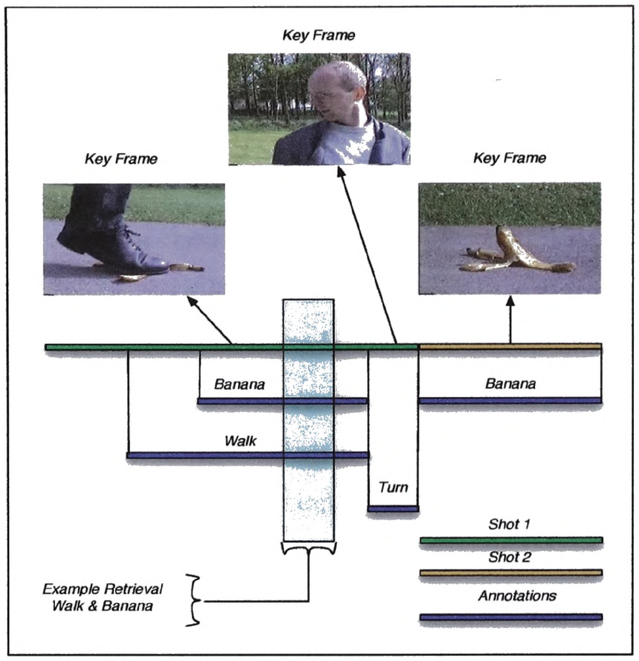


Figure 2.18: An example of retrieval in Auteur

2.7.1 Representing colour

The choice of colour space is important from the point of view of similarity based retrieval and it is significant that whilst from the perspective of retrieval, considerable attention has been given to colour, little attention has been given to an effective scheme for the semantic description of colour. The area of colour analysis is one of the few where significant work has been carried out by Berlin and Kay [29] to examine the relationship between commonly used semantic categories such as red, black, white etc and image features. This work is significant because it gives insight into the mappings between semantic colour terms and colour samples from the Munsell [197] [198] colour space, not only amongst North American English speakers but also amongst speakers of other languages. However later authors such as Wyler[296] have noted that the 11 English linguistic categories adopted do not map onto the linguistic categories in other languages. Of particular interest is the that the number of basic colour categories (11) is small: white, black, red, yellow, green, blue, brown, purple, pink, orange, grey. It is also noteworthy that in languages with a smaller number of colour terms this can contract to as few as

2 (black and white). It is also interesting to note that there have been arbitrary choices of colour as the basic colour terms by key figures in the history of colour theory including Newton [207] and Goethe [288] amongst those who quote small but differing maximum numbers. This contrasts with the arbitrary naming of a large number of colour charts including the well known X-Window system and the reliance of an arbitrary numbering scheme in the Pantone System [81]. The colour terminology problem was also noted by Hjelmslev [128] in the lack of correspondence between English and Welsh colour terms in the context of the development of his semiotic approach where gwyrrd:green, glas:blue, llwyd:gray/brown do not have a one to one correspondence in their spectral mappings. Other examples of this come from Lyons [182]. Neither Lyons nor Hjelmslev reference Berlin or give a sources for experimental basis for this lack of correspondence between colour terms.

An alternative approach to that of Berlin and Kay *op.cit.* of segmenting a colour space by colour terms is the investigation of colour as a linguistic phenomenon as exemplified by Wyler *op.cit.* Mapping between semantic categories and image colours is significant because it provides a basis for comparing colour retrieval with other feature types and for contrasting the similarity-based image retrieval techniques with what can be termed image categoric based techniques utilising semantic terms such as red car or black car for example. In conclusion, the semantic description of colour, considered by many authors above to be fundamental in describing images, is almost entirely arbitrary, meaning that assigning a meaningful, objectively verifiable, descriptive term to a visual object's colour automatically, is an area fraught with difficulty. Therefore, a more effective approach is to provide a representation of colour in the user interface which is readily achieved, if colour similarity based retrieval is required.

2.7.2 Representing texture

If an analysis of colour semantics is problematic a comparable analysis of texture is almost intractable at present since so few terms exist. i.e. rough, smooth. Also the term 'texture' is in fact, used as an analogy for spatial frequency. This is amply illustrated by the reliance on the Brodatz [44] photographic collection of textures as the foundation for most work on texture metrics for image retrieval and segmentation. There appears to be no comparable work to Berlin's *op.cit.* in the texture domain. Therefore, it is proposed to adopt the same approach for texture as colour and represent it in the user interface if textural similarity-based retrieval is needed.

2.7.3 Representing edges

Whilst the procedure proposed by Marr [190] might be attractive from the point of view of reconstructing 3 dimensional representations of the 2 dimensional objects seen on the screen this is not the approach adopted here. Spatial edge frequency can be represented visually, but is more useful as a mechanism for representing object boundaries, recognising that an object from the observer's point of view may be made up of a number of edge defined expression plane segments. This indicates a direction for further work that will be introduced in section 2.32 and is further discussed in section 5.4.3.3.

2.7.4 Representing objects

Object representation has been systematically investigated by other authors in an extensive literature represented by Parkes *op.cit.*, Nack *op.cit.* and Bloch *op.cit.* in the context of moving picture content representation. Motivated by discussions later in this chapter, parts of these authors' approaches are adopted for the AVA content model. From these analyses it will be shown that the knowledge base needs to contain representations of the following entities related to still images and their interrelationships. There is also a need to represent class membership as discussed in the next section 2.7.5; other relationships are developed in section 2.9.2. The issue of whether the still image is the appropriate level of temporal decomposition for object representation is taken up in section 2.8.6.

1. Object identification

2. Object position

3. Object outline
4. Object classification into:
    - 4.1 Foreground
        - 4.1.1 Actors
        - 4.1.2 Props
    - 4.2 Background
        - 4.2.1 Actors
        - 4.2.2 Props
5. Object Relational Hierarchies

The need to separate the image into foreground and background derives from an analysis of Arijon's [14] work in section 2.4.4 and its application by Bloch [36] and Nack's [200] more extensive treatment in Auteur. In effect, this identifies the need to represent what are the *object* portions of the existents in Chatman's [54] terminology. The representation of the scenes or setting will be deferred until after the analysis of temporal decomposition. Static objects are taken to be part of the scene and moving objects to be taking part in events.

2.7.5 Class membership

The need to represent object-relational hierarchies is identified above and amongst these object-relations is that of class membership. It can be readily appreciated that there is a significant gap analogous to the famous semantic gap between machine code and high-level languages [76], between the various image features described in Chapter 3 and a knowledge base or a classification hierarchy that contains disparate objects such as aeroplanes, fish or frogs. It is clear from DNA analysis in the botanical and zoological domains that many objects that are visually homomorphic are in fact not related at all. Of significance in this discussion is the issue in developmental psychology of class creation and recognition, such as when a child differentiates between their *own mother*, *other's mother*, and the class of *mothers.* However each instance of an *other's mother* will share one or more feature sets with *own mother*. The notions of a semantic gap and the need to address it is now an active area of image retrieval research from the perspective of clustered keywords but not from a formal knowledge representation point of view as exemplified by [305] [112]. This however does not fully expose the difficulties in finding a methodology for mapping visual features onto semantic class hierarchies which receive little attention in the image retrieval literature. These problems have a long history in the AI knowledge representation literature with Sowa [266] giving a thorough review of the subject and Brachman [41] reprints classic papers in the field.

The problem can be exemplified by considering the objects that are classified as instances of *chair* in design books such as, Fiel and Fiel [91] and Byars [192] particularly when the extreme examples of chairs depicted by Warhol [120] are also included. Whilst it may be argued that an instance of *chair* may have a particular colour, texture, shape or combination thereof a generic classification and retrieval scheme capable of distinguishing these objects from instances of *bed* or *stool* seems fraught with difficulty. This was highlighted recently by Sommerville [124] in an email discussion in relation to furniture choice for a new facility. It will be evident from the discussion above that the human ability to abstract a generic classification from a series of instances is reliant on substantial *a-priori* cultural and functional knowledge. This knowledge is wholly cognitive and incapable, at present, of being derived bottom up from feature sets and rules. So, it must be embodied either explicitly or implicitly in the recognition system.

The AVA approach differs substantially from the normal image retrieval or video browsing approaches, which rely on embodying implicit knowledge about the domain of discourse in the system architecture. The AVA approach seeks to make the capture and use of this knowledge explicit and exploits this explicit representation in the system design. This assertion is further illuminated by Swain and Ballard's [278] reading of Chapman [63], to justify the assertion that colour is a much more important feature for routine behaviour, than was considered to be the case in the work of Marr [190], whose model of vision emphasised complete 3D object reconstruction. They also discuss the relationship of colour to the class identity of an object together with the significance of resolution to shape retrieval. It can be considered that a particular shape set will only hold true over a particular range of scales. Again reference to the Extended Semiotic model helps to support the difficulty that Swain and Ballard have with Biederman's opinion that:

> ... "surface characteristics such as colour and texture will typically have only secondary roles in primal access ... we may know that a chair has a particular colour and texture simultaneously with its volumetric description but it is only the volumetric description that provides efficient access to the representation of *chair*." Biederman [31].

Also of note is that when Lyons [182] compares kinship relationships in Russian with those in English he identifies that even fundamental relationships such as these diverge across different languages.

From this discussion it will be apparent that meaningful class membership identification will have to be defined explicitly and these relationships will have to be independent of *computational* object-inheritance hierarchies. Less obvious is that similar feature sets are likely to be bound to distantly related objects in the relationship hierarchies. This points to a requirement for separate search spaces for features and descriptions. This point is also returned to in chapter 5.

2.7.6 Representing which time?

The semantics of time are well understood from the foundation work by Allen [147] which can be readily represented using figure 2.19. This modelling does not expose the full implications of describing time in the context of multimedia, so an analysis comparable to that conducted into the semiotics of content must be carried out. This is due to the use of moving picture content to represent events happening in some real world, in the past, present or future, or in some completely imaginary world, where many of the laws of physics as we currently understand them have been conveniently repealed. An extreme example of such an imaginary narrative world is provided by Galaxy Quest, (1999).

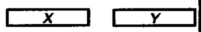
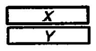
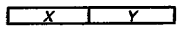
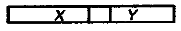
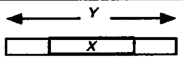
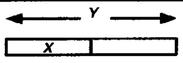
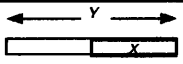
Relation	*Symbol*	*Inverse Symbol*	*Pictorial Represeantation*
X before Y	<	>	
X equal Y	=	=	
X meets Y	*m*	*mi*	
X overlaps Y	*o*	*oi*	
X during Y	*d*	*di*	
X starts Y	*s*	*si*	
X finishes Y	*f*	*fi*	

Figure 2.19: Allen's binary temporal relationships

Therefore it is necessary (as a minimum) to distinguish the following temporal domains

1. Content **capture** time
2. Content **production** time
3. Content **expression** time
4. Content **distribution** time
5. Content **narrative** time
6. Content **represented** time
7. Content **referred-represented** time
8. Content **description** time

Each of these temporal domains will now be defined

Definition 1. *Content **capture** time is; the time at which the content is originally captured i.e. the time of recording or filming etc. There may be a many to one relationship between event recording and the recorded event instance used in the work as distributed.*

Definition 2. *Content **production** time is the time at which the content item is edited. This time is decomposable into many workflow operations in a commercial enterprise. Workflow is not considered to be in scope in the current discussion. Multiple versions of the same content item may be produced. i.e several versions of the film True Romance (1993) exist on video tape, all with minor editing changes.*

Definition 3. *Content **distribution** time is the time at which the content item is distributed, again there may be many distribution instances for a commercial content item.*

Definition 4. *Content **expression** time is the time at which the playback of the content item occurs. Clearly there may be many separate instances of expression for any instance of a content item. For any given expression instance the expression time for any given frame or picture in a content item will occur in decompressed picture order. That is in a picture sequence $P_t^1, \dots P_t^n$, P_t^i is always **expressed** at time $P_t^{i-1}, < P_t^i < P_t^{i+1} \quad \forall\{P\} \quad i = 1, \dots, n$. Where t is the content **represented** time.*

Definition 5. *Content **narrative** time is the chronological ordering of events in the narrative. This is the natural ordering if each event were fully represented and they were expressed in the order they occur. It is clearly difficult in conventional text for events occurring in parallel (or at* equal*) times to be depicted so conventionally there is always some ordering applied. In film there are often instances of* equal *depiction using split screen techniques notably in the opening sequences of Grand Prix, (1966) and The Thomas Crown Affair, (1968).*

Definition 6. *Content **represented** time is; the time represented in the content this may be the time "now" for the actors in the moving picture sequence. It may also be in the past or future for the actors in the main narrative. This derives from the observation that the temporal ordering of the events depicted i.e. the*

narrative *time in a content item may not follow the actual chronological order of those events in the narrative world. That is, the time-line of the events depicted at* ***expression*** *time is different from the narrative time ordering. This* ***represented*** *"now" may also be in the past present or future for the observer. An excellent example of this is Stanley Kubrick's 2001, (1968) which was set in the future when released, is now set in the past and has a narrative extent spanning millennia.*

This demonstrates the need for a more thorough analysis of narrative which is provided in section 2.3.

Definition 7. *Content* ***referred-represented*** *time is a time referred to in a moving picture sequence. A common visual clich for this is to depict a photograph or a "home movie" belonging to one of the actors in the sequence showing a childhood scene or event.*

Definition 8. *Content* ***description*** *time is the time at which the content is described.*

It is the **expression, narrative, represented, referred-represented, and description** times that are of concern in the current context. The solution to managing these differing time lines is to present the user with two time lines in the user interface each capable of presenting material in a manner comparable to a project planner or a non-linear editing program. In some respects this is after the manner of Butler *et al.* [47]. It is clearly beyond the capability of current techniques to automatically resolve such temporal inconsistencies and it is anticipated that whilst systems should be capable of *representing* these relationships the *identification* of them will be describer determined.

2.8 AVA content model development

The objective in developing the AVA content model is that of finding an effective mechanism to describe content in terms of both meaningful descriptive terms and feature sets that can be extracted from the media data that can be combined in a bi-directional approach. The development of the AVA content model begins by examining the revisions to the semiotic model that provided conceptual underpinnings for MPEG-7 by distinguishing between content and expression. Inherent to this task is the development of a set of relationships between; content, expression, data, compressed data and description plane entities that represent the meaningful characteristics of the first 4 in the content description. These revised and extended models are then developed further by introducing Chatman's cross cutting of form and substance to provide the conceptual framework for the development of the AVA content model proper. This begins by examining a representation of the classical content model in UML and continues by making comparisons with the CLORIS model. The setting construct is considered as the minimal unit of content description and replaced with the existent-context and then the hierarchical decomposition of content for the AVA model is completed. The approach to the presentation of the theoretical background in chapter 4 and the analysis of previous content models in this chapter has been relatively abstract. The remainder of this section is more concrete in that it combines the background material with insights gained from the implementation of the AVA demonstrator and methodology described in a later chapter.

2.8.1 Revising semiotic content models

The basic semiotic model introduces the relationship between the expression and denotation of a sign system. The distinction semiotics makes between content and expression was judged to provide a valuable analytic basis for discussing multimedia content. This representation was extended by the current author to overcome the difficulties in identifying the separate areas of concern when description of multimedia content that may or not be compressed was under discussion. The analysis described in Hartley *et al.* [123] centres around reiterating some of Parkes' analysis of shot and scene break terminology which was then of particular significance to the MPEG-7 temporal decomposition terminology with a view to identifying the minimal units of content representation. The model was useful in the context of the MPEG-7 requirements definition process as it adds elements to the basic semiotic model to represent data, expression, content, description and the viewer/listener. The conceptual underpinning the model still provides to MPEG-7 has already been noted in the introduction. The viewer/listener is referred to as the observer in this and subsequent models. This model is beneficial because it allows clear distinctions to be made between the following:

1. content, i.e. what is invoked in the observer's mind by the
2. expression *through*
   - 2.1 rendering on to screen
   - 2.2 processing to loud speaker
3. of data *for*:
   - 3.1 natural image,
   - 3.2 natural video,
   - 3.3 natural audio,
   - 3.4 synthetic image,
   - 3.5 synthetic video, or
   - 3.6 synthetic audio
4. with description *of:*
   - 4.1 content
   - 4.2 data
   - 4.3 compressed data

The revised semiotic model is illustrated in figure 2.20 where the modelling was heavily influenced by the object orientated approach to media of MPEG-4 [78] which provides compression and transport technologies for *synthetic* and *natural* content. This terminology was developed in response to the difficulties encountered when using the term *meta data* in the context of multimedia. The difficulties become more apparent when the need to start referring to descriptions of content description as *meta* meta data, which rapidly produces recursive definitions.

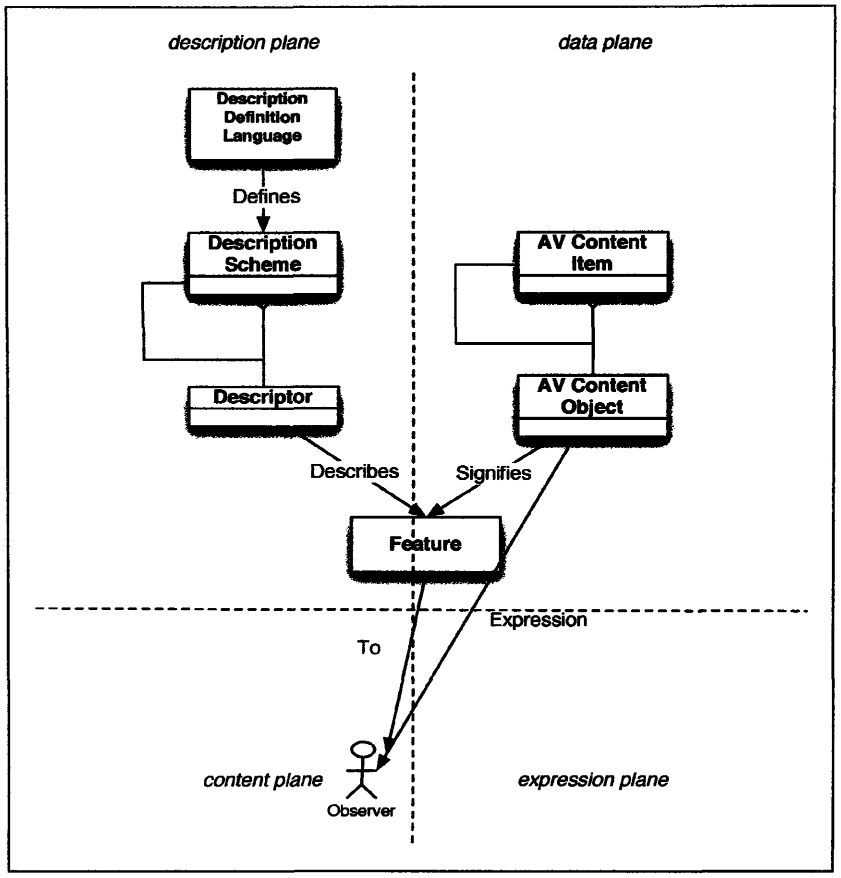


Figure 2.20: Revised Semiotic Model

2.8.2 Applying the semiotic model

The application of the semiotic model is illustrated in figure 2.21, where the diagram shows a single frame of a well-known moving picture sequence rocket.mpg. In the diagram the MPEG-1 compressed video data (compressed data plane) portion of the file is shown. This corresponds to the image data (data plane) shown. In turn the image rendered on screen from this data (expression plane) is shown and an outline conceptual description of the image (description plane) is given, the feature sets characterising the image segments are omitted.

2.8.3 The extended semiotic model

The extended semiotic model shown in figure 2.22 developed by the present author was introduced to clarify the distinction between data, compressed data, descrip-

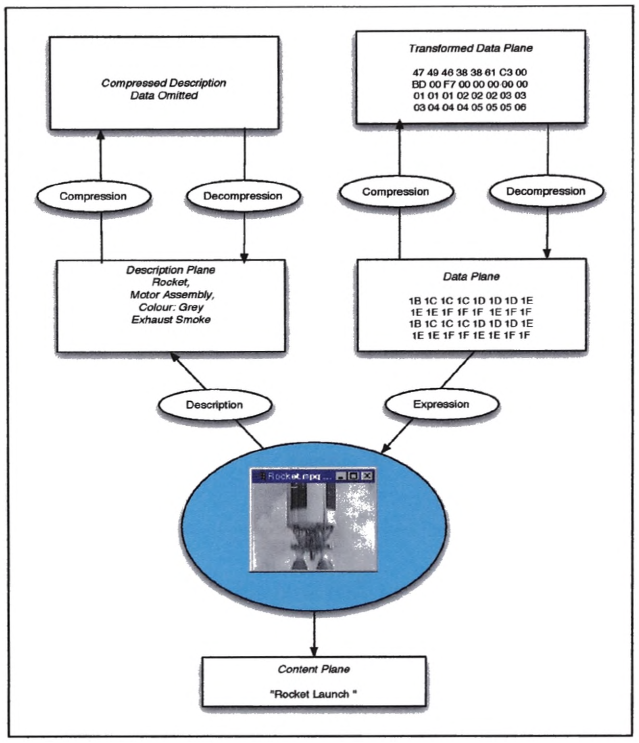


Figure 2.21: Application of the semiotic model

tions and compressed descriptions. When this version of the model was developed the choice of MPEG-7 DDL had not been made and the need for a binary interchange format for MPEG-7 descriptions had not fully emerged. The development of the MPEG-7 BiM [138] validates the view that binary descriptions would be necessary and other subsequent MPEG-7 developments support the utility of the model.

2.8.4 The R2 semiotic model

Further revisions of the semiotic model have been made to accommodate the need to distinguish; the feature sets used to *characterise* an object perceived by the viewer, the audio or visual object itself; the concept invoked in the viewers mind and in MPEG-7 terminology the descriptors and description schemes used to describe the object illustrated in figure 2.21. In fact the MPEG-7 terminology is completely discarded and a new terminology developed. The revised model in figure 2.20 fails to provide an effective mechanism to model the relationships be-

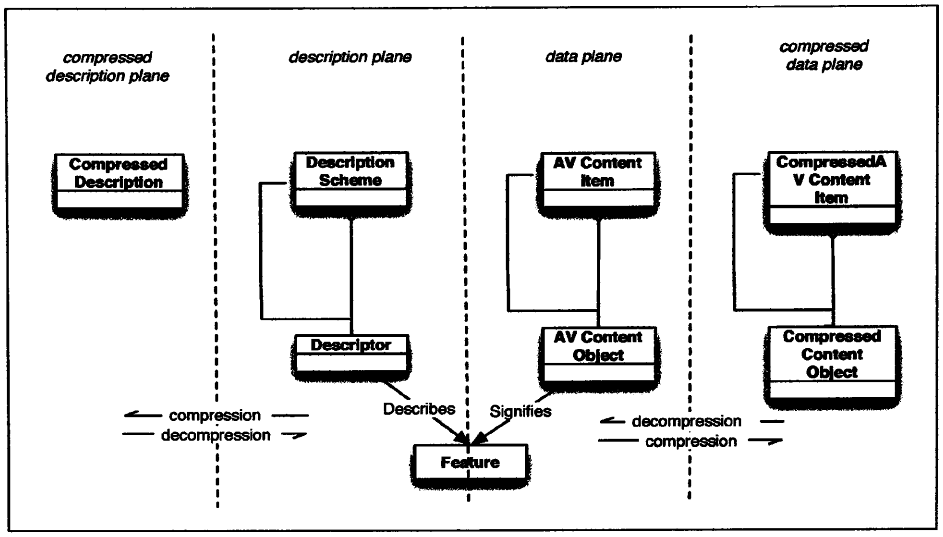


Figure 2.22: Extended semiotic model

tween; the audio visual content object, it's expression, conceptual invocation in the observers mind, description, characterisation by audio visual feature and the observer. Audio visual features are taken to be some numeric data set *characterising* the audio-visual content object at some level of decomposition of the audio-visual content item. This is in contrast to words or categories *describing* the audio-visual content object in any semantically meaningful sense. It has become apparent that to provide effective meaningful descriptions of audio-visual objects both audio-visual features *characterising* the audio-visual object and meaningful terms for *describing* the audio-visual object are needed. These different descriptive components are then *bound* together either by the human describer or by the description system. The concept of *binding* features together with semantic or meaningful descriptions needs some reinforcement since this will be treated as an *overt* step in the AVA modelling and architectural considerations that follow. This should be contrasted between the *implicit* binding between semantic descriptive categories that normally takes place in image processing and Bayesian recognition systems.

This leads to a further revision of the semiotic model, which is shown diagrammatically in figure 2.23. This model will be referred to as the R2 semiotic model in later discussions to distinguish it from the revised model. It will be noted that the implementation of the model described here is restricted to visual objects and that

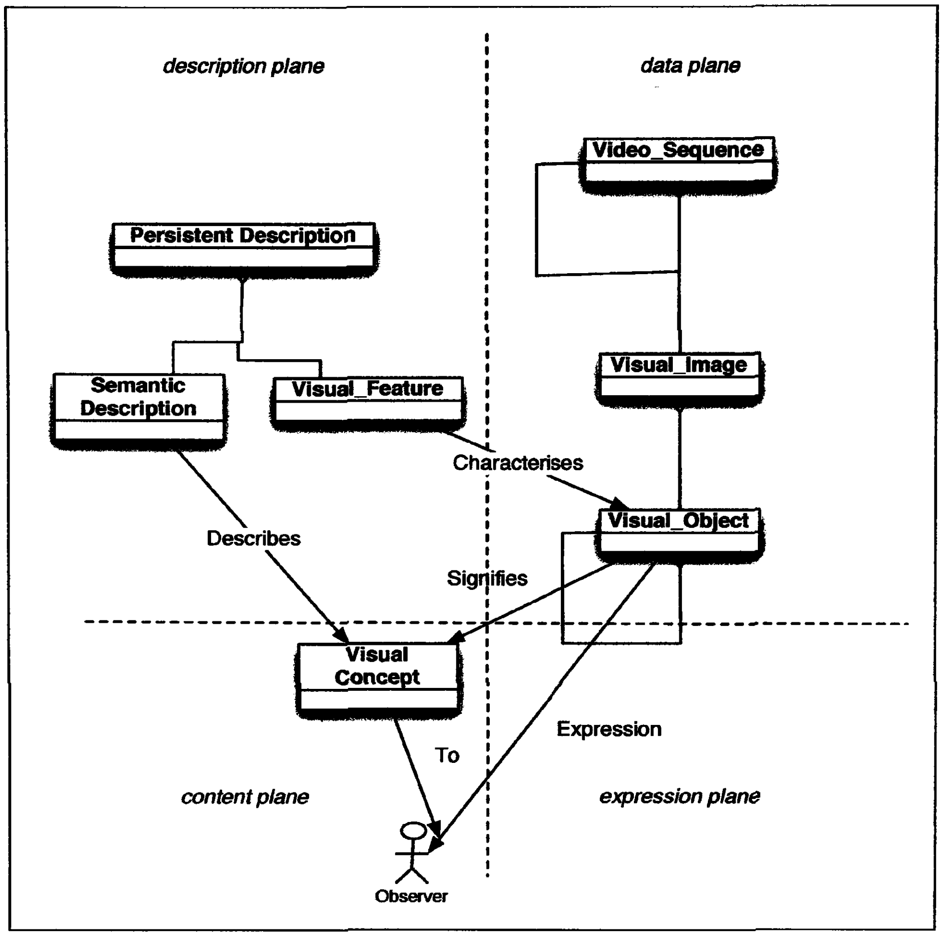


Figure 2.23: R2 semiotic model

it is necessarily limited by temporal extents. This limitation is readily exemplified by considering that in the soccer domain David Beckham sometimes wore a red shirt when playing for Manchester United and sometimes wore a white shirt when playing for England.

2.8.5 The classical content model

To develop the decomposition of moving picture content further the classical model from Aristotle [15] is formally considered. The classical model is chosen because of its simplicity, antiquity and because it forms the basis of most analyses of narrative. It will also be noted that the story plot terminology is now adopted. The classical model is recast into UML in figure 2.24 and the relationship between

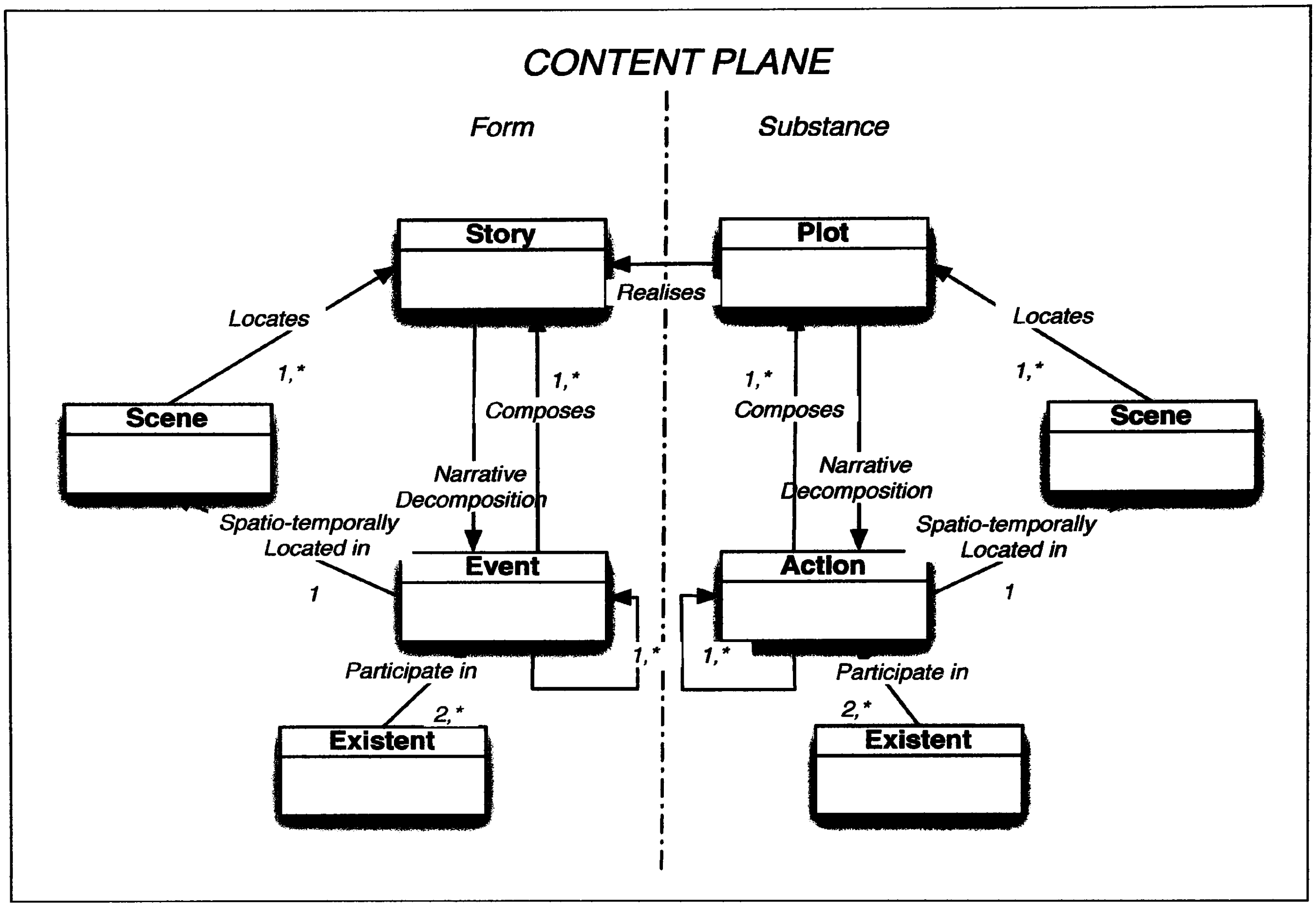


Figure 2.24: The classical content model recast in UML

the form and substance of content is considered. To simplify the diagram and discussion it is assumed here that no elision or temporal non-linearities occur and the term 'existents' is taken to mean all the objects, actors and setting components in Chatman's sense present in events.

It is in the interrelationships between plot, story, scene and event/action that the classical model and the CLORIS model diverge. If the distinction developed in the revised semiotic model between the data plane and the content plane is now applied to the substance of the narrative and its moving picture realisation, the UML diagram in figure 2.25 can be obtained. The term 'sequence' is omitted since it is generally used to mean a collection ≥ 1 frames that may, or may not, span shot, and optical boundaries. This model exposes some of the problems

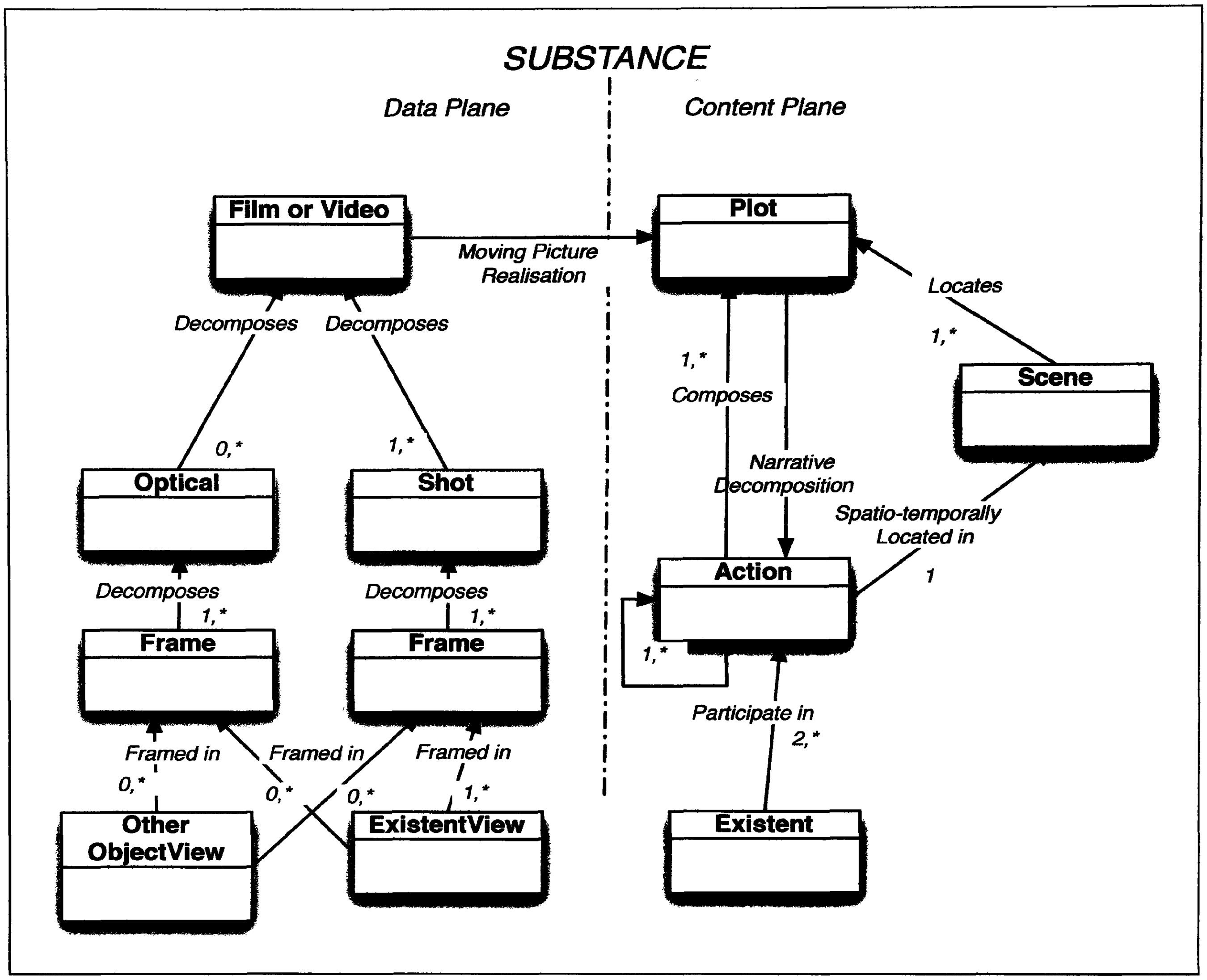


Figure 2.25: The substance of the content and data of narrative in UML

with the CLORIS model because of the assumed decompositional relationships between the sequence, scene and shot can no longer be seen to hold. This is because whilst the CLORIS model captures the containment relationships between events, it places the scene directly into the data plane decomposition hierarchy and creates a decompositional relationship between scenes and events (recall the account Mandler's analysis in section 2.3.5). This, in turn, invalidates the conceptual basis for the setting, for all its usefulness from a modelling perspective. The setting in the CLORIS model provided a "bridge" between the denotative aspects of what are here referred to as the expression plane and the description plane. In effect the implementation of the CLORIS demonstrator implemented a series of "binding" points between visual objects present in the micrometer film and their semantic descriptions. A diagram showing a possible approach to automating the task of instantiating setting like descriptions that contributed to this analysis is shown in figure 2.26. Since the model was discarded details of the diagram are omitted. Alternate bridging concepts are considered and compared for the AVA model in sections 2.9.6 to 2.9.7 and illustrated in figures 2.34, 2.36 and 2.35. The contribution the binding process makes to the modelling is developed further in figure 4.16 when the AVA methodology is described.

In diagram 2.25 film, shot and frame take their conventional meanings and an optical is taken to be an optical effect in Arijon's sense. An Existent view is a view on an existent found within a frame. The term Other object view is used to define "non-existent" objects in view i.e. views on things that do not exist in the plot such as titling or masks. The remaining terms in the content plane; plot, scene action and existent retain the meanings developed in the analysis of narrative.

2.8.5.1 Form and substance in the description plane

To develop the AVA-content model further it is necessary to consider the form and substance of the description plane. In effect, this means taking what is the form and substance of content from figure 2.24 and identifying what is the form and substance of the corresponding description plane entities. This will be done by initially taking the relatively straightforward uppermost and lowest levels of the model. So the story:plot and the existent-view:existent form and substance relationships will be considered in the content and description planes. The story:plot relationships are shown in figure 2.27. The important concept that this diagram captures is that it is not merely a description of the content plot that is being considered but also the content's expression during the film or video showing. It is necessary then, to consider the data and expression planes at this level as in figure 2.28 effectively this is the narrative equivalent of the revised semiotic model in figure 2.20.

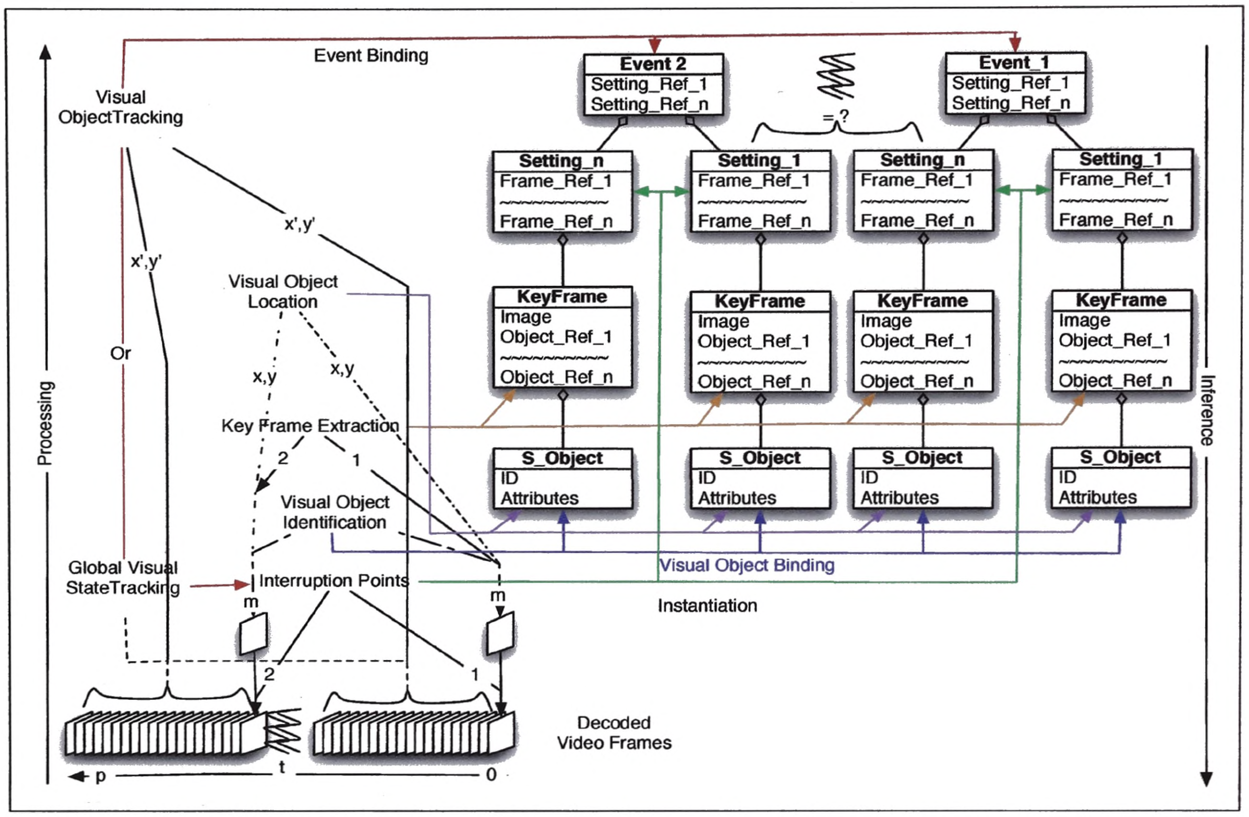


Figure 2.26: Visual to semantic class binding points in CLORIS like model

2.8.6 Description of existents

Some more terminology must be introduced at the lowest temporal decomposition level i.e. the video frame, to relate content plane existents to their corresponding description plane entities.

1. Existent

   The *existent* is defined as an object in the *content* plane *c.f.* Chatman.

2. Existent description

   The *existent description* is defined as the entity corresponding to the existent in the *description* plane. It also corresponds to the *framed existent view* of the same *framed existent* in the *expression plane.*

3. Framed existent

   The *framed existent* is the *data plane* entity corresponding to the *existent* in

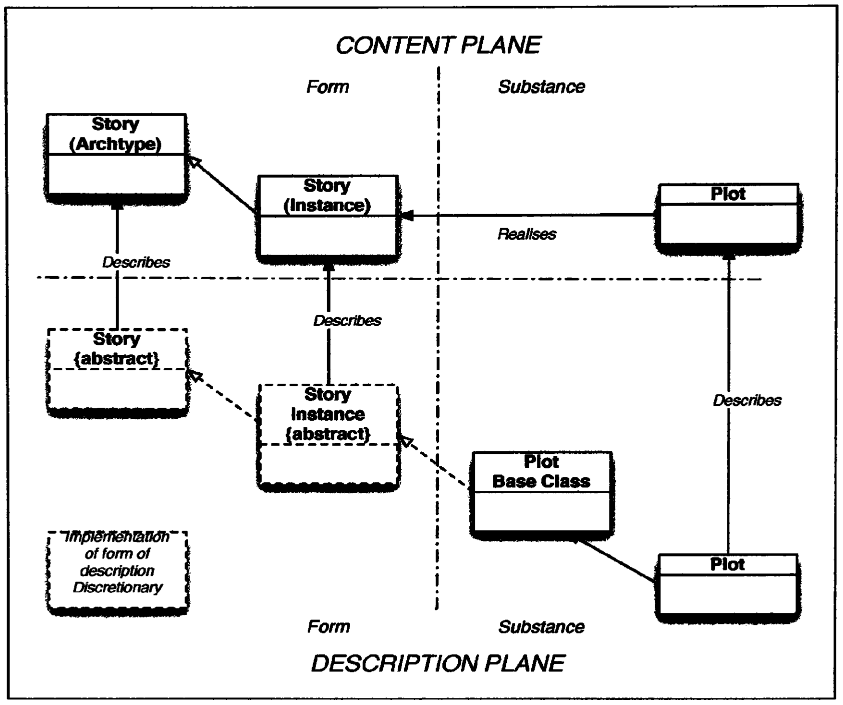


Figure 2.27: The content and description of story and plot in UML

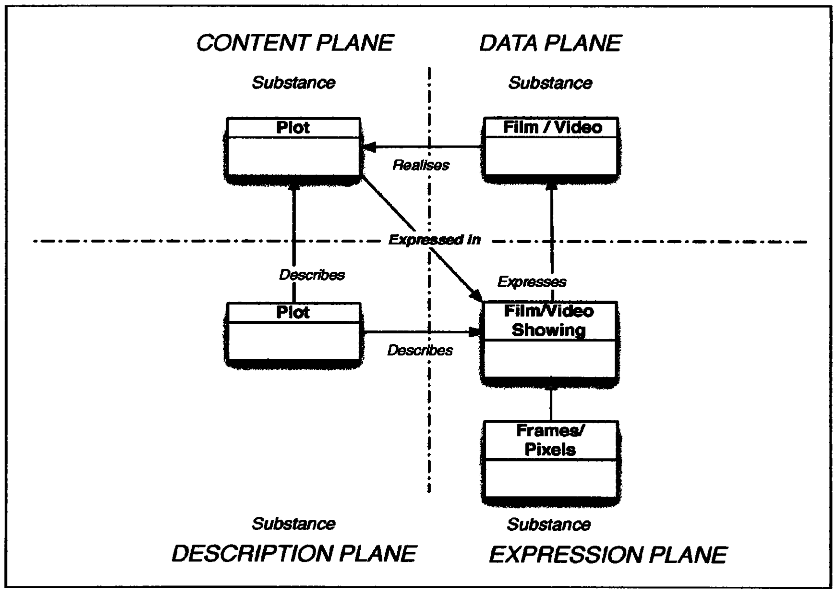


Figure 2.28: The substance of the content, data, description and expression planes of story-fabula and plot-syuzhet in UML

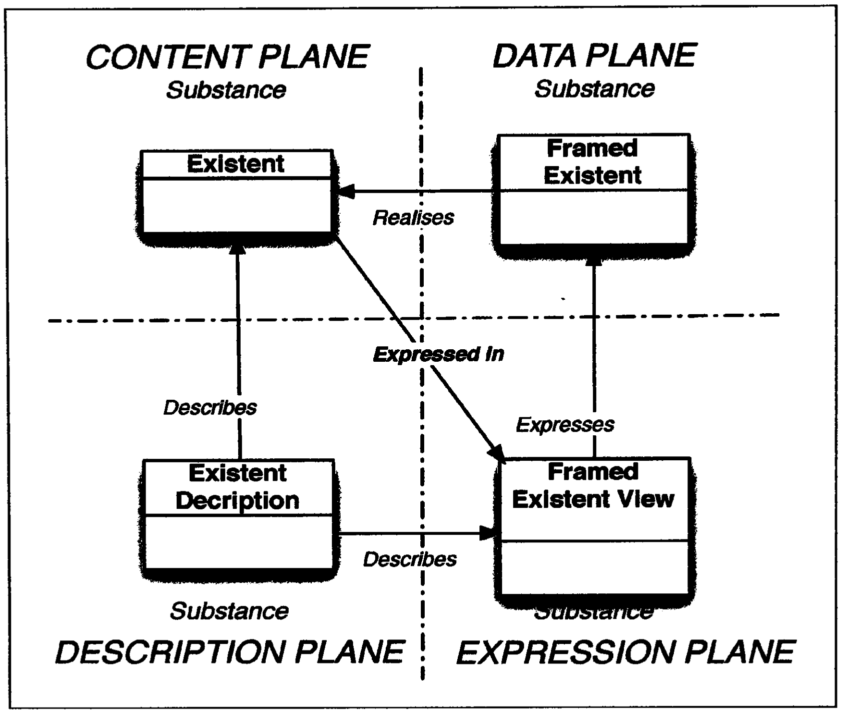


Figure 2.29: The substance of the content, data, description and expression planes of existents in UML

one video frame in the *data* plane.

4. Framed existent view

   The *framed existent view* is the *expression* plane entity expressing the *content* plane *existent*.

So now, considering the lowest level of the classical model in the same four planes produces the diagram in figure 2.29 where the framed existent view in the expression plane is equivalent to the visual object of the revised and R2 semiotic models. This alternate rendering of these models completes the preliminaries needed to build the AVA content model.

2.9 Structure of the AVA content model

The possibility of relating an existent view with Pasolini's [271] cineme or Eco's semes and Schank's [238] picture producer has already been identified. These two

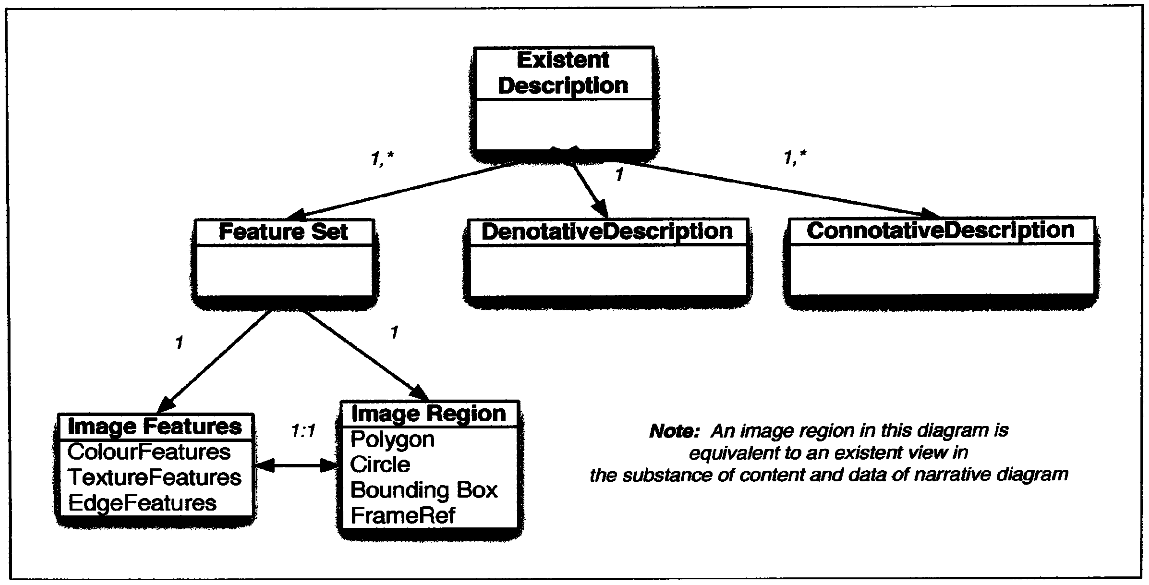


Figure 2.30: The AVA description of existents in UML

concepts potentially provide a conceptual framework to link a denotative description of a visual object with a single existent view. It is necessary to allow multiple views on the same object to be linked to the same denotative description. It is also desirable that connotative descriptions can be added to the description as well. This leads to the diagram in figure 2.30 which serves as a base class definition for the description of the existent allowing further specialisation into object, actors and setting[5] and their component relationships.

2.9.1 Description to image feature and region binding

The concept of "binding" between image feature sets and a denotative description has already been introduced to create the **bound** *existent description*. This concept is extended here to allow the image region that corresponds to the feature set to also bind to the description. This results in a 1 to 1 relationship between each image-region:feature-set tuple and a many to one relationship between the feature-set:image-region tuples and the corresponding denotative descriptions. Existent descriptions are allowed to exist without corresponding image information

[5]Setting is now used in Chatman's [54] sense not Parkes' [213]

in which case they will be referred to as **unbound**.

2.9.2 Existent relationships

The CLORIS model identified the need for relationship identification and representation between what are now termed existents in the AVA model. The minimum set of relationships that are needed at this stage are set membership and part-of relationships. Is-a or set membership can be provided through object orientated programming paradigms and modelled through the use of UML inheritance. It is assumed therefore that an existent definition in a domain schema can inherit from a super class defining overall qualities i.e Twinky is an instance of a teletubby with an attribute of having a "favourite thing". The attribute "favourite thing" would be a public data member of the teletubby class.

Ideally, the model should be flexible enough to allow new relationships to be defined in an ordered manner. In addition the system architecture should facilitate implementation of multiple relationships. A primitive relationship class definition can implement this functionality and by allowing all existent object classes across the hierarchies to inherit from it, the capability can be propagated to all instances as required. The relationship class needs to define methods to create relationship types and a corresponding relationship container that is initially empty. Whilst constraining the relationship types available the class allows controlled extension through the usual object orientated mechanisms. This approach allows class introspection methods that iterate over the relationship object instances, to traverse what are in effect relationship graphs. The basis for implementation of set membership and part-of relationships, as conceptual graphs after Sowa [267], comparable to the approach adopted in the CLORIS model is therefore provided. Also existents can now be decomposed into constituent parts which means that some of the image-feature: image-region tuples may refer to both the main part and the component and others only to the main part. Here the level of decomposition is dependent on the application and therefore at the discretion of the describer. It is sufficient that the content model can accommodate this decomposition. This leads to a refinement of existent description modelling shown in figure 2.31. The issue of

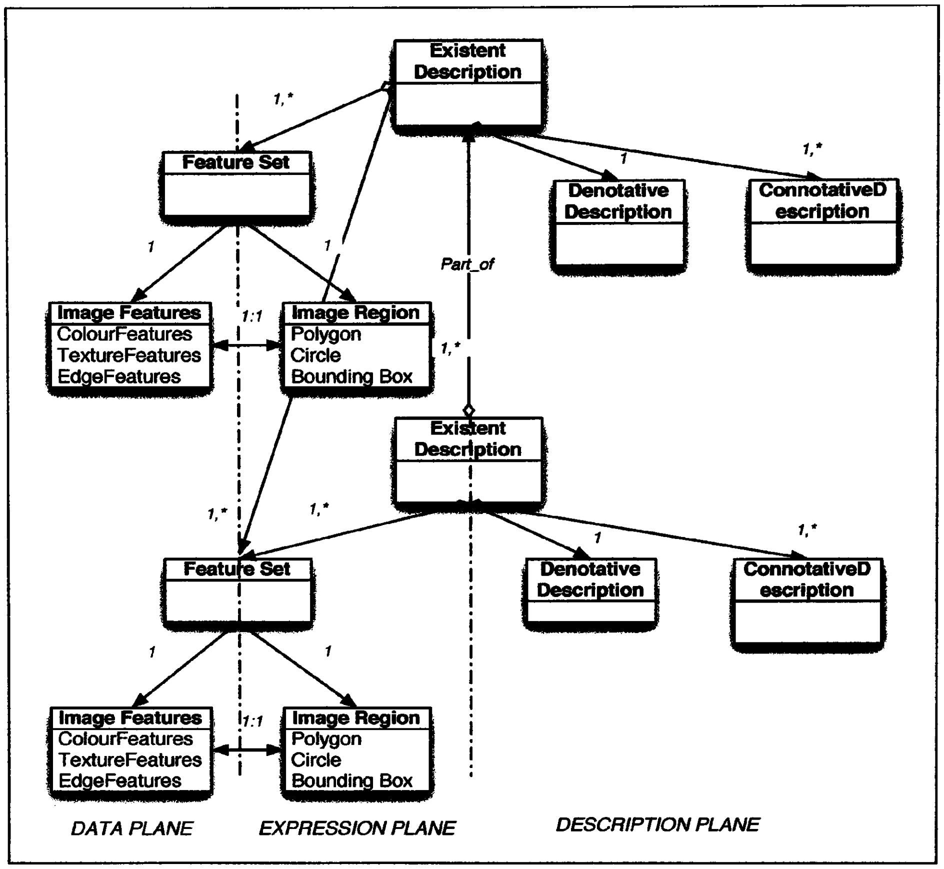


Figure 2.31: AVA existent relationships in UML

whether the sub-part feature set to super-part existent binding should be achieved explicitly or inferentially is left to further work described in the conclusions 5.4.3.1.

2.9.3 The existent context

Having identified that a bound-existent description is one where a denotative description has been bound to a feature set, the existent context can be restated as being a sequence of frames over which a binding between feature set and denotative description holds true for *some* pair of visual-object identification and recognition algorithms. In this definition, image region transformations are allowed within limits determined by the recognition algorithm which provides the basis of the AVA annotation methodology described later. This suggests adopting the condensation approach of Blake and Isard [34] for the task of tracking object

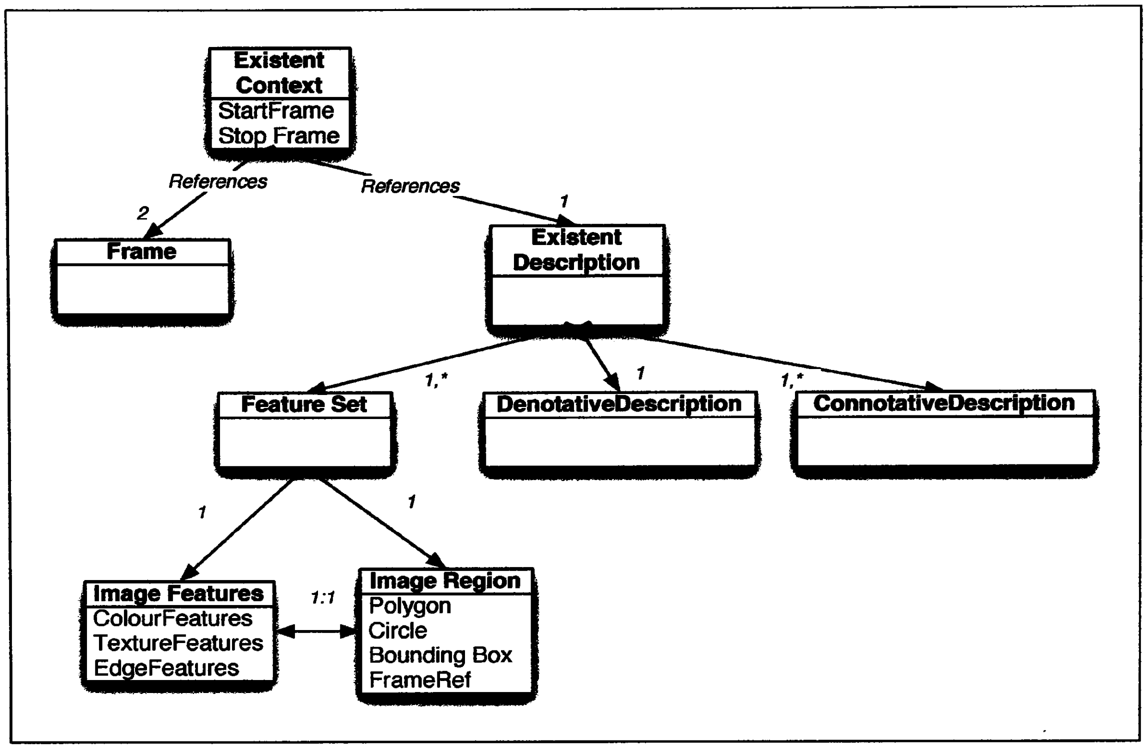


Figure 2.32: The existent context in UML

deformations for the duration of the existent context. Extension of the current work in this direction is considered in section 5.4.3.3. This emphasises both the failure led nature of the binding process and that initial binding occurs at a single frame, corresponding to the interruption concept from the CLORIS model and the AVA methodology. As such an existent context is equivalent to a proposition, as its name suggests of *existence* in the CLORIS model setting construct. The frame at which recognition occurs can be also referenced in the existent description. The UML representation of the existent context is shown in figure 2.32.

2.9.4 Spatial and temporal decomposition

The conceptual difficulty that arises at this point in developing the modelling hierarchy in most other approaches stems from the intertwining of the different spatial and temporal decompositions present in data and content planes shown in figure 2.25. Here the temporal terminology introduced earlier in this chapter can be usefully applied to disentangle the hierarchies. From the spatio-temporal decomposition point of view *content* the data plane is a container for expression time temporal relationships[6]. The content plane is clearly where the content lies, so there are two distinct decompositions which are, in some sense orthogonal to one another. It should be noted that any moving picture portrayal of an object inherently results in time elapsing at expression time, irrespective of whether the sequence portrays a stasis or a process statement c.f. section 2.3.3. Furthermore it will be recalled from section 2.3.6 that any expression time moving picture object portrayal implies that the object is located in a scene however rudimentary. So, entities comparable to the CLORIS setting must be introduced. These are needed to provide the description plane bridge between the data, expression and content planes, for the intermediate levels of the model which is undertaken in the next section. The intermediate levels are those between the existent description and the overall plot description.

2.9.5 Existent context set

In any but the most sparse animation such as Chuck Jones' animation The Dot and the Line (1965) adapted from the book [151] of the same title, or matte sequences there will more than one object expressed. This means that a descriptive container must be identified for sequences with more than one visible object. The existent context set is introduced to provide this functionality and it's UML representation is shown in figure 2.33. The existent context set is defined as a sequence of frames in which a number ≥ 2 existents are expressed. This entity also has the capability of being bound to temporal video features used to detect the existent context set boundaries when these have been identified.

2.9.6 A replacement for the setting

Conceptually, it is at this level that entities that fulfil the same roles as the setting and master setting are required. Adoption of the term setting is deprecated because

[6]This ignores the temporal relationships imposed by the data file or stream syntax discussed later

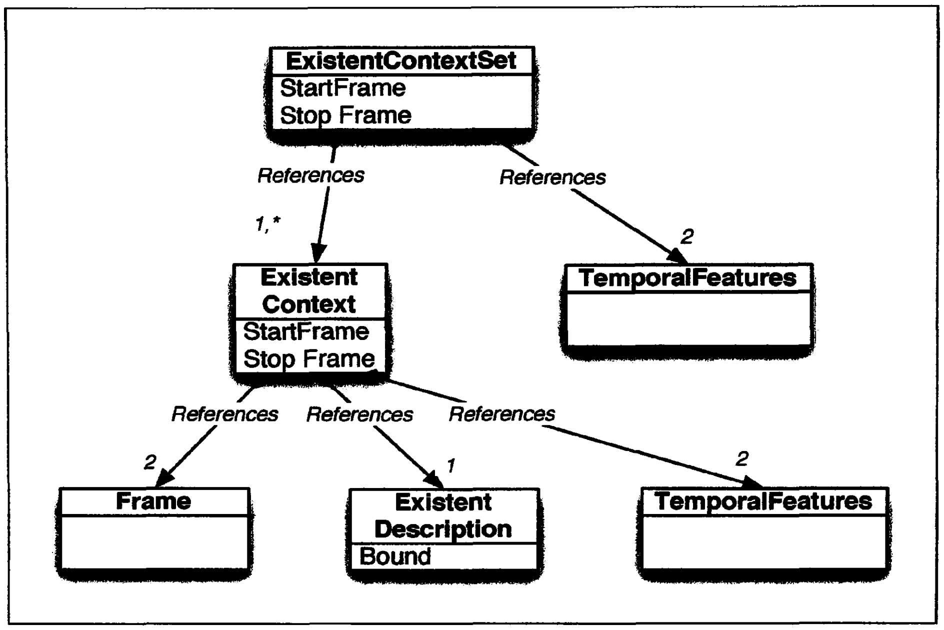


Figure 2.33: The existent context set in UML

of its earlier use and strict definition given within the CLORIS model, the term's overloaded meaning from other narrative domains and its use by other writers. In the AVA model there will be a deliberate mapping between *expression* time and content or *narrative* time. Also inferential relationships between existent contexts will be allowed within the existent context through mechanism such as Parkes' [215] transitivity tables for pans and zooms etc.. A "straw-man" entity for modelling at this level the *narrative context* is now introduced, and a UML representation provided in figure 2.34. Also the distinction between actor, object and setting [7] existents become important, since above this level the description of actions in scenes begins. So a narrative context is considered to be a container for existent contexts. Not all existent contexts need to be actually expressed during the duration of the narrative context. A narrative context will have a stop and start frame number and it may have narrative stop and start times attached but this will be describer-determined. The narrative context in the model provides the start and end points for actions and scenes.

[7] Again setting is used here in Chatman's sense.

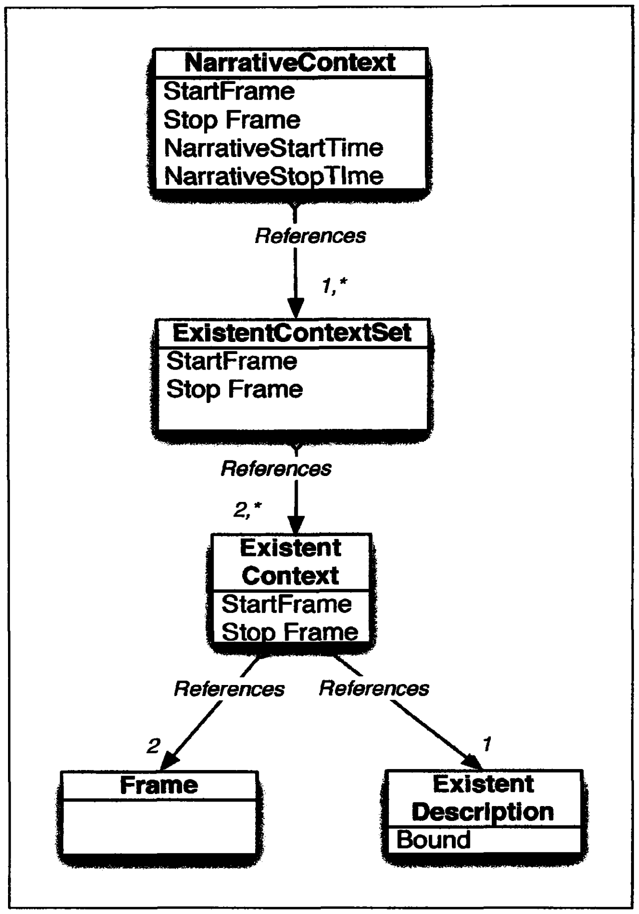


Figure 2.34: The narrative context in UML

2.9.7 Representing actions in scenes

In figure 2.25 the relationship between the substance of content and data was shown. This can now be augmented with elements from figure 2.33 and the narrative context set from figure 2.34 is introduced to bridge the content and data planes then figure 2.36 is obtained. This piece of modelling is compelling but it fails to capture all the subtleties of the spatio-temporal relationships between the entities at this level. It is also desirable that the concept of the scene as a spatio-temporal container for actions and existents is retained and that existents can now be differentiated into setting (Chatman's sense), actor and object existents. There is also an important conceptual deficiency in this model, illustrated with figure 2.35 which arises when trying to differentiate between narrative contexts that reference only sequences containing setting or actor/object existent contexts. This means that the narrative context in its defintion in figures 2.36 and 2.35, has much the same flaws that were earlier attributed to Parkes' setting in section 2.8.5 and section 2.8.5 This becomes particularly apparent when the overall plot-syuzhet description is also introduced into the modelling. It should also be possible to define a stereotypical event independently of an instance and independently of its moving picture realisation. The narrative context concept is rejected and a script based representation of actions is introduced.

2.10 Applying Schank's conceptual dependency theory

To this end an analysis of the applicability of Schank's conceptual dependency (CD) theory is carried out to establish which aspects of the CD are applicable to automating moving picture annotation. The CD defines a set of conceptual categories defined as: Picture Producer (PP), real world actions (ACT), Picture Aider or attributes (PA), Action Aider (AA) again effectively action attributes, Times (T) and Locations (LOC). Several primitive actions and a set of primitive relationships are defined over the actions, objects, locations and times. Parkes [212] shows the applicability of the Schank's script metaphor to moving picture annotation although he adopts Sowa's [267] [265] notation. The script is essentially an aggregate of CDs. The evolution of scripts to thematic organisation packets (TOPs) and memory organisation packets (MOPs) and story skeletons is described in appendix C. It will be sufficient here to indicate how the script formalism can be integrated into the content model. Typically, sequences from a moving picture content item will represent a single track from a script, unless the content has been specifically developed for an educational or ludic application. It will typically also be heavily elided. These elisions are those where the content is completed by inference based on the observers narrative world knowledge.

2.10.1 Notation

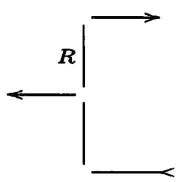
Schank developed his conceptual dependency theory as an aid to natural language understanding. In CD a diagrammatic notation is developed with the following conventions:

$\leftrightarrow$ denotes the relation between actor and action

$\leftarrow$ O denotes the relation between action and object

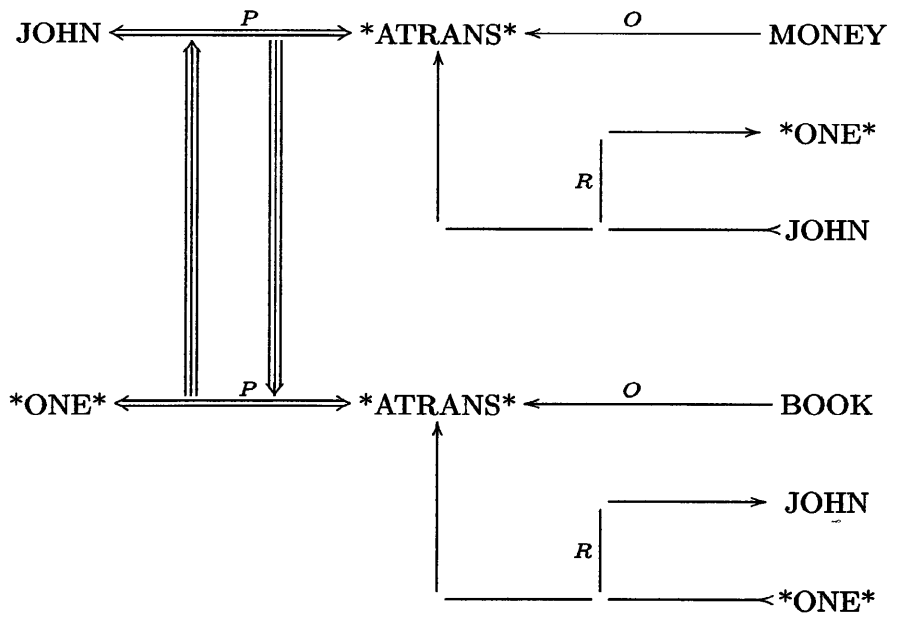
$\Longleftarrow$

denotes causality dependence, and:



denotes the relation between action, object, recipient and donor

This notation was used by a natural language understanding program to unpack the knowledge inferred in a sentence such as:
"John bought a book"
which results in the production of the following diagram:



2.10.2 Conceptual categories in conceptual dependency

In his analysis of linguistic inference in this and other work Schank and his collaborators define six conceptual categories:

1. PP picture producer; i.e. real world objects,
2. ACT real world actions,
3. PA picture aider; attributes of objects,
4. AA action aider; attributes of actions,
5. T Times,
6. LOC locations.

2.10.2.1 PP picture producer

It should be noted that Schank most usefully from the perspective of this and subsequent analysis uses PP as an acronym for **picture producer**. It is this observation that is critical to subsequent discussions of inference in relation to moving picture content. From the perspective of moving picture understanding we are confronted with the task of *infering* the meaning of the pictures in front of the observer. The challenge is that this meaning relies on (*at least*) as much real world knowledge as is involved in the analysis of the simple sentence shown above. It is not claimed that moving picture content can be automatically understood. This notation does provide a systematic approach that can be adapted to represent moving picture content at the level of primitive events. These primitive events can be composed into larger narrative structures through reference to Schanks' scripts, the revisions to Parkes' setting formalism resulting in the definition of the visual context and the visual-object feature-set binding process that are developed further in chapter 4. For the time being it will be assumed that a Schankian picture producer can be interpreted as a visual object in a segment of moving picture content.

The picture producer concept also provides a point where the considerations of film language and in particular Passolini's question about the existence of a *cineme* and Eco's analysis of the seme as the basis of any putative film language can be considered from a machine processing perspective.

2.10.2.2 PA picture aider

A key observation here is that objects in the real world are regarded as picture producers not as attributes of a picture producer. So that a PA is, as one would expect from the natural language understanding origins of the technique, generally equivalent to an adjective. This means that it is possible to separate consideration of the objects of interest in a visual scene and the qualities that describe them. Therefore returning to the R2 semiotic model the image features that characterise a visual object can be considered as picture aiders. This is extremely useful as other pieces of the CD notation denote ownership, movement etc.

2.10.2.3 ACT action

The primitive real world actions are divided into 12 category names that were sufficient to enable Schank and his collaborators to represent the action part of a large number of natural language sentences. These will be enumerated and discussed in the context of what can be seen in a moving picture sequence.

1. ATRANS: abstract relationship transfer i.e. possession, ownership or control.
2. PTRANS: physical control transfer over an object.
3. PROPEL: physical force applied to an object.
4. MOVE: movement of an animal body part.
5. GRASP: grasping of an object by an actor.
6. INGEST: taking into the body of an object by an animal.
7. EXPEL: expulsion from the body of an object by an animal.

8. MTRANS: mental information transfer between or within an animal [8].

9. CONC: conceptualising or thinking about an idea by an animal.

10. MBUILD: construction of new information from old by an animal.

11. ATTEND: directing a sense organ towards an object.

12. SPEAK: producing sounds from the mouth.

2.10.2.4 CD actions applied to annotation

PROPEL. MOVE and GRASP are directly observable from moving picture sequences although in some sequences they are likely to be synchronous to some extent dependent on the domain. In the teletubbies sequence MOVE, GRASP and PROPEL are synchronous as the ball is picked up and a GRASP continues as the ball is held and this is followed by further MOVE, GRASP, PROPEL conjunctions. In some sports i.e. soccer MOVE and PROPEL conjunctions predominate and others i.e. cricket there are MOVE, GRASP and PROPEL conjunctions.

CONC, ATTEND and SPEAK can also be conveniently considered through their synchronicity in visual sequences. CONC and ATTEND are potentially unresolvable from a purely visual perspective. SPEAK is available if the camera is directed towards the speaker since if the camera is directed in this direction lip-movement can be detected assuming the speaker has lips. It is here that the results of Li and Luo are relevant [172] relying on combined approaches to audio and visual segmentation are applicable in the recognition of multiple speaker intercut sequences.

MTRANS ATRANS and MBUILD can be eliminated from consideration since it will be apparent that generally these 3 actions are not immediately observable from a moving picture sequence. Whilst an MTRANS or MBUILD might be represented in a documentary about cognitive science or education theory, a specialist ontology

[8] Schank [244] here refers to a 3 part cognitive model: CP conscious processing, LTM long term memory and sense organs. Later work [240] and [242] modifies and develops this model with reference to memory organisation packets MOPs and thematic organisation packets TOPs.

would be built to represent this knowledge and some *directly* visual representation would be provided for the viewer when direct representation of these concepts were necessary. Similar observations about ATRANS in the context of company ownership in financial news coverage can be made.

INGEST and EXPEL are potentially available through direct observation of visual information: INGEST from the co-location in the frame of objects over time and EXPEL similarly. It is probable *thankfully* that EXPEL will be of limited application outside the wildlife domain hence it was not considered in the current implementation.

Currently there are limits on what can be directly determined from the visual track, however the content describer may wish to have a full representational capability available in which case *all* of these primitives need to be supported in the application. There is also clearly scope to map between natural language terms using Wordnet [89].

2.10.2.5 Action aider AA

Action aiders denote attributes of actions in conceptual dependency such as force; they are considered to be equivalent to adverbs in natural language. This opens up the feasibility of considering action aiders in relationship to motion features extracted from the video.

2.10.2.6 Temporal and other relationships in conceptual dependency

In this notation temporal relationships are considered to be equivalent to linguistic tenses and as such are modifications of the main link between actor and action ($\Leftrightarrow$), or a link between an object and it's state. Due to this relationship between link modifiers and natural language tenses, link modifiers are concerned with denoting conceptual interrelationships other than the purely temporal ($\Longleftrightarrow$). The link modifiers Schank defines in [244] are:

1. past
2. present
3. future
4. ts=x, begin a relation at time x
5. tf=x end a relation at time x
6. conditional
7. negation
8. question

2.10.2.7 CD temporal relationships applied to annotation

It will be noted that 1 to 5 above are temporal and can readily be mapped onto a graphical representation such as Allen's discussed in section 2.7.6 It has already been noted that there is a need to distinguish between several timelines in the representation of moving picture content. There are two other key observations that can be drawn from these tense relationships; the first is that there is a need to represent temporal relationships other than a way that strictly adheres to "wall clock time" although narrative world "wall clock time" may play a role such as in the film High Noon (1952). The second is that there is need to represent non-temporal event interrelationships dynamically. This is because the causal relationships portrayed at one point in the expression of a moving picture sequence may be changed later for dramatic effect. Rebecca (1940) is film that shows this well.

Only the first of these points will be considered now, but it is anticipated that a variant of the approach discussed here will provide a solution. It has already been noted that the provision of multiple "time line" representations of content can resolve these temporal relationship issues. Effectively this maps between interval $t_1^e, \ldots, t_n^e$ and $t_\alpha^n, \ldots, t_\nu^n$ where e and n are expression and narrative times respectively, $1 \ldots n$ the expression time interval and $\alpha \ldots \nu$ the narrative time interval. Time line representations are a familiar feature of the interfaces to video non linear editors such as Final Cut Pro [281] and project planning software such as Microsoft Project [226]. So whilst this feature has not been implemented in the current interface it is assumed that temporal relationship *representation* but not *description* is merely an implementation issue.

The other non temporal relationships remaining in CD are not considered on this occasion hence they represent work for the future.

2.10.2.8 LOC location in CD

Location in CD is the location noun in the sentence that the dependency describes. It is unlikely that this can be inferred directly from the visual features in a moving picture sequence unless a textual location specifier is given, such as "San Francisco" or the location spoken by an actor if the audio track is analysed in a multi-modal environment. Many cities have iconic visual features such as San Francisco's bridge. Inevitably though, the mapping of location name to visual scene must initially be a manual one. The relationship between location in CD, the visual scene and visual context will be returned to in chapter 5.

2.10.2.9 Conceptual syntax rules

Schank then defines 16 conceptual syntax in [238] rules 14 in [239] and reiterates 9 in [244] The treatment in [239] is outlined here since the representation of events and their temporality in software is of concern here. Note in Schank [239] rules 12 and rule 13 are classed as variants.

$PP \Longleftrightarrow ACT$

1. indicates that an actor performs a certain act.

$PP \Longleftrightarrow PA$

2. indicates that an object has a given set of attributes.

$PP \Longleftrightarrow PP$

3. indicates set membership between two PPs.

$PP \longleftarrow PP$

4. indicates reference to a PP or its attribute previously defined.

$PP1 \Longleftarrow PP2$

5. indicates that PP2 is a part of *or* possessor of PP1 *or* PP1 located in PP2.

$ACT \xleftarrow{O} PP$

6. indicates the object of an action.

$$ACT \xleftarrow{\quad} \overset{R}{\left| \begin{array}{ll} \longrightarrow PP & \text{indicates the recipient of an object} \\ & \\ \longleftarrow PP & \text{indicates the donor of an object} \end{array} \right.}$$

7. within this action.

$$ACT \xleftarrow{\quad} \overset{D}{\left| \begin{array}{l} \longrightarrow PP \\ \\ \longleftarrow PP \end{array} \right.}$$

8. indicates the direction of an object within an action.

$A \xleftarrow{I} \Updownarrow$

9. indicates the instrumental conceptualisation of an action.

$$\begin{array}{c} X \\ \Uparrow \\ Y \end{array}$$

10. indicates X caused conceptualisation Y when written with a c this denotes that X could cause Y.

$$\begin{tikzcd}
& & PA2 \\
PP & {} \arrow[l] \arrow[ur, "R"] & \\
& & PA1 \arrow[ul]
\end{tikzcd}$$

11. indicates a state change of an object.

$$\begin{tikzcd}
\text{PP1} & {} \arrow[l, Leftarrow] \arrow[r, Rightarrow] & \text{ACT1} \\
\text{PP2} & {} \arrow[l, Leftarrow] \arrow[r, Rightarrow] \arrow[u, Rrightarrow] & \text{ACT2}
\end{tikzcd}$$

12. indicates PP2 ⇔ ACT2 is caused by PP1 ⇔ ACT1

$$\begin{tikzcd}
\text{PP1} & {} \arrow[l, Leftarrow] \arrow[r, Rightarrow] & \text{ACT1} & \\
& & & \text{PA2} \\
\text{PP2} & {} \arrow[uu, Rrightarrow] & {} \arrow[ll, Leftarrow] \arrow[ur] & \\
& & & \text{PA1} \arrow[ul]
\end{tikzcd}$$

13. indicates PP1 ⇔ ACT1 causes a change in PP2

$$\begin{tikzcd}
& T \arrow[d] & \\
{} & {} & {} \arrow[ll, Leftrightarrow]
\end{tikzcd}$$

14. shows the temporality of one conceptualisation w.r.t. another i.e. tomorrow.

$$\begin{tikzcd}
\text{PP1} & {} \arrow[l, Leftarrow] \arrow[r, Rightarrow] \arrow[d, "T"'] & \text{ACT1} \\
\text{PP2} & {} \arrow[l, Leftarrow] \arrow[r, Rightarrow] & \text{ACT2}
\end{tikzcd}$$

15. indicates one conceptualisation can be the time of another i.e. *while.*

$$\begin{tikzcd} & PP \arrow[d, Rightarrow] & \\ {} \arrow[rr, Leftrightarrow] & {} & {} \end{tikzcd}$$

16. indicates the conceptualisation must occur in some location.

2.10.2.10 CD syntax rules applied to annotation

Rules 1, 6, 4 and 8 define: an action, the object of an action, an object donor and recipient and the direction of an object respectively. These are all appropriate for direct incorporation into the model and potentially can be directly inferred from the visual track of moving picture content within the restricted set of ACTs discussed above. There is a need to maintain knowledge about object and actor state over time and allow changes in the underlying knowledge base to occur.

Rule 4 involves attribute reference, visual references are commonplace in moving picture content. Whether a statement is in fact a visual reference or restatement however is a matter of content interpretation. Whether references can be discriminated automatically from restatements might potentially be determined from cutting patterns. It is desirable in either case that the content model should support a relationship representation. Rule 11 is directly applicable to content from three perspectives. The first is that the knowledge base must be able to support changes in the visual attributes or features bound to a *visual object* at multiple scales. The second is that the annotation system must be able to distinguish between changes that occur due to scale and viewpoint changes and those that result from ACTS. The third is that not all of these attributes will be known at the point an object instance is initially created. Object or actor attributes can be implemented and changed through normal attribute set and get methods in an object orientated environment. The need to associate multiple visual feature sets with a single object or actor suggests a referencing based approach.

Rule 3 clearly needs to be represented and implemented. This is the workaday stuff of object orientated programming. It is noteworthy that even simple narrative is likely to result in logical ambiguity particularly in the humour domain. Sowa's conceptual graph theory can be adopted to represent class membership is-a, part-of relationships.

Rule 5 is concerned with the location of events, objects and actions. Here Schank typically refers to locations like New York or Home which have echoes of traditional stage direction or film direction terminology such as:

> **INT-HANGAR-DAY**
> We are still moving through light planes, but now we are inside the hangar. Some of the planes have their engine covers open , parts strewn around. Others are partially covered with tarps or have sections missing. There is even a sleek executive jet parked in one corner.
>
> As we float past the planes we notice a woman leaning against the wing of a Piper Cub, her chest against the wing's trailing edge, her arms spread out to each side, as though flying herself. Cronenberg [62] (p.3), see also Crash (1996)

The observer of a moving picture sequence clearly has a concept of "here and now" at the present location c $L_c^{t=e}$, at a narrative time t_c^n, for a given expression time point t_c^e . To develop a knowledge base about the content there is a need to maintain knowledge about locations objects and events at those locations and the temporal relationships.

Rules 9, 10, and 14 to 16 are perhaps the easiest to consider from the perspective of automated annotation. Whilst there is a need to represent these conceptual dependencies in the content model it is well beyond the current scope to consider how inferences could be made from purely visual information about these conceptual dependencies. Alternate link designation in the time line diagram outlined above would serve to adequately represent these relationships.

Rules 12 and 13 depict causal dependency relationships between conceptualisations so for the present these need only to be considered from a representational point of view.

2.11 Completing the AVA model

This is an appropriate point in the modelling to reconsider Chatman's [54] event blocks and his distinction between kernel, satellite and spur events. These distinctions are also introduced into the model illustrated in figure 2.37. A departure is now made from the classical model of narrative. The introduction of the Scene Description concept does allow a classical interpretation of the scene to be retained typified by:

> "**SCENE:** The camp of the Greeks before Troy; in the background, the smoking ruins of the city. At the entrance of one of the tents Hecuba is stretched on the ground. " Euripides in Hadas [88] (p 257)

These relationships effectively capture the propositional nature suggested by Garnham described in section C and the containment relationships identified by Mandler [186] discussed in section 2.3.5.

A complete illustration of the model is provided on the end papers.

2.12 Conclusions

This chapter has considered the spatio-temporal decomposition of video from several viewpoints, semiotic, narrative film theoretic, and descriptive. The lengthy development of the background material provides the conceptual framework for the development of the AVA content mode. Some conclusions, in the context of the current work, on the film as language debate will first be made. This is followed by some overall conclusions about the applicability of the terminology.

Buckland [46] provides a review of film semiotics and in particular how Metz's position developed from [194] to [195] in response to Eco's work on cinema [79]. This is in effect Metz changing his position developed in [194] that there is no minimal unit of film articulation. It will be seen that from a representational view-point, identifying the minimal unit of cinematic meaning with a view on a Schankian picture producer, can provide the minimal unit of meaningful articulation in the analyses of Eco and Passolini. It was well observed by Parkes [213] that from his point of view it was irrelevant whether film represented a language or not so long as it was possible to represent the content in terms of a description language. This point has been amply made by subsequent research by Parkes [212], Nack [202], Hartley [123] and by the MPEG-7 enterprise [188]. Parkes' statement predates the more rigorous semiotic analysis presented here and Nack's work *op.cit.* on theme based editing, nevertheless it is significant in that useful results have been obtained by adopting this position.

The adoption of the additional cross-cutting of the distinction between substance and form allows further separation of concerns to be achieved when applying the R2 semiotic model to multimedia content. The application by Chatman of semiotic theory to narrative together with the cross-cutting enables the methodology of the R2 model to be extended from the level of existents to the level of events and narrative. It will be noted that the form of the expression of moving pictures is the moving picture language, and the form of the content is the relationships between the language components. Irrespective of whether a *moving picture* or *film* language exists, what is of interest from the perspective of the current developments is the substance and form of the description plane language. That is the language used to *describe* the content as perceived by the observer through the expression plane. Finally, not only should the description language be capable of describing the content but it should also be capable of being computationally implemented. Tthis is developed in chapter 4 after discussion of the AVA low level image and video analysis techniques.

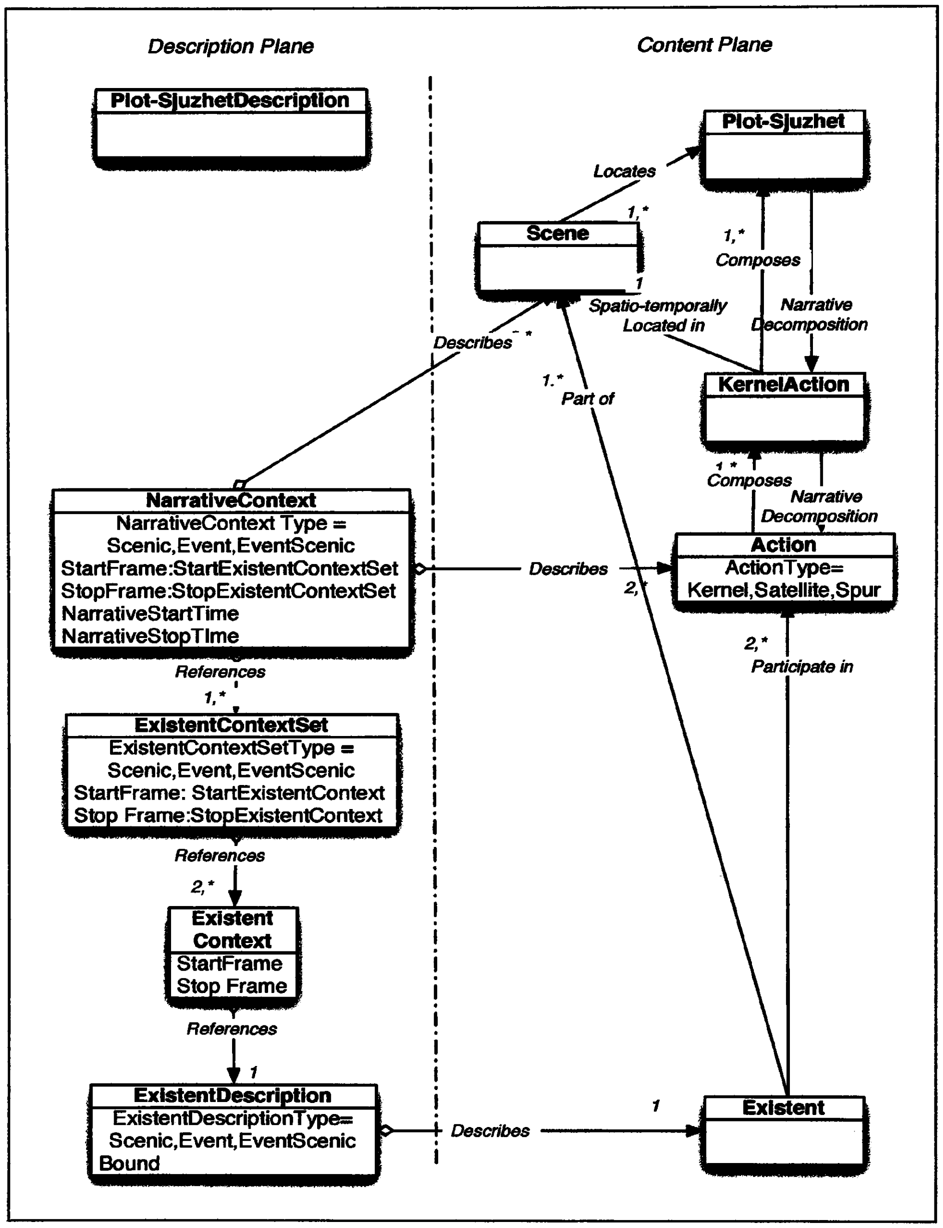


Figure 2.35: Narrative bridging by description revised in UML

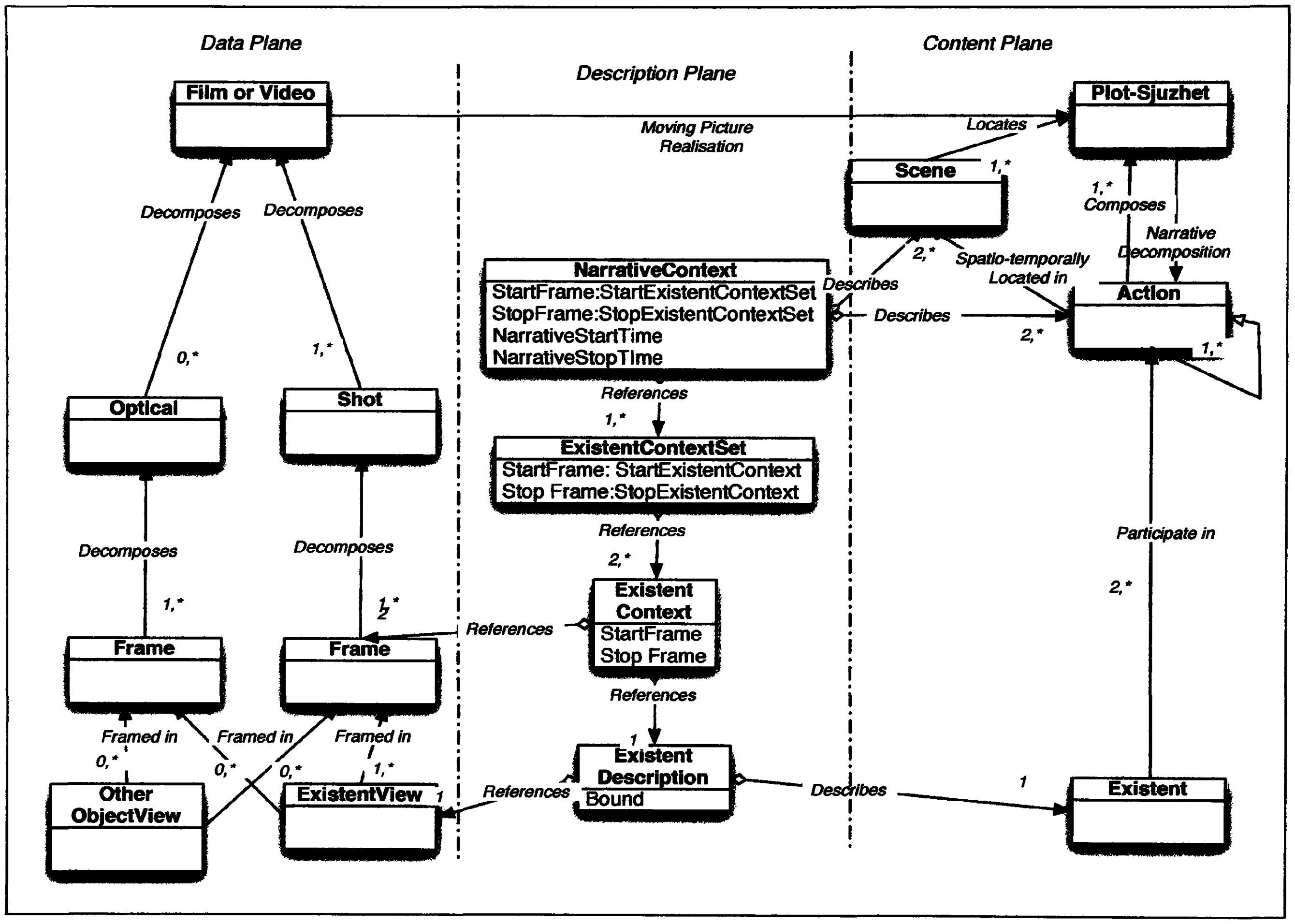


Figure 2.36: The narrative content and data bridged by description in UML

Chapter 3

Image and video analysis

3.1 Introduction

This chapter is concerned with techniques to provide the bottom-up image and video analysis portions of the AVA bi-directional approach and contribute to the instantiation of the lower layers of the model in goal 2. In general there is a wide range of techniques that can be applied to achieve this goal, however space permits only a small selection of these to be examined in detail. The interested reader is referred to the last known comprehensive survey of the field carried out by Rosenfeld [233] in 1999 which references 1700 papers and collections such as Chen *et al.* [55]. Other useful background material is available in; Rabiner and Gold [227] for digital signal processing, Gonzalez and Wintz [109], and Sonka *et al.* [264] for image processing, Rao and Yip [228] for the discrete cosine transform (DCT), Batchelor *et al.* [22] is representative of the machine vision perspective and the collection edited by Lew [171] is useful for visual information retrieval.

Chapter overview: In this chapter concern centres on those techniques with potential for immediate application in satisfying the objectives of identifying and locating objects in images, achieving temporal segmentation and identifying optical effects. First, a preliminary discussion of colour spaces is given, since different colour spaces are used in expression, compression and in image analysis. This is followed by some background material on the MPEG compression schemes since they form an integral part of some approaches to temporal segmentation and in the subsequent description of the present author's work on video decoder modifications and DCT based feature extraction. This leads into an examination of some existing techniques for each of the identification, location and temporal segmentation tasks in the continuous and transform domains.

Filtering in the DCT domain developments carried out by the present author are then described in three sections; The first is a method to produce multiple independent image streams from a single MPEG encoded video stream. The second describes a novel feature extraction method applicable to DCT compressed still images followed by a novel method for applying other techniques in the DCT domain. The third section describes the application of the methods from the second section to one or more of the multiple streams described in the first section for object identification, location and temporal segmentation.

Application of either of the existing or the novel sets of techniques derives sets of image and video features. These image feature sets are then available for *binding* into the existent descriptions introduced in chapter 2 section 2.8.4. The temporal segmentation techniques can also be used to identify the interruption points in the AVA annotation methodology. These techniques are now described to meet the AVA requirements and satisfy the low-level image and video analysis goal.

3.2 Preliminaries

3.2.1 Colour spaces

The use of image colour analysis is a fundamental technique for visual object location retrieval and temporal segmentation and is achieved by quantifying the distribution of colour across an image or a sequence of images. The basis of this approach is correlation modelling between the physical measurement of light intensity and the response of the visual system to such stimuli. Topics such as colour management and calibration in imaging systems are beyond the scope of this work, however extensive literature exists on the application of these techniques in digital video see Poynton [223]. This analysis depends on the choice of colour space, i.e. the choice of a mathematical representation of colour in terms of base

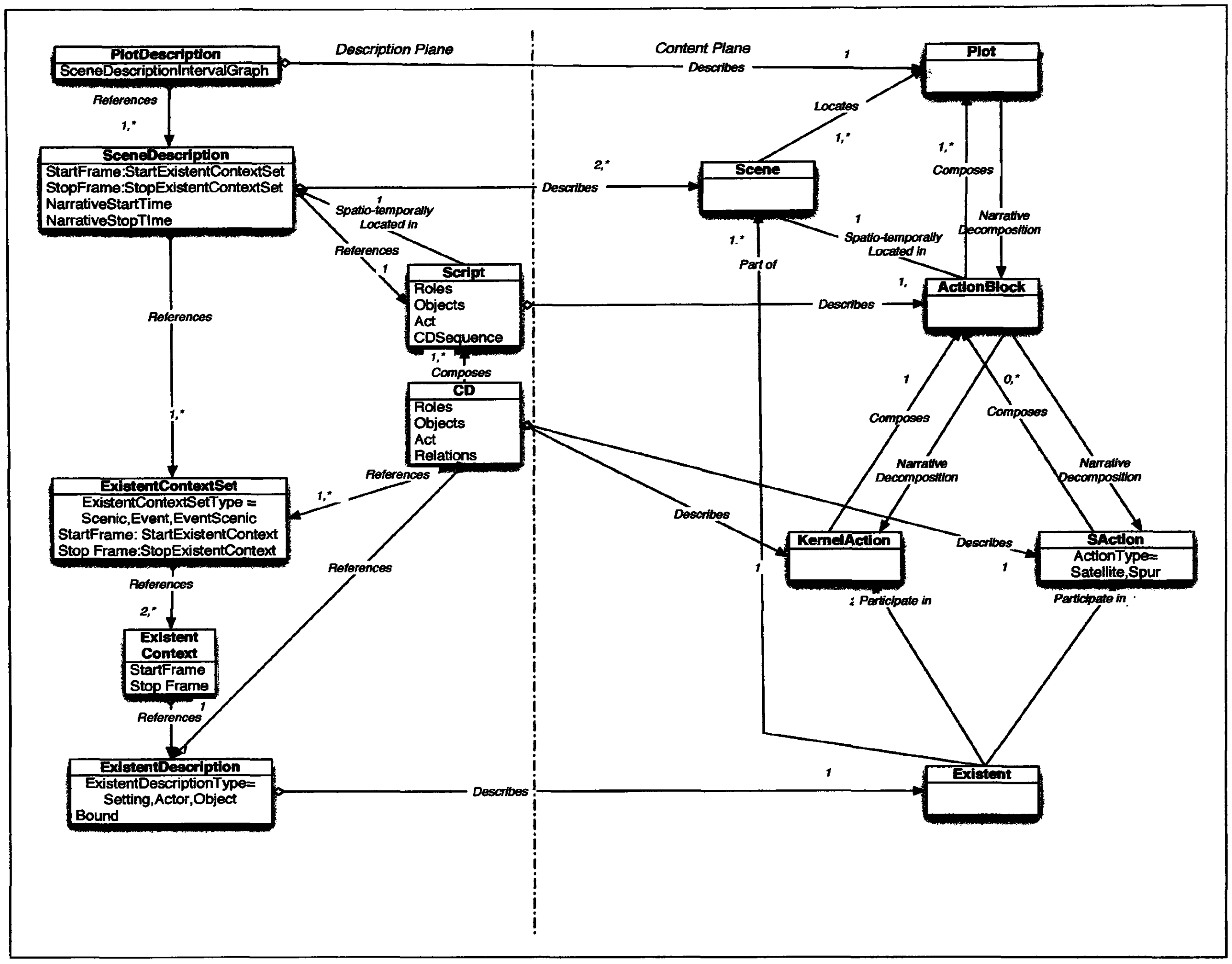


Figure 2.37: Narrative data and content bridging by description in UML

vectors such as the familiar RGB triple or the less familiar HMMD 5-tuple (Hue Max Min Difference and Sum) used in parts of the MPEG-7 [140] standard. An account of the relationship between the RGB colour space, Munsell colour chips and the Committee Internationale d'Eclarage (CIE) colour models, is given by Sharma *et al.* [248]. Other colour spaces are available such as L*a*b and L*u*v that more closely model the response of the human visual system. The dichotomy between mathematical colour representation and the *semantics* colour description is explored in section 2.7.1.

Swain and Ballard adopted an opponent colour space in Ballard and Brown [18] which is shown in equation 3.1.

$$\begin{pmatrix} rg \\ by \\ wb \end{pmatrix} = \begin{pmatrix} 1 & -1 & 0 \\ -1 & -1 & 2 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} r \\ g \\ b \end{pmatrix} \tag{3.1}$$

This is comparable with the YCrCb and YIQ colour spaces used in compression given in 3.2 and 3.3 and of interest because of the linear relationship between the components and RGB.

$$\begin{pmatrix} Y \\ I \\ Q \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.1140 \\ 0.596 & -0.274 & -0.322 \\ -0.211 & -0.523 & 0.312 \end{pmatrix} \begin{pmatrix} r \\ g \\ b \end{pmatrix} \tag{3.2}$$

The YIQ colour space was used in H261 [134] and H263 [135] standards developed for video conferencing.

$$\begin{pmatrix} Y \\ Cr \\ Cb \end{pmatrix} = \begin{pmatrix} 0.2990 & 0.5870 & 0.1140 \\ 0.5000 & -0.4187 & -0.0813 \\ -0.1140 & -0.3313 & 0.5000 \end{pmatrix} \begin{pmatrix} r \\ g \\ b \end{pmatrix} \tag{3.3}$$

In lossy JPEG and the MPEG compression schemes the YCrCb colour space (intensity, red difference, blue difference) is used which is similar to the YIQ. These colour spaces have been widely used in compression standards with most of the recent attention being given to YCrCb. This colour space has been used to support a variety of colour sub-sampling schemes in JPEG, MPEG-1, MPEG-

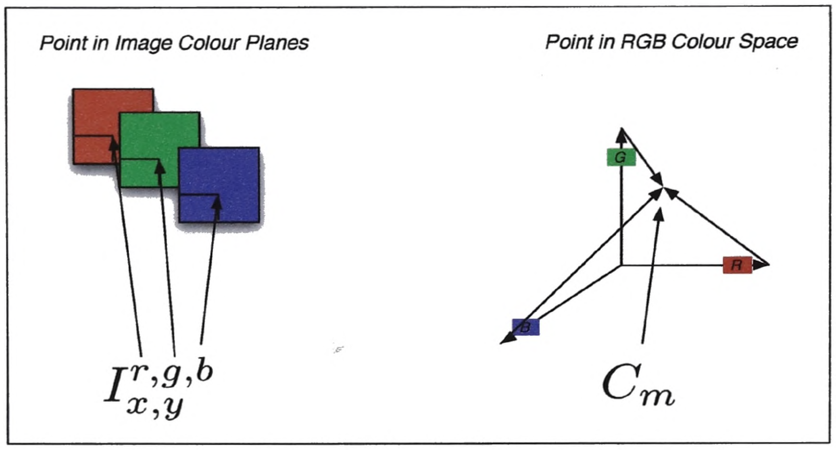


Figure 3.1: Image colour planes and colour space

2 and MPEG-4 at ratios of 4,1,1 blocks [1] to full 4,4,4 blocks per colour plane respectively.

It should now be noted that there is a distinction between a point x, y of a given colour c in an image I^c_{xy} and the point in the *colour space* given by the *r:g:b* tuple as illustrated by figure 3.1. Therefore, an image colour histogram can be considered to be the colour probability distribution of the image.

The colour distribution can be expressed as:

$$h_m = \Sigma^x_{i=0}\Sigma^y_{j=0}I_c \quad where \quad I_{x,y} = \begin{cases} 1: & if \quad I^c_{x,y} = C_m \\ 0: & otherwise \end{cases} \tag{3.4}$$

and where assuming an R,G,B colour space:

$$C_m = R_\alpha, G_\beta, B_\gamma \quad and \quad \alpha, \beta, \gamma \quad are\ the\ colour\ indices\ for\ bin\ m.$$

The colour space of interest for retrieval and location is the HSV colour space, which has several advantages over the spaces described above. developed in the late 1970s by A.R. Smith [253] who represented the digitally generated colour space of television tubes as a cube with orthogonal RGB axes. He describes the

[1]A block of pixels is normally 8 pixels by 8

approximate non trigonometric conversion method into Hue Saturation and Value (HSV) [253], where values of RGB and HSV are usually assumed to be normalised.

Let:

$$V = \max(R, G, B) \quad X = \min(R, G, B) \tag{3.5}$$

then

$$S = \frac{V - X}{V} \quad \text{If} \quad S = 0 \quad \text{return} \tag{3.6}$$

else

$$r' = \frac{V - R}{(V - X)} \quad g' = \frac{V - G}{(V - X)} \quad b' = \frac{V - B}{(V - X)} \tag{3.7}$$

and

$$H = \begin{cases} 5 + b & if \quad r = V \quad and \quad g = X \\ 1 - g & if \quad r = V \quad and \quad g \neq X \\ 1 + r & if \quad g = V \quad and \quad b = X \\ 3 - b & if \quad g = V \quad and \quad b \neq X \\ 3 + g & if \quad b = V \quad and \quad r = X \\ \quad 5 - r & otherwise \end{cases} \tag{3.8}$$

Where varying H is equivalent to traversing the circle from Red to Blue to Green. Decreasing S results in de-saturating the colour or increasing the amount of white in the colour, decreasing V or devaluation increases the blackness. He provides a geometrical analogy for the transform as conversion from RGB cubic space to an HSV hexcone. Subsequently this has been represented by J.R Smith [256] as a cylinder as shown in 3.2. It can be seen to be derived from the Munsell [197] and the CIE work on colour definition already discussed. It will be seen that YCrCb terms can be substituted into equations 3.5 to 3.8 to allow a direct conversion between the YCrCb and HSV colour spaces.

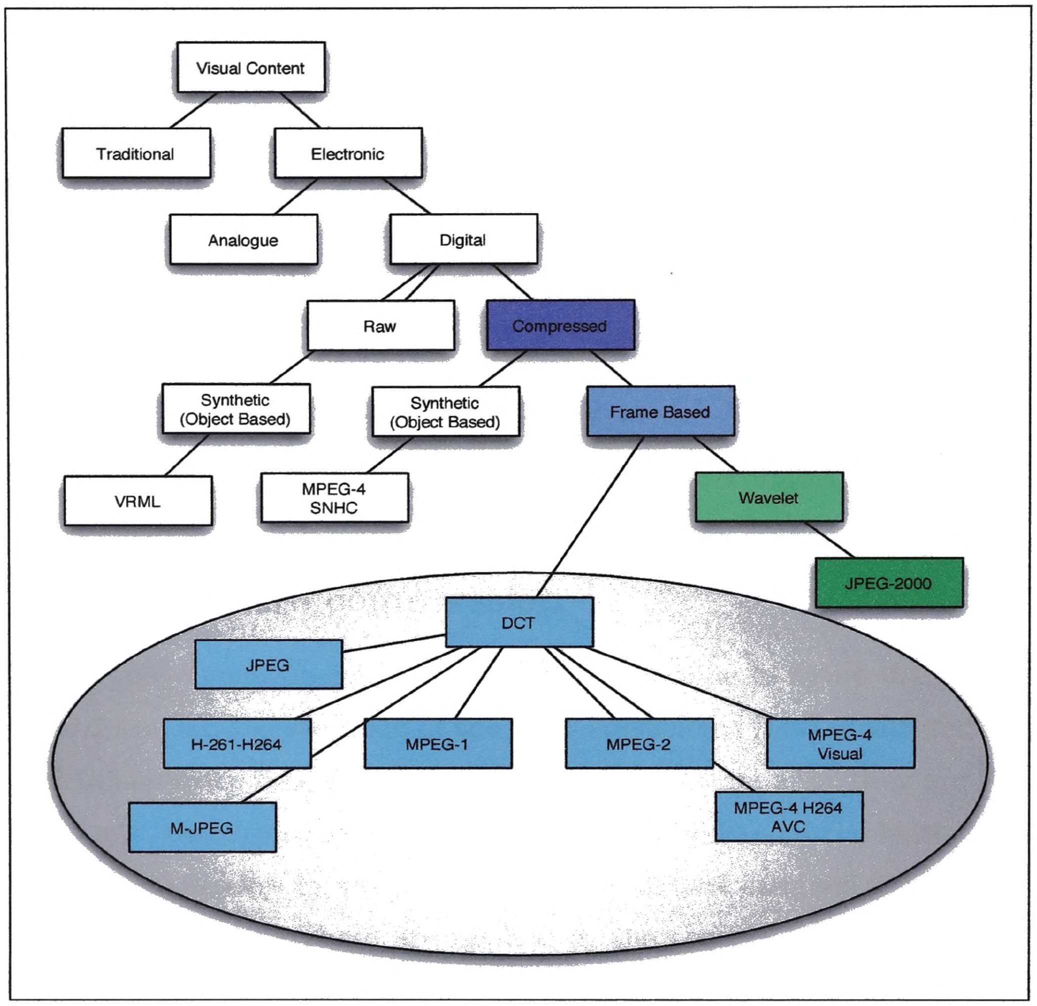


Figure 3.3: A taxonomy of moving picture technologies

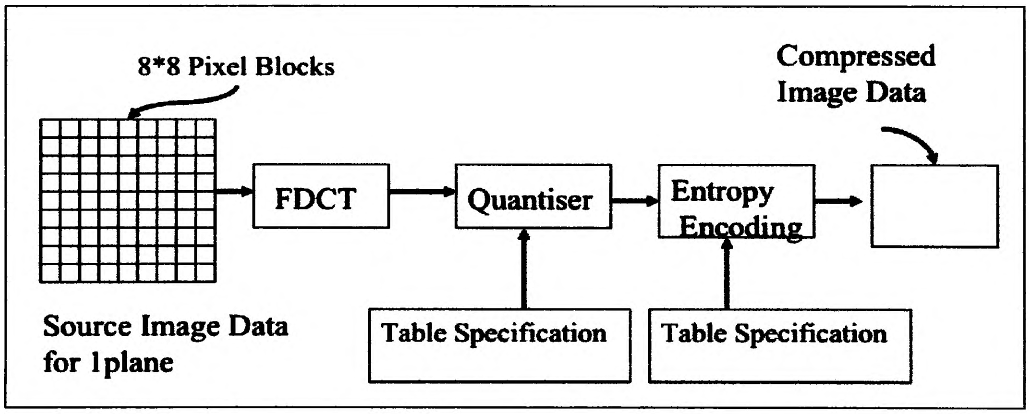


Figure 3.4: JPEG image compresssion

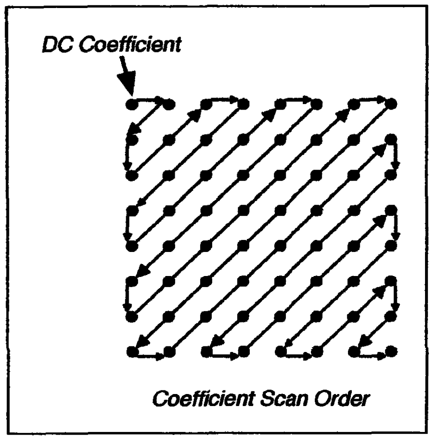


Figure 3.5: Zig-zag DCT coefficient scanning

3.3.2 Still picture compression

The JPEG image compression scheme shown in outline in figure 3.4, utilises the DCT transform, zig-zag scanning of the DCT coefficients [157] illustrated in figure 3.5 , which facilitates run-length coding of the coefficients after they are Huffman variable length, entropy coded [130] sometimes referred to as variable length coding (VLC). VLC is a process by which shorter codewords are assigned to more frequent symbols in the source. These are applied in the sequence shown in figure 3.4. The zig-zag scanning order is shown in figure 3.5. This synthesis of a number of efficient compression algorithms operating in concert accounts for the success of this approach in the still image domain.

3.3.3 Moving picture compression

The MPEG committees during development that has stretched over more than 10 years made a suite of codecs available targeted at applications of increasing sophistication. An outline of the compression approach adopted in MPEG-1 will be given and an account of the differences between it and MPEG-2 and MPEG-4 will be noted.

Syntactic Layer	Function
Sequence Layer	Random Access Unit: Video
Group of Pictures	Random Access Unit: video
Picture Layer	Primary Coding Unit
Slice Layer	Resynchronisation Unit
Macroblock Layer	Motion Compensation Unit
Block Layer	DCT Unit

Table 3.1: Syntax of video layers in ISO/IEC 1117-2:1993 (MPEG-1)

3.3.4 MPEG-1 visual

Video in MPEG-1 terminology is decomposed from the sequence into a Group of Pictures (GOP) which in turn is further decomposed into the Picture, Slice, Macro-block and Block. The syntax of the MPEG-1 stream is defined in terms of a set of headers for each of these structures as shown in table 3.1. The headers are decoded in turn and the relevant storage structures constructed along with quantisation, motion compensation and Huffman coding tables used in decoding the stream. Some of these tables may be redefined in the stream, in which case the stream syntax also provides headers identifying these structures as well. These layers' functionality is now outlined:

3.3.4.1 Sequence layer

The sequence layer is defined as a random access unit and usually only one is present in a stream or disc file. This layer defines horizontal and vertical picture size, pel aspect ratio, picture rate and the quantisation tables. Whilst quantisation can be changed during a stream there are few, if any, instances of picture size changing in a stream or file.

3.3.4.2 Group of pictures (GOP)

The GOP is defined as the minimum random access unit; the pictures present in the GOP have interdependencies varying with GOP size. The I or intra-coded picture depends only on it's own coded pixels. P or predictive-coded pictures depend on their own pixels and also pixels from the preceding I picture. The B or bi-directional coded pictures depend on their own pixels and both pixels from the preceding I picture and on pixels from the succeeding P picture. The P and B picture dependencies on the I picture do not extend beyond the current GOP. On this basis a decoder may readily skip GOPs during playback or rewind but must always restart playback at the beginning of a GOP. This has significant implications in player implementation: if frame by frame step forward or rewind is required as in the AVA player implementation. It will also be appreciated that GOP size has implications on the amount of storage needed to decode the stream since storage must be allocated for each of the dependent pictures in the GOP. This layer implements motion compensation which is a psycho-physical approach to efficient motion-video compression which exploits the similarity of adjacent images in the source stream. This is exploited in turn in some of the temporal analysis schemes described below.

3.3.4.3 Picture layer

In MPEG-1 the picture consists of 3 rectangular arrays of 8 bit numbers, the luminance and 2 chrominance matrices. These correspond to a grey-scale image (Y) and two colour difference components (Cr and Cb). The I,P and B picture types have already been described, but note that a fourth picture type was originally defined in the MPEG-1 standard namely: the D or DC picture. This was provided for fast scanning of streams, but since this is readily reconstructed from the DC DCT coefficients, support for this picture type was dropped although the approach is still used in temporal segmentation.

3.3.4.4 Slice layer

The role of the slice layer is to allow resynchronisation in the event of transmitter and receiver de-synchronisation. This is readily appreciated by anyone with a digital video broadcast, set top box. In this case a short noise burst results in the loss of one or more slices within a number of dependent pictures inside the GOP. It is sufficient here to duplicate the header parsing and slice decoding portions of the decoder.

3.3.4.5 Macroblock layer

In MPEG-1 the macroblock is made up of 4 luminance and 2 chrominance 8 by 8 bit blocks. The storage and decoding stages for each of these must be duplicated. This stage in MPEG-1 decoding produces an immediate 2:1 reduction in data rate. The differing YCrCb block relationships and colour depths available in MPEG-2 and MPEG-4 have already been noted. This stage must be fully duplicated.

3.3.4.6 Block layer

The block layer is the primary coding layer of the MPEG stream and minimal element of compression which is lossyly compressed using an equivalent scheme to that adopted in lossy JPEG encoding. Decoding at this layer involves decoding the run length coding, Huffman decoding, inverse quantisation, inverse discrete cosine (IDCT) and colour conversion into RGB colour space and each of these stages must be duplicated. The inverse DCT in the MPEG-1 standard is defined as:

$$F_{uv} = 1/4 C_u C_v \Sigma_{x=0}^{7} \Sigma_{y=0}^{7} f(x,y) \cos\frac{\pi(2x+1)u}{16} \cos\frac{\pi(2x+1)v}{16} \tag{3.9}$$

where $u, v, x, y = 0, 1 \ldots 7$

x, y are spatial co-ordinates in the pixel domain

u, v are the co-ordinates of the DCT coefficients in the transform domain

$C_u, C_v = 1/\sqrt{2}$ for $u = 0, v = 0$ or -1 otherwise

The separable property of the DCT and IDCT, which means that the DCT can be applied in the x direction and the in the y direction to the results of the previous transformation step is well-known. This property is exploited in the MPEG standards their software implementation and in the work of this thesis.

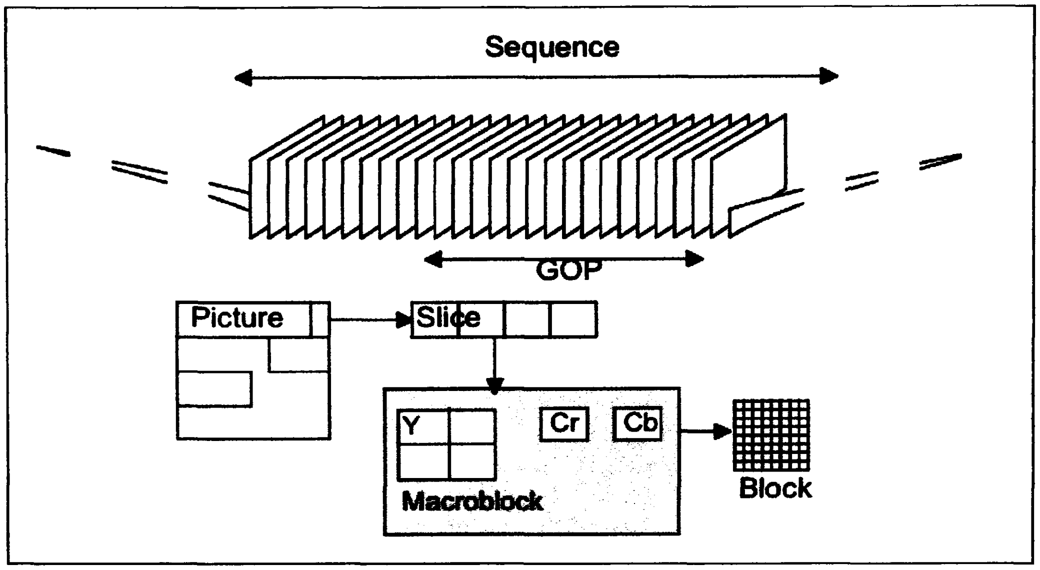


Figure 3.6: MPEG-1 video decomposition hierarchy

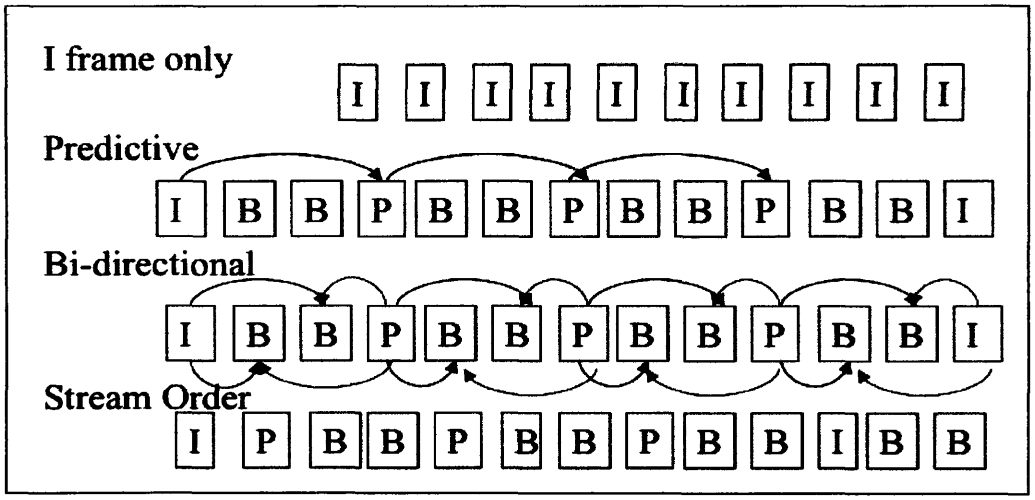

I frame only
I I I I I I I I I I
Predictive
I B B P B B P B B P B B I
Bi-directional
I B B P B B P B B P B B I
Stream Order
I P B B P B B P B B I B B

Figure 3.7: MPEG-1 MPEG-1 picture types

3.3.5 MPEG-2 visual

MPEG-2 differs from MPEG-1 in providing a larger number of data rates, colour depths and macro-block coding coding approaches. These variations are managed through a series of profiles and levels which represent conformance points for encoder and decoder technologies. For example 4:2:2 studio profile at main level, which as its name suggests, is targeted specifically at studio applications and has a bit rate of around 18 Mb/s. It uses 2 Cr and 2 Cb blocks for every 4 Y blocks to make up a macro-block as opposed to the 4:1:1 ratio used in MPEG-1 and MPEG-2 main profile at main level. Several other compression parameters can be varied in MPEG-2. Essentially the same overall compression scheme adopted in MPEG-1 is used in MPEG-2, so modifying an MPEG-2 decoder in the manner described in detail below is a relatively straightforward task.

3.3.6 MPEG-4 visual

The MPEG-4 Visual part [141] provides a more flexible set of coding tools than MPEG-2 supporting a wide range of bit rates and also provide scaleable coding. This is distinct from the advanced coding scheme described in detail below but it serves to illustrate how the principles developed in MPEG-1 and MPEG-2 have been extended to give improved compression ratios. With improved compression ratios comes considerably increased decoder and by implication encoder complexity. This in turn increases the complexity of making the modifications outlined above to an MPEG-4 decoder.

3.3.7 MPEG-4 advanced video coding

MPEG-4 Advanced Video Coding supports a variety of options that are again managed through profiles and levels representing conformance points for interoperability. Typically these place constraints on the syntax elements that can be present in the bit-stream. This standard supports field based frame coding of video, the separate fields in the picture being denoted by the terms upper and lower. Thus interlace scanned video as well as progressive scanned video is supported in the standard. Either or both interlaced fields may be variably coded to make up pictures for use in subsequent stages of the process. In addition in this coding scheme 8 picture types are available, three of which are Instantaneous Decoded Refresh Picture (IDR) types which are coded using only Intra (I) or slice inter (SI) marco-blocks respectively. The other 5 picture types are made up of various combinations of the slice types outlined below according to the table 3.2 reproduced from the standard [143]. Some of these picture types are formally

Primary picture type	Valid slice type values in primary coded pictures
0	I
1	I, P
2	I.P.B
3	SI
4	SI, SP
5	I, SI
6	I, SI, P, SP
7	I, SI, P, SP, B

Table 3.2: Valid slice types in MPEG 4 pictures after [143]

named in the standard, notably Predictive (P) are coded using inter prediction from previously decoded reference pictures using 1 motion vector; Bi-Predictive (B), uses intra prediction using 2 motion vectors. Unlike MPEG-1 and 2 a previously decoded predicted picture may be used as a reference picture for subsequent pictures. I and SI slices are encoded using prediction of samples from within the same slice, the samples being quantised in the case of the SI slice. P and SP slices are encoded using inter picture prediction from previously decoded reference pictures with up to one motion vector or reference index to predict the sample values for each block. B slices are coded using inter prediction or intra prediction with up to two motion vectors or reference indices to predict the sample values for each block. The DCT transform is applied, the results quantised and after this either VLC or arithmetic coding is applied followed by run length coding as in MPEG-1 and JPEG. Arithmetic coding is a statistical alternative to the VLC technique. Inter coding can be carried out at the block, macroblock, slice or picture level as opposed to just the picture level as in MPEG-1. It also allows inter frame macroblock and inter frame sub macroblock predictive motion compensation which uses from 0 to 4 partitions of each 16*16 block of luma samples and each 8*8 block of chroma samples. This enables motion compensation to be carried out at a finer resolution than in MPEG-1 and 2 where motion compensation is carried out at the macroblock level between pictures. This is done to provide improved image quality and it obviously results in greater codec complexity. From this discussion it can be readily seen that the approach to annotation and the novel decoder discussed below are applicable to this coding scheme.

3.4 Visual object recognition and location

Spatial image analysis methods are now examined, this is followed by temporal methods on the basis that the temporal techniques are often modifications of those used for spatial segmentation. When the spatial techniques are not used for *similarity-based* retrieval they are used to detect picture *differences* to provide temporal picture sequence boundary detection by use of modified metrics. This ordering reverses the natural stream processing order for compressed video but is more natural for the explanation of the techniques. The approach considered here is based on the use of colour histograms to perform the two distinct tasks of visual object *identification* and *location.* This distinction is adopted from Swain and Ballard [278]. The need for task separation derives from brain function analysis showing that separate brain areas are specialised in humans for each of these tasks. This is considered further in the evaluation section 5.5 in the light of newer results from functional Nuclear Magnetic Resonance Imaging (fNMRI). The visual object recognition and visual object location aspects of still image processing will now be discussed.

3.4.1 Visual object recognition

Swain and Ballard show that colour histogram based retrieval has a high level of immunity to changes in scale and rotation [277]. Histogram template matching has subsequently been utilised in a number of large scale demonstrator systems, notably: IBMs Query by Image Content (QBIC) and Query by Video Content (QVIC) [94], and Columbia University's VisualSEEk [255]. These are characterised by Smith [256], in his work on retrieval systems as instances of Query by Visual Content (QBVC) systems. Schiele [245] is another, more recent proponent of the technique. Eigenvector approaches were used in the MIT Photobook system [219], the commercial product Virage [113] and the MOODS multimedia management system [111]. The use of a region based semantic classification by feature set scheme in the Simplicity system is described by Wang *et al.* [293].

The two authors Smith [256] and Schiele [245] are particularly useful in that they separately provide solutions to the lighting invariance problem with different texture metrics; Smith with the Haar [162] transform and Schiele with steerable filters (a term coined by Freeman and Adelson [99]). Steerable filters can be based on Gaussian derivatives and are investigated in the context of compressed domain feature extraction later in section 3.7.2.

There are some obvious potential problems with the histogram techniques in that two separate objects or scenes may have the same histogram and yet still be totally different visually. The view is held that in the closed domains under consideration in the current case vulnerability to this issue is reduced. More recently joint histogram techniques have been proposed to overcome this problem in large databases applications where the problem becomes critical [216].

3.4.2 Image histograms

Image histograms are defined as n-dimensional vectors $H_j, j = 1, \dots, n$, (where n is the number of bins representing the number of grey levels or colours and H_j is the number of pixels from the frame with colour j. Histogram Intersection is defined on a pair of histograms that can be designated the Image I and the Model M each containing n bins represented as:

$$\sum_{j=0}^{n} \min\left(I_j, M_j\right) \tag{3.10}$$

This gives an estimate of the number of pixels of colour j in the model that have pixels of the corresponding colour in the image. A fractional value appropriate for threshold, or confidence value estimation can be obtained by normalizing the number of pixels in the model histogram as follows:

$$H\left(I_j, M_j\right) = \frac{\sum_{j=1}^{n} \min\left(I_j, M_j\right)}{\sum_{j=1}^{n} M_j} \tag{3.11}$$

Swain and Ballard [278] noted that the histogram match value is not reduced by the presence of distracting pixels in the background. This is of particular relevance in the AVA annotation methodology where distracting pixels are likely to be present. Latterly both Smith *op.cit.* and Schiele *op.cit.* have compared the performance of histogram intersection with other more sophisticated distance metrics. These can give improved retrieval effectiveness.

The HSV colour space has been partitioned into 162 colour and 4 grey bins [256] and into 256 bins or binary sub multiples thereof as proposed in MPEG-7 [140]. The rationale for the former is to combine minimisation of the number of bins with symmetric colour space partitioning, intuitively capable of being related to RGB. The rationale for the latter choice of 256 is the ability to support rapid indexing when the histograms are subjected to Haar [162] transform processing [140]. Here comparison can be performed by an exclusive or operation on the transformed histogram values rather than the more complex algorithm discussed below. There is continued interest in colour spaces and their transformation as can be seen from recent work on reflectance spectra and the Munsell colour space [65].

3.4.3 Visual object location

Smith *op.cit.* and Schiele *op.cit.* also usefully independently assess the effectiveness of a number of back-projection algorithms. Schiele's work is significant for two reasons. His use of steerable Gaussian derivatives gives validation for the use of the transform domain techniques proposed later in section 3.7. In confirmation of Swain and Ballard's *op.cit.* results he is able to show that the histogram technique has immunity to changes in: scale, shape and to a limited extent occlusion. This in turn contributed to the need for a reconsideration of Parkes' [214] definition of the setting formalism in the development of the AVA model. The method for locating an object in the approach adopted by Swain and Ballard is referred to as histogram back projection. In this technique a ratio histogram is first defined as follows:

$$R_i = \min\left(\frac{M_i}{I_i}, 1\right) \tag{3.12}$$

In the image the pixel values are replaced by the values of R that they index. i.e each pixel value. This is the back projection of the ratio histogram onto the image. This algorithm has been successfully used by Ennesser and Medioni [85], and Smith *op.cit.* and Schiele *op.cit.*

3.5 Temporal segmentation

Temporal segmentation has been investigated in considerable depth in recent years, generally with the aim of generating video indices and supporting fast browsing. The techniques employed can be used to detect shot and scene breaks in either the compressed or uncompressed domains. Survey papers are provided by Aas *et al.* [1], Brunelli *et al.* [45], Koprinska and Carrato [160] and Lienhart [173]. Whilst a simple technique is adopted in the current implementation of the AVA architecture, its component based implementation readily allows investigation into other approaches. An outline summary of some of the available techniques is now given and some observations made about their applicability to the AVA architecture and implementation. The discussion is not exhaustive since the volume of research in the area is considerable.

3.5.1 Temporal segmentation of uncompressed video: cut detection

Segmentation in the uncompressed domain divides into three categories based on comparison of pixels, blocks and histograms. Typically this relies on the definition of a similarity measure between successive images. If two successive images are sufficiently dissimilar a cut is identified. This can be readily observed by considering the graphs shown in figure 4.8 where the graph shows marked transitions corresponding to cuts in the video.

3.5.1.1 Pixel comparison

The measurement and threshold comparison of the absolute sum of pixel differences is amongst the simplest of pixel based comparison methods. The method is described by Kikukawa [155] for grey scale images and defined as:

$$D(i, i+1) = \frac{\sum_{x=1}^{X} \sum_{y=1}^{Y} \mid P_i(x,y) - P_{i+1}(x,y) \mid}{XY} \tag{3.13}$$

and for colour images:

$$D(i, i+1) = \frac{\sum_{x=1}^{X} \sum_{y=1}^{Y} \sum_{c} \mid P_i(x, y, c) - P_{i+1}(x, y, c) \mid}{XY} \tag{3.14}$$

where i and $i+1$ are the two successive frames under consideration with dimensions X, Y and $P_i(x, y)$ and $P_i(x, y, c)$ are the intensity or colour values at pixel location x, y. A disadvantage of this technique is that the threshold may be reached both in cases of a large change over a small area *and* a small change over a large area. This can be considered beneficial in the context of AVA because in both these cases the changes may relate to a change in the existent context.

This approach can be improved by counting the number of pixels that change more than a threshold and comparing the total against a second threshold as described by Zhang *et al.* [303] and Nagasaka [205] and shown in equation 3.15:

$$DP(i, i+1, x, y) = \begin{cases} 1 & : \quad \text{if} \mid P_i(x, y) - P_{i+1}(x, y) \mid > T_i \\ 0 & : \quad \text{otherwise} \end{cases}$$

$$D(i, i+1) = \frac{\sum_{x=1}^{X} \sum_{y=1}^{Y} DP(i, i+1, x, y)}{XY} \tag{3.15}$$

If $DP(i, i+1, x, y)$, the percentage of changed pixels, is greater than a threshold T_2, a cut is said to have been detected. This can have the effect of filtering out a proportion of the spurious cuts detected, though there is still sensitivity to object and camera motion.

3.5.1.2 Block-based comparison

Block based approaches rely on localised characteristics that improve immunity to camera and object motion. Frame i is divided into a number of blocks b that are compared with each succeeding frame $i+1$. The difference between i and $i+1$ is measured by:

$$D(i, i+1) = \sum_{k=1}^{b} c_k DP(i, i+1, k) \tag{3.16}$$

and c_k is a predetermined coefficient for block k and $D(i, i+1)$ is a partial match value for the kth block in frame i and $i+1$. Likelihood ratio matching has also been investigated by Katsuri and Jain [153]. This clearly leads itself to implementation in the compressed domain and integration into the AVA architecture. This method has higher immunity to small and slow object motion. Another approach using coarse divisions of the frames and non-linear statistics is described by Shahraray [247] which can further suppress effects of camera and motion effects. Xiong describes methods based on spatial and temporal sub-sampling of the image space [298] and [297].

3.5.1.3 Histogram comparison for temporal segemtnation

The invariance of image histograms to scale and rotation variation has already been noted in section 3.4.1 above. These factors can also be exploited when considering temporal segmentation to gain further immunity from object and camera motion. However the same flaws when the histogram approach is applied to retrieval and location are present in their application to temporal segmentation. Cut detection depends on the absolute sum of histogram differences between successive frames $D(i, i+1)$ being greater than threshold T.

$$D(i, i+1) = \sum_{j=1}^{n} \mid H_i(j) - H_{i+1}(j) \mid \tag{3.17}$$

where $H_i(j)$ is the histogram value for bin, j of the ith frame and n the total number of bins. Zhang *et al.* [303] describe a technique that uses only the upper two bits of the three colour intensities to reduce the number of bins used for comparison from 2^{24} to 64. Extensions of this method are available using the MPEG-7 scaleable colour descriptor [140] [188]. Nagasaki *op.cit.* proposes using the χ^2 test for histogram comparison of successive frames. Unfortunately not only does the approach emphasise the differences across a cut but it also serves to emphasise motion and is more computationally intensive. The study includes localisation extensions *c.f.* section 3.5.3 that give superior results.

A comparative study of the use of the RGB, HSV, YIQ, L*a*b, L*u*v and Munsell colour spaces was conducted by Gargi *et al.* [103], when $\delta H_i(j), H_{i+1}(j)$, histogram intersection and weighted bin difference comparisons were used. It is concluded that the YIQ, L*a*b and Munsell colour spaces perform better than

HSV and L*u*v which in turn are better than RGB. This has implications that can be exploited when considering the transform domain in section 3.5.4.

3.5.2 Temporal segmentation of uncompressed video: other transitions

The problems that object and camera motion introduce when transitions other than cuts are considered have already been mentioned. This is because the changes due to non-cut transitions are comparable to, or smaller than, those due to object motion or lighting changes. The methods outlined below attempt to overcome these deficiencies by adopting more sophisticated techniques.

3.5.2.1 Twin comparison method

This technique described by Zhang [303] adopts a multi-pass comparison approach to measure the accumulated differences between frames. The first pass detects cuts through comparison against a high threshold T_h. The second pass uses a lower threshold T_l to detect the start frame f_s of a potential transition. If $f_{s+1,\ldots,s+n} < T_l$ the end frame f_e of the potential transition is found, then provided $\sum_{i=s}^{f} \delta(f_i, f_{i+1}) > T_h$ a valid transition is said to have been detected. If $\delta(f_s, f_e) > T_h$ then there is a valid transition; if not the process is restarted. Further refinements allow some frames in the range $f_{s+1,\ldots,s+n}$ to be $< T_l$ before transition analysis is abandoned. Boreczky's [38] analysis shows that this relatively simple technique performs well. It is also readily integrated into the AVA architecture given that the MPEG decoder library utilised in AVA supports the use of multiple frame pointers, although the start-up latency would be adversely affected.

3.5.3 Histogram comparison methods

3.5.3.1 Block based histograms

The block based histogram method integrates the histogram approach from section 3.5.1.3 with the block based approach of section 3.5.1.2. The frame to frame histogram difference (equation 3.17) is calculated on a block by block basis (equation 3.5.1.2). Swanberg *et al.* [279] used this technique in the RGB colour space with values of $C_k = \frac{1}{b}$ for all blocks. Variations on this technique using the HSV colour space are described by Ip and Lee [49], with selective processing being implemented to overcome the instabilities encountered in this colour space.

3.5.3.2 Feature-based approach

Mai *et al.* [216] describe an approach for temporal segmentation based on edge feature layout analysis. This technique is based on the observation that during cuts and other transitions new edges occur far from old edges. This allows cuts, fades and dissolves to be detected by counting edges entering and exiting the frame. However false positives may be detected due to the presence of multiple moving objects and rapid changes in brightness. MPEG-7 visual *op.cit.* descriptors could be readily used to implement this approach by using a combination of the grid-layout container and the edge descriptor.

3.5.3.3 Model based approaches

The techniques described so far are considered to be strictly bottom-up in the video indexing community. The approaches that are based on mathematical models are sometimes referred as top down in this context. Although this is not considered to be top down from the knowledge representation perspective of AVA, Aigrain and Joly [4] model content in terms of three factors: a) Gaussian noise, b) pixel intra-shot change probability due to object motion, camera motion, angle change, focus change and lighting change and c) a shot transition factor. The latter either replaces, in the case of cuts, or combines with (b) in the case of transitions. Between 75% and 100% detection accuracy is reported using this technique when applied to "difficult" VHS content.

Gradual transitions such as fades and dissolves can be modelled in terms of intensity changes, as in the technique proposed by Yu *et al.* [300]. On a first pass, cuts are detected using histogram comparison and on a subsequent pass transition detection is carried out using a model of image edge pixel change for inter-cut frames. This methodology has low immunity to object motion.

Another model based approach, developed by Hampapur *et al.* [121] divides transitions into four classes: i) an identity class composed only of hard cuts ii) a spatial class composed of wipes, slide page turns and iris effects, iii) a chromatic class where color space transformations occur as in dissolves and fades and iv) a spatio-chromatic class combining spatial and colour space transformations typified by morphing operations. This can be considered to be a superset of the classification in the preceding paragraph.

Spatial and temporal separation classification was developed by Lienhart *et al.* [174], where the transitions are characterised in terms of contrast and colour based spatial gradients. At a cut or fade, a new statistical process is considered to be governing the video signal in the case of cuts or a simple mathematical function is in progress as in the case of fades.

Hidden Markov Model (HMM) methods have been adapted by Foote *et al.* [96] for use in temporal segmentation. The states in the model are the states of the video i.e., shot cut, dissolve, pan etc. The transitions in the model are valid transitions between states such as shot to cut, dissolve to shot. They used a multi-modal feature set based on i) grey level histogram inter-frame difference, ii) acoustic interval difference across the transition iii) inter-frame object motion. The Baum-Welch, [71] forward-backward, training algorithm is used to learn the parameters of the HMM. This method gives better results than those reported using pure threshold-based techniques in the target application domain namely: segmenting video recorded conference proceedings by speaker. Finite state models have been used by Mahadi *et al.* [183] and extended further by Chen *et al.* [56] through the incorporation of audio cue detection.

3.5.3.4 Camera operation recognition

Camera motion can be detected through analysis of the motion present in the uncompressed video by utilising optical flow analysis techniques such as those described by Horn and Schunk [129]. This method can also detect object motion. Another image domain technique is described by Zhang *et al.* [304] which relates points in image space $U(X,Y)$ to points in space (x,y,z), i.e:

$$X = F\frac{x}{z} \qquad\qquad Y = F\frac{y}{z} \tag{3.18}$$

A zoom is taken to result from a change in the camera's focal length not accompanied by a camera movement. This can be represented as a change of co-ordinates in image space:

$$U' = \frac{F'}{F}U = f_z U \tag{3.19}$$

Where F, F' are the focal length before and after the zoom and f_z is referred to as the zoom factor. In contrast a pan is caused by a camera rotation about an axis parallel to the image plane, which results in changes to both the camera and image space co-ordinates of an object. This can be expressed as:

$$U' = U + \begin{pmatrix} \frac{p'_x}{p_x} \end{pmatrix} = U + \mathbf{p} \tag{3.20}$$

$\mathbf{p}$ is referred to as the pan vector. The combination of camera pan and zoom is then;

$$U' = f_z U + \mathbf{p} \tag{3.21}$$

These can be derived using a motion field calculated from two frames, δt, apart. There is considerable computational cost associated with both these techniques so the compressed domain techniques outlined in section 3.5.4 are more efficient.

Camera motion can also be detected though analysis of the motion vectors in the compressed video and is described in depth by Zhang *et al.* [304].

3.5.4 Temporal segmentation of MPEG video

The structure of the MPEG stream has already been described earlier in section 3.3.3. The approaches based on using MPEG compressed video fall into the following categories of increasing analytical complexity that can be applied to DCT coded video. Since several of these were investigated prior to the widespread deployment of MPEG coding additional, notes are given on some straightforward improvements that could be made if the video is encoded in this format.

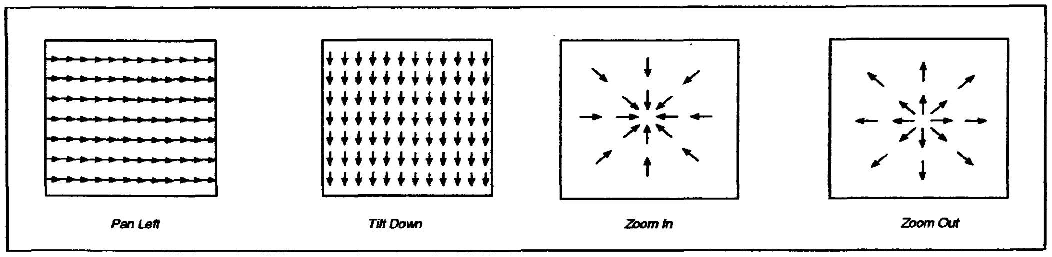


Figure 3.8: Camera movement motion field patterns

1. DC term comparison methods and their extensions
2. DCT coefficient methods and their extensions
3. DC term and macro-block coding mode methods
4. Macro-block coding mode and motion vector methods
5. DC term, macro-block coding mode and motion vector methods
6. Macro-block coding mode and bit-rate information methods

3.5.4.1 DC term comparison methods and their extensions

A simple approach using the $0th$ term in the blocks is used to construct what Yeo and Liu [299] refer to as a DC image where the DC term represents a pixel in a reduced dimension image. From the analysis in section 3.7.2 it will be apparent that this is equivalent to applying a sharp low pass filter to the images. These authors get improved results using pixel-to-pixel difference metrics on the DC terms in comparison with the uncompressed domain. Scene transitions are detected by the use of a sliding window to examine m successive frame differences. Cuts are identified when the conditions in equations 3.22 and 3.23 are satisfied:

$$\delta(f_i, f_{i+1}) = \max_{\#1}(f_{i-m}, \ldots f_{i+m-1}) \tag{3.22}$$

$$\delta(f_i, f_{i+1}) = \max_{\#2}(f_{i-m}, \ldots f_{i+m-1}) \tag{3.23}$$

A modelling approach is then applied using 4 parameters to identify linear transitions and a further 7 to differentiate gradual transitions from those with intermediate plateau conditions and lighting spikes.

A technique based on comparing inter-frame DC term histograms is described by Shen *et al.* [249] where the DC terms for P and B frames are interpolated using a weighted sum of the DC coefficients from the I frames. Static threshold techniques are used to detect cuts and windowed averaging comparison is used for gradual transition detection. Again the method fails to discriminate between transitions and fast object motion.

Further extensions are proposed by Taskiran and Delp [280] who develop a method based on a two dimensional feature vector. The first vector component is derived from the luminance histogram intersection obtained in a similar way to the method described in section 3.10 of the two DC images. The second is obtained from the absolute difference of the standard deviations σ of the luminance component $\mid \sigma_i - \sigma_{i+1} \mid$. The combination is described as the *generalised sequence plot* and overcomes the deficiencies of both the pixel based and histogram based approaches when used alone.

A further refinement uses only the DC components of I frames described by Patel and Sethi [217]. Comparison of DC images is carried out using three different statistics χ^2, test, the Yakimovsky likelihood ratio and Kolmorgorov-Smirnov test.

The technique has been extended to provide textual annotation by Jain and Chaudhuri [132]. Here key frames are obtained from the extracted temporal segments by comparing histogram difference against a threshold. The key frames extracted are then matched against a database of key previously annotated key frames.

3.5.4.2 DCT coefficient comparison methods and their extensions

DCT coefficient comparison was the first technique to be investigated by Arman *et al.* [16] in relation to the analysis of compressed domain temporal segmentation. It is immediately obvious by considering the relationships between frames in the coding scheme that in detecting cuts only I frames need to be considered. A temporal vector is constructed from a subset of DCT coefficients in a subset of macro-blocks from I frames only as $V_i = \{c_i, c_i + 1 \ldots, c_n, \}$. This has reduced complexity in comparison with using all the DCT coefficients. The normalised inner product is then calculated and a cut identified if:

$$D(i, i+n) = \frac{V_i \cdot V_{i+n}}{|V_i||V_{i+n}|} > 1 + T \tag{3.24}$$

If this inequality holds a cut is detected. False positive detection is reduced by applying a second threshold, T_2, such that $(0 < T_1 < T_2 < 1)$. If $T_1 < D(i, i+n) < T_2$ the two frames are decompressed and histogram comparison applied as in section 3.17.

A refinement described by Zhang [304] is to compare inter-frame DCT coefficients in corresponding blocks. A difference metric summing the coefficient differences corresponding to the pixel based approach in section 3.5.1.1 is then used. The approach as described compares all 64 coefficients. A transition is detected if the block difference exceeds T_1 and the number of changed blocks exceeds T_2.

3.5.5 DC term and macro-block coding mode methods

In the technique described by Meng *et al.* [193], cut detection is achieved by computing ratios of forward, backward and intra-coded motion vectors:

$$R_p = \frac{intra}{forw} \qquad R_b = \frac{back}{forw} \qquad R_f = \frac{forw}{back} \tag{3.25}$$

Gradual transitions are detected by determining the variance of the DC image sequence made up from I and P frames and testing for parabolic curves. This relies on gradual transitions being a linear combination of two sequences, s_1 and s_2 with intensity variances, σ_1, σ_2, characterised by $s(t) = s_1(t)[1 - \alpha(t)] + s_2(t)\alpha(t)$ where $\alpha(t)$ is a linear parameter. This results in a variance curve which is parabolic:

$$2\sigma^2(t) = (\sigma_1^2 + \sigma_2^2)\alpha(t) - 2\sigma_1^2. \tag{3.26}$$

3.5.5.1 Motion vector comparison methods

A DC-term and macro-block coding mode with motion vector comparison method was developed by Zhang *et al.* [302]. This method is similar to the twin comparison method described in section 3.5.2.1 applied in the DCT domain. The DC coefficients of the DCT are used in a first-pass and a comparison of the motion vectors in P and B frames for the second-pass. This combination provides an effective method of detecting cuts in the absence of motion, and pan and zoom operations can be discriminated from slow transitions. However static frames give false positives and it has not been shown to be capable of discriminating between gradual transitions and object movement.

A macro-block coding mode and motion vector comparison method to detect cuts, fades and dissolves by only using motion vectors from P and B frames in combination with a neural-rule-based approach was adopted by Koprinska and Carrato [161]. They adopt a two-pass approach; the first detects peaks in the number of intra coded macro-blocks in P pictures, the second applies rule based decision making to the sharp peaks in the I macro-blocks. These rules quantify the number of backward and forward and interpolated macro-blocks to determine simple transitions such the cuts and fades to black. The neural module detects complex transitions such as stationary, pan, zoom and object motion learnt from pre-classified examples.

3.6 MPEG decoder modifications

A series of modifications to the Berkeley MPEG-1 decoder [28] have been made in the course of this research to allow playback of video onto two rendering surfaces. These modifications go deeper into the player source code than is necessary to merely duplicate the decoded image on screen. The modifications involve duplicating the portions of the video stream decoder software concerned with video stream: parsing, decoding, storage and video picture reconstruction. The modifications were carried out to demonstrate the feasibility of using this approach to support the creation of an edge picture stream running in parallel with the image stream. It will be shown in section 3.7.2 that an edge picture can be efficiently derived from the DCT coefficients of the pictures in the compressed video stream. Also in section 3.7.3 it will be shown that a set of image features can be derived from the DCT coefficients in a comparable manner. This approach can be carried forward and applied to MPEG-2 and MPEG-4 video decoders albeit with greater implementation complexity. An MPEG-1 decoder was chosen for this exercise because source code for this player was stable and available at the time this portion of the work was carried out. To understand the modifications made to the MPEG-1 player some further description of the MPEG-1 coding scheme must be given for each layer.

1. **Sequence** Since the sequence layer defines the horizontal and vertical picture size, GOP size, pel aspect ratio, picture rate and the quantisation tables, it will be readily concluded that the picture size parameters will be used to create duplicate reconstructed picture storage structures, rendering buffers *and* sufficient storage for the dependent pictures in the GOP.

2. **Group of Pictures GOP** GOP size varies with codec type and stream definition so it will be concluded that the picture size parameters will be used to create duplicate reconstructed picture storage structures, rendering buffers *and* sufficient storage for the dependent pictures in the GOP. This is due to the interdependency of the pictures resulting in stream order being different from display order. In MPEG-1 GOP size was usually 5 but MPEG-2 and MPEG-4 encoders have typically used significantly larger GOP sizes as memory costs have fallen. Some of these issues are platform, programming language and windowing toolkit-dependent. There are significant platform differences for example in this area between the X-Windows and the Win32 architectures. In particular, some framework and language combinations are unable to support two rendering surfaces.

3. **Picture** At the picture level there will be three image planes each corresponding to a colour in the reconstructed image. Pictures are reconstructed according to picture type and macroblock type as follows:

   3.1 I pictures:
   are reconstructed only from macroblocks originally encoded within the captured picture they are encoded from; these are known as I macroblocks. So when decoding the I picture there will be one macroblock for each 16 by 16 block of pixels in the original image. Following an I picture header the MPEG stream contains a number of slice headers

each containing a number of I macroblocks.

3.2 P pictures:
are reconstructed from P and I macroblocks. P macro blocks correspond to 16 by 16 block of pixels in the original image and from I macroblocks. In the case of the I macroblock in the P picture the stream contains a reference to the I macroblock in the I picture and a copy is made when the P picture containing it is reconstructed.

3.3 B pictures:
are similar to P pictures where B macro blocks correspond to 16 by 16 block of pixels in the original corresponding captured image and the remainder of the B picture is made up of I and P macroblock copies corresponding to the references present in the MPEG stream. The macro block copies may be made from either forward or backward references.

This referencing and copying of macroblocks between pictures accounts for the *compressed data plane* picture order differing from the *data plane* picture order. The interdependency of the different picture types in MPEG-1 means that care must be taken during duplication of the picture reconstruction code to ensure that all macroblocks are correctly placed in the appropriate picture types based on the motion vectors present in the P and B pictures. Figure 3.9 shows an intermediate phase in the development of the dual player when only I frames reconstruction was being duplicated. Analysis of the DC coefficients and motion vectors form the basis of some of the approaches to temporal segmentation in the compressed domain as discussed in chapter 3 section 3.5.4.1.

4. **Slice** The slice is only significant from the resynchronisation point of view. However the slice data structures need to be duplicated for parallel decoding implementation.

5. **Macroblock** The macroblock is primarily present to support greater or lesser amounts of colour sub-sampling as described earlier. Each block in the macro block contributes to one of the YCrCb colour planes. The contri-

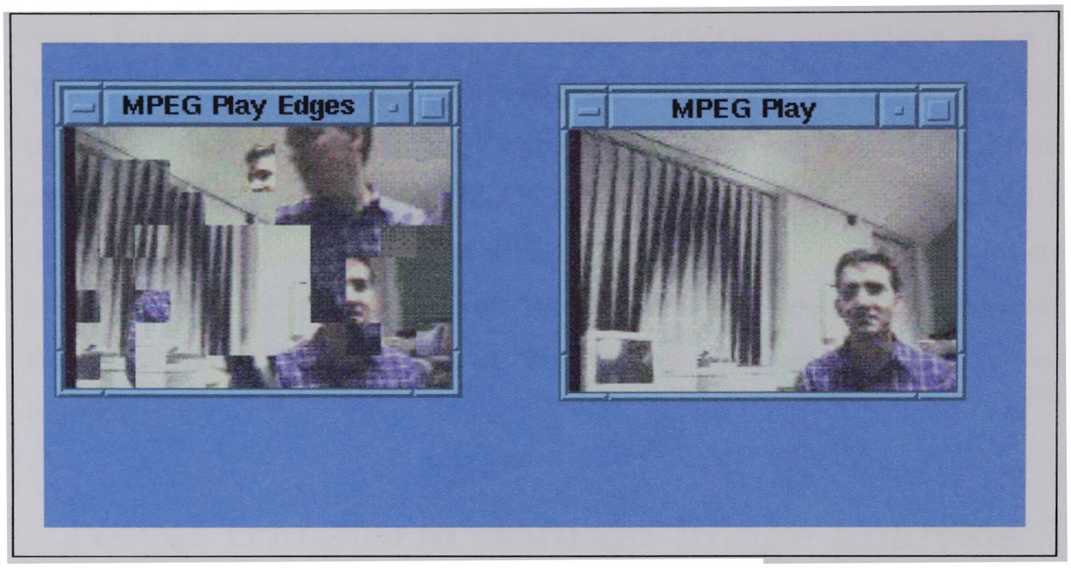


Figure 3.9: MPEG dual player showing I frame only reconstruction

bution that each block in the macro-block makes to the reconstructed picture must be duplicated to achieve dual decoding. Motion compensation takes place at this layer since a particular macroblock may be used in more than one picture.

6. **Block** At the lowest layer the *block* each pixel in the data plane corresponds to a DCT coefficient in the compressed data plane according to the relationship shown in equation 3.9, assuming no colour sub-sampling is being carried out. The one to one relationship after *encoding* is achieved by truncating the number of terms produced by the DCT algorithm to 8. To achieve further levels of compression these DCT coefficients are then quantised and Huffman encoded i.e. instead of the value of the DCT coefficient being present in the stream, a symbol corresponding to the quantised DCT coefficient value is placed in the stream as the coefficients in the block are processed according to the zig-zag scanning order shown in figure 3.5. The Huffman codes are also run length encoded to further increase the compression ratio. Note that there are up to 12 blocks per macroblock. So each block must be decoded from the stream to the point of reconstructing the 8 by 8 DCT coefficient array before separate processing; for image reconstruction or transform do-

main feature extraction or use by other detection algorithms takes place. The DCT contributes significantly to the efficiency of the decoder and the potential efficiency of the edge detector and texture feature extractor described next. It is *after* the block is returned from the stream and *before* performing the inverse DCT that the duplicate set of blocks must be created identical to those extracted from the stream. The second set of blocks can then be filtered or masked according to the schemes described in sections below.

3.7 Novel transform domain image analysis

3.7.1 Preliminaries

The behaviour of linear systems in the time domain considers the operation of transfer functions representing the behaviour of these systems on the signals to obtain the output of the system. Of the many texts that exist describing this behaviour Raven [229] is typical. The output of such systems can be obtained by evaluating expressions such as 3.27 where:

$$Y(s) = G(s)F(s) \tag{3.27}$$

The fourier series expansion for $x_a t$, is

$$x_a(t) = \sum_{k=-\infty}^{k=+\infty} c_k e^{jk\Omega_0 t} \tag{3.28}$$

where $c_k, k = 0, k = \pm 1, k = \pm 2, \cdots$ are arbitrary complex constants and $\Omega = 2\pi f$. More usefully a transform to obtain the Fourier coefficients can be derived, which exists provided the Dirilicht conditions hold for $x(t)$:

$$X(\Omega) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} x(t) e^{-j\Omega} dt \tag{3.29}$$

The discrete time equivalent is known as the discrete fourier transform (DFT) and this takes the form:

$$X(\omega) = \sum_{n=0}^{L-1} x(n) e^{j\bar{\omega} n} \quad for \quad 0 \leq \bar{\omega} \leq 2\pi \tag{3.30}$$

where $x(n)$ is the finite duration sequence of signal samples of length L and the $X(\omega)$ are the L equidistant frequency samples. Computation of the DFT is of $O(n^2)$ in the number of data points for which the series is being evaluated. There are several computationally more efficient algorithms that achieve at least $O(\log n)$. The discrete N point sine and cosine transforms can be derived by taking the real and imaginary parts of the $2N$ point sequence. This is the conceptual basis of the DCT, there also exist fast methods of evaluating this function. These are described by Elliot and Rao [82]. Taking their derivation the one dimensional DCT can be defined as:

$$X_c(k) = \sum_{n=0}^{N-1} x(n) \cos \frac{\pi n k}{N} \tag{3.31}$$

In addition there is a discrete time equivalent to convolution in the continuous domain which can be defined as

$$y(n) = \sum_{k=-\infty}^{k=\infty} x(k)h(n-k) \tag{3.32}$$

where $x(k)$ is the input signal and $h(n)$ is the impulse response of the linear time invariant system (LTIS). This can be thought of as stepping the filter transfer function across the signal transfer function and performing an element-by-element multiplication at each step, followed by product summation. This is computationally complex. However, in the discrete time domain it can be shown analogously to equation 3.27 in the continuous domain that

$$x(n) = x_1(n) * x_2(n) \overset{\longleftrightarrow}{_F} X(w) = X_1(w)X_2(w) \tag{3.33}$$

where $n(n)$ is the input signal, $*$ is the convolution operator and F the DFT and Inverse DFT operator dependent on direction of the transform. That is, convolution in the time domain is equivalent to multiplication in the frequency domain. This is obtained from the relationship between the DFT and the z transform being the discrete time equivalent of the Laplace or s transform in the continuous domain. It can also be shown that taking the z transform of a function is equivalent to performing the DFT (subject to convergence restrictions). Hence it is beneficial to take advantage of the fact that the MPEG image data is already in the transform domain which provides the motivation for this part of the work.

This does not complete this part of the analysis since equation 3.33 is defined in terms of of the DFT of both the signal and the impulse response of the system. The signals of interest in the MPEG stream are defined in terms of the DCT, so it will be necessary to show that there is an equivalent relationship that applies to the DCT domain. This is shown in equations 3.34 and 3.35 developed by Chitpraset and Rao [57]

If

$$Y_c(k) = X_c(k)H_f(k) \qquad (3.34)$$
$$where \quad k = -N, -N+1, \cdots, N-1$$

then

$$y(n) = \hat{x}(n) * h(n) \qquad (3.35)$$
$$where \quad n = 0, 1, \cdots, N-1$$

where $\hat{x}$ is the sequence folded about $n = -\frac{1}{2}$. This means that the convolution multiplication property holds if the multiplication is performed over $2N$ data points. If the transfer function of the edge detector is real and even, then edge detection can be performed on the positive data present in the MPEG stream. Chitpraset and Rao's paper [57] shows results obtained from applying high and low pass filters to DCT transformed data of the whole image.

It should be noted that the method described in [57] uses an overlap and add approach to applying the filter to 8 bit by 8 bit blocks of data obtained by applying the DCT to the *entire image.* Then the filter is applied to one block, stepped by less than a block, applied again and the results added and re-scaled. It will be recalled that the DCT is applied on an 8 bit by 8 bit *block basis* to the image in the MPEG coding scheme decribed in 3.3.3. Extension of this approach to filtering is readily achieved in 2 dimensions due to the separable nature of the DCT as shown in depth by Rao and Yip [228]

3.7.2 Edge detection

The application of an edge filtering approach in the transform domain can be developed from a consideration of the well-known Sobel [264] and Canny [52] operators such as those shown in matrix 3.37, deriving vertically and horizontally from the generic operator 3.36.

$$\begin{matrix} x_{1,1} & x_{1,2} & x_{1,3} \\ x_{2,1} & x_{2,2} & x_{2,3} \\ x_{1,3} & x_{2,3} & x_{3,3} \end{matrix} \tag{3.36}$$

$$\begin{matrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{matrix} \qquad \begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix} \tag{3.37}$$

These are in fact approximations to the gradient or Laplacian operator ∇ which can find the gradient of an image $f(x,y)$ at a point (x,y) which can be defined as a two dimensional vector

$$G\left[f(xy)\right] = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{df}{dx} \\ \frac{df}{dy} \end{bmatrix} \tag{3.38}$$

The Laplacian operator is normally used in conjunction with the Gaussian operator because the Laplacian generally produces too many edge points. The application and biological implications of this filter have been extensively discussed by Marr [190]. Marr shows the following relationship for evaluating the Laplacian of the Gaussian:

$$\nabla^2 G_{(r)} = \frac{-1}{\pi\sigma^4}\left(1 - \frac{r^2}{2\sigma^2}\right) e^{\frac{-r^2}{2\sigma^2}} \tag{3.39}$$

This is evaluated in figure 3.10 for a small range of useful values and the fast fourier transform of these results is shown in figure 3.11. An approximation to the Fourier coefficients of the filter was obtained by Ersoy and Hansen[87] as follows:

$$H(k) = e\sigma^2 k^2 e^{-\sigma^2 k^2} \tag{3.40}$$

$$\text{where} \quad 0 \leq k \leq N-1$$

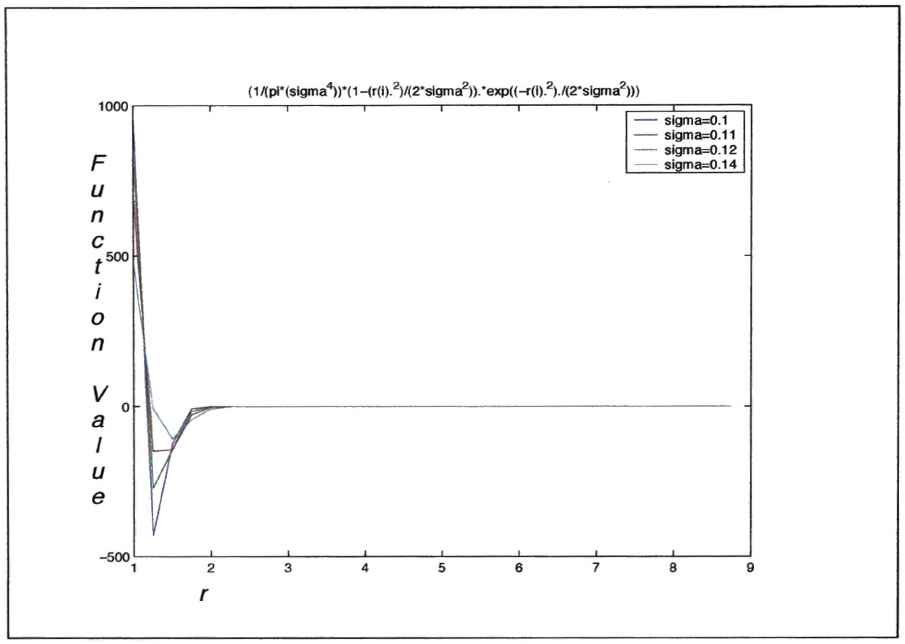


Figure 3.10: $\nabla_2 G$ curves for varying σ

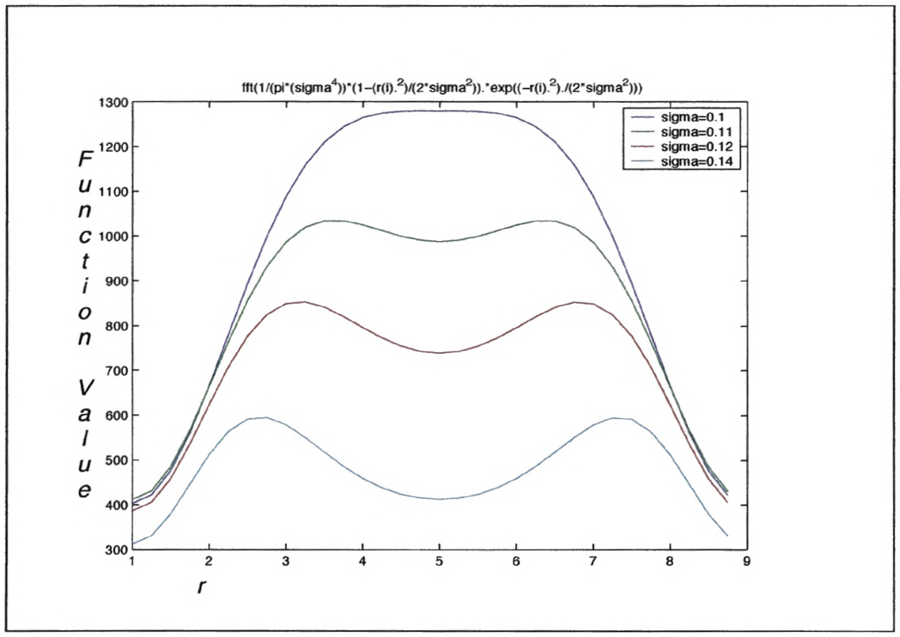


Figure 3.11: FFT($\nabla^2 G$) curves for varying σ

where σ controls the spread of the impulse response, k indexes the coefficient number and N is the number of coefficients. The approximation is readily extended into 2 dimensions by substituting $(k_1^2 + k_2^2)$ for $0 \leq k_1, k_2 \leq N-1$. In the one dimensional case the Laplacian of the Gaussian produces the family of curves illustrated in figure 3.11 whilst equation 3.40 produces the set of curves in figure 3.12.

In another paper Aghagolzadeh and Ersoy [13] show how modifying the approximation with a mask can provide an edge detector when applied on a block by block basis to DCT transformed images.

$$\begin{aligned} H_{I,J} = ku^2 e^{-\sigma^2 k^2} \quad \text{where} \quad u = \left(u_i^2 + u_j^2\right)^{\frac{1}{2}} \\ \text{and} \quad u_k = \quad \text{for} \quad k = 0, 1, \ldots, N-1 \end{aligned} \tag{3.41}$$

They simulate this behaviour using block sizes of 16 and 32 pixels which is larger than the block sizes used in the major compression algorithms that were part of the discussion in chapter 2. This work has been repeated and extended. On the basis of which it can be shown that the approximation defined in equation 3.41 can be used to approximate to the FFT of the Laplacian of the Gaussian.

Figures 3.12 and 3.13 show 1 dimensional representations of the filter developed by Aghagolzadeh and Ersoy over two ranges of values of sigma that have been considered for filtering. Figure 3.14 shows the results of applying the filter in the DCT domain and the resultant inverse transformed image.

3.7.3 DCT edge filter and steerable gaussians

The discussion above shows that edge filtering is feasible in the transform domain. The use of the filter can be extended beyond it's use as an *edge detector*. It can also be used as a texture feature extractor since the Gaussian itself is known to be steerable from work by Freeman and Adelson [99] and Simoncelli [250] amongst others. It is already known that the DCT is separable so a texture feature can be obtained by applying a linear combination of the approximations given in the

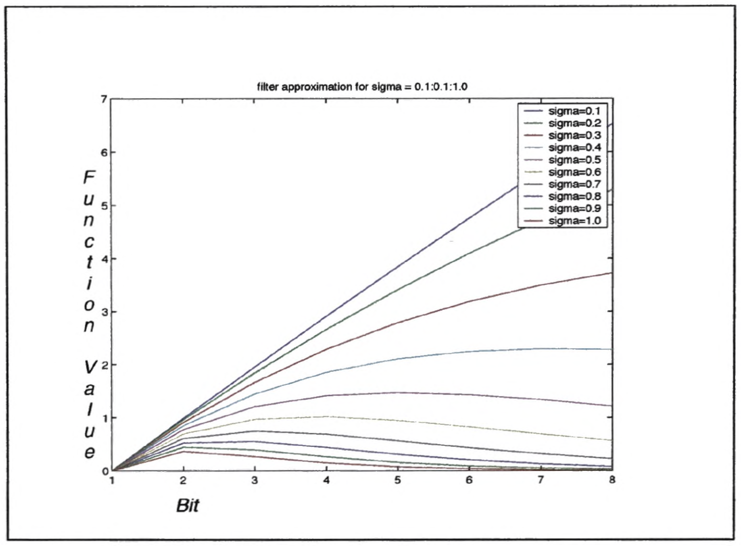


Figure 3.12: Approximation to $FFT\nabla_2 G$ varying σ 0.1 *to* 1.0

preceding section at different values of σ. This is clearly an opportunity for further work and is discussed in section 5.3.1

3.7.4 Spatial frequency texture mapping

A new approach applying a set of binary masks to the DCT coefficients according to the pattern shown in figure 3.15 is proposed in this section. The masks were obtained by considering the zigzag coding of the DCT coefficients described in chapter 2 and exploiting what is effectively the spatial and directional distribution of the coefficients that the zig-zag coding produces. This gives an alternative approach to the steerable Gaussians described above and enables spatial frequency feature extraction to be carried out in the DCT transform domain. Clearly other choices about the regions can be made which are considered further in the conclusions 5.3.1. This represents a preliminary analysis. In effect 10 binary masks can be defined; each mask is one in the corresponding region shown. Clearly region 1 will produce a sub-sampled grey-scale image. Applying each mask in turn produces the results shown in figure 3.16 for region 6, a complete set of results have been obtained for one image. It will be readily appreciated that an analysis comparable with that in section 3.1 can be applied in the spatial frequency domain

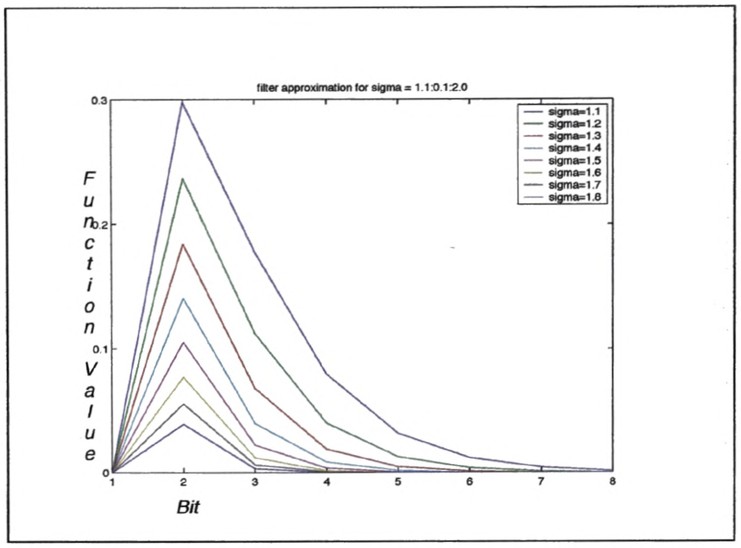


Figure 3.13: Approximation to $FFT\nabla_2 G$ varying σ 1.1 *to* 2.0

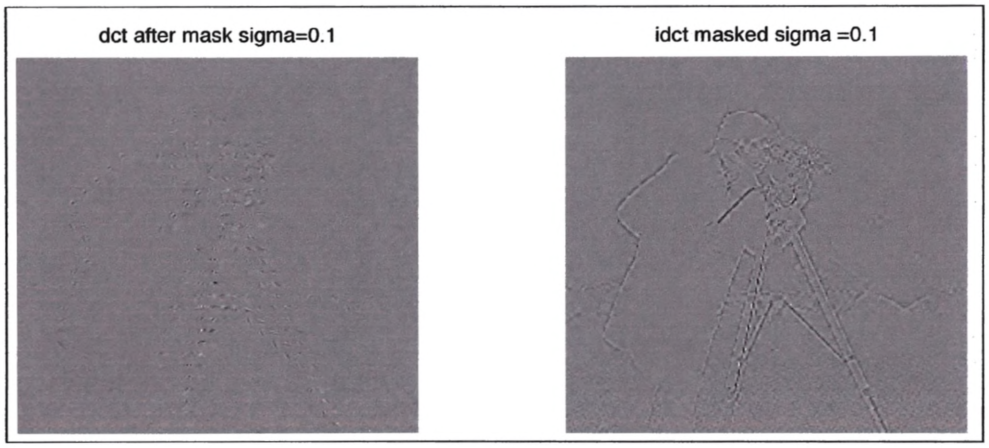


Figure 3.14: Results of applying filter $\sigma = 0.1$

as shown in figure 3.17. In effect the image plane is transformed by this operation into a block region spatial frequency map. Considering that the DCT coefficients are quantised in the MPEG stream as discussed in chapter 2, a map created in this way would also be quantised. Therefore it is a relatively simple task to obtain this histogram of block region intensities from the MPEG stream since an 8 or 4 level quantisation can be simply obtained from the default quantisation tables. This produces an 80 or 40 bin histogram and need only be obtained from the Y blocks in each image. Some further economies of processing may be obtained by restricting

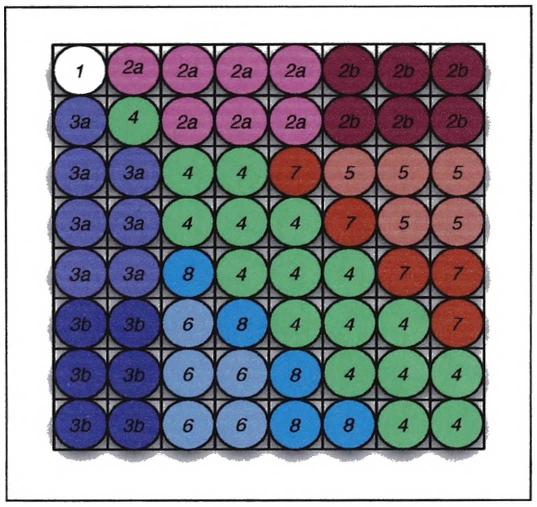


Figure 3.15: Regions in DCT block for texture mapping

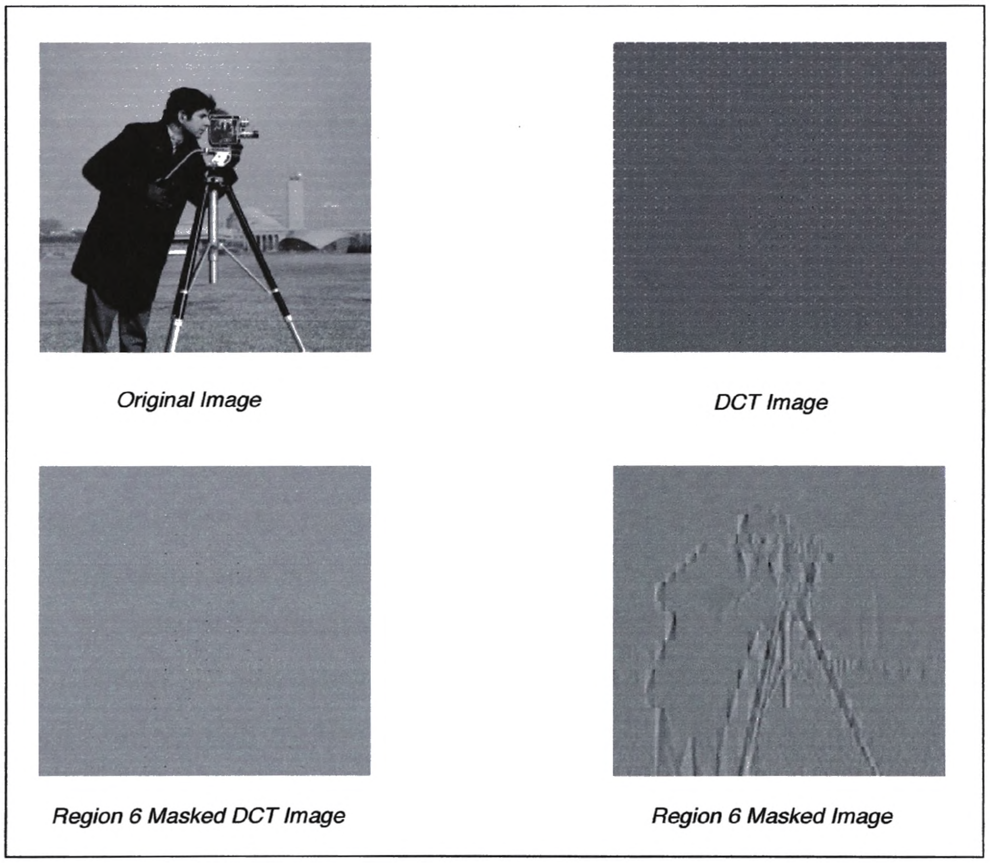


Figure 3.16: Result of applying binary mask region 6

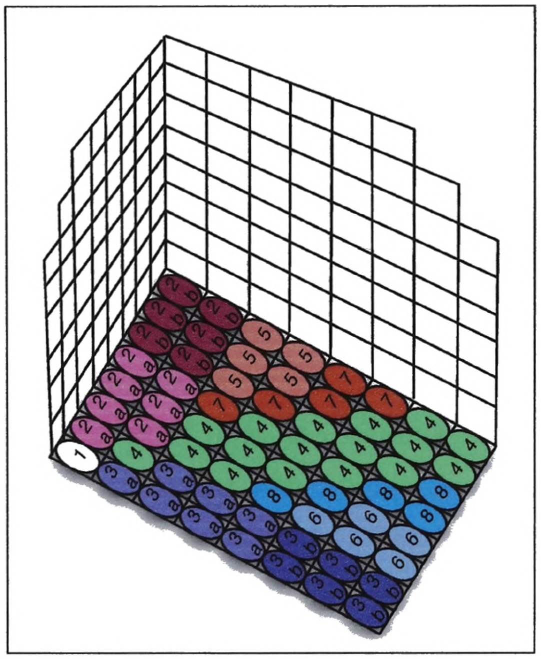


Figure 3.17: Texture space

the picture types considered to I or I, P instead of I, P *and* B, excluding the DC contribution which is considered in the next paragraph. A further restriction to consideration of only the AC terms reduces the number of bins to 72 or 36.

3.7.5 Transform domain colour histogram

It has already been noted above that the 162-bin colour portion of Smith's [256] 166-bin colour histogram is obtained in HSV space by spatially sub-sampling the colour image. In effect the DC coefficients of the DCT coefficient block supply an image spatially sub-sampled. So a simple substitution of the YCrCb relationship of equation 3.3 into the definition of the relationship of the HSV colour space with RGB from section 3.2 will allow a simple derivation of a sub-sampled HSV image that is easily obtained from the DCT image.

3.7.6 Grey level contribution

It will be noted that the grey component of the 166 bin combined histogram is derived when $R = G = B$. This suggests that 4 or 6 bins be allocated to grey level information *c.f.* Smith *op.cit.*

3.7.7 Transform domain edge histogram

Considering that a transform domain histogram is being constructed from spatial frequency information which is derived from an AC component contribution of 72 or 36 bins and a colour component contribution of 162 bins, it is useful to evaluate if edge distribution can be considered as well. There is some motivation to consider the upper limit on the number of bins as 256 since scaleable histograms can be developed through the application of the Haar [162] transform for histograms of size 2^n, *c.f.* MPEG-7 scalable colour descriptor [188]. Depending on the resolution of the spatial frequency histogram this would leave a potentially useful number of bins for this information.

3.7.8 DCT domain image analysis conclusions

The capability to extract edge images from the MPEG stream is demonstrated above in section 3.7.2. It is shown how a compact heterogeneous combined histogram can be obtained from the DCT coefficients in the video stream in section 3.7.3. There is clearly considerable scope for the evaluation of the retrieval effectiveness of the schemes proposed above. A methodology for the quantitative evaluation of the retrieval effectiveness of this approach is described in section 5.3.

3.7.9 DCT domain temporal analysis

There is clearly considerable scope to apply the methodology outlined earlier in this chapter in sections 3.7.7 and 3.7.5 and figure 3.17, to the temporal segmentation task. It is also possible to consider the temporal analysis of changes in the *spatial* distribution of edges derived from the edge image over time when used alone or in combination with the macroblock motion vector approaches discussed in section 3.5.5.1. The low computational cost of deriving the edge images and DCT coefficient histograms makes this approach an attractive proposition representing an area for future work that is discussed in the conclusions 5.2.5. Mandal *et al.* [184]

give a critical analysis of image and video indexing techniques in the transform domain that references no techniques comparable to those described here.

3.8 Conclusions

Still and moving picture processing and their associated pattern analysis techniques are an extremely active area of research as over 1700 papers were published in 1999 [233]. Most of the techniques in use for image analysis can potentially be applied to the moving picture domain. The approach here has been to adopt techniques that could be implemented within a reasonable time-frame to demonstrate the feasibility of the approach developed in the following chapters.

The analysis above on visual object recognition and visual object location serves to demonstrate that effective techniques exist to perform both visual object recognition visual and object location in still images. The relationship of these techniques to the overall AVA architecture is also discussed in chapter 5.

It can be seen from the sections discussing temporal segmentation that many classes of transitions can be automatically detected with varying degrees of success. It is often not recognised by proponents of these techniques that there is effectively an "arms race" going on in video production circles in developing increasingly sophisticated transition effects. As a result it is contended that the transition detection techniques can only be applied effectively to archive material. This is appropriate since it is this author's view that the effective use of metadata and annotation in the production cycle will result in this information being available without any requirement to compute the presence of transitions.

It will also be noted that the majority of the optical boundary detection methods have difficulty discriminating between object motion and slow transition effects. There seems to be a strong bias in this type of work towards the detection of shot boundaries and a failure to recognise that in the context of content description, object motion analysis is as significant a factor as shot-boundary detection. In effect a temporal boundary detection algorithm that is over aggressive from the perspective of strictly detecting production boundaries will in fact be detecting what Parkes referred to as changes in setting. It should also be noted that there are several examples of movies where there are extremely long sequences without any apparent visual transitions that are made up of several shots, notably Hitchcock's Rope, 1948. This brings into consideration issues surrounding temporal segmentation terminology that are discussed by Parkes [213] and Hartley *et al.* [123]. It is also not apparent why these approaches all appear to be based on first order comparisons rather than second order.

Finally a novel technique for generating dual parallel partially decoded MPEG streams has been described. This allows feature extraction and image rendering to be carried out in parallel. A novel image feature extraction technique is also described that exploits block based DCT coded images in it's approach. It is shown that this technique can be applied to one or more of the partially decoded MPEG streams obtained from the modified MPEG player. These developments are aimed at direct integration into the lower levels of the AVA architecture to provide feature extraction for visual object recognition and location, and temporal segmentation.

Chapter 4

AVA architecture and methodology

4.1 Overview

The AVA Architecture provides a coherent approach to integrating a number of technologies controlled through an effective user interface. It achieves this through an implementation based on the use of the Python [24] object orientated scripting language interfaced via automatically or hand generated, wrappers to underlying C [154] and C++ [276] libraries. The AVA architecture has supported an exploratory approach to these investigations. The developments to date have successfully integrated the majority of the technologies needed into the architecture. Where they have not been completely integrated experiments have been carried out to show the feasibility of integration. It should also be noted that within each technology area a number of alternate integration approaches have been investigated before adopting the approach described here, these are outlined in the conclusions. The technologies adopted are:

1. User interface described in section 4.2

2. Still and moving picture codecs described in section 4.3

3. Numerical processing described in section 4.4

4. Still and moving picture feature extraction described in section 4.5

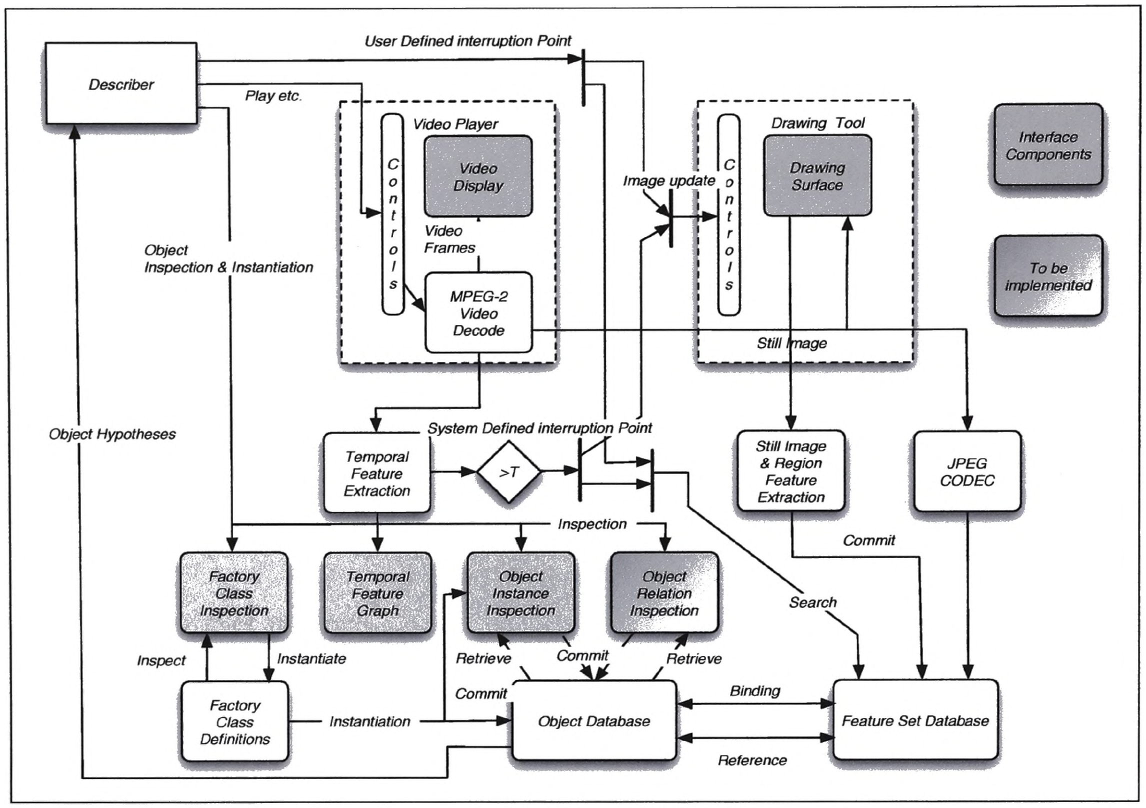


Figure 4.1: AVA architecture overview

5. Video content representation object factories described in section 4.6
6. Knowledge and feature databases described in section 4.7
7. Inference system(s) described in section 4.8
8. AVA system controller described in section 4.9
9. XML Schema serialisation described in section 4.10

The architectural approach and integration status of each of these elements is described in turn. This is followed by a description of the AVA annotation methodology in section 4.11. An overview of the AVA system components and their interrelationships is shown in figure 4.1. Major message paths are shown and visible interface surfaces are shown in grey. The distinction between manual and automatic interruption points is also shown.

4.2 AVA User Interface

Central to the AVA GUI design is the separation of concerns and realisation of those concerns as different interface components. Each is implemented as an independent object. This has often enabled each major interface component to be instantiated and tested as a standalone application before integration into the application. This has enabled a separation of control buttons, visualisation surfaces, interaction surfaces and other controls. The AVA GUI is made up of components that implement the functionality shown in the UML diagram in figure 4.2 which shows the class interrelationships. Implementation is achieved through the integration of components implementing the following functions:

1. Video player surface
2. Second player surface and image drawing
3. Controls for:
    - 3.1 Video player
    - 3.2 Drawing surface
4. Information Views on
    - 4.1 Graphing
    - 4.2 Object hierarchy inspection
    - 4.3 Object instance inspection
    - 4.4 Image feature display
    - 4.5 Temporal decomposition
5. Forms interface to description attributes

4.2.1 The video player

The video player has a full complement of conventional VCR style controls for fast-forward, rewind to start, stop, single step forward and backward; these are

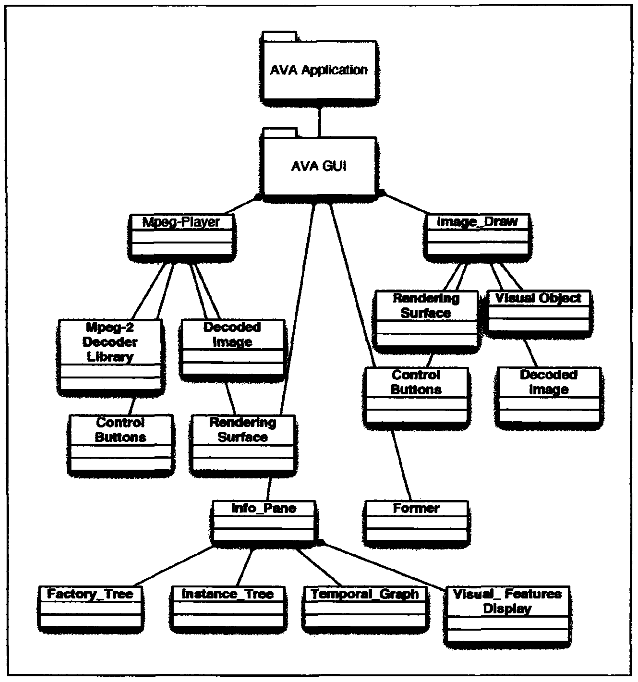
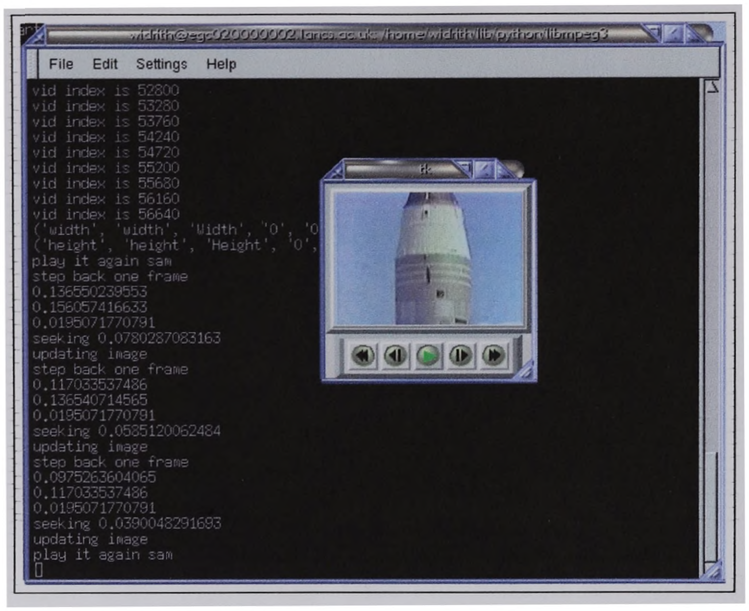


Figure 4.2: The AVA user interface

implemented as a simple state machine. The full complement of controls is necessary to give the user the opportunity to select a more appropriate frame in the video than the one automatically selected by the system when automatic interruption point selection is invoked. It is important that the selected frame is a representative frame for the existent-content set. In common with other components in the architecture this component is implemented as a set of Python [24] classes inherited from a automatically generated SWIG [23] interface around a third party MPEG-2 [145] decoder library. The MPEG decoder is described further below in section 4.3. This approach allows integration of additional codecs should the need arise. The MPEG player component is shown in figure 4.3 instantiated as an independent application. The class hierarchy for this component is not shown to conserve space.

4.2.2 The still image drawing area

The image drawing area supports manual update of the image for drawing in from the current frame displayed in the video player window when a manual interruption


widrith@egc020000002.lancs.ac.uk: /home/widrith/lib/python/libmpeg3
File
Edit
Settings
Help
tk

```
vid index is 52800
vid index is 53280
vid index is 53760
vid index is 54240
vid index is 54720
vid index is 55200
vid index is 55680
vid index is 56160
vid index is 56640
('width', 'width', 'Width', '0', '0
('height', 'height', 'Height', '0',
play it again sam
step back one frame
0.136550239553
0.156057416633
0.0195071770791
seeking 0.0780287083163
updating image
step back one frame
0.117033537486
0.136540714565
0.0195071770791
seeking 0.0585120062484
updating image
step back one frame
0.0975263604065
0.117033537486
0.0195071770791
seeking 0.0390048291693
updating image
play it again sam
```

Figure 4.3: The video player component screen shot

point is invoked. This area also allows automated image update at an interruption point determined by the system based on temporal video analyis. The main objective of this part of the interface is to allow a user to create outlines around regions that correspond to existents in the still image. After outlining the existents the initial binding of the denotative existent description to an image region is carried out by the user invoking the annotation operation. The image features are then calculated and together with the region co-ordinates they form the bound existent description. This mechanism creates the image region histograms providing bound existent description database entries for existent identification and location at later *system* determined interruption points. The user is provided with bounding box, ellipse, and polygon drawing tools. An image of the drawing component independently instantiated as an application is shown in figure 4.4. A further image showing the image drawing component integrated into the overall AVA application and the MPEG player advanced by a number of frames from the frame rendered on the drawing surface is shown in figure 4.8. Figure 4.5 shows both the factory and the instance hierarchies discussed below in an enhanced version of the image drawing application.

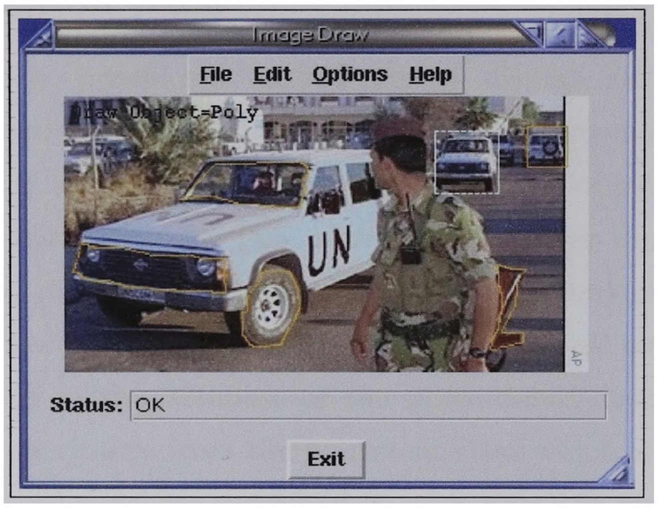
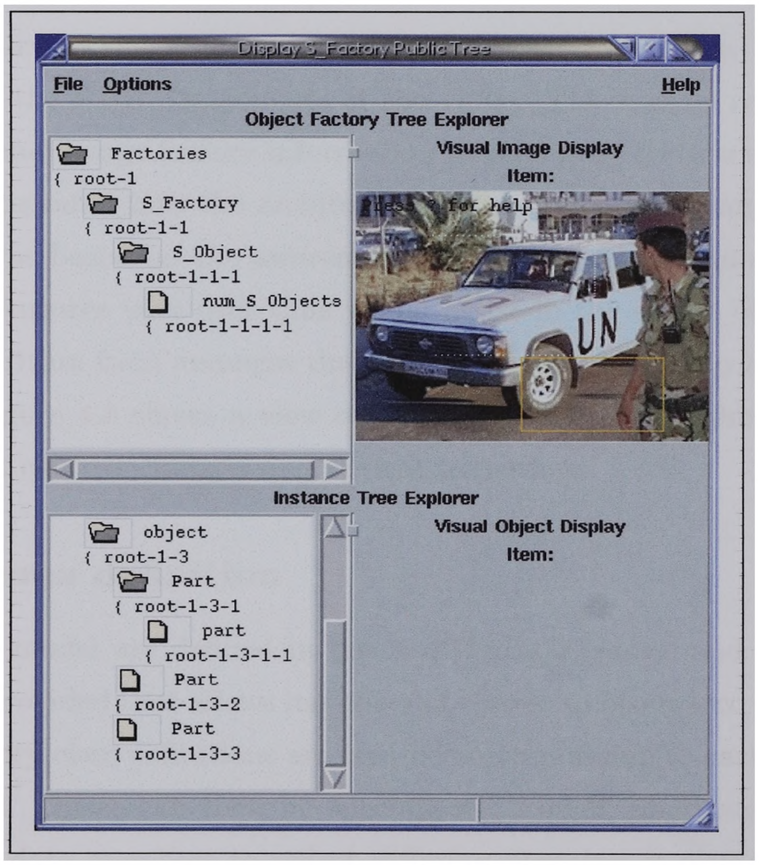


Figure 4.4: The image draw component Screen Shot



```
Factories
{ root-1
    S_Factory
  { root-1-1
        S_Object
    { root-1-1-1
            num_S_Objects
      { root-1-1-1-1
```

```
    object
  { root-1-3
        Part
    { root-1-3-1
            part
      { root-1-3-1-1
        Part
    { root-1-3-2
        Part
    { root-1-3-3
```

Figure 4.5: The Image Draw Component Screen Shot

4.2.3 Information pane

The information pane provides a tabbed area to display the graph of dynamic playback characteristics associated with the manual or automatic detection of interruption points. It also provides views of the abstract object factories and the description object instance tree. This allows the describer during the image annotation process to invoke instantiation of an abstract object instance from an object factory to create a description instance. The overall interface is shown in figure 4.6. Inclusion in the database of the instantiated description instance is achieved through inheritance from the persistent class which is part of the ZODB [269] object orientated database described in more detail in section 4.7.

4.2.3.1 Graph interface

The graph interface is intended to provide the describer with a view on the dynamic behaviour of a set of image features as they vary over time. At present this is restricted to the RGB components of the video. This can be readily extended to show additional image feature information as the image feature extraction components are integrated into the architecture. The objective of implementing this functionality was based on the awareness that more detailed human observation of how image features vary over time would assist in the choice of temporal segmentation algorithm from amongst the many that were known to be available *c.f.* section 3.5. Figure 4.8 shows a view of the graph of the Teletubby episode after several existent context changes and optical transitions.

4.2.4 Schema definition

Entities in the model are defined in terms of Python classes and a factory class mechanism is provided that allows instances of classes to be created dynamically at run time. Factory class definitions are created corresponding to each type of entity to be modelled. These can then be specialised through inheritance mechanisms and additional data members to reflect differing domains, it also allows separate indexes to be kept for instance counting. Extensions to the knowledge base are easily created by adding additional class definitions.

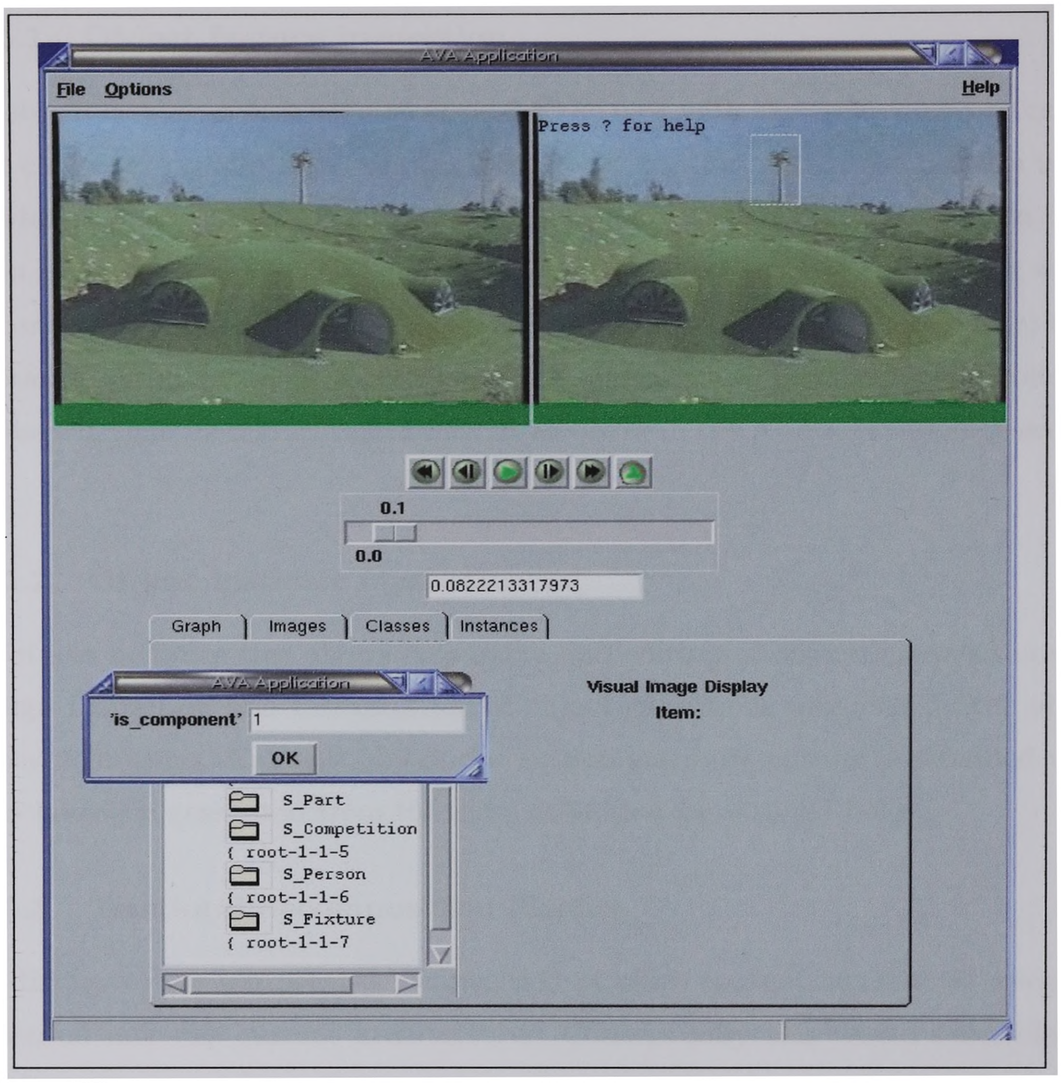


Figure 4.6: AVA interface showing object inspection

4.2.5 Knowledge base

A pair of tools to allow hierarchical browsing of the object hierarchy and the instance hierarchy are provided. These support existent context definition to image segment binding. This methodology also supports the binding of moving picture sequences to a conceptual dependency (CD) and higher level concept definitions. The object instance tree allows inspection and editing of object instance attributes.

4.2.5.1 Object factory inspection

The object factory definition view provides the user with the opportunity to review what classes are available to use for content description within the domain under consideration. Additional factory class definition files can be loaded at run time. If this is done these can also be inspected. Clicking on an object definition whilst the corresponding image region is "twinking" (shown in white in the figure) after annotation has been invoked causes an object instance to be created. The binding process will then be carried out as described below in the AVA methodology section 4.11.

4.2.5.2 Object instance inspection

The object instance tree allows inspection and editing of object instances created through inspection and instantiation of object definitions provided by the object factory definitions. Existent object description attribute editing is provided via a forms interface generated from the class definition as outlined below.

4.2.5.3 Temporal decomposition display

The display of the temporal decomposition of video content has not yet been implemented into the current architecture. Conceptually this would resemble the time line displays in non-linear video editors as already noted. It will also be appreciated from figure 4.16 that the frame references in the existent context set descriptions and existent descriptions form a series of hierarchical overlapping temporal decompositions that represent a temporal view on the same information in the object instance tree. This can be represented as a series of interval graphs [93] each representing a different decomposition.

4.2.6 The forms interface

The user interface also provides the forms interface to allow the user to modify the abstract object attributes at object instantiation time. The forms interface is provided by the "former" class, which provides the object introspection capabilities for the factory class hierarchies. The former class generates a form from selected

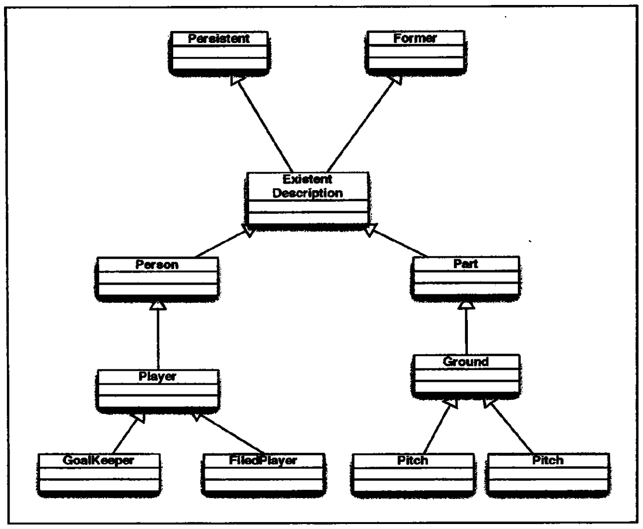


Figure 4.7: A portion of a description hierarchy

attributes of the instance object class definition These may be filled or amended by the *describer*. This means that every object in the hierarchy that has user modifiable attributes inherits from the former class. This inheritance and it's relationship to the persistent behaviour is shown in figure 4.7. Viewing of visual features can also be supported through the info pane though at the time of writing this is yet to be implemented.

4.3 Still and moving picture codecs

The architecture incorporates an MPEG-2 decoder as described earlier in the section on the MPEG player component 4.2.1, a JPEG codec and additional codecs for still image formats. The latter are provided by a wrapper around standard C libraries from the Python imaging library (PIL) [180], which provides effective image input and output. The conversion between image formats from one library to another in turn utilises functionality provided by the Python numerical library.

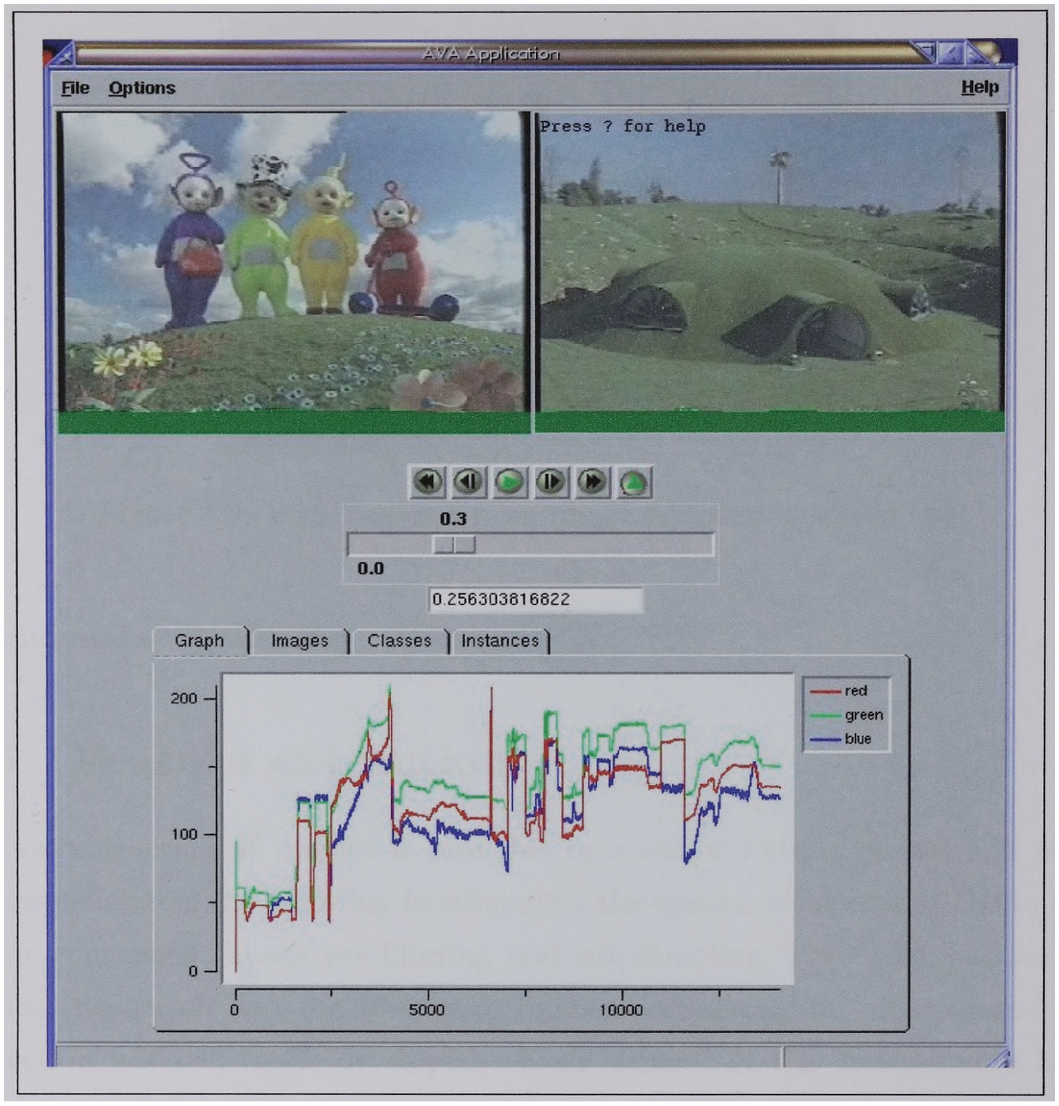
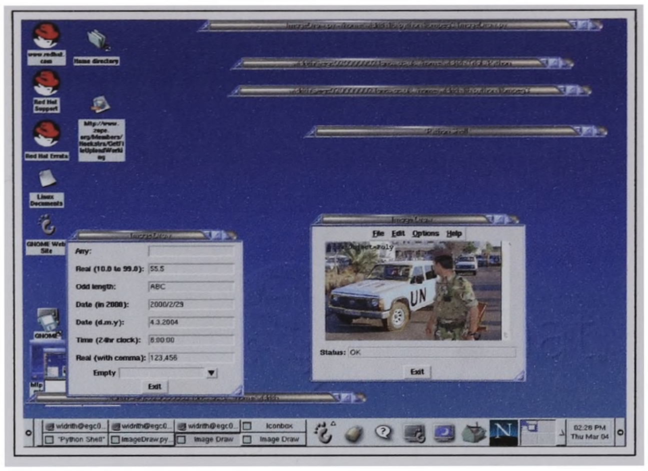


Figure 4.8: AVA user interface showing the graph pane

4.4 Numeric processing

Integral to the implementation of the architecture is the ability to incorporate effective numerical processing. Several versions of the library have been used and it is fully described in its manual [110]. This library provides a coherent mechanism to implement array processing which interfaces to the PIL library referred to above and provides convenient mechanisms to address sub arrays. It is on top of these capabilities that the image processing functionality is built. The image processing capability provides implementations of the histogram evaluation and intersection



Any:
Real (10.0 to 99.0): 55.5
Odd length: ABC
Date (in 2000): 2000/2/29
Date (d.m.y): 4.3.2004
Time (24hr clock): 8:00:00
Real (with comma): 123,456
Empty
Status: OK

Figure 4.9: Form invoked from image annotation screen shot

algorithms described earlier.

4.5 Feature extraction

Feature extraction at present is provided by a set of Python classes and methods implementing the following functionality, the continuous domain RGB to HSV colour conversion, image pre-filtering and sub-sampling, HSV colour space histogram evaluation and the histogram intersection algorithm. Histogram back-projection has still to be developed. Each element of this functionality has a separate test harness written around it to assist algorithm verification. It is envisaged that now that the feasibility of the DCT transform methodology has been demonstrated that this would be the direction of future developments. This is described further in the 5.4.2

4.6 Video content representation object factories

The current *implementation* is based on a more naive content model than that described in chapter 5. The naive model is shown in figure 4.10. Evaluation of the naive model and in particular implementation of the lower level portions of the

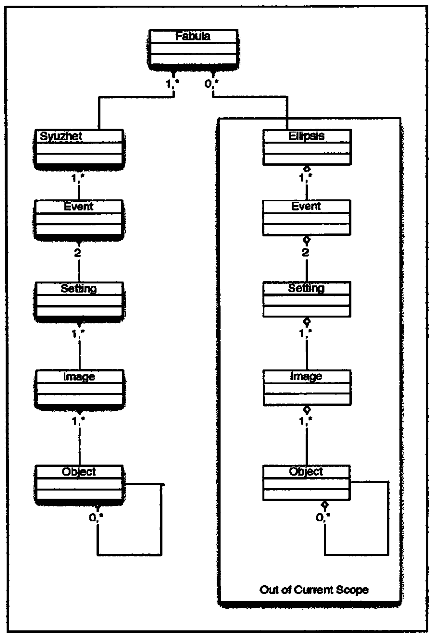


Figure 4.10: The naive AVA content model

model together with a detailed consideration of the capabilities of the low level feature extraction functionality available contributed to the reappraisal of Parkes' setting construct that forms a crucial part of that analysis. This illustrates the effectiveness of the iterative refinement approach to content modelling based on implementation and empirical observation of content and the behaviour of the image analysis methods. Underlying the currently implemented model are the extensible set of object factories that have already been described above.

4.7 Knowledge and feature databases

The database chosen for this implementation is the Zope Object Orientated Database (ZODB) [269]. The present ZODB implementation is layered on the physical implementation of the BTree storage structures. It should be noted that other physical implementations could be supported in the future such as RTrees [114] or R+Trees [246]. The overall interrelationships between the storage structures, the database connection and the persistent class within the ZODB are shown in figure 4.12. It is noteworthy that R+Trees give superior performance for spatial data and are

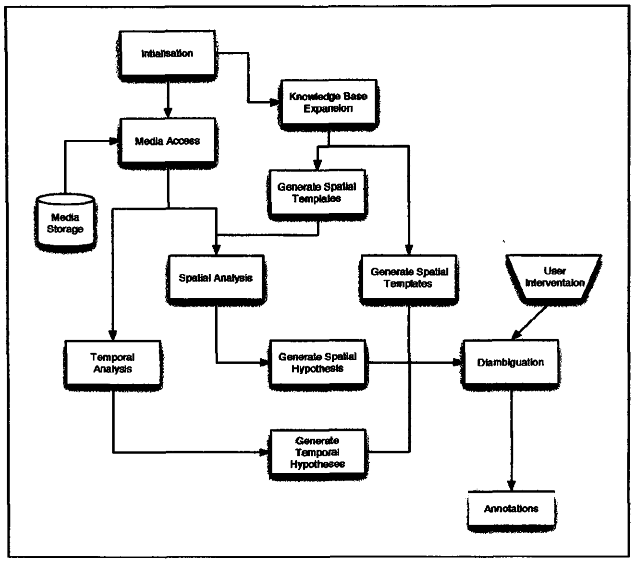


Figure 4.11: An early overview of the annotation process

widely used in the Geographical Information Systems (GIS) field[232]. The suitability of this type of physical storage of the data for video annotation requires further research. It has already been shown that video content representation needs to support both temporal and spatial information and providing optimised storage for this data will have to deal with the non orthogonal nature of time representation compared to the spatial dimensions. Conceptually the architecture needs to be able to support generic knowledge that is needed for all domains such as cinematographic rules and basic classes of entities, their attributes and relationships together with their roles in basic event structures. It must also be cable of creating instances of these entities bound to one or more feature sets. The basic approach to achieving this is given in diagram 4.11. In addition feature data can be held in a more appropriate tabular form using the efficient HDF5 [206] data storage format developed by the NCSA for large numerical data-sets. Again a python wrapper is available for this library developed by Alted [8]. Further analysis shows that the Network Common Data Format CDF file format [285] developed by University Corporation for Atmospheric Research for meteorological data is likely to be a more effective format for storage of features for time varying media. Network CDF has the capability to support multi dimensional data along a variable dimension

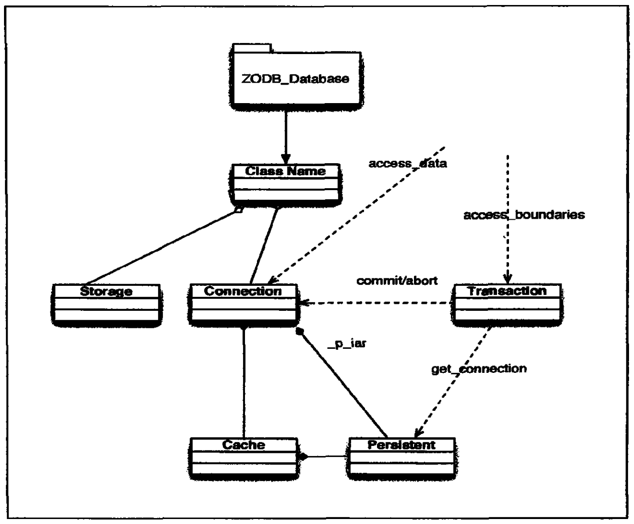


Figure 4.12: ZODB database in AVA (after Kuchling)

that generally represents time. Python wrappers are available for this library also. Preliminary investigations have demonstrated the feasibility of this approach. The adoption of a hybrid tabular and object orientated database format is discussed further in section 5.4.1.1.

4.8 Inference system(s)

Separate experiments have shown that both the CLIPS [106] expert systems shell and a Prolog [59] interpreter can be integrated into the architecture. Either of these approaches can be adopted for inference based processing. However another approach has emerged that suggests that inference over the object database may be achievable without recourse to the use of an external inference engine and the overhead inherent in passing class definitions between independent processing paradigms. This is the subject of further discusison in section 5.4.1.2

4.9 AVA system controller

At present the main system controller is in effect the event loop within the user interface controller. This in turn passes control to the video player which runs in a separate thread decoding frames and rendering them on screen until an interruption point occurs. The present implementation relies on the user making the decision about when an interruption point should occur. This is readily augmented by the adoption of a simple threshold approach based on colour difference techniques or one of the more sophisticated temporal segmentation schemes described in chapter 3. Section 4.11 gives a detailed description of the different phases in the automation of the annotation methodology.

The current implementation is multithreaded and the current design allows for additional threads controlling inference systems to be integrated into the system. It is anticipated that these would incrementally take over from the user the burden of binding instantiated existent object descriptions to feature sets as the knowledge about the narrative world of the particular video grew. This is described further in the worked Children's TV example below 4.13. It is also anticipated that further experimentation with the temporal segmentation algorithms outlined in chapter 3 can identify an algorithm(s) that can detect visual feature set transitions effectively. This is discussed further in section 5.4.3.

4.10 XML schema serialisation

An underlying assumption throughout this development has been that it should be primarily concerned with the *internal* representation and processing of still and moving picture features and content description. XML serialisation has not as yet been integrated into the implementation. However, serialisation of the object database present in the current integration should be readily achieved since serialisation of Python classes to XML is relatively straightforward as a set of wrappers exist to the Xerces [170] XML library developed by the Apache community. It has already been noted that MPEG-7 is a technology that allows exchange of metadata between applications in either text or binary formats [138].

4.11 Annotation methodology

The automation of the annotation procedure discussed in section 4.12 is considered to be capable of progressive development in five phases of increasing difficulty:

1. Manual annotation of manually segmented images,
2. Manual annotation of semi-automatically segmented images,
3. Semi-automatic annotation of automatically segmented images,
4. Automatic annotation of automatically segmented images in restricted domains with describer provided knowledge
5. Automatic annotation of automatically segmented images in restricted domains
6. Automatic annotation of automatically segmented images without domain constraint.

It will be recognised that the CLORIS prototype and AUTEUR relied on the first approach. It is argued that the AVA application demonstrator is an instance of an approach that spans stages 2 and 4. It is also recognised that the 5th and final stages are likely to remain practically unrealisable for the foreseeable future.

4.12 Annotation procedure

A brief description of the AVA annotation procedure will now be given: The annotation process is divided into two phases. The first is the definition of the factory classes for the domain under consideration, if a suitable factory class does not already exist in the knowledge base. This phase creates the *a-priori* information that the system needs to enable the creation of the description instances or annotations in the second phase. In the second phase, two annotation modalities are covered, referred to as either static or dynamic. In the static mode object instances are created through reference to ancillary information about the domain the video represents. Parkes' creation of an object relation model of a micrometer's constituent components in his use of the micrometer film, represents an example of such modelling. This component model is subsequently associated with an image area in the dynamic process described below. In the second modality object models are created by the user at video play back time along with the image area associations. The exchange of messages between objects is illustrated in figure 4.17. The video decoder decodes each image in the video file in turn. After decoding the first image, a manual interruption point is said to have occurred. The occurrence of an interruption point results in the image drawing pane being updated by the user or by the system. The correspondence between multiple interruption points in the video and key frames, is shown in figure 4.13. In addition it shows how several low level image analysis methods could be combined to augment the procedure outlined here. The next step is for the user to identify visual objects in the updated drawing area. After the interesting objects in the image have been outlined, the annotation process is initiated. The outline of each interesting object is now "twinked" to indicate to the user that this is the visual object that should now be bound to an existent factory object category. Class introspection is utilised to enable the user to inspect and modify the attributes of the existent objects as they are instantiated. This attribute modification is carried out through a forms interface that is created from the object's attributes by the former class, from which factory objects inherit this behaviour. The mechanisms that achieve this are described above in the forms section of the user interface 4.2.6

This process of assignment of visual objects to semantic object category instances is known as the binding process. In the illustration only the instantiation of a single visual object is shown. Finally each bound visual object to semantic object pair is committed to the object database for storage. Overall the AVA architecture can be considered to be an implementation of a semi-automated system to carry out binding between the physical spatio-temporal segmentation described already and the semantic spatio-temporal decomposition object hierarchy. A generic diagram of this process is shown in figure 4.15. Examples of specific instances of the process are given later. Further images in the video file are now decoded and the user may define interruption points or, interruption points may be automatically defined, according to one of the algorithms discussed in chapter 3. The definition

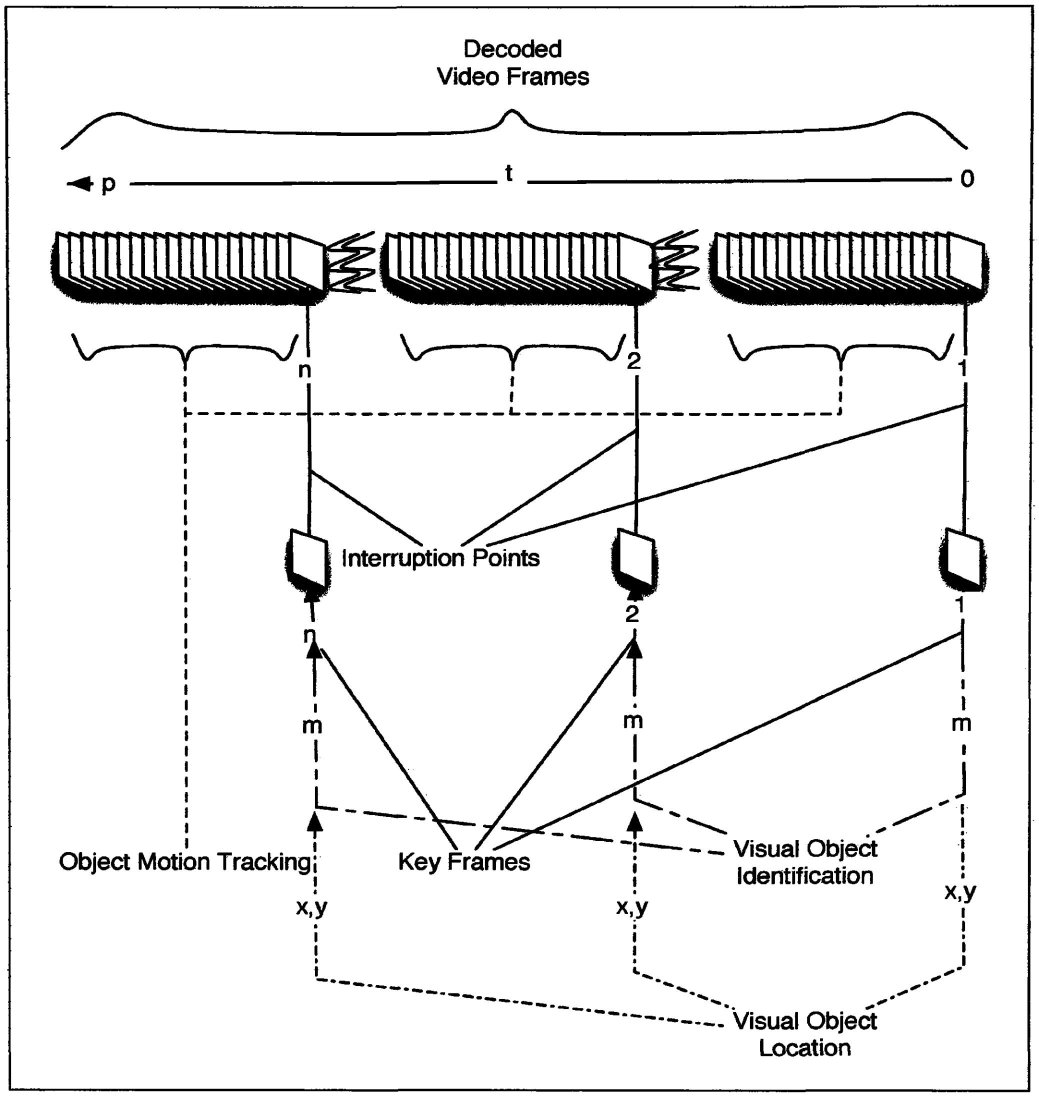


Figure 4.13: Spatio-temporal decomposition

of an interruption point results in the image drawing pane being updated, followed by object instantiation binding and commitment to storage as discussed earlier.

4.12.0.1 Definition of a semi-automated annotation algorithm

This can more formally be expressed as an algorithm as follows:

For some procedure δ that detects optical motion or optical effects let pictures p

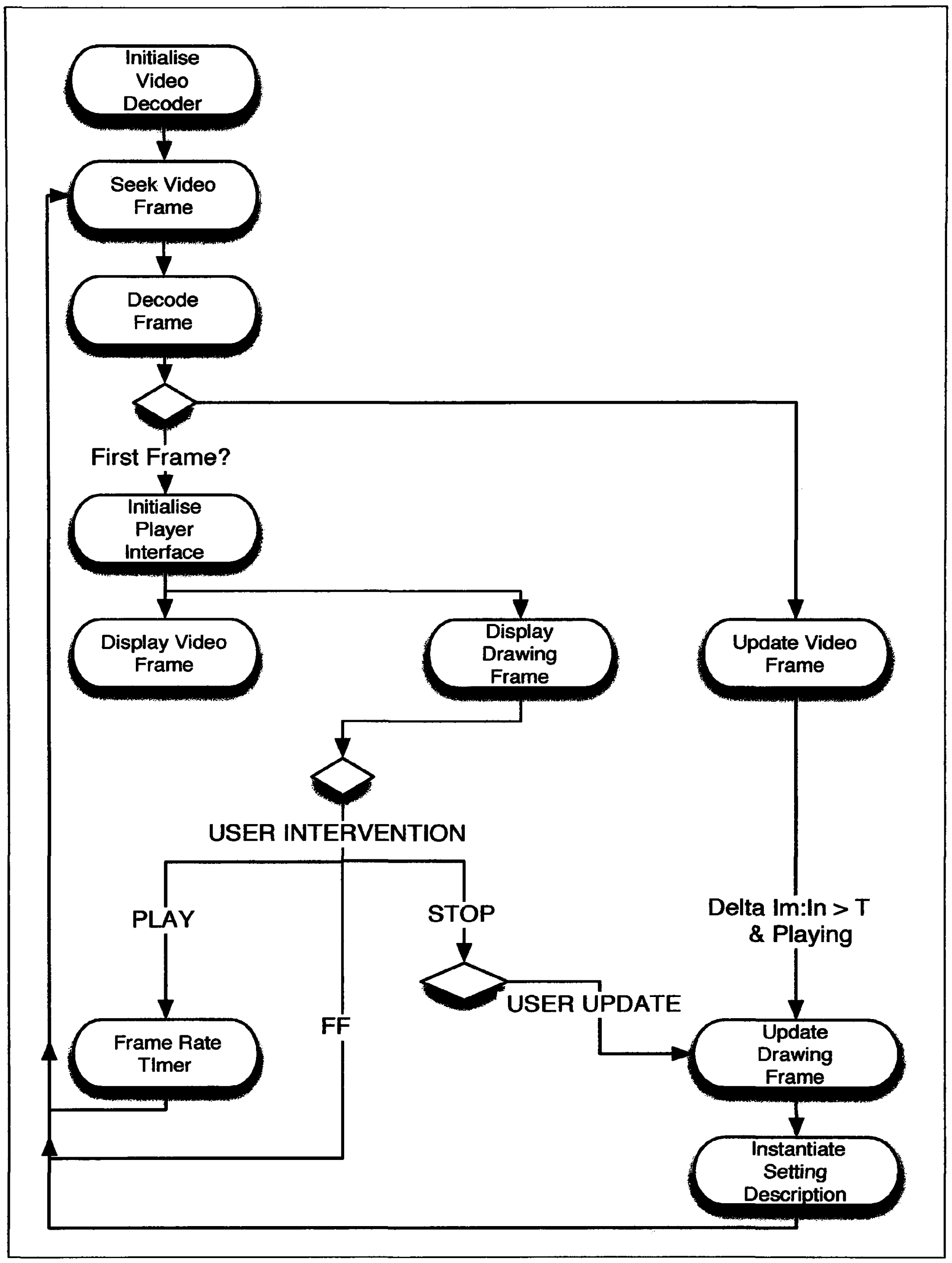
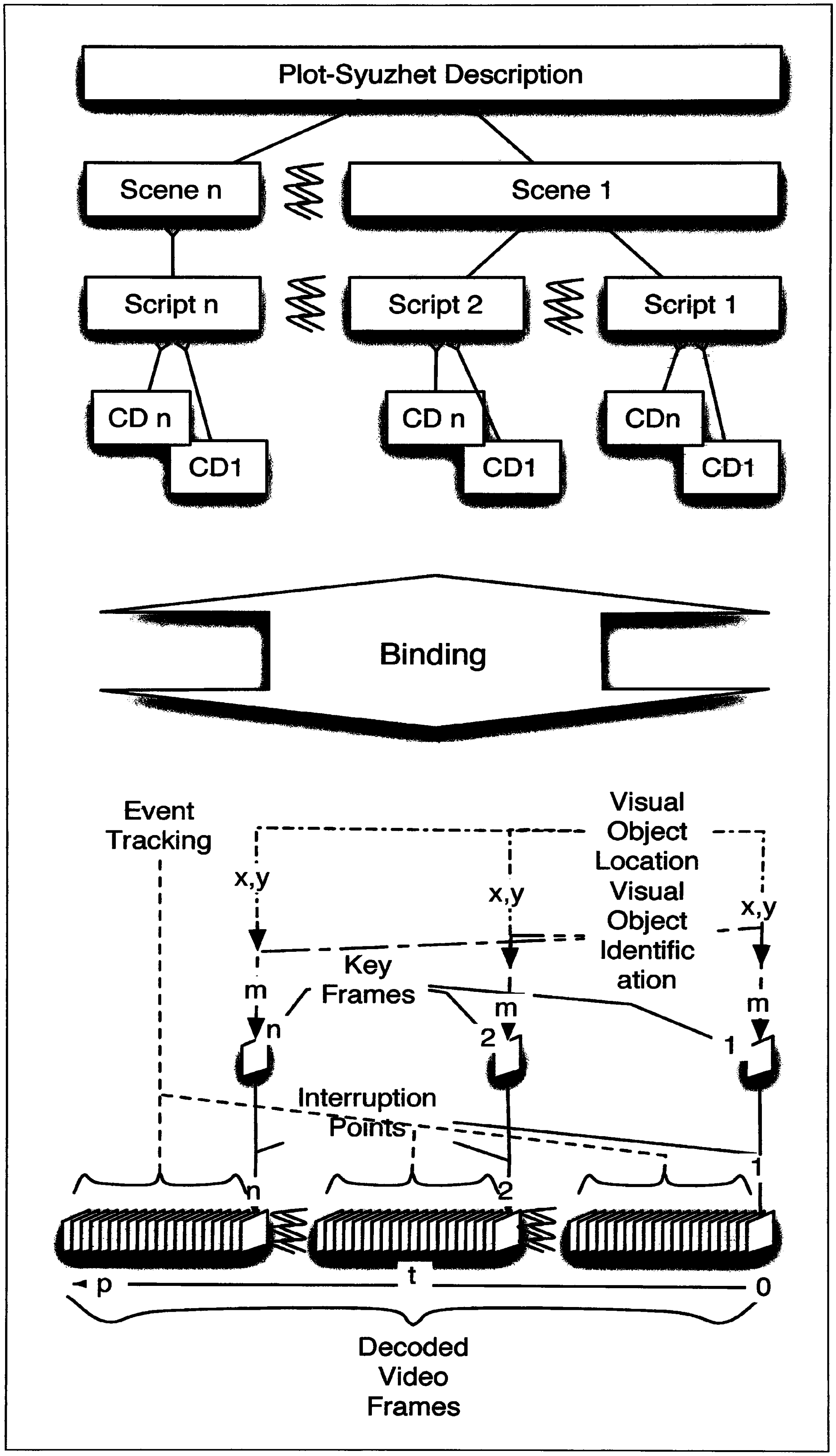


Figure 4.14: Flow diagram of the interruption process


Plot-Syuzhet Description
Scene n
Scene 1
Script n
Script 2
Script 1
CD n
CD1
CD n
CD1
CDn
CD1
Binding
Event Tracking
Visual Object Location
Visual Object Identification
x,y
x,y
x,y
Key Frames
m
m
m
n
2
1
Interruption Points
n
2
1
p
t
0
Decoded Video Frames

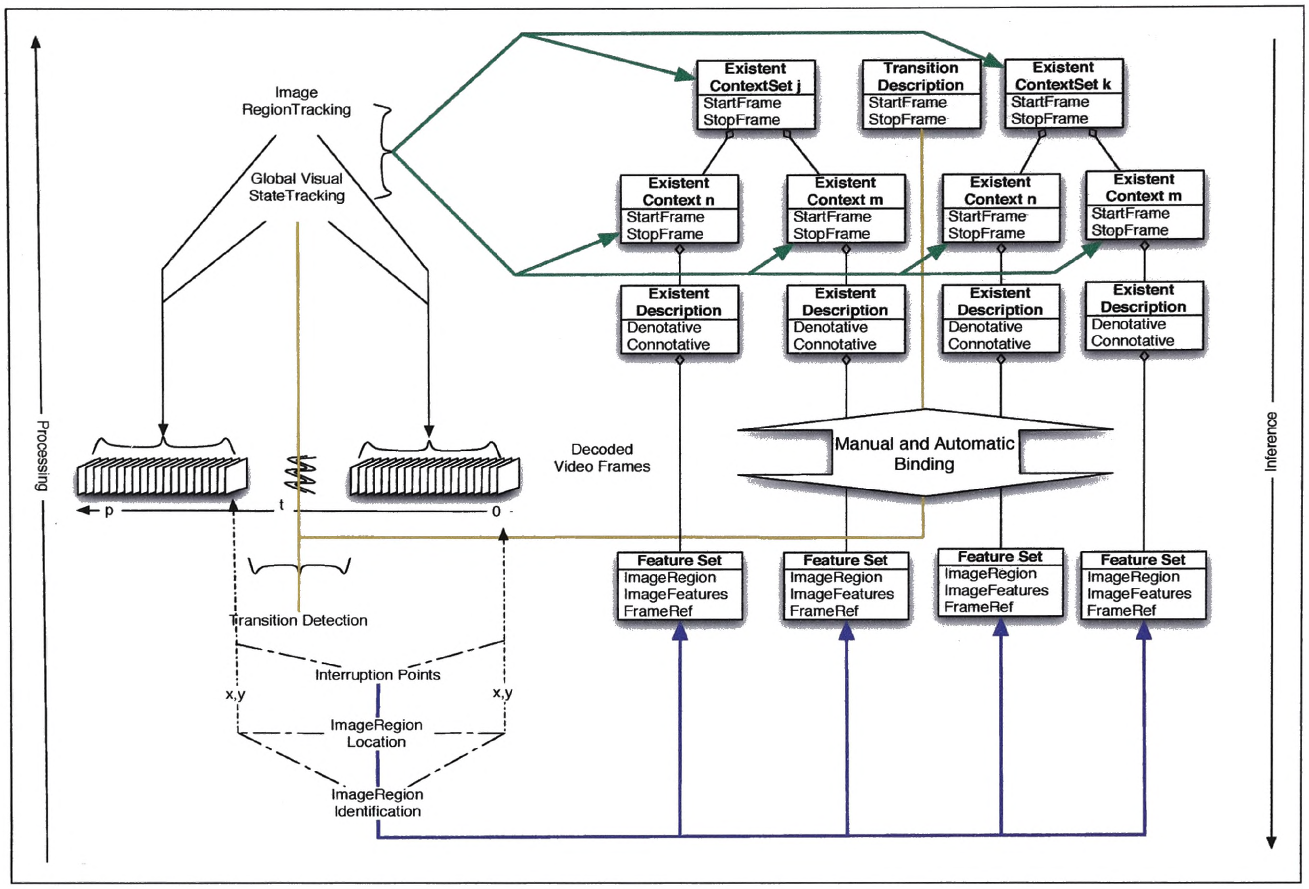


Figure 4.16: Existent feature to description binding points

be defined such that:

$p_{(1\ldots i)}$ are the pictures before the optical or existent context transition

$p_{(0\ldots j)}$ are the pictures during the optical or existent context transition

$p_{(1\ldots k)}$ are the pictures after the optical or existent context transition

where

$$i \geq 1, j \geq 0, k \geq 1$$

and

b = bound existent instances.

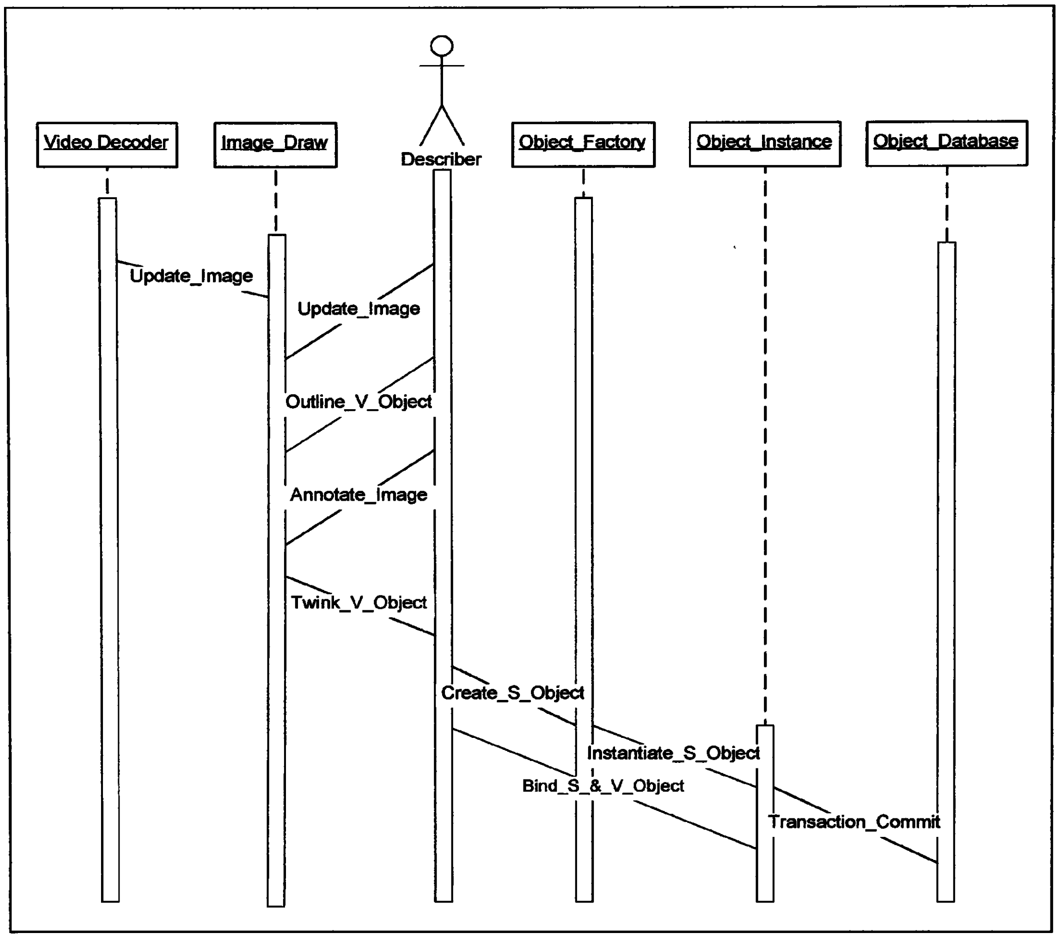


Figure 4.17: The description object creation cycle

Then

b_s^f is defined such that

s = the number of bound existents

f = the number of feature sets for existent b_s.

Then if transition detected at some picture p_k:

```
for $n = 0; n \geq s; n++$
    for $m = 0; m \geq f; m++$
        do object recognition
        if bound existent $b_n$ recognised
            do object location
            mark image pixels in $p_k$

if $p_k$ marked pixels $\leq$ threshold $T$
    invoke manual binding
```

4.13 A childrens' TV example

The crux of this discussion is the amount of world knowledge needed to support the AVA modelling approach. A short section of the "Favourite Things" Teletubbies episode made by Rag Doll productions for BBC Worldwide[1] was chosen because it decomposes into 4 similar event sequences that follow a simple story structure. This superficially simple example was chosen because it reveals many of the issues that arise when describing narrative. Also it is relatively short and as other authors such as Binsted [131] have noted it is best to begin with small steps when considering the application of AI techniques to complex problems.

The basic story structure is a quest in which the story is illustrated with key frames in appendix D. The key frames are representative of interruption points created by a temporal segmentation method. All of the teletubbies have a favourite thing. They are located together and the viewer is alerted by a *narrator* that a teletubby is lacking their favourite thing. This is emphasised by the teletubbies gesturing. There are audio and visual signs that can be interpreted as sadness. So the quest begins and the non lacking teletubbies look across 7 visible locations and

[1]The representative frames are copyright BBC/Ragdoll Productions Ltd 1996 and reproduced with permission

1 that can be inferred, for the missing favourite thing. It is found by a non lacking teletubby and the 3 meet and return the missing thing to the teletubby who lost her favourite thing. The finder then hugs the teletubby who lost her favourite thing.

The first question that arises is how much world knowledge is needed about teletubby world to enable the event sequences to be adequately represented at a reasonable level of granularity? In the following discussion of that issue a particular instantiation of the object factories as outlined in section 4.6, the feature database described in section 4.7 and the inference system described in section 4.8 is described. To begin with the straight forward information, we need to know the names of the objects and the actors:

Objects = {Hat, Handbag, Scooter, Ball }
Actors ={Tinky Winky, Laa Laa, Dipsy, Po}

The locations of the actions also need to be identified. These are arbitrarily chosen to be reasonably short verbal descriptions able to uniquely designate the locations without resorting to numbers, or letters:

Locations= { Top of Hill, Tree Line, Flower Patch, Slope of Hill, Teletubby House Entrance, Little Flower Valley, Around Teletubby House, Finders Tree, Inside TT House}

Teletubby World $\equiv$ {*Locations*}

The last location is inferred as, we never go into Teletubby House in this episode though many episodes of this series do show the interior.

What can be observed in the moving picture sequences can now be examined.

1. individual actors

2. actors in the same frame *and* close visual proximity

3. actors visually overlapping *or* being overlapped by objects

4. actors changing spatial location from frame to frame

5. actors changing overlapping relations with objects from frame to frame.

The relationships between actors' favourite thing objects are designated here as tuples:

<Po:Scooter>, <Laa Laa:Ball>, <Dipsy:Hat>, <Tinky Winky:Handbag>

An actor relationship object list of lists is an alternate implementation approach that has also been investigated.

It is when we come to interpret the meaning of the visible states that the difficulties arise and inference is necessary. Some effort is needed to identify this information formally in text but it actually does not take a great deal longer to create software class definitions for these object categories that inherit from prototype location, object and actor classes. In this analysis, since it is of necessity static, each illustrative frame is representative of an existent context or existent context-set dependent on the number of objects of interest in the frame (see also appendix E and appendix F for implementation issues).

In frames 1 to 4 in Appendix D each teletubby is conveniently framed and named by the narrator. A description of the temporal extents represented by these frames is created and manual binding of the actor existent description for each named teletubby to the boundary of the teletubby's pixels within each frame is carried out. Then the visual features are calculated and the database entries for each bound existent created. The location of the teletubby in Teletubby World must be manually annotated, using the tools discussed in section 4.2 and the annotation mechanism described in sections 4.11 and 4.12.

Frames 5 and 6 in Appendix D group the 4 teletubbies. The identification and location algorithms can be applied to these frames to locate and identify the actors automatically creating further description instances. The location of the teletubby must again be manually annotated, using the tools discussed in section 4.2 and the annotation mechanism described in sections 4.11 and 4.12.

Frames 7,8,9 and 10 in Appendix D show pairs of teletubbies. Again, the identification and location algorithms are applied to these frames to locate and identify the actors automatically creating further description instances. Here the teletubbies are sufficiently similar to previous frames for the recognition and location algorithms to succeed so annotations would be created automatically. The location of the teletubbies in Teletubby World must again be manually annotated.

Frames 10 and 11 in Appendix D show a group of the 4 teletubbies. Here again the identification and location algorithms are applied to the frames to locate and identify the actors in the frame and automatically create further description instances.

Frame 12 in Appendix D shows the title of the episode overlaid onto the landscape of the Teletubby World. This information can be utilised by the user to identify the description. The text can be described in terms of classes inheriting from the *other object* definition portion of the content model. described in chapter 2.

Frame 13 in Appendix D shows the teletubbies with their favourite things at the top of Teletubby Hill location. The associations between teletubbies and favourite things are described in terms of the conceptual dependency. The narrator points out that Laa Laa has no ball. At this frame the additional existent definitions for the Handbag, Hat and Scooter have to be manually instantiated and bound to the corresponding pixels in the frame to create their bound descriptions. The addition of these visual objects to the frame does not change the pixel counts associated with the teletubbies to any great extent. On the basis of the results reported by Smith [256] and Schiele [245], the recognition algorithms are robust enough to recognise the teletubbies, which results in automated annotation of these objects being created. This establishes the basis for the overall narrative that will now be acted out. Essentially a quest story structure is being enacted.

Frame 14 in Appendix D shows Laa Laa in medium shot with no ball. Here Laa Laa will be automatically recognised but Dipsy and Po will not. This is due to

Dipsy and Po being substantially occluded by the frame boundary, so additional feature sets will have be manually bound to these existent descriptions for the presence of these teletubbies in this temporal extent to be described.

Frame 15 in Appendix D shows the teletubbies in long shot starting to look for the missing ball. The teletubbies have rotated their positions but this does not preclude the identification algorithm from succeeding and additional annotations being automatically created since the scale of the teletubbies is similar to frame 13.

Frame 16 in Appendix D shows Tinky Winky with the Handbag at the Tree Line location. The scale of the teletubby is sufficiently similar to that in frame 1 for automatic recognition and location to succeed and a description to be automatically created. Similarly the handbag being held is sufficiently similar in scale to that in frame 13 to also be recognised. The location of the teletubby at the Tree Line in Teletubby World however, must be manually annotated.

Frame 17 in Appendix D shows Dipsy with Hat at the Slope of Hill location. The teletubby and favourite thing will be recognised on the basis that the teletubbies' scale is similar to that in frame 2 and the favourite thing's scale is similar to frame 13 albeit with rotations. Were the rotations too great additional feature sets would have to be manually added to the bound existent description definitions since the location and recognition algorithms would fail. The location of the teletubby at Flower Patch in Teletubby World must be manually annotated as before.

Frame 18 in Appendix D shows Po on the Scooter at the entrance to Teletubby House location. The scale of the teletubby is sufficiently similar to that in frame 4 for automatic object recognition to operate and corresponding descriptions to be automatically created. Similarly the scooter is sufficiently similar in scale to that in frame 13 to also be recognised. The location of the teletubby at Teletubby House entrance in Teletubby World must be manually annotated.

Frame 19 in Appendix D shows Laa Laa again with no ball. The scale of the teletubby is sufficiently similar to that in frame 3 for automatic object recognition to operate and corresponding descriptions to be automatically created. It will be noted that no favourite thing is recognised for instantiation into a CD instance. The location of the teletubby at Little Flower Valley in Teletubby World must be manually annotated.

Frame 20 in Appendix D shows Dipsy with Hat in long shot outside Teletubby House. Here the object recognition algorithm would fail since the change of scale between the earlier views of Dipsy and this frame is in the order of 4 to 1 and additional binding of feature sets to existent descriptions would have to take place for each existent. The location of the teletubby in the world must be manually annotated.

Frame 21 in Appendix D shows the missing ball at the Finders Tree location. A new manual existent description to feature set binding operation would have to take place at this frame since the ball has not been observed in the video before and no description of it instantiated. The location of the ball in the Teletubby World must be manually annotated.

Frame 22 in Appendix D shows the missing ball and Dipsy with Hat. Dipsy and favourite thing will be recognised on the basis that the teletubby's scale is similar to that in frame 2 and the favourite thing's scale is similar to frame 13. These will also be identifiable by the recognition and location algorithms and annotations automatically created. The location in the Teletubby World must be manually annotated.

Frame 23 in Appendix D shows the other two non-lacking teletubbies, Dipsy and the missing Ball. The teletubbies, their favourite things *and* now the ball will all be recognised and located by the object recognition and location algorithms and so corresponding descriptions will now be created.

Frame 24 in Appendix D shows Laa Laa with no ball and again this would be automatically annotated on the basis of similarities of scale with previous frames so a further description would be automatically created.

Frame 25 in Appendix D shows Laa Laa with the ball and the other teletubbies at the location already defined. The scale of the teletubbies is sufficiently similar to that of previous frames for the object recognition and location algorithms to be successful and corresponding descriptions to be automatically created.

Frame 26 in Appendix D shows Laa Laa occluded by the ball and the other three teletubbies. The three together with their favourite things are sufficiently similar in scale to their representation in previous frames to be recognised and located by the algorithms and corresponding annotations automatically created. Whether the identification algorithm will succeed in identifying Laa Laa with this level of occlusion needs to be further evaluated, however if it does not an additional feature set would be manually bound to the description of the teletubby as defined in section 4.12.0.1.

Frame 27 in Appendix D shows all the teletubbies with their favourite things but Tinky Winky lacks the handbag. So the scenario starts again with different actors in the roles of seekers and lacking.

In the remainder of this episode of Teletubbies the lost ball scenario is repeated for each of the other teletubbies in turn with their own favourite thing missing. On the basis of the described behaviour of the image object location and identification algorithms discussed above existent description recognition and binding operations can be carried out by the system for the rest of the sequences and so descriptions can be created automatically for them.

4.14 Conclusion

A set of tools for the annotation of video has been developed. These tools support video playback, still image interaction, description object creation, description instance creation, temporal decomposition of video, description schema creation and description instance attribute editing. The implementation integrates still and moving image codecs, numerical processing for image and video feature extraction, numerical processing, knowledge and feature databases together with capabilities to inference over the knowledge bases and search the image feature database.

The tools for video description described above are then applied to the task of automating video annotation. On the basis of the availability of the tools for video annotation just described, the content model developed in chapter 2, the still image object recognition and location algorithms and approaches to temporal segmentation of video described in chapter 3 an annotation methodology is developed and described. This annotation methodology is further specified in terms of an algorithmic definition. Through a worked example based on a relatively simple piece of video content the methodology and algorithm are shown to be capable of semi-automating the annotation process.

Chapter 5

Evaluation, Future Work and Conclusions

5.1 Overview

The major developments and analyses carried out in each of the preceding chapters are evaluated in section 5.2. This is conducted by reviewing the degree to which the relevant requirements developed in chapter 1 section 1.3.3 derived from the objectives described in section 1.3.1, have been met. Specifically this analysis is carried out as follows: the content model in section 5.2.1, the visual object location and recognition methods in section 5.2.2, video temporal segmentation in section 5.2.3, the MPEG decoder modifications in section 5.2.4, the transform domain feature extraction methods in section 5.2.5, and the architecture and methodology in section 5.2.6. In addition the development methodology is reviewed in section 5.2.7 and the implementation approach in section 5.2.8. The application of software engineering techniques is reviewed in section 5.2.9 with the development process and participation in standards development being reviewed in section 5.2.7. The psychological plausibility of the AVA content model is discussed in section 5.5

An outline of a set of quantitative analyses is given in section 5.3 which addresses the performance of the transform domain feature extraction algorithms and their application to visual object identification and location in section 5.3.1, and their application to temporal segmentation in section 5.3.2. A series of proposals for future work are made in section 5.4. Finally conclusions are drawn about the overall work in section 5.7.

5.2 Evaluation against requirements

5.2.1 Content model

The AVA content model, as described in chapter 2, was developed based on the analysis of semiotics, narrative and film theories. It overcomes the deficiencies in the CLORIS model and adapts and refines aspects of other earlier approaches. The model's effectiveness is illustrated by the worked example shown in chapter 4 section 4.11. This is not to say it represents the *only* content model, however the model has a wider range of application than other models, overcomes the conceptual problems associated with spatio-temporal decomposition in the CLORIS model and is more grounded in the semiotic distinction between denotative and connotative description, expression content and data. This makes the model more closely adapted to the exploitation of the visual object identification, location and the temporal segmentation algorithms detailed in chapter 3. It also has the potential to exploit visual object tracking and the Vitterbi algorithms that are discussed in sections 2.9.5 and 5.4.3.2. It meets the content model requirements defined in section 2 as follows:

5.2.1.1 Content model requirements

The requirements identified that the model needed to be able to represent entities at the following levels of description:

1. The narrative level.

   The narrative level is addressed through it's decomposition into events in scenes developed through the analysis in chapter 2 section 2.3 and the AVA content model development described from section 2.8 forward where the upper narrative layers of the model are developed.

2. The event level.

   The event level is modelled through the use of Schank's scripts and CD which derives from the analysis of the conceptual dependency formalism in section 2.10.

3. The visual object level.
   This is provided through the definition of the existent and existent context. The requirements also state that the model should also support description of the following:

4. Spatial relationships between entities in the model.

5. Temporal relationships between entities in the model.
   The two requirements above are met through provision in the AVA model, for mechanisms to separate location from temporal information. This is achieved by the separate use of scene descriptions and temporal decomposition of the narrative into events as discussed in section 2.9.4. Indeed it is one of the strengths of the AVA model that it emphasises this distinction.

6. Attributes of entities in the model.
   In terms of the AVA model this is provided by public data members in the class definitions forming the schema for the domain existents in the model. It is modelled by defining the data members for the existents as described in section 2.9.2.

7. Typing of entities in the model (i.e. is-a relationships).
   This is provided by basing the AVA model development on object orientated programming paradigms as discussed in section 2.9.2.

8. Compositional relationships between entities in the model (i.e. is part-of relationships).
   This is also provided through the mechanisms discussed in 2.9.2.
   The requirements state that instantiating these entities should be capable of being carried out by utilising techniques from the image-processing based analysis of images and video, so the model should also contain the following:

9. Video temporal feature references.

10. Video temporal feature cluster references.

    These two requirements can be met by binding the temporal decomposition feature references to the existent content sets in addition to the stop and start frames as in the discussion of the existent context set 2.9.5.

11. Image features.

12. Image feature clusters.

    These two requirements are met through the image feature set and the denotative description binding process described in section 2.9.1 by creating instances of bound existents. Image feature sets can exist independently of any existent in the model until an existent is defined to be bound to it.

On this basis it can be said that all the requirements for the AVA content model have been met by the developments described in chapter 2.

5.2.2 Feature extraction

Swain and Ballard [278], Smith [256] and Schiele [245] clearly show that the basic visual object identification and location algorithms are capable of operating successfully over a range of scales and rotations and in the presence of limited amounts of occlusion. These are the typical conditions that occur in moving picture content. The existent context construct described in section 2.9.3 is designed to encapsulate this behaviour. Whilst integration of these techniques into the AVA application demonstrator has yet to be achieved, the histogram intersection algorithm has been implemented and so further demonstration of this aspect of the work is a matter of *integration* and does not require demonstrable algorithm alteration. It does however alter the application of the algorithms from that of object retrieval to that of iterative knowledge acquisition. The assumption is therefore made that the change in the application of the algorithms is valid. This section addresses and meets the visual object recognition and location parts of requirement 6 for image processing based analysis of moving picture content and the corresponding part of goal 6.

5.2.3 Temporal segmentation

The wide range of temporal segmentation algorithms discussed in chapter 3 could initially give the impression that this task is readily achieved. The analysis in this chapter shows that most if not all of these algorithms fail to a greater or lesser extent in the task that they seek to achieve, namely the non guided detection of shot cuts and other optical boundaries that occur in moving picture content. It has already been identified that the cause of this failure is due to the inability of the algorithms to distinguish between slow moving visual objects and gradual optical transitions. The occurrence of false positives is considered beneficial in the current approach since the false positives represent an opportunity to test the existence of known visual objects in the scene. This effectively determines whether the same existent context set is valid at the temporal boundaries detected by the temporal segmentation algorithm used. This in turn provides the basis for experimental design to identify algorithms that detect existent context set boundaries. This is more useful from the point of view of creating fine grained moving picture content descriptions than shot and optical boundary detection which should be available from other sources of metadata. In turn this also points to the application of the Condensation and Vitterbi algorithms discussed in sections 2.9.5 and 5.4.3.2. This section meets the continuous and transform domain temporal segmentation part of goal 6 and the corresponding parts of requirement 6.

5.2.4 Video decoder modifications

The work described in chapter 2 shows that the novel modifications to the MPEG-1 decoder support the delivery of two image streams to the rendering surface. Chapter 2 also maintains that the same approach can be applied to MPEG-2 and MPEG-4 decoders albeit with increased implementation complexity. Clearly this assertion can only be verified by further implementation, however the existence of an instance of a modified decoder shows the validity of the approach. Whilst the implementation of a dual stream MPEG-2 decoder for integration into the current architecture is an implementation exercise requiring some effort it does not substantially depart from the approach already described. This is readily appreciated by considering the proposals for a hardware implementation of the decoder modifications outlined in section 5.4.4 and in figures 5.1 and 5.2. This provides novel capabilities and mechanisms to support the DCT domain visual object recognition, location and temporal segmentation parts of goal 6 and the corresponding part of requirement 6.

5.2.5 Transform domain feature extraction

These developments identify novel mechanisms to carry out edge detection, visual object location and recognition in the DCT transform domain to meet the transform domain part of goal 6 and corresponding part of requirement 6. A more extensive approach to carry out a more complete quantitative evaluation of these techniques is described below. The three sections above, 5.2.2, 5.2.3 and 5.2.4 show how the analysis and developments carried out in chapter 3 meet the requirements defined in section 1.3.3 requirement 6 for image processing based image video analysis and the objectives for continuous and DCT transform domain based analysis defined in section 1.3.2 objective 6.

5.2.6 Architecture and annotation methodology

Some discussion of the evolution of AVA application in architecture is needed to carry out it's evaluation. The current architecture has evolved from an initial concept of 8 main subsystems that were designated

1. Video and Image Codecs
2. Video and Image Analysis
3. Video Content Representation
4. Knowledge Database(s)
5. Inference System(s)
6. Graphical User Interface
7. System Controller

8. Black Board

In many respects the initial concept has proved to be remarkably robust. The main subsystems identified in this overview can still be related to the current architecture described in chapter 4. The architectural decomposition into modules has enabled subsystems to be evaluated, redesigned and independently tested before reintegration into the overall implementation. In particular this applies to some of the "development spurs" where an implementation approach that at first sight appeared promising was later found to be unsuccessful. The architecture and it's implementation in the AVA demonstrator have facilitated the reappraisal of both the naive content model and Parkes' CLORIS model as described earlier. As such the architecture and it's implementation have met one of the main goals of their design and implementation to enable content model evaluation and still provide the basis for further development and investigations.

5.2.6.1 Semantic annotation requirements

The AVA architecture implements the high level knowledge representation capabilities by providing mechanisms to support the goals identified in section 1.3.2 objective 5 which in turn provide capabilities corresponding to the requirements defined in section 1.3.3 requirement 5 as follows:

1. The high-level hierarchy should support "objective" descriptions.
   This is supported by the "public" data members of the Python class definitions for objects in the representation schema for the domain under consideration.

2. The high-level hierarchy should support "subjective" descriptions.
   This is supported by allowing free text fields to be attached to any entity in the decomposition hierarchy particularly the existent definitions.

3. The high-level hierarchy should support the decomposition video content according to the requirements in section 1.3.3 requirement 2 of the AVA model.
   This is provided through class definitions corresponding to the entities in the AVA content model.

4. The high-level decomposition hierarchy should support decomposition of video material into semantically meaningful entities.
   Again this is met through the AVA content model.

5. The high-level decomposition hierarchy should be capable of representing objects and their relationships.
   This is met through the class membership and link mechanisms discussed in the model definition section 2.9.2 and their implementation as Python classes in the architecture.

6. The high-level hierarchy should be capable of representing temporal relationships between entities.
   This is met through the mechanisms discussed in the development of the AVA content model and it's implementation in terms of Python classes described in section 4.2.5.3.

7. The high-level hierarchy should support the decomposition of moving picture content into events.
   This is modelled through the conceptual dependency mechanism and can be implemented in terms of events inheriting from a small selection of generic Python event classes.

8. The high-level hierarchy should support the decomposition of moving pictures into semantically meaningful visual objects.
   This is met by the overall AVA content model.

From the analysis above it can be seen that all of objective 5 and requirement 5 have been met by the AVA content model and are capable of being integrated into the architecture.

5.2.6.2 Software architecture requirements

The architecture has to also meet the following software architecture requirements defined in section 3:

1. Investigation into the bi-directional methodology for video annotation from requirement 1 above.
   The architecture has clearly fulfilled this requirement having provided the basis for the AVA model developments already extensively described. It is through access to an extensible and flexible implementation of the architecture that the flaws in the naive and CLORIS models were identified.

   It was identified that the architecture should also support the integration of the following capabilities defining the entities and relationships identified in requirement 2 for the AVA content model above.

2. Knowledge representation schema.
   The knowledge representation schema should support the following capabilities:

   2.1 Provision of generic schema elements for video content representation.
   This capability is provided through the Python class factory definitions described in chapter 4 section 4.2.4.

   2.2 Storage of the knowledge representation schema.
   Storage of the knowledge representation schema is provided through disk files that can be dynamically loaded into the application at run time as described in section 4.2.5. The schema definitions are inspected through the object factory inspector and the instances are inspected through the instance inspector. Storage of the schema instances is made available in the ZODB database as described in section 4.7.

   2.3 Extension of the knowledge representation schema for specialised content domains.
   This is also provided through the mechanisms descried in 4.2.4.

   2.4 Creation of of generic schema entity instances.

   2.5 Creation of specialised schema entity instances for specialised content domains.
   These two requirements are met through the ability to create instances

of the generic and specialised Python classes descried in section 4.2.4.

The architecture requirements also specify that the following user interface capabilities should be provided:

3. Moving picture decoding, playback and display.
   This is met by the capabilities described in section 4.2.1.

4. Image processing based analysis of moving picture content.

5. Storage of numeric features representing moving picture characteristics.

6. Image processing based analysis of still picture content.
   Image processing based analysis of moving and still picture content mechanisms have been identified and at the time of writing have still to be completely implemented and integrated. The architecture is flexible enough to allow integration of these capabilities when they are completed and the status is described in section 4.5.

7. Storage of numeric features representing still picture characteristics.
   Storage of image features can be supported through the mechanisms described in section 4.7 utilising the HDF-5 or NetCDF storage capabilities and again this has not been integrated.

8. Provide a user interface for interaction with content and descriptions. The introduction also defines these user interface requirements:

   8.1 Provide video playback with capabilities and controls for

      8.1.1 normal speed forward playback.

      8.1.2 variable speed forward playback.

      8.1.3 variable speed backward playback.

      8.1.4 single frame stepped playback.

   8.2 Provide frame reference display.

   8.3 Provide video still image display.

   8.4 Provide capabilities for video still image interaction.

8.5 Provide knowledge representation (k-r) schema display and exploration.

8.6 Provide k-r schema instance display and exploration.

The capabilities described in section 4.2 show that all of these interface requirements have been met.

The outline above of the capabilities implemented and investigated during the developments described in previous chapters shows that the architecture has the capability to support of the requirements identified in the introduction. Whilst not all of the capabilities have been implemented the majority have been and that integration of these capabilities is an implementation issue rather than a fundamental research issue.

5.2.6.3 AVA annotation methodology

The requirements for the AVA annotation methodology are defined in section 1 as follows: A methodology for video content description should be developed that combines the following:

1. High-level knowledge representation based decomposition of moving picture content.

2. Low-level image processing based analysis of moving picture content.

The AVA methodology described in section 4.11 meets the requirements defined above through the binding process and is formally defined in the algorithm definition in section 4.12.0.1. Overall implementation is lagging architectural design which in turn is lagging the content model development status. This is natural since the main *research* area has been content model and the associated annotation methodology development.

5.2.7 Development process

It is desirable that an evaluation of work of this nature should evaluate the development process as well as the results of the work. As with all development activities the process is inevitably bounded by both time and financial constraints. In the case of this work funding was linked by the project reviewers to participation in the MPEG-7 standardisation process. This is particularly significant since the progress of this work conflicted with the development of the MPEG-7 standard for a large proportion of the available time. This linkage had both detrimental and positive effects on the progress of this work. A few words are necessary about the MPEG process itself since this attempts to create a fusion of the formal International Standards Organisation/ International Telegraph Union (ISO/ITU) process with the Internet Engineering Task Force (IETF) approaches. In essence a key to the success of these approaches is that they are grounded in reference implementations around which the standard is subsequently written. This has much to recommend it as a standardisation process when it is able to harvest results of existing research and achieve consensus between competing commercial interests. The distinction should be made however, between harvesting existing research in a synthesis of well-known techniques, and an attempt to direct research through a standardisation agenda. A key aspect of the MPEG process is the call for technologies. An experimental model is then developed that forms the basis of the reference software. MPEG has then typically standardised the syntax and conformance points associated with bit-streams as outlined in chapter 2 for the visual parts of the MPEG-1, 2 and 4 with reference software providing an implementation of the decoder. When attempting to standardise in a domain as disparate as multimedia metadata, this process could result in the many good ideas presented during the call for technologies being assembled together in a somewhat ad-hoc manner without any coherent framework defining their interrelationships. It is this author's opinion that this is one of the outcomes of the MPEG-7 activity where the standard lacks an obvious content model underpinning the MDS [139] part of the standard.

5.2.8 Implementation approach

Realising the functionality of the architecture described here has proved to be a significant challenge, in the present context of almost continuously evolving tools, languages and environments. After a number of false starts the implementation strategy adopted has been the utilisation of the scripting language Python [24], whose environment from the users perspective has many characteristics akin to the older concept of execution and expert system shells developed in LISP [295] and Prolog [59] and POP-11 [251] in that exploratory programming is readily supported. Prolog has inference built in with it's implementation of the resolution algorithm [6] whereas LISP has typically had inferencing layered on top of it's basic list processing functionality. CLIPS adopts LISP syntax but has the RETE algorithm [98] as it's core mechanism for inference. The capabilities of the earlier systems are considerably enhanced in Python as the language was designed to provide support for object orientated programming from the outset and also supports multiple inheritance. A frequently cited objection to the use of interpreted languages is that they suffer from runtime performance penalties. Python however readily overcomes these criticisms since interfaces to C/C++, functions and libraries can be constructed utilising the SWIG [23] [26] automated wrapper generator tools. Of particular relevance to this discussion is the application described by Beazley [25] that uses a Python user interface and front end to control supercomputers carrying numerically intensive simulations of molecular dynamics. In this application domain particular emphasis is placed on the ability to rapidly reconfigure the application to investigate emerging behaviour. This is the approach adopted in the implementation of the architecture of the AVA demonstrator. Specifically a SWIG interface wrapper definition was developed around a C++ [276], MPEG-2 [145] decoder library. When processed with the SWIG tool set with Python options enabled, a C [154] wrapper file is produced which was then compiled to generate an MPEG video decoder module capable of being accessed in Python. The Python module generated in this manner has enabled a collection of video decoder based applications to be built with relative ease. In addition a third party wrapper [150] to an image-processing library[133] has been investigated in part of the work together with an interface to the CLIPS and Fuzzy CLIPS expert system shells. An added benefit of this approach is that it can be used to support future research directions into video annotation and suggests that the approach can be readily scaled in complexity.

5.2.9 Software engineering considerations

The development of the architecture of the AVA demonstrator was carried out by considering functionality at a decomposition level considerably above that of the object orientated programming class definition. Significant use of the UML [235] was made at this stage. It will be evident that UML has been used as an aide to problem understanding from the initial definition of the extensions to Eco's semiotic model described in detail earlier in chapter 2 and in the analysis of existing content models. It was also used to clarify the structure of the application during development, although it has to be said that initially the structure of the application was determined through the adoption of a combination of exploratory and extreme programming styles. Whilst this is regrettable from the perspective of formal development methodologies, it is often necessary in work of this nature, since it is impossible to consider the definition of a complete set of requirements [263] or even have available a sufficiently well constrained problem definition prior to commencement of implementation. Indeed implementation can be seen as a tool to aide requirements refinement in this context.

5.3 Quantitative evaluation proposals

5.3.1 Transform domain video feature extraction

A more extensive investigation into retrieval effectiveness of the transform domain feature extraction algorithms is clearly necessary. The basic methodology for this is described in the work of Smith [256] and Schiele [245]. It entails identifying an image database of significant size containing images of various scales rotations and levels of occlusion. A series of similarity based retrieval experiments are then carried out taking each image in the database and verifying that the correct images are returned and the statistics of resulting intersection and location values obtained. In repeating this quantitative analysis there would be the opportunity to compare the effectiveness of the transform domain algorithms with the continuous domain algorithms. Since retrieval effectiveness analysis forms a significant part of Smith's and Schiele's investigations and was not the main direction of this work it is left for future work to carry out this exercise. This approach can be applied to the colour features extracted in the transform domain i.e. the steerable Gaussian proposals in section 3.7.3, the choice of DCT regions in the region based image analysis proposals in section 3.7.4 and the analysis of edge distribution in the edge images in section 3.7.7.

5.3.2 Transform domain temporal segmentation

The approach to evaluating the effectiveness of temporal segmentation algorithms has typically involved a manual assessment of where the temporal segment boundaries and optical transitions occur. This is then followed by the application of one or more of the temporal segmentation algorithms to the same piece of content followed by a comparison of the boundaries detected. This can readily be achieved and the approach applied to the evaluation of the application of the transform domain features described in chapter 3. This is facilitated by the AVA architecture's inherent support for manual temporal segmentation of moving picture content.

5.4 Future work

5.4.1 Database extensions

Three aspects of database implementation have considerable potential for further exploitation in relation to the storage of content descriptions and image and temporal features. These are object-relational hybrid databases, object-tabular hybrid databases and the integration of inference systems into object databases.

5.4.1.1 Hybrid database

There is considerable interest in the literature at present in hybrid relational object databases see Stonebreaker [275] for a general overview. Considerable relevant work has been done in this area that can be applied to automated video annotation. In particular the application of an XML representation of temporal intervals [176] and the management of temporal intervals in [116] are of particular interest. In addition because of the potentially large amounts of image feature data that may need to be stored the use of HDF5 tabular storage [206] has already been considered and shown to be capable of integration into the current AVA architecture. There is clearly considerable scope for further investigation into this aspect of system implementation.

5.4.1.2 Inference in object orientated databases (OODB)

The capability of adding functional programming formalisms to OODB was discussed by Hillebrand *et al.* [127]. A beneficial side effect of the objects in the class hierarchy inheriting from the former class is that this provides mechanisms for object reference, and object attribute introspection, allowing the states of the object attributes in the database to be inspected and modified. This mechanism can be extended to support reasoning over the objects in the database so that rule based processing based on the state of the bound descriptions that are members of the current existent context could be implemented with the rules being fired when an interruption point occurs. This would remove the necessity for the use of an external rule based processing engine of the types already referenced. This is beneficial since an external inference mechanism inherently implies multiple representations of the same description fragments and introduces concurrency issues.

5.4.2 Further integration

It has already been noted above that MPEG-7 [188] and SMPTE [260] have been developing standards for content description interchange. This interchange capability has not yet been integrated into the AVA architecture. It is an essential further step in this development that such an interchange capability is added to the architecture in order to effectively exploit the descriptions created by the demonstrator application.

5.4.3 Existent, existent context and context set identification

5.4.3.1 Existent descriptions and relationships

In chapter 2 section 2.9.2 whether the mechanism for part and sub-part description decomposition in existent descriptions would be defined by the content *describer* or automatically determined, was left open. The transitivity tables described by Parkes [215] provide inference rules that allow the relationships between CLORIS settings to be inferred from their visual interrelationships. In effect the table represents a set of visual states that have transitions between. This in turn suggests the application of Finite State (FSM) or Hidden Markov Model (HMM) models to state identification and transition in a transitivity table. In turn this would allow the part sub-part relationships between Existent Descriptions to be semi - automatically determined. The issue of whether the sub-part feature set to super-part existent binding should be achieved explicitly or inferentially can also be more effectively addressed if reasoning over the object and feature set databases is available directly. Clearly further work is needed in this area to establish the viability of this approach.

5.4.3.2 Existent contexts

Establishing the temporal extent of an existent context suggests an approach to temporal segmentation akin to that used in speech recognition and natural language processing where forward backward probabilistic algorithms such as the Viterbi algorithm have been used. In speech these are used to establish the temporal extent of phonemes and in natural language processing they are used to resolve the parse with the highest probability amongst a set of possible sentence parses.

5.4.3.3 Existent context set

It has already been shown that edge images can be obtained from the MPEG stream, the manual binding process can effectively assign a set of edge boundaries

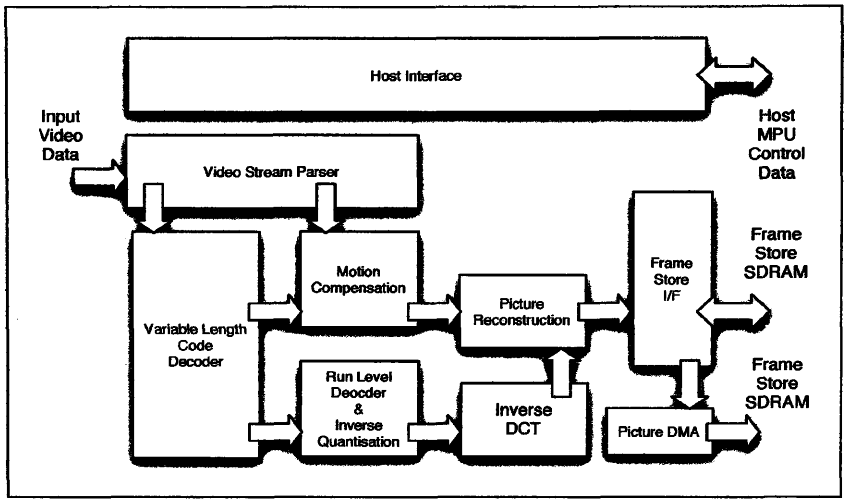


Figure 5.1: ASIC core for MPEG-2 decoding

to an image region and the relationship to a denotative object description has already been discussed in section 2.7.3 and section 2.9.3. This suggests investigating the application of active contour condensation approaches such as that described by Blake and Isard [34]. This is based on affine transformations of object contours so these algorithms have the potential to be applied to the problem of tracking object translation and deformation in the visual context set.

5.4.4 Hardware implementation

The modifications described in chapter 2 to the MPEG-1 decoder and the algorithms described in chapter 3 for transform domain edge image and feature extraction, naturally lend themselves to hardware implementation. Since field programmable gate array cores are commercially available for MPEG-2 [10] and MPEG-4 decoders [9], the methodological approach to the modifications to the MPEG-1 decoder described in chapter 3 can also be applied to a hardware implementation of the MPEG-2 and MPEG-4 decoders. This is illustrated conceptually in a comparison of figure 5.1 with figure 5.2. This should clearly not be undertaken before the quantitative evaluations proposed in section 5.3 have been successfully completed.

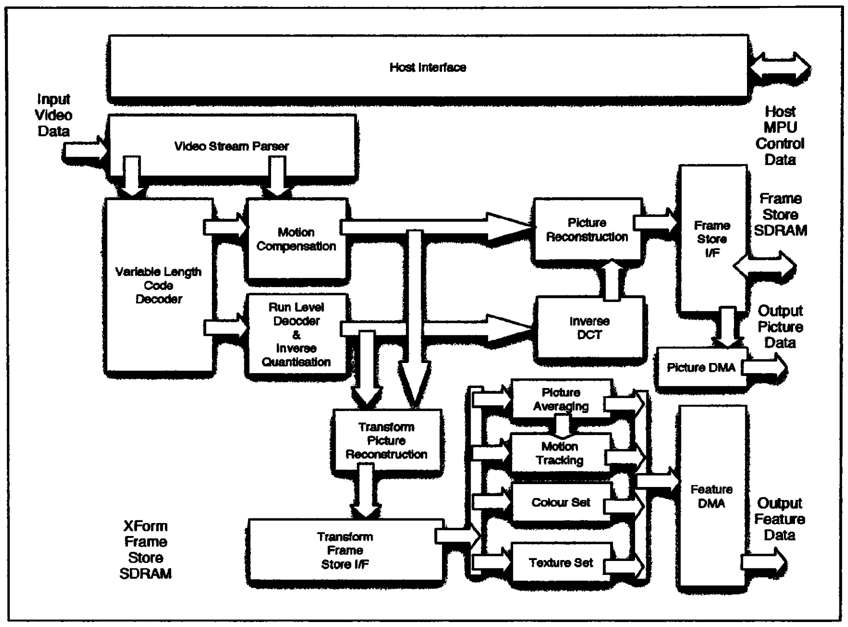


Figure 5.2: ASIC core for MPEG-2 decoding and feature extraction

5.5 Psychological plausibility of the model and fNMRI

Current functional nuclear magnetic resonance imaging (fNMRI) and positron emission tomography (PET) techniques provide support for the view that the AVA architecture is at least psychologically plausible and as such, represents a potential cognitive architecture for describing visual scenes in closed domains. A useful overview of functional neuroimaging technologies and associated areas of active research is given by Cabeza and Kingstone [50].

It is widely accepted in biological psychology [66] and in theories of human vision, [190] that specialised groups of cells are responsible for different functions within the human brain. This specialisation in processing functions allows parallel processing of different types of visual information. Therefore aspects of vision are perceived at the same time, such as depth and orientation, upward and downward, and leftward and rightward motion. It has also been demonstrated that the perception of colour and motion [301], objects and faces [152], are all perceived in different areas of the brain. Although the case for specialisation between faces, and facial expressions is less clear [222], it is notable that the individual perceiving the stimulus is not consciously aware of these differences except at times of high stress, as noted by Leach [168]. The mental imaging of faces has been shown to involve different brain areas to those involved in the mental imaging of places and objects [270] . There is an interesting link between the imagining and imaging of faces and places where imaging activates the same stimulus-specific brain regions as imagining faces and places. [210]. It has also been shown that the areas that image motion are also involved in imagining motion [108]. This linkage process is further reinforced by work that shows that eye movements made when an image is initially perceived are reproduced when the image is subsequently imagined. These observations provide good support for the arguments propounded by Schank in his Conceptual Dependency [238] theory and contributed to the development of the CD related aspects of the model. Conway's [61] fNMRI results on personal episodic memory as the basis for long term storage of life events also point to support for this approach.

5.6 Standards for content description exchange and markup

So far the development of the AVA model has been concerned with the *internal* representation of content. There is clearly a role for *external* representation of the models. Various technologies have been evaluated for this role. The adoption by MPEG of XML schema as the language for description definition resulted from an analysis of the then available technologies for document markup and knowledge representation. The annotation of text has a long history from scholarly jottings in margins noted by DeRose and Durand [72], to the development of Goldfarbs Generalised Markup Language (GML) with IBM, followed by the standardisation of Standard Generalised Markup Language) (SGML) [136] and the use of SGML for text document markup is described by van Herwijenen [287]. The development of eXtensible Markup Language (XML) [290] from SGML is discussed in detail by

Bradley [42] and Pitts-Moulis and Kirk [221]. The motivation for this development was largely due to the difficulties encountered in parsing SGML. The relationship of HTML to SGML is well known as are uses, capabilities and impact that HTML [291] has had in recent years which will not be reiterated here. In addition there is the known problem that extensions to HTML introduce presentation capabilities in addition to the desired representation capabilities. XML was developed to overcome the difficulties that implementers had faced when developing tools for SGML-based document processing. XML schema fulfils the dual role of providing some of the capability of a Document Type Definition (DTD) described by St Laurent and Bigger [166] in particular it allows the definition of types in the conventional computer science sense, in contrast to XML's support solely for character types. The need to support the definition of types other than characters should be self evident from the earlier discussions. The Dublin Core Metadata [77] set of elements have been incorporated into schema defined by both MPEG and SMPTE for multimedia information exchange.

In addition to the activities of the bodies identified above, the broadcast and the motion picture industries have also been independently active through the aegis of the European Broadcast Union (EBU) and Society of Motion Picture and Television Engineers (SMPTE). The SMPTE metadata framework published in a joint EBU SMPTE report [167] and the subsequent development of MXF [257] represent some of the outcomes of these activities. It is also symptomatic of the tendency to fragmentation that initially developed between the media industries, the MPEG-7 standard in its effort to be extensible and comprehensive, and the immediate needs of commercial practitioners to manage and exchange media. Now portions of MPEG-7 are being implemented in MXF and vice-versa. A prior attempt to develop a standard for time based media mark-up is HyTime more formally known as the HyperMedia/Time Based Structuring Language which was an earlier ISO activity [137]. This language has suffered from difficulties associated with parser development that can be largely attributed to its conceptual origins in SGML rather than XML. The specification defines the capability of HyTime and a discussion of its capabilities are given by DeRose and Durand [72]. Other notable initiatives were the Knowledge Interchange Format (KIF) [105] and complementary

Knowledge Query and Manipulation Language (KQML) [92]. These were attempts to standardise mechanisms for knowledge interchange between expert systems and to define a common query mechanism for agents. KIF can now be considered to be an RDF schema instance and KQML to be a language for agent communication.

It is also interesting to note that there has been some effort to standardise a textual notation for Sowas [268] conceptual graph formalism. There is a relationship between this effort and the development of the Semantic Web that is discussed by Berners-Lee in one of his roadmap articles [30]. In conclusion, there are a wide number of choices for meta-data interchange. These standardisation activities say little about the internal structure of the content they attempt to represent or whether any internal model of the content is available to the application. This leaves the application free to develop it's internal model as appropriate. The internal representation used by the application can then be serialised in an appropriate standard form.

5.7 Conclusions

A novel set of modifications to an MPEG-1 player that support dual image stream decoding have been developed in section 3.6. It is shown that these modifications can be extended to MPEG-2 and MPEG-4 decoders and implemented in software or hardware. Novel DCT transform domain edge image decoding is described in section 3.7.2 and novel transform domain feature extraction is described in section 3.7.3. These have been simulated for the still image case. It is shown that the MPEG player modifications allow edge images and image features to be decoded from the MPEG stream with improved efficiency over the continuous domain approaches normally adopted. A thorough review of semiotic narrative and film theories relevant to moving picture content description has been carried out. This results in novel revisions and extensions to the semiotic model which are described in sections 2.8.1, 2.8.3 and 2.8.4. These models are then applied to still images and moving picture narrative. The computational application of narrative theory to the problem of content description supports the development of a coherent and novel model of moving picture content. The AVA model developed in chapter 2 is capable of describing content from the narrative level to the image feature level. The novel AVA content model incorporates several new constructs such as: the existent description in section 2.8.6, the existent context in section 2.9.3, the mechanism of "binding" in section 2.9.1, the existent context set in section 2.9.5. The existent context is capable of representing complex relations between visual objects in a manner comparable with Sowa's conceptual graphs [267]. A novel mechanism for representing moving picture actions in scenes is developed in section 2.9.7 based on Schank's [238] conceptual dependency and script formalisms [243] given in appendix 2.10. The AVA model has a substantially greater content breadth over which it can be applied than previous implementations, and a description of the novel AVA application demonstrator is given. The AVA demonstrator has several substantial areas of functionality which it combines in a novel manner, with a novel user interface. The implementation of the AVA architecture contributed to the development of the AVA content model and the architecture and is robust, flexible and extensible. It is envisaged that the AVA implementation will form the basis for further experimentation into moving picture content description. A novel methodology for semiautomatic moving picture annotation is proposed and defined as an algorithm and a worked example provided that shows the applicability of the technique. The implementation completed to date shows that a content description author's tool can be developed from the approach adopted for the demonstrator. Considerable scope for further development and research has been identified and outlined. It has therefore been demonstrated that the work substantially meets the goals, objectives and requirements defined in chapter 1.

Appendix A

MPEG input documents

A.1 Technology Input Documents

A review of MPEG-7 Terminology. Hartley E., Nack F., and Parkes A. P., ISO/IEC JTC1/SC29/WG11 M2634, MPEG 41, Fribourg CH

Introduction of MPEG-7 Systems Requirements Document Hartley E., Ohm J.R., ISO/IEC JTC1/SC29/WG11 M4006, MPEG 45 Atlantic City US

Report of the Ad-hoc Group on MPEG-7, Requirements, Nack F., Hartley E., ISO/IEC JTC1/SC29/WG11 M4011, MPEG 46, Rome IT 1998

Report of the Ad Hoc Group on MPEG-7 Requirements, Nack F., Hartley E., ISO/IEC JTC1/SC29/WG11 M4291, MPEG 47, Seoul KR 1999

Considerations for DS Development (m2634 revised), Hartley E., ISO/IEC JTC1/SC29/WG11 M4879, MPEG 48, Vancouver CA 1999

Tutorial on the SMPTE Metadata Dictionary, Hartley E., Wilkinson J. ISO/IEC JTC1/SC29/WG11 M5201, MPEG 49, Melbourne AU 1999

DDL Parser Development Status Notes, Hartley E., ISO/IEC JTC1/SC29/WG11

M5202 , MPEG 49, Melbourne AU 1999

Report of CE on Semantic DS, Bugatti A., Benitez A., Mehrotra, R., Hasida K., Rising H., Joergensen C., Leonardi R., Hartley E., Tekalp M., ISO/IEC JTC1/SC29/WG11 M6355, MPEG 53, Beijing CH 2000

Linguistic DS Considerations and Recommendations, Hartley E. and Rayson P., ISO/IEC JTC1/SC29/WG11 M6797 MPEG 54, Pisa IT 2001

Proposal for Editing and Revision of MDS draft, Hartley E., ISO/IEC JTC1/SC29/WG11 M6822, MPEG 54, Pisa IT 2001

MPEG-4 Reference Software Architecture Working Draft, Chang W., Purnhagen H., Hartley E., Han M. and Lifshitz Z.,ISO/IEC JTC1/SC29/WG11 M6944, MPEG 55, Singapore SG 2001

MPEG-7 Report on CE on Graph Rules, Classification Schemes, and Restrictions, Rising H.K. and Hartley E., ISO/IEC JTC1/SC29/WG11 M7138 MPEG 55 Singapore SG 2001

A.2 Procedure and Process Input Documents

Report of the Ad-hoc Group on MPEG-7 Evaluation Logistics Vetter M., Hartley E., and Salembier P., ISO/IEC JTC1/SC29/WG11 M4582 MPEG 47 Seoul KR 1999

UK National Body Comments on the structure of MPEG, for MPEG-7 Development Hartley E., ISO/IEC JTC1/SC29/WG11 M4852, MPEG 48 Vancouver CA 1999

UK National Body Comments on DDL Development, Hartley E., ISO/IEC JTC1/SC29/WG11 M4851 MPEG 48 Vancouver CA 1999

UK National Body Position on Provision of Web and FTP Services, Hartley E. (For UK National Body), ISO/IEC JTC1/SC29/WG11 M5496 MPEG, 50 Maui US 1999

UK National Body Position on Source Code for Normative and Non Normative Tools Hartley E. (For UK National Body), ISO/IEC JTC1/SC29/WG11 M5494 MPEG 50 Maui US 1999

UK National Body Position on Document Management Technologies, Hartley E. (For UK National Body), ISO/IEC JTC1/SC29/WG11 M5495 MPEG 50 Maui US 1999

UK National Body Position on MPEG-7, Hartley E. (on behalf of UK National Body) , ISO/IEC JTC1/SC29/WG11 M6821, MPEG 54 Pisa IT 2001

Appendix B

Formal definition of the CLORIS model

Parkes content model was originally defined in terms of a number of formal definitions, assumptions and corollaries that are extracted from [213] and reproduced below:

Parkes' Definition 1. *a **sequence** is a succession of* $\geq$ **1** *scenes forming a subordinate unity, within a film, of conception and purpose.*

Parkes' Definition 2. *a **scene** is a succession of* (≥ 1) *shots depicting events which are taking place in some constant location, and over a continuous period of time.*

Parkes' Definition 3. *a **shot** is a succession of* $\geq$ **2** *frames depicting events which are taking place in some constant location, and over a continuous period of time.*

The setting formalism is developed by Parkes (1988), to provide the basis for his spatio-temporal hierarchical decomposition of film, in preference to the shot, sequence or scene, largely because of the ambiguities inherent in the use of these terms. The **setting** is initially defined as follows.

Parkes' Definition 4. *a **setting** in a moving film is the unit of film associated with the longest time-interval over which the visual content of the film can be objectively described by the same conjunction of formulae.*

Parkes' Definition 5. *a **setting description** is such a conjunction of formulae.*

This defintion is subsequently generalised to:

Parkes' Definition 6. *a **setting** is a set of images sharing the same objectively describable visual state. The setting description is the objective description.*

These definitions depend on a number of assumptions about the decomposition of moving picture content. Parkes' contends that on this basis the shot disappears from view in terms of the description of events. As events map onto settings; settings are constituents of scenes and settings share location characteristics with scenes. This leads to the development of the structure shown in figure 2.15. Parkes relies on Carrolls [53] structural analysis in developing his model.

Parkes' Assumption 1. *the narrative film is realised as* sequences *of* scenes, *each of which is a series of shots.*

Parkes' Assumption 2. *the nature of the realisation of event structures by film sequences makes the following requirements of a CCV description language.*

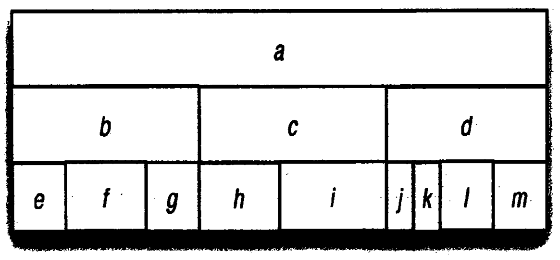
1. ***(Paraphrase)** The language registers the commonality (i.e. equivalent event structures) of diversely realised film sequences*

2. ***(Ambiguity)** (Components of) film sequences are described in such a way as to facilitate (a) their use in film sequence realisations of diverse event structures, and (b) their discussion as being representative of diverse event structures.*

3. ***(Deletion)** The overall narrative is described in such a way that it caters for the fact that events in the overall narrative may not feature visually in a film sequence realisation of that narrative.*

Parkes' Assumption 3. *The setting is the **minimal described unit of the film sequence at the level of events** i.e. the constituent below which descriptions at the level of events are not attached.*

Parkes' Assumption 4. *To state that an event, $\boldsymbol{e}$, takes place over a series of settings, $\boldsymbol{s_1, .. s_n}$ when they are displayed in a moving film implies:*



$E = \{a, \ldots m\}$

Where each rectangle is a time interval

And each letter is an event description proposition

WED(E) = a, and the disjoint non-empty subsets which are all abstract event descriptions are:

$E_1 = \langle \{b,e,f,g\}, s, f, m, d \rangle$

$E_2 = \langle \{c,h,l\}, s, f, m, d \rangle$

$E_3 = \langle \{d,j,k,l,m\}, s, f, m, d \rangle$

Figure B.1: Abstract event description

1. *(a) None of the* $s_i, 1 \leq i \leq n$ *necessarily depict the visual aspects of* **e**, *i.e. the* agent, instrument, object *etc., the visual characteristics of these, and the visual relationships between them that give rise to the visual manifestation of the event* **e**.

2. *(b) The event* **e** *is realised if some entity with the requisite knowledge of* **e** *who viewed* $s_1, ..s_n$ *as a film sequence is convinced that there is sufficient visual evidence present to indicate that* **e**. *is actually taking place.*

Parkes gives two corollaries to assumption 4 as follows:

Corollary 4.1. *The system will only know in event description terms of those events which someone decides to attribute to the sequences: the more knowledge* *the describer has of those events which someone decides to attribute to the sequences the better; The more knowledge the describer has of the narratives in a domain the better and*

Corollary 4.2. *A person (e.g. learner) who subsequently* views *the described material may not have the requisite knowledge, and thus may need to have the significance of the* visual *evidence explained. The latter is particularly important since the evidence whilst visual, may be* understood *by the describer on the basis of non-visually orientated knowledge that the describer possesses- the system would need this knowledge also, to make these explanations.*

This broader definition allows the minimum length of a setting to be a single frame and allows the setting description to be applied to what would be now known as key frame databases. Parkes then notes that these key frames are necessarily *partial* in that they are incomplete as a result of being subject to *spatial* deletion and *selective* being derived from one observer's point of view at the expense of all others; *depth ambiguous.* This leads to the following assumption:

Parkes' Assumption 5. *The still picture is object determined to a describer, but may not be to a subsequent viewer, who may have insufficient knowledge to form the correct object hypothesis on the basis of that picture alone.*

Parkes then considers the distinction between what can be inferred from a still image when considered in the context of a moving picture sequence and what can be inferred from a still image when considered arbitrarily. He concludes that the still image has extra levels of meaning when seen as part of a moving picture sequence. It follows that the status of the frame, in relation to event representation, is as a result of its position in relation to the other frames in the sequence leading to the further assumption:

Parkes' Assumption 6. *The still frame has levels of meaning, ranging from the purely objective, called* ***object-determined****; through* ***event-selected*** *when given a still frame, could depict an instant of is assigned as being in progress in that frame -this reflects the* ***event*** *- and/or* ***state-ambiguity*** *of the still image; to* ***event-determined****: the film was interrupted when* **e** *was taking place, so* **e** *is* ***in-progress.***

A representation language for photographic material needs to be able to maintain the distinction (for physical objects, at the very least) between **concepts** and **instances** of concepts.

Parkes' Assumption 7. *A representation language for photographic material must be able to maintain a distinction between and commonalities of, information* ***about*** *concepts and information* ***about*** *specific instances of those concepts.*

Parkes' Assumption 8. *A representation language for photographic material needs to be able to maintain a definitional type hierarchy which includes the concepts which have instances, depicted in the visual material.*

Parkes' Assumption 9. *Any type label used in the initial description applied to a setting (or image) is not to be replaced by any subtype of that type, unless it can be independently established that the truth of the resulting setting description, with regard to the objectively-visible nature of the setting, is preserved.*

Appendix C

Story Grammars

Story structure analysis was carried out by Propp [225] and Lakoff [165] in the first half of the 20th on folk and fairy tales as part of the Russian formalist movement. Propps analysis cantered on a folk-tale corpus collected by Aarne and Thompson [282]. Informal analysis was also carried out in the West by Cambell [51]and still continues with McKnee [191]. After the publication of Propps work in the 1960s interest developed in relating story recall ability to story structure typified by Mandlers work [185]. Following this considerable energy was expended in creating formal story grammars based on most of the levels in the Chomsky hierarchy [58] by Colby [60] and van Dijk [286] as related by Anderson and Slator [11]. AI interest in story grammars begins with Rumelhart's [236] use of a context free grammar (CFG) which was subsequently refined by Mandler and Johnson *op.cit.* , who used CFGs and later a universal rewrite system (URS) [148]. Story grammars were subsequently refuted by Black and Wilensky [33] and Black and Bower [32] which was countered in a subsequent riposte by Mandler and Johnson [187] and other proponents of the theory.

Black and Wilensky's, Black and Bower's and Garnhams objections will now be reconsidered. Black and Wilensky, and Black and Bower, both object to story grammars because of their recursive definition in terms of their own terms. The infinite recursion and cross definition properties of the Johnson and Mandler grammars can be resolved by adding additional terms and rule modification, familiar tasks with formal grammar development. A more serious objection comes from

Garnham [104] who observes that story grammars are, couched in terms of production rules defining sets of propositions about the narrative world and not elements that correspond to items in a lexicon. This is due to the absence of a lexicon of propositions. In Garnham's argument the production rules, operate on sets of propositions about the states of actors, objects and their attributes in a spatio-temporal location. In effect this prescribes a set of transformations in the state of the narrative world. Johnson-Laird [149] and Garnham [104] note that two principle factors influence text processing namely *referential continuity* and *plausibility.* A text is referentially continuous if the text refers to a set of common objects, and its anaphoric phrases can be resolved using inferential processes, such as bridging. They claim that such theories overcome problems with story grammars since they rely on motivational structures and goal frameworks. Once the propositions in a text are discovered it suffices to place them into a framework generated from stored knowledge and so associating propositions with structural classes is not necessary in this framework. This in turn can be countered by considering the following: The development of a soccer ontology is a relatively straight forward task and the use of such ontologies in games applications shows that finite sets of propositions can be constructed in *closed-domains* for production rule operation. For the time being it will be left open whether this invalidates Garnham's assertion that,

> If story parsers are like sentence parsers then they must be able to identify units of story analysis corresponding, roughly, to lexical items in the analysis of sentences. Fortunately story grammarians are agreed that the unit of story analysis is the proposition. But unlike the set of lexical items the set of propositions is not finite. The latter set must be defined recursively, since propositions can be embedded within each other to an arbitary depth. Garnham *et al.* p. 148 [104]

The abstraction of a general model of team-sports competition from a formal model is a relatively simple step. The validity of this approach is reinforced by the success reported by Assfalg et al. [17]. These results might preclude the use of story grammars as a representational tool and point to the use of Schanks [238] CD to construct scripts and consideration of their later integration into the memory organisation packet (MOP) and thematic organisation packet (TOP) structures in [240] It is well known that some classes of story follow predictable patterns since we can find multiple instances of the same story e.g. the multiple versions of the Romeo and Juliet story that exist on film with different locations media etc. William Shakespeare's Romeo + Juliet (1996), West Side Story (1961) and Romanoff and Juliet (1961) and other examples all follow the same basic story line but are recast in different social millieux.

From Conway's [61] fNMR results on *personal narrative*, it can be shown that story recall is improved if a structure is imposed. However, evidence for reliance on the use of a formal structuring mechanism remains inconclusive [209]. Anderson and Slator [11] conclude their review of the story grammar theory by noting that the use of story grammars is considered useful in teaching learning disabled children and may still be considered to have explanatory power. It is useful to note that story structure is being widely taught in UK Primary Schools [97] through a methodology that is recognisably related to Propp's functions and as a Schankian [241] story skeletons and classic horror movies such as Jaws (1975) can be readily fitted to this model.

So in conclusion structure in stories aids their recall, the structure represents sets of propositions about the state of the narrative world. These sets of propositions are capable of being related formally by a grammar, such a grammar may or may not be psychologically valid. Processing sets of propositions is feasible as evidenced by game applications. Whether story grammars have a role in automating annotation systems remains to be seen.

Appendix D

Teletubbies Favourite Things Representative Frames

The representative frames from Teletubbies Favourite Things are copyright BBC/Rag-doll Productions Ltd 1996 and reproduced with permission.

Favourite Things D.1:

Favourite Things D.2:

Favourite Things D.3:

Favourite Things D.4:

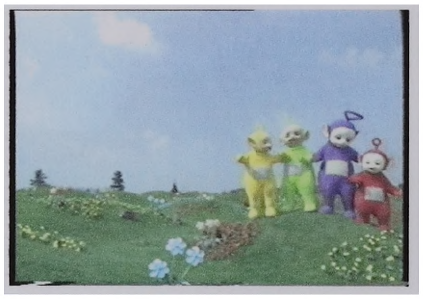
Favourite Things D.5:

Favourite Things D.6:

Favourite Things D.7:

Favourite Things D.8:

Favourite Things D.9:



Favourite Things D.10:

Favourite Things D.11:

Favourite Things D.12:

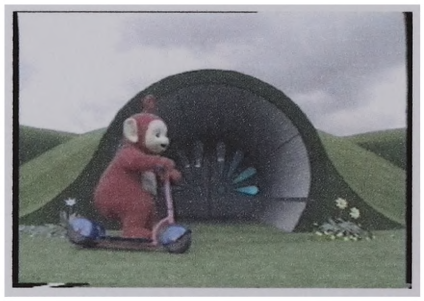
Favourite Things D.13:

Favourite Things D.14:

Favourite Things D.15:

Favourite Things D.16:

Favourite Things D.17:


Favourite Things D.18:

Favourite Things D.19:

Favourite Things D.20:

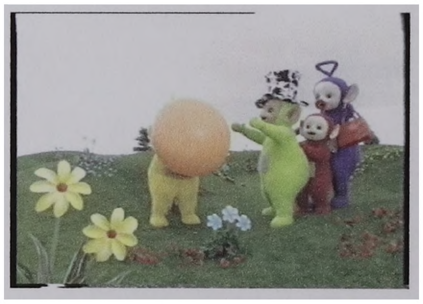
Favourite Things D.21:

Favourite Things D.22:

Favourite Things D.23:

Favourite Things D.24:

Favourite Things D.25:


Favourite Things D.26:

Favourite Things D.27:

Favourite Things D.28:

Appendix E

Python source code listings

E.1 Video Decoding

Listing E.1: Basic MPEG player Python code

```
from Tkinter import *
from ImageTk import *
import libmpeg3
import Numeric
import Image

class Mpeg_play1:
        "Simple MPEG-2 Player"
        def open_file(self, fname = None):
                global fna
                if fname == None:
                        fna = libmpeg3.mpeg3_open("/home/widrith/Mandrake/pympeg_04/
                            rocket.mpg")
                else:
                        try:
                                fna = libmpeg3.mpeg3_open(fname)
                        except IOError, e:
                                print "unable to open 'fname' : ", e
```

```
def get_pic_size(self):
        global fna
        self.vid_width = libmpeg3.mpeg3_video_width(fna, 0)
        self.vid_height = libmpeg3.mpeg3_video_height(fna, 0)
        self.vid_depth =  3

        self.max_frames = libmpeg3.mpeg3_video_frames(fna, 0)
        self.frame_rate = libmpeg3.mpeg3_frame_rate(fna, 0)

        self.pic_size = (self.vid_width, self.vid_height, self.vid_depth)

def __init__(self, fname = None, surface = None):
        global fna
        global percent_absolute
        fna = None
        self.open_file(fname)
        self.get_pic_size()
        if self.max_frames == 1:
                self.percent_absolute = 1
        else:
                self.percent_absolute = 0


        row_py = libmpeg3.ptrcreate("char *", "NULL", self.vid_height)
        self.framep = Numeric.ones([self.vid_width, self.vid_height, self.
            vid_depth], 'b')
        self.row_p_p_char = libmpeg3.setup_pointers(self.framep, row_py,
            self.vid_width, self.vid_height, 3)


        self.img1 = Image.new('RGB',(self.vid_width, self.vid_height))
        self.img2 = Image.new('RGB',(self.vid_width, self.vid_height))
        self.stat = libmpeg3.mpeg3_read_frame(fna, self.row_p_p_char, 0, 0,
            self.vid_width, self.vid_height, self.vid_width, self.vid_height
```

```
                , 3, 0)
            self.img1.fromstring(self.framep.tostring(),'raw','RGB',0)
            self.igm1 = PhotoImage(self.img1)
            self.igm2 = PhotoImage(self.img1)
            if surface == None:
                    self.img1.show()
            return None

    def pic_size():
            return self.pic_size

    def get_image(self):

            self.stat = libmpeg3.mpeg3_read_frame(fna, self.row_p_p_char, 0, 0,
                self.vid_width, self.vid_height, self.vid_width, self.vid_height
                , 3, 0)
            self.img1.fromstring(self.framep.tostring(),'raw','RGB',0)
            img = self.img1

    def seek_percent(self, position = None):
            global fna
            print 'seeking', position
            self.stat = libmpeg3.mpeg3_seek_percentage(fna, position)
            self.stat = libmpeg3.mpeg3_read_frame(fna, self.row_p_p_char, 0, 0,
                self.vid_width, self.vid_height, self.vid_width, self.vid_height
                , 3, 0)
            self.img1.fromstring(self.framep.tostring(),'raw','RGB',0)
            img = self.img1

    def wind(self):
            global fna
            print 'seeking', frameno
            self.stat = libmpeg3.mpeg3_seek_percentage(fna, 1)
            self.stat = libmpeg3.mpeg3_read_frame(fna, self.row_p_p_char, 0, 0,
                self.vid_width, self.vid_height, self.vid_width, self.vid_height
                , 3, 0)
            self.img1.fromstring(self.framep.tostring(),'raw','RGB',0)
            img = self.img1
            print new_pos

    def rewind(self):

            global fna
            self.stat = libmpeg3.mpeg3_seek_percentage(fna, self.position)
            self.stat = libmpeg3.mpeg3_read_frame(fna, self.row_p_p_char, 0, 0,
                self.vid_width, self.vid_height, self.vid_width, self.vid_height
                , 3, 0)
            self.stat = libmpeg3.mpeg3_read_frame(fna, self.row_p_p_char, 0, 0,
                self.vid_width, self.vid_height, self.vid_width, self.vid_height
                , 3, 0)
            self.img1.fromstring(self.framep.tostring(),'raw','RGB',0)
            img = self.img1

    def get_frame_no(self):
            global fna
            frameno = libmpeg3.mpeg3_get_frame(fna,0)
            return frameno

    def get_percentage(self):
            global fna
            percentage = libmpeg3.mpeg3_tell_percentage(fna)
            return percentage

    def get_frame():
            return self.framep

if __name__ == '__main__':
        root  = Tk()
        root.withdraw()
        myplay = Mpeg_play1()
        root.mainloop()
```

Listing E.2: MPEG Decoder Driver Program

```
from Tkinter import *
from ImageTk import *
import libmpeg3
import Numeric
import Image
fna = libmpeg3.mpeg3_open("/home/widrith/Mandrake/pympeg_04/rocket.mpg")
vid_width = libmpeg3.mpeg3_video_width(fna, 0)
vid_height = libmpeg3.mpeg3_video_height(fna, 0)
max_frames = libmpeg3.mpeg3_video_frames(fna, 0)

frame_rate = libmpeg3.mpeg3_frame_rate(fna, 0)

row_py = libmpeg3.ptrcreate("char *", "NULL",vid_height)
framep = Numeric.ones([160,120,3],'b')
row_p_p_char = libmpeg3.setup_pointers(framep, row_py, vid_width, vid_height, 3)



root = Tk()

frame = Frame(root, width=160, height=120, bg= 'gray50', relief =RAISED, bd =4)

label = Label(frame)
frame.pack()
label.place(relx=0.5, rely=0.48, anchor=CENTER)
img = Image.new('RGB',(160,120))
img2 = Image.new('RGB',(160,120))
```

```
stat = libmpeg3.mpeg3_read_frame(fna, row_p_p_char, 0, 0, vid_width, vid_height,
    vid_width, vid_height, 3, 0)
img.fromstring(framep.tostring(),'raw','RGB',0)
igm = PhotoImage(img)
label['image'] = igm

stat = libmpeg3.mpeg3_set_frame(fna, 20, 0)

stat = libmpeg3.mpeg3_read_frame(fna, row_p_p_char, 0, 0, vid_width, vid_height,
    vid_width, vid_height, 3, 0)
img2.fromstring(framep.tostring(),'raw','RGB',0)
igm2 = PhotoImage(img2)
label['image'] = igm2

libmpeg3.mpeg3_read_frame(fna, row_p_p_char, 0, 0, vid_width, vid_height, vid_width,
     vid_height, 3, 0)



img3 = img.rotate(180)
igm = PhotoImage(img3)
label['image'] = igm


img4 = img.convert("L")
igm = PhotoImage(img4)
label['image'] = igm


img4 = img.convert("CMYK")
igm = PhotoImage(img4)
label['image'] = igm
```

```
draw = ImageDraw.Draw(img)
draw.line((0,0) + img.size, fill = 128)
draw.line((0, img.size[1], img.size[0], 0), fill = 128)
del draw
igm = PhotoImage(img)
label['image'] = igm


enhancer = ImageEnhance.Sharpness(img)
enhancer.enhance(.35)
img5 = enhancer.enhance(.35)
igm3 = PhotoImage(img5)
label['image'] = igm3




import ImageFilter

im7 = img2.filter(ImageFilter.EDGE_ENHANCE)
igm7 = PhotoImage(im7)
label['image'] = igm7
im7 = img4.filter(ImageFilter.EDGE_ENHANCE)
igm7 = PhotoImage(im7)
label['image'] = igm7
im7 = img4.filter(ImageFilter.EDGE_ENHANCE_MORE)
igm7 = PhotoImage(im7)
label['image'] = igm7


im7 = img4.filter(ImageFilter.FIND_EDGES)
igm7 = PhotoImage(im7)
label['image'] = igm7
```

E.1.1 Semantic and visual factory classes

Listing E.3: Semantic persistent factory Python code

```
import types
from Tools import __Lister, _Former
import Persistence
import ZODB







def factory(aClass, *args):
    return apply(aClass, args)


def new_factory(aClass, *args, **kwargs):
    return apply(aClass, args, kwargs)




class S_Object:

    num_S_Objects = 0
    def __init__(self):
        S_Object.num_S_Objects +=1
        self.__S_serial = S_Object.num_S_Objects

class S_Object_f(_Former,Persistence.Persistent):
    num_S_Objects_f = 0
    def __init__(self):
        S_Object_f.num_S_Objects_f +=1
        self.serial = S_Object_f.num_S_Objects_f


class S_Person(S_Object, __Lister,Persistence.Persistent):



    num_S_Persons = 0
    name = ""
    role = ""
    settinglist = []
    def __init__(self, name= None, role = None, setting = None ):
        self.settinglist = S_Person.settinglist
        S_Person.num_S_Persons +=1
        S_Object.num_S_Objects += 1
        self.__s_object_serial = S_Object.num_S_Objects
        self.__s_person_serial = S_Person.num_S_Persons

        if name is not None:
            self.name = name
        else:
            self.name = S_Person.name
        if role is not None:
            self.role = role
        else:
            self.role = S_Person.role
        if setting is not  None:
            self.settinglist.append(setting)
    def __getitem__(self, index):
        if self.settinglist != []:
            return self.settinglist[index]
        else:
            return None



class S_Player(S_Person,__Lister,Persistence.Persistent):

    num_S_Players = 0
    role = ""
    team = ""
    position = ""
    shirt_number = ""
    def __init__(self, name= None, role = None, setting = None , position = None,
        shirt_number = None ):
        S_Object.num_S_Objects += 1
        S_Person.num_S_Persons += 1
        S_Player.num_S_Players += 1
        self.__s_object_serial = S_Object.num_S_Objects
        self.__s_person_serial = S_Person.num_S_Persons
        self.__s_player_serial = S_Player.num_S_Players

        if name is not None:
            self.name = name
        else:
            self.name = S_Person.name
        if role is not None:
            self.role = role
        else:
            self.role = S_Person.role
        if position is not None:
            self.position = position
        else:
            self.position = S_Player.position
        if shirt_number is not None:
            self.shirt_number = shirt_number
        else:
            self.shirt_number = S_Player.shirt_number
        self.settinglist = S_Person.settinglist
        if setting is not  None:
            self.settinglist.append(setting)

    def __getitem__(self, index):
        if settinglist != []:
            return settinglist[index]
        else:
            return None




class S_Team(S_Object, __Lister,Persistence.Persistent):

    num_S_Teams = 0
    team_name = ""
    domain = ""
    def __init__(self, team_name = None, domain = None,  players = None , extras =
        None):
        S_Object.num_S_Objects += 1
        S_Team.num_S_Teams += 1
        self.__s_object_serial = S_Object.num_S_Objects
        self.__s_teams_serial = S_Team.num_S_Teams
        self.playerslist = []
        self.extraslist = []

        if team_name is not None:
            self.team_name = team_name
        else:
            self.team_name = S_Team.team_name
        if domain is not None:
            self.domain = domain
        else:
            self.domain = S_Team.domain
        if players is not None:
            for player in players:
                self.playerslist.append(players)
        else:
            self.playerslist = []
        if extras is not None:
            for extra in extras:
                self.extraslist.append(extra)
        else:
            self.extraslist = []

    def __getitem__(self, list, index):
        return self.list[index]

class S_Part(S_Object, __Lister,Persistence.Persistent):

    num_S_Parts = 0
    name = ""
    is_component = False
    def __init__(self, name = None, is_component = False):
        S_Object.num_S_Objects += 1
        S_Part.num_S_Parts += 1
        self.__s_object_serial = S_Object.num_S_Objects
        self.__s_part_serial = S_Part.num_S_Parts
        if name is not None:
            self.name = name
        else:
            name = S_Part.name
        if is_component is not None:
            self.is_component = is_component
        else:
            pass




class S_Competition(S_Object, __Lister,Persistence.Persistent):

    num_S_Competitions = 0
    comp_name = ""
    def __init__(self, comp_name = None):
        S_Object.num_S_Objects += 1
        S_Competition.num_S_Competitions += 1
        self.__s_object_serial = S_Object.num_S_Objects
        self.__s_competition_serial = S_Competition.num_S_Competitions
        if comp_name is not None:
            self.comp_name = comp_name
        else:
            self.comp_name = S_Competition.comp_name

class S_Fixture(S_Object, __Lister,Persistence.Persistent):

    num_S_Fixtures = 0
    away_team = ""
    home_team = ""
    match_name = ""
    num_S_Fixtures = 0
    competition = S_Part(None)
    def __init__(self, home_team = None, away_team = None , match_name = None,
        competition = None):
        S_Object.num_S_Objects += 1
        S_Fixture.num_S_Fixtures +=1
        self.__s_object_serial = S_Object.num_S_Objects
        self.__s_fixture_serial = S_Fixture.num_S_Fixtures
        if home_team is not None:
            self.home_team = home_team
        else:
            self.home_team = S_Fixture.home_team
        if away_team is not None:
            self.away_team = away_team
        else:
            self.away_team = S_Fixture.away_team
        if match_name is not None:
            self.match_name = match_name
        else:
            self.match_name = S_Fixture.match_name
        if competition is not None:
            self.competition = competition
        else:
            self.competition = S_Fixture.competition
```

Listing E.4: Visual persistent factory Python code

```
import Image
import pickle
import os
import types
from Tools import __Lister, _Former
import Persistence
import ZODB
```

```
def v_factory(i_image=None, *i_args):

    if i_image is not None:
        if type(i_image) == str:
            if os.path.isfile(i_image):

                fptr = open(i_image)

                try:
                    img = Image.open(fptr)
                    data   = img.tostring("raw", img.mode)
                    size   = img.size
                    mode   = img.mode
                    format = img.format
                    sys_ref = i_image
                    fptr.close()
                    l_args = (size, mode, format, data, sys_ref)
                    args   = i_args + l_args
                    return apply(V_Image, args)
                except IOError, e:
                     print "Unable to read Image Data"
        else:


            try:
                data   = i_image.tostring("raw", i_image.mode)

                size   = i_image.size
                mode   = i_image.mode
                format = i_image.format
                sys_ref = i_image
                l_args = (size, mode, format, data, sys_ref)
                args   = l_args + i_args
                return apply(V_Frame, args)
            except IOError, e:

                print "Unable to read Image Data"

def factory(aClass, *args):
    instance = None
    if aClass == V_Image:
        instance = v_factory(*args)
        return instance
    elif aClass == V_Frame:
        instance = v_factory(*args)
        return instance
    else:
        return apply(aClass, args)



def new_factory(aClass, *args, **kwargs):
    return apply(aClass, args, kwargs)



def im_factory(i_image=None):

    if i_image is not None:
        fptr = open(i_image)
        img = Image.open(fptr)
        size   = img.size
        mode   = img.mode
        format = img.format
        data   = img.tostring("raw", img.mode)
        sys_ref = i_image
        fptr.close()
        args = (size, mode, format, data, sys_ref)
    return apply(V_Image, args)
```

```
class V_Object(__Lister, Persistence.Persistent):
    num_V_Objects = 0
    def __init__(self, image_ref = None, path = None, path_type = None):
        V_Object.num_V_Objects +=1
        self.__V_Object_serial = V_Object.num_V_Objects


class v_Region(V_Object, __Lister,Persistence.Persistent):

    num_v_Regions = 0
    def __init__(self, image_ref = None, path = None, path_type = None):
        V_Region.num_V_Regions +=1
        V_Object.num_V_Objects +=1
        self.__v_Region_serial = v_Region.num_v_Regions
        self.__v_Object_serial = v_Object.num_v_Objects
        if image_ref is not None:
            self.image_ref = image_ref
        else:
            self.image_ref = None
        if path is not  None:
            self.image_ref = image_ref
        else:
            self.image_ref = NOne
        if path_type is not  None:
            self.path_type = path_type
        else:
            self.path_type = None
```

```
class V_Image(V_Object, __Lister):

    num_V_Images = 0
    def __init__(self, size= None, mode= None, format=None, data=None, sys_ref=None)
        :
        V_Image.num_V_Images +=1
        self.__v_Image_serial = V_Image.num_V_Images
        if size is not None:
            self.size    = size
        else:
            self.size    = None
        if mode is not None:
            self.mode    = mode
        else:
            self.mode  = None
        if format is not None:
            self.format = format
        else:
            self.format = None
        if data is not None:
            self.__data__ = data
        else:
            self.__data__    = None
        if sys_ref is not None:
            self.sys_ref = sys_ref
        else:
            self.sys_ref = None

class V_Frame(V_Object, __Lister, Persistence.Persistent):

    num_V_Frames = 0
    def __init__(self, size= None, mode= None, format=None, data=None, sys_ref=None,
                video=None, frame_num=None ):
        V_Object.num_V_Objects +=1
        self.__v_Object_serial = V_Object.num_V_Objects
        V_Frame.num_V_Frames +=1
        self.__v_Frame_serial = V_Frame.num_V_Frames
        if size is not None:
            self.size    = size
        else:
            self.size    = None
        if mode is not None:
            self.mode    = mode
        else:
            self.mode   = None
        if format is not None:
            self.format = format
        else:
            self.format = None
        if data is not None:
            self.data = data
        else:
            self.data    = None
        if sys_ref is not None:
            self.sys_ref = sys_ref
        else:
            self.sys_ref = None
        if video is not None:
            self.video = video
        else:
            self.video = None
        if frame_num is not None:
            self.frame_num = frame_num
        else:
            self.frame_num = None
```

```
class V_Setting(__Lister,Persistence.Persistent):

    num_v_Settings = 0
    def __init__(self, size= None, mode= None, format=None, data=None, sys_ref=None,
          video=None, start_frame=None, end_frame=None ):
        V_Object.num_V_Objects +=1
        self.__v_Object_serial = V_Object.num_V_Objects
        V_Setting.num_v_Settings +=1
        self.__v_Setting_serial = V_Setting.num_v_Settings
        if size is not None:
            self.size    = size
        else:
            self.size    = None
        if mode is not None:
            self.mode    = mode
        else:
            self.mode  = None
        if format is not None:
            self.format = format
        else:
            self.format = None
        if data is not None:
            self.data = data
        else:
            self.data    = None
        if sys_ref is not None:
            self.sys_ref = sys_ref
        else:
            self.sys_ref = None
        if video is not None:
            self.video = video
        else:
            self.video = None
        if start_frame is not None:
            self.start_frame = start_frame
        else:
            self.start_frame = None
        if end_frame is not None:
            self.end_frame = start_frame
        else:
            self.end_frame = None

class V_Event(__Lister,Persistence.Persistent):

    num_v_Events = 0
    def __init__(self,  video=None, start_setting=None, end_setting=None ):
        V_Event.num_v_Events +=1
        self.__v_Event_serial = V_Event.num_v_Events
        V_Object.num_V_Objects +=1
        self.__v_Object_serial = V_Object.num_V_Objects
        if video is not None:
            self.video = video
        else:
            self.video = None
        if start_setting is not None:
            self.start_setting = start_setting
        else:
            self.start_setting = None
        if end_setting is not None:
            self.end_setting=end_setting
        else:
            self.end_setting=None



if __name__=='__main__':
    img = im_factory('/home/widrith/Images/_38243165_1inspectors300.jpg')
    print img
```

Appendix F

Interface and C source code listings

F.1 SWIG interface

Listing F.1: SWIG Interface File; libmpeg3.i for Libmpeg3

```
%module libmpeg3
%include pointer.i
%{
#include "libmpeg3.h"
#include "mpeg3demux.h"
#include "mpeg3protos.h"
#include "help.h"
%}


%include "help.i"
%include "numpy.i"
%include "typemaps.i"
%include "libmpeg3.h"
%include "mpeg3demux.h"
%include "mpeg3protos.h"
```

F.2 Helper functions

Listing F.2: Helper Function Interface File help.i

```
%typemap(python, in) mpeg_uchar * {
        PyArrayObject *arry;
        if ((arry = (PyArrayObject *) PyArray_ContiguousFromObject($source,
            PyArray_UBYTE, 0, 0)) ==NULL) return NULL;
        $target = (mpeg_uchar *)arry->data;
        $source = (PyObject *)arry;
}


%inline %{
#include "arrayobject.h"
unsigned char **new_row_buffer( int height){
        unsigned char  **row_buffer = malloc(height*sizeof(unsigned char *));
        return row_buffer;
}



unsigned char *new_vid_frame(int height, int width, int depth){
        unsigned char *vid_frame_p;
        unsigned char *vid_array = malloc(((width *( height * depth)) +4) * sizeof(
            unsigned char ));
        return &vid_array;
}



unsigned char **setup_pointers(mpeg_uchar *vidframe, char **row_buffer, int width,
    int height, int depth){
        int vid_index=0;
        int i=0;
        int mtype;
        unsigned char **rowc =NULL;
        row_buffer[i] =&vidframe[0];
        for(i=1; i < height ; i++ ){
                printf("vid index is %i\n", vid_index);
                vid_index =  i * width * depth;
                row_buffer[i] =&vidframe[vid_index];

                }
        rowc = row_buffer;
        return rowc;
}



int printframe(int height, int width, int depth, unsigned char* vid_array) {
        int i;
        int size;
        size = height * (width * depth);
        for(i=0; i < size ; i++ ){
                printf("%x",vid_array[i]);
        }
        printf("size =%i\n", size);
        printf("\n");
        return 0;
}



int set_vidarray(int height, int width, int depth, mpeg_uchar *arry){
  PyArrayObject *arrptr;
  int stat;




  stat = 0;




  arry[0]=0;
  return stat;
}



static int array_defs( PyArrayObject *array_obj){

      PyArrayObject* array;
      array =&array_obj;
      printf("number of dimensions =%i \n", array->nd);
      printf("width =%i \n", array->dimensions[0]);
      printf("height =%i \n", array->dimensions[1]);
      printf("depth =%i \n", array->dimensions[2]);
      return  1;
}
```

```
%}
%name(delfv) void free(void *);
```

Listing F.3: Helper Function Header File help.h

```
typedef unsigned char mpeg_uchar;
```

F.3 Video Decoding

Listing F.4: SWIG generated Python Wrapper File for Libmpeg3

```
# This file was created automatically by SWIG.
import libmpeg3c
class mpeg3_demuxer_t:
    __setmethods__ = {}
    for _s in []: __setmethods__.update(_s.__setmethods__)
    def __setattr__(self,name,value):
        if (name == "this"):
            if isinstance(value,mpeg3_demuxer_t):
                self.__dict__[name] = value.this
                if hasattr(value,"thisown"): self.__dict__["thisown"] = value.
                    thisown
                del value.thisown
                return
        method = mpeg3_demuxer_t.__setmethods__.get(name,None)
        if method: return method(self,value)
        self.__dict__[name] = value

    __getmethods__ = {}
    for _s in []: __getmethods__.update(_s.__getmethods__)
    def __getattr__(self,name):
        method = mpeg3_demuxer_t.__getmethods__.get(name,None)
        if method: return method(self)
        raise AttributeError,name

    __setmethods__["file"] = libmpeg3c.mpeg3_demuxer_t_file_set
```

```
__getmethods__[”file”] = libmpeg3c.mpeg3_demuxer_t_file_get
__setmethods__[”raw_data”] = libmpeg3c.mpeg3_demuxer_t_raw_data_set
__getmethods__[”raw_data”] = libmpeg3c.mpeg3_demuxer_t_raw_data_get
__setmethods__[”raw_offset”] = libmpeg3c.mpeg3_demuxer_t_raw_offset_set
__getmethods__[”raw_offset”] = libmpeg3c.mpeg3_demuxer_t_raw_offset_get
__setmethods__[”raw_size”] = libmpeg3c.mpeg3_demuxer_t_raw_size_set
__getmethods__[”raw_size”] = libmpeg3c.mpeg3_demuxer_t_raw_size_get
__setmethods__[”packet_size”] = libmpeg3c.mpeg3_demuxer_t_packet_size_set
__getmethods__[”packet_size”] = libmpeg3c.mpeg3_demuxer_t_packet_size_get
__setmethods__[”do_audio”] = libmpeg3c.mpeg3_demuxer_t_do_audio_set
__getmethods__[”do_audio”] = libmpeg3c.mpeg3_demuxer_t_do_audio_get
__setmethods__[”do_video”] = libmpeg3c.mpeg3_demuxer_t_do_video_set
__getmethods__[”do_video”] = libmpeg3c.mpeg3_demuxer_t_do_video_get
__setmethods__[”data_buffer”] = libmpeg3c.mpeg3_demuxer_t_data_buffer_set
__getmethods__[”data_buffer”] = libmpeg3c.mpeg3_demuxer_t_data_buffer_get
__setmethods__[”data_size”] = libmpeg3c.mpeg3_demuxer_t_data_size_set
__getmethods__[”data_size”] = libmpeg3c.mpeg3_demuxer_t_data_size_get
__setmethods__[”data_position”] = libmpeg3c.mpeg3_demuxer_t_data_position_set
__getmethods__[”data_position”] = libmpeg3c.mpeg3_demuxer_t_data_position_get
__setmethods__[”data_allocated”] = libmpeg3c.mpeg3_demuxer_t_data_allocated_set
__getmethods__[”data_allocated”] = libmpeg3c.mpeg3_demuxer_t_data_allocated_get
__setmethods__[”reverse”] = libmpeg3c.mpeg3_demuxer_t_reverse_set
__getmethods__[”reverse”] = libmpeg3c.mpeg3_demuxer_t_reverse_get
__setmethods__[”error_flag”] = libmpeg3c.mpeg3_demuxer_t_error_flag_set
__getmethods__[”error_flag”] = libmpeg3c.mpeg3_demuxer_t_error_flag_get
__setmethods__[”next_char”] = libmpeg3c.mpeg3_demuxer_t_next_char_set
__getmethods__[”next_char”] = libmpeg3c.mpeg3_demuxer_t_next_char_get
__setmethods__[”time_offset”] = libmpeg3c.mpeg3_demuxer_t_time_offset_set
__getmethods__[”time_offset”] = libmpeg3c.mpeg3_demuxer_t_time_offset_get
__setmethods__[”read_all”] = libmpeg3c.mpeg3_demuxer_t_read_all_set
__getmethods__[”read_all”] = libmpeg3c.mpeg3_demuxer_t_read_all_get
__getmethods__[”titles”] = libmpeg3c.mpeg3_demuxer_t_titles_get
__setmethods__[”total_titles”] = libmpeg3c.mpeg3_demuxer_t_total_titles_set
__getmethods__[”total_titles”] = libmpeg3c.mpeg3_demuxer_t_total_titles_get
```

```
__setmethods__["current_title"] = libmpeg3c.mpeg3_demuxer_t_current_title_set
__getmethods__["current_title"] = libmpeg3c.mpeg3_demuxer_t_current_title_get
__getmethods__["astream_table"] = libmpeg3c.mpeg3_demuxer_t_astream_table_get
__getmethods__["vstream_table"] = libmpeg3c.mpeg3_demuxer_t_vstream_table_get
__setmethods__["total_programs"] = libmpeg3c.mpeg3_demuxer_t_total_programs_set
__getmethods__["total_programs"] = libmpeg3c.mpeg3_demuxer_t_total_programs_get
__setmethods__["current_program"] = libmpeg3c.
    mpeg3_demuxer_t_current_program_set
__getmethods__["current_program"] = libmpeg3c.
    mpeg3_demuxer_t_current_program_get
__setmethods__["current_timecode"] = libmpeg3c.
    mpeg3_demuxer_t_current_timecode_set
__getmethods__["current_timecode"] = libmpeg3c.
    mpeg3_demuxer_t_current_timecode_get
__setmethods__["current_byte"] = libmpeg3c.mpeg3_demuxer_t_current_byte_set
__getmethods__["current_byte"] = libmpeg3c.mpeg3_demuxer_t_current_byte_get
__setmethods__["transport_error_indicator"] = libmpeg3c.
    mpeg3_demuxer_t_transport_error_indicator_set
__getmethods__["transport_error_indicator"] = libmpeg3c.
    mpeg3_demuxer_t_transport_error_indicator_get
__setmethods__["payload_unit_start_indicator"] = libmpeg3c.
    mpeg3_demuxer_t_payload_unit_start_indicator_set
__getmethods__["payload_unit_start_indicator"] = libmpeg3c.
    mpeg3_demuxer_t_payload_unit_start_indicator_get
__setmethods__["pid"] = libmpeg3c.mpeg3_demuxer_t_pid_set
__getmethods__["pid"] = libmpeg3c.mpeg3_demuxer_t_pid_get
__setmethods__["transport_scrambling_control"] = libmpeg3c.
    mpeg3_demuxer_t_transport_scrambling_control_set
__getmethods__["transport_scrambling_control"] = libmpeg3c.
    mpeg3_demuxer_t_transport_scrambling_control_get
__setmethods__["adaptation_field_control"] = libmpeg3c.
    mpeg3_demuxer_t_adaptation_field_control_set
__getmethods__["adaptation_field_control"] = libmpeg3c.
    mpeg3_demuxer_t_adaptation_field_control_get
```

```
__setmethods__["continuity_counter"] = libmpeg3c.
    mpeg3_demuxer_t_continuity_counter_set
__getmethods__["continuity_counter"] = libmpeg3c.
    mpeg3_demuxer_t_continuity_counter_get
__setmethods__["is_padding"] = libmpeg3c.mpeg3_demuxer_t_is_padding_set
__getmethods__["is_padding"] = libmpeg3c.mpeg3_demuxer_t_is_padding_get
__getmethods__["pid_table"] = libmpeg3c.mpeg3_demuxer_t_pid_table_get
__getmethods__["continuity_counters"] = libmpeg3c.
    mpeg3_demuxer_t_continuity_counters_get
__setmethods__["total_pids"] = libmpeg3c.mpeg3_demuxer_t_total_pids_set
__getmethods__["total_pids"] = libmpeg3c.mpeg3_demuxer_t_total_pids_get
__setmethods__["adaptation_fields"] = libmpeg3c.
    mpeg3_demuxer_t_adaptation_fields_set
__getmethods__["adaptation_fields"] = libmpeg3c.
    mpeg3_demuxer_t_adaptation_fields_get
__setmethods__["time"] = libmpeg3c.mpeg3_demuxer_t_time_set
__getmethods__["time"] = libmpeg3c.mpeg3_demuxer_t_time_get
__setmethods__["audio_pid"] = libmpeg3c.mpeg3_demuxer_t_audio_pid_set
__getmethods__["audio_pid"] = libmpeg3c.mpeg3_demuxer_t_audio_pid_get
__setmethods__["video_pid"] = libmpeg3c.mpeg3_demuxer_t_video_pid_set
__getmethods__["video_pid"] = libmpeg3c.mpeg3_demuxer_t_video_pid_get
__setmethods__["astream"] = libmpeg3c.mpeg3_demuxer_t_astream_set
__getmethods__["astream"] = libmpeg3c.mpeg3_demuxer_t_astream_get
__setmethods__["vstream"] = libmpeg3c.mpeg3_demuxer_t_vstream_set
__getmethods__["vstream"] = libmpeg3c.mpeg3_demuxer_t_vstream_get
__setmethods__["aformat"] = libmpeg3c.mpeg3_demuxer_t_aformat_set
__getmethods__["aformat"] = libmpeg3c.mpeg3_demuxer_t_aformat_get
__setmethods__["program_association_tables"] = libmpeg3c.
    mpeg3_demuxer_t_program_association_tables_set
__getmethods__["program_association_tables"] = libmpeg3c.
    mpeg3_demuxer_t_program_association_tables_get
__setmethods__["table_id"] = libmpeg3c.mpeg3_demuxer_t_table_id_set
__getmethods__["table_id"] = libmpeg3c.mpeg3_demuxer_t_table_id_get
__setmethods__["section_length"] = libmpeg3c.mpeg3_demuxer_t_section_length_set
```

```
    __getmethods__["section_length"] = libmpeg3c.mpeg3_demuxer_t_section_length_get
    __setmethods__["transport_stream_id"] = libmpeg3c.
        mpeg3_demuxer_t_transport_stream_id_set
    __getmethods__["transport_stream_id"] = libmpeg3c.
        mpeg3_demuxer_t_transport_stream_id_get
    __setmethods__["pes_packets"] = libmpeg3c.mpeg3_demuxer_t_pes_packets_set
    __getmethods__["pes_packets"] = libmpeg3c.mpeg3_demuxer_t_pes_packets_get
    __setmethods__["pes_audio_time"] = libmpeg3c.mpeg3_demuxer_t_pes_audio_time_set
    __getmethods__["pes_audio_time"] = libmpeg3c.mpeg3_demuxer_t_pes_audio_time_get
    __setmethods__["pes_video_time"] = libmpeg3c.mpeg3_demuxer_t_pes_video_time_set
    __getmethods__["pes_video_time"] = libmpeg3c.mpeg3_demuxer_t_pes_video_time_get
    def __init__(self,*args):
        self.this = apply(libmpeg3c.new_mpeg3_demuxer_t, args)
        self.thisown = 1
    def __del__(self, destroy= libmpeg3c.delete_mpeg3_demuxer_t):
        if getattr(self,'thisown',0):
            destroy(self)
    def __repr__(self):
        return "<C mpeg3_demuxer_t instance at %s>" % (self.this,)

class mpeg3_demuxer_tPtr(mpeg3_demuxer_t):
    def __init__(self,this):
        self.this = this
        if not hasattr(self,"thisown"): self.thisown = 0
        self.__class__ = mpeg3_demuxer_t
libmpeg3c.mpeg3_demuxer_t_swigregister(mpeg3_demuxer_tPtr)
ptrvalue = libmpeg3c.ptrvalue

ptrset = libmpeg3c.ptrset

ptrcreate = libmpeg3c.ptrcreate

ptrfree = libmpeg3c.ptrfree
```

```
ptradd = libmpeg3c.ptradd

new_row_buffer = libmpeg3c.new_row_buffer

new_vid_frame = libmpeg3c.new_vid_frame

setup_pointers = libmpeg3c.setup_pointers

printframe = libmpeg3c.printframe

set_vidarray = libmpeg3c.set_vidarray

delfv = libmpeg3c.delfv

MPEG3_RGB565 = libmpeg3c.MPEG3_RGB565
MPEG3_BGR888 = libmpeg3c.MPEG3_BGR888
MPEG3_BGRA8888 = libmpeg3c.MPEG3_BGRA8888
MPEG3_RGB888 = libmpeg3c.MPEG3_RGB888
MPEG3_RGBA8888 = libmpeg3c.MPEG3_RGBA8888
MPEG3_RGBA16161616 = libmpeg3c.MPEG3_RGBA16161616
MPEG3_601_RGB565 = libmpeg3c.MPEG3_601_RGB565
MPEG3_601_BGR888 = libmpeg3c.MPEG3_601_BGR888
MPEG3_601_BGRA8888 = libmpeg3c.MPEG3_601_BGRA8888
MPEG3_601_RGB888 = libmpeg3c.MPEG3_601_RGB888
MPEG3_601_RGBA8888 = libmpeg3c.MPEG3_601_RGBA8888
mpeg3_check_sig = libmpeg3c.mpeg3_check_sig

mpeg3_open = libmpeg3c.mpeg3_open

mpeg3_open_copy = libmpeg3c.mpeg3_open_copy

mpeg3_close = libmpeg3c.mpeg3_close

mpeg3_set_cpus = libmpeg3c.mpeg3_set_cpus
```

```
mpeg3_set_mmx = libmpeg3c.mpeg3_set_mmx

mpeg3_has_audio = libmpeg3c.mpeg3_has_audio

mpeg3_total_astreams = libmpeg3c.mpeg3_total_astreams

mpeg3_audio_channels = libmpeg3c.mpeg3_audio_channels

mpeg3_sample_rate = libmpeg3c.mpeg3_sample_rate

mpeg3_audio_samples = libmpeg3c.mpeg3_audio_samples

mpeg3_set_sample = libmpeg3c.mpeg3_set_sample

mpeg3_get_sample = libmpeg3c.mpeg3_get_sample

mpeg3_read_audio = libmpeg3c.mpeg3_read_audio

mpeg3_reread_audio = libmpeg3c.mpeg3_reread_audio

mpeg3_read_audio_chunk = libmpeg3c.mpeg3_read_audio_chunk

mpeg3_has_video = libmpeg3c.mpeg3_has_video

mpeg3_total_vstreams = libmpeg3c.mpeg3_total_vstreams

mpeg3_video_width = libmpeg3c.mpeg3_video_width

mpeg3_video_height = libmpeg3c.mpeg3_video_height

mpeg3_aspect_ratio = libmpeg3c.mpeg3_aspect_ratio

mpeg3_frame_rate = libmpeg3c.mpeg3_frame_rate
```

```
mpeg3_video_frames = libmpeg3c.mpeg3_video_frames

mpeg3_set_frame = libmpeg3c.mpeg3_set_frame

mpeg3_get_frame = libmpeg3c.mpeg3_get_frame

mpeg3_seek_percentage = libmpeg3c.mpeg3_seek_percentage

mpeg3_tell_percentage = libmpeg3c.mpeg3_tell_percentage

mpeg3_previous_frame = libmpeg3c.mpeg3_previous_frame

mpeg3_end_of_audio = libmpeg3c.mpeg3_end_of_audio

mpeg3_end_of_video = libmpeg3c.mpeg3_end_of_video

mpeg3_get_time = libmpeg3c.mpeg3_get_time

mpeg3_read_frame = libmpeg3c.mpeg3_read_frame

mpeg3_read_yuvframe = libmpeg3c.mpeg3_read_yuvframe

mpeg3_drop_frames = libmpeg3c.mpeg3_drop_frames

mpeg3_read_video_chunk = libmpeg3c.mpeg3_read_video_chunk

mpeg3demux_copy_titles = libmpeg3c.mpeg3demux_copy_titles

mpeg3_new_css = libmpeg3c.mpeg3_new_css

mpeg3_new_demuxer = libmpeg3c.mpeg3_new_demuxer

mpeg3_delete_demuxer = libmpeg3c.mpeg3_delete_demuxer
```

```
mpeg3demux_read_data = libmpeg3c.mpeg3demux_read_data

mpeg3demux_length = libmpeg3c.mpeg3demux_length

mpeg3_get_demuxer = libmpeg3c.mpeg3_get_demuxer

mpeg3demux_tell = libmpeg3c.mpeg3demux_tell

mpeg3demux_tell_percentage = libmpeg3c.mpeg3demux_tell_percentage

mpeg3demux_get_time = libmpeg3c.mpeg3demux_get_time

mpeg3demux_eof = libmpeg3c.mpeg3demux_eof

mpeg3demux_bof = libmpeg3c.mpeg3demux_bof

mpeg3demux_start_reverse = libmpeg3c.mpeg3demux_start_reverse

mpeg3demux_start_forward = libmpeg3c.mpeg3demux_start_forward

mpeg3_new_title = libmpeg3c.mpeg3_new_title

mpeg3_new_timecode = libmpeg3c.mpeg3_new_timecode

mpeg3_new_atrack = libmpeg3c.mpeg3_new_atrack

mpeg3_delete_atrack = libmpeg3c.mpeg3_delete_atrack

mpeg3_new_vtrack = libmpeg3c.mpeg3_new_vtrack

mpeg3_delete_vtrack = libmpeg3c.mpeg3_delete_vtrack

mpeg3audio_new = libmpeg3c.mpeg3audio_new
```

```
mpeg3audio_delete = libmpeg3c.mpeg3audio_delete

mpeg3video_new = libmpeg3c.mpeg3video_new

mpeg3video_delete = libmpeg3c.mpeg3video_delete

mpeg3video_read_frame = libmpeg3c.mpeg3video_read_frame

mpeg3_new_fs = libmpeg3c.mpeg3_new_fs

mpeg3_delete_fs = libmpeg3c.mpeg3_delete_fs

mpeg3io_open_file = libmpeg3c.mpeg3io_open_file

mpeg3io_close_file = libmpeg3c.mpeg3io_close_file

mpeg3io_read_data = libmpeg3c.mpeg3io_read_data

mpeg3bits_new_stream = libmpeg3c.mpeg3bits_new_stream

mpeg3bits_getbits = libmpeg3c.mpeg3bits_getbits
```

Listing F.5: Portion SWIG generated C Wrapper File for Libmpeg3

```
#define SWIGPYTHON


#include <string.h>

#if defined(_WIN32) || defined(__WIN32__)
#       if defined(_MSC_VER)
#               if defined(STATIC_LINKED)
#                       define SWIGEXPORT(a) a
#                       define SWIGIMPORT(a) extern a
#               else
#                       define SWIGEXPORT(a) __declspec(dllexport) a
#                       define SWIGIMPORT(a) extern a
#               endif
#       else
#               if defined(__BORLANDC__)
#                       define SWIGEXPORT(a) a _export
#                       define SWIGIMPORT(a) a _export
#               else
#                       define SWIGEXPORT(a) a
#                       define SWIGIMPORT(a) a
#               endif
#       endif
#else
#       define SWIGEXPORT(a) a
#       define SWIGIMPORT(a) a
#endif

#ifdef SWIG_GLOBAL
#define SWIGRUNTIME(a) SWIGEXPORT(a)
#else
#define SWIGRUNTIME(a) static a
#endif

#ifdef __cplusplus
extern "C" {
#endif

typedef void *(*swig_converter_func)(void *);
typedef struct swig_type_info *(*swig_dycast_func)(void **);

typedef struct swig_type_info {
  const char               *name;
  swig_converter_func       converter;
  const char               *str;
  swig_dycast_func          dcast;
  struct swig_type_info    *next;
  struct swig_type_info    *prev;
  void                     *clientdata;
} swig_type_info;

#ifdef SWIGNOINCLUDE

SWIGIMPORT(swig_type_info *) SWIG_TypeRegister(swig_type_info *);
SWIGIMPORT(swig_type_info *) SWIG_TypeCheck(char *c, swig_type_info *);
SWIGIMPORT(void *)           SWIG_TypeCast(swig_type_info *, void *);
SWIGIMPORT(swig_type_info *) SWIG_TypeDynamicCast(swig_type_info *, void **);
SWIGIMPORT(swig_type_info *) SWIG_TypeQuery(const char *);
SWIGIMPORT(void)             SWIG_TypeClientData(swig_type_info *, void *);

#else

 static swig_type_info *swig_type_list = 0;



SWIGRUNTIME(swig_type_info *)
SWIG_TypeRegister(swig_type_info *ti)
{
  swig_type_info *tc, *head, *ret, *next;

  tc = swig_type_list;
  while (tc) {
    if (strcmp(tc->name, ti->name) == 0) {

      head = tc;
      next = tc->next;
      goto l1;
    }
    tc = tc->prev;
  }
  head = ti;
  next = 0;


  ti->prev = swig_type_list;
  swig_type_list = ti;


 l1:
  ret = head;
  tc = ti + 1;

  while (tc->name) {
    head->next = tc;
    tc->prev = head;
    head = tc;
    tc++;
  }
  head->next = next;
  return ret;
}


SWIGRUNTIME(swig_type_info *)
SWIG_TypeCheck(char *c, swig_type_info *ty)
{
  swig_type_info *s;
  if (!ty) return 0;
  s = ty->next;
  while (s) {
    if (strcmp(s->name,c) == 0) {
      if (s == ty->next) return s;

      s->prev->next = s->next;
      if (s->next) {
        s->next->prev = s->prev;
      }

      s->next = ty->next;
      if (ty->next) ty->next->prev = s;
      ty->next = s;
      return s;
    }
    s = s->next;
  }
  return 0;
}


SWIGRUNTIME(void *)
SWIG_TypeCast(swig_type_info *ty, void *ptr)
{
  if ((!ty) || (!ty->converter)) return ptr;
  return (*ty->converter)(ptr);
}


SWIGRUNTIME(swig_type_info *)
SWIG_TypeDynamicCast(swig_type_info *ty, void **ptr)
{
  swig_type_info *lastty = ty;
  if (!ty || !ty->dcast) return ty;
  while (ty && (ty->dcast)) {
     ty = (*ty->dcast)(ptr);
```

```
      if (ty) lastty = ty;
   }
   return lastty;
}


SWIGRUNTIME(swig_type_info *)
SWIG_TypeQuery(const char *name) {
   swig_type_info *ty = swig_type_list;
   while (ty) {
      if (ty->str && (strcmp(name,ty->str) == 0)) return ty;
      if (ty->name && (strcmp(name,ty->name) == 0)) return ty;
      ty = ty->prev;
   }
   return 0;
}


SWIGRUNTIME(void)
SWIG_TypeClientData(swig_type_info *ti, void *clientdata) {
   swig_type_info *tc, *equiv;
   if (ti->clientdata) return;
   ti->clientdata = clientdata;
   equiv = ti->next;
   while (equiv) {
      if (!equiv->converter) {
         tc = swig_type_list;
         while (tc) {
            if ((strcmp(tc->name, equiv->name) == 0))
               SWIG_TypeClientData(tc,clientdata);
            tc = tc->prev;
         }
      }
      equiv = equiv->next;
```

```
  }
}
#endif

#ifdef __cplusplus
}

#endif




#include <stdlib.h>
#include "Python.h"

#ifdef __cplusplus
extern "C" {
#endif

#define SWIG_PY_INT     1
#define SWIG_PY_FLOAT   2
#define SWIG_PY_STRING  3
#define SWIG_PY_POINTER 4
#define SWIG_PY_BINARY  5


typedef struct swig_const_info {
    int type;
    char *name;
    long lvalue;
    double dvalue;
    void   *pvalue;
    swig_type_info **ptype;
} swig_const_info;
```

```
#ifdef SWIGNOINCLUDE

SWIGEXPORT(PyObject *)        SWIG_newvarlink();
SWIGEXPORT(void)              SWIG_addvarlink(PyObject *, char *, PyObject *(*)(void
    ), int (*)(PyObject *));
SWIGEXPORT(int)               SWIG_ConvertPtr(PyObject *, void **, swig_type_info *,
     int);
SWIGEXPORT(int)               SWIG_ConvertPacked(PyObject *, void *, int sz,
    swig_type_info *, int);
SWIGEXPORT(char *)            SWIG_PackData(char *c, void *, int);
SWIGEXPORT(char *)            SWIG_UnpackData(char *c, void *, int);
SWIGEXPORT(PyObject *)        SWIG_NewPointerObj(void *, swig_type_info *,int own);
SWIGEXPORT(PyObject *)        SWIG_NewPackedObj(void *, int sz, swig_type_info *);
SWIGEXPORT(void)              SWIG_InstallConstants(PyObject *d, swig_const_info
    constants[]);
#else



typedef struct swig_globalvar {
  char          *name;
  PyObject *(*get_attr)(void);
  int           (*set_attr)(PyObject *);
  struct swig_globalvar *next;
} swig_globalvar;

typedef struct swig_varlinkobject {
  PyObject_HEAD
  swig_globalvar *vars;
} swig_varlinkobject;

static PyObject *
swig_varlink_repr(swig_varlinkobject *v) {
  v = v;
  return PyString_FromString("<Global variables>");
}

static int
swig_varlink_print(swig_varlinkobject *v, FILE *fp, int flags) {
  swig_globalvar  *var;
  flags = flags;
  fprintf(fp,"Global variables { ");
  for (var = v->vars; var; var=var->next) {
    fprintf(fp,"%s", var->name);
    if (var->next) fprintf(fp,", ");
  }
  fprintf(fp," }\n");
  return 0;
}

static PyObject *
swig_varlink_getattr(swig_varlinkobject *v, char *n) {
  swig_globalvar *var = v->vars;
  while (var) {
    if (strcmp(var->name,n) == 0) {
      return (*var->get_attr)();
    }
    var = var->next;
  }
  PyErr_SetString(PyExc_NameError,"Unknown C global variable");
  return NULL;
}

static int
swig_varlink_setattr(swig_varlinkobject *v, char *n, PyObject *p) {
  swig_globalvar *var = v->vars;
  while (var) {
    if (strcmp(var->name,n) == 0) {
      return (*var->set_attr)(p);
    }
    var = var->next;
  }
  PyErr_SetString(PyExc_NameError,"Unknown C global variable");
  return 1;
}

statichere PyTypeObject varlinktype = {
  PyObject_HEAD_INIT(0)
  0,
  (char *)"swigvarlink",
  sizeof(swig_varlinkobject),
  0,
  0,
  (printfunc) swig_varlink_print,
  (getattrfunc) swig_varlink_getattr,
  (setattrfunc) swig_varlink_setattr,
  0,
  (reprfunc) swig_varlink_repr,
  0,
  0,
  0,
};


SWIGRUNTIME(PyObject *)
SWIG_newvarlink(void) {
  swig_varlinkobject *result = 0;
  result = PyMem_NEW(swig_varlinkobject,1);
  varlinktype.ob_type = &PyType_Type;
  result->ob_type = &varlinktype;
  result->vars = 0;
  result->ob_refcnt = 0;
  Py_XINCREF((PyObject *) result);
  return ((PyObject*) result);
}

SWIGRUNTIME(void)
SWIG_addvarlink(PyObject *p, char *name,
            PyObject *(*get_attr)(void), int (*set_attr)(PyObject *p)) {
  swig_varlinkobject *v;
  swig_globalvar *gv;
  v= (swig_varlinkobject *) p;
  gv = (swig_globalvar *) malloc(sizeof(swig_globalvar));
  gv->name = (char *) malloc(strlen(name)+1);
  strcpy(gv->name,name);
  gv->get_attr = get_attr;
  gv->set_attr = set_attr;
  gv->next = v->vars;
  v->vars = gv;
}


SWIGRUNTIME(char *)
SWIG_PackData(char *c, void *ptr, int sz) {
  static char hex[17] = "0123456789abcdef";
  int i;
  unsigned char *u = (unsigned char *) ptr;
  register unsigned char uu;
  for (i = 0; i < sz; i++,u++) {
    uu = *u;
    *(c++) = hex[(uu & 0xf0) >> 4];
    *(c++) = hex[uu & 0xf];
  }
  return c;
}
```

```
SWIGRUNTIME(char *)
SWIG_UnpackData(char *c, void *ptr, int sz) {
  register unsigned char uu = 0;
  register int d;
  unsigned char *u = (unsigned char *) ptr;
  int i;
  for (i = 0; i < sz; i++, u++) {
    d = *(c++);
    if ((d>= '0') && (d<= '9'))
      uu = ((d - '0') << 4);
    else if ((d>= 'a') && (d<= 'f'))
      uu = ((d - ('a'-10)) << 4);
    d = *(c++);
    if ((d>= '0') && (d<= '9'))
      uu |= (d - '0');
    else if ((d>= 'a') && (d<= 'f'))
      uu |= (d - ('a'-10));
    *u = uu;
  }
  return c;
}


SWIGRUNTIME(int)
SWIG_ConvertPtr(PyObject *obj, void **ptr, swig_type_info *ty, int flags) {
  swig_type_info *tc;
  char  *c;
  static PyObject *SWIG_this = 0;
  int     newref = 0;

  if (!obj) return 0;
  if (obj == Py_None) {
    *ptr = 0;
    return 0;
  }
#ifdef SWIG_COBJECT_TYPES
  if (!(PyCObject_Check(obj))) {
    if (!SWIG_this)
      SWIG_this = PyString_InternFromString("this");
    obj = PyObject_GetAttr(obj,SWIG_this);
    newref = 1;
    if (!obj) goto type_error;
    if (!PyCObject_Check(obj)) {
      PyDECREF(obj);
      goto type_error;
    }
  }
  *ptr = PyCObject_AsVoidPtr(obj);
  c = (char *) PyCObject_GetDesc(obj);
  if (newref) PyDECREF(obj);
  goto cobject;
#else
  if (!(PyString_Check(obj))) {
    if (!SWIG_this)
      SWIG_this = PyString_InternFromString("this");
    obj = PyObject_GetAttr(obj,SWIG_this);
    newref = 1;
    if (!obj) goto type_error;
    if (!PyString_Check(obj)) {
      PyDECREF(obj);
      goto type_error;
    }
  }
  c = PyString_AsString(obj);

  if (*c != '_') {
    *ptr = (void *) 0;
    if (strcmp(c,"NULL") == 0) {
      if (newref) { Py_DECREF(obj); }
      return 0;
    } else {
      if (newref) { Py_DECREF(obj); }
      goto type_error;
    }
  }
  c++;
  c = SWIG_UnpackData(c,ptr,sizeof(void *));
  if (newref) { Py_DECREF(obj); }
#endif

#ifdef SWIG_COBJECT_TYPES
cobject:
#endif

  if (ty) {
    tc = SWIG_TypeCheck(c,ty);
    if (!tc) goto type_error;
    *ptr = SWIG_TypeCast(tc,(void*) *ptr);
  }
  return 0;

type_error:
  if (flags) {
    if (ty) {
      char *temp = (char *) malloc(64+strlen(ty->name));
      sprintf(temp,"Type error. Expected %s", ty->name);
      PyErr_SetString(PyExc_TypeError, temp);
      free((char *) temp);
    } else {
      PyErr_SetString(PyExc_TypeError,"Expected a pointer");
    }
  }
  return -1;
}


SWIGRUNTIME(int)
SWIG_ConvertPacked(PyObject *obj, void *ptr, int sz, swig_type_info *ty, int flags)
    {
  swig_type_info *tc;
  char  *c;

  if ((!obj) || (!PyString_Check(obj))) goto type_error;
  c = PyString_AsString(obj);

  if (*c != '_') goto type_error;
  c++;
  c = SWIG_UnpackData(c,ptr,sz);
  if (ty) {
    tc = SWIG_TypeCheck(c,ty);
    if (!tc) goto type_error;
  }
  return 0;

type_error:

  if (flags) {
    if (ty) {
      char *temp = (char *) malloc(64+strlen(ty->name));
      sprintf(temp,"Type error. Expected %s", ty->name);
      PyErr_SetString(PyExc_TypeError, temp);
      free((char *) temp);
    } else {
      PyErr_SetString(PyExc_TypeError,"Expected a pointer");
    }
  }
  return -1;
}



SWIGRUNTIME(PyObject *)
SWIG_NewPointerObj(void *ptr, swig_type_info *type, int own) {
  PyObject *robj;
  if (!ptr) {
    Py_INCREF(Py_None);
    return Py_None;
  }
#ifdef SWIG_COBJECT_TYPES
  robj = PyCObject_FromVoidPtrAndDesc((void *) ptr, (char *) type->name, NULL);
#else
  {
    char result[1024];
    char *r = result;
    *(r++) = '_';
    r = SWIG_PackData(r,&ptr,sizeof(void *));
    strcpy(r,type->name);
    robj = PyString_FromString(result);
  }
#endif
  if (!robj || (robj == Py_None)) return robj;
  if (type->clientdata) {
    PyObject *inst;
    PyObject *args = Py_BuildValue((char*)"(O)", robj);
    Py_DECREF(robj);
    inst = PyObject_CallObject((PyObject *) type->clientdata, args);
    Py_DECREF(args);
    if (own) {
      PyObject *n = PyInt_FromLong(1);
      PyObject_SetAttrString(inst,(char*)"thisown",n);
      Py_DECREF(n);
    }
    robj = inst;
  }
  return robj;
}

SWIGRUNTIME(PyObject *)
SWIG_NewPackedObj(void *ptr, int sz, swig_type_info *type) {
  char result[1024];
  char *r = result;
  if ((2*sz + 1 + strlen(type->name)) > 1000) return 0;
  *(r++) = '_';
  r = SWIG_PackData(r,ptr,sz);
  strcpy(r,type->name);
  return PyString_FromString(result);
}


SWIGRUNTIME(void)
SWIG_InstallConstants(PyObject *d, swig_const_info constants[]) {
  int i;
  PyObject *obj;
  for (i = 0; constants[i].type; i++) {
    switch(constants[i].type) {
    case SWIG_PY_INT:
      obj = PyInt_FromLong(constants[i].lvalue);
      break;
    case SWIG_PY_FLOAT:
      obj = PyFloat_FromDouble(constants[i].dvalue);
      break;
    case SWIG_PY_STRING:
      obj = PyString_FromString((char *) constants[i].pvalue);
      break;
    case SWIG_PY_POINTER:
      obj = SWIG_NewPointerObj(constants[i].pvalue, *(constants[i]).ptype,0);
      break;
    case SWIG_PY_BINARY:
      obj = SWIG_NewPackedObj(constants[i].pvalue, constants[i].lvalue, *(constants[
          i].ptype));
      break;
    default:
      obj = 0;
      break;
    }
    if (obj) {
      PyDict_SetItemString(d,constants[i].name,obj);
      Py_DECREF(obj);
    }
  }
}

#endif

#ifdef __cplusplus
}
#endif




#define  SWIGTYPE_p_unsigned_char swig_types[0]
#define  SWIGTYPE_p_p_unsigned_char swig_types[1]
#define  SWIGTYPE_p_mpeg3_t swig_types[2]
#define  SWIGTYPE_p_double swig_types[3]
#define  SWIGTYPE_p_mpeg3video_t swig_types[4]
#define  SWIGTYPE_p_mpeg3_bits_t swig_types[5]
#define  SWIGTYPE_p_mpeg3_demuxer_t swig_types[6]
```

```
#define SWIGTYPE_p_float swig_types[7]
#define SWIGTYPE_p_mpeg3_fs_t swig_types[8]
#define SWIGTYPE_p_void swig_types[9]
#define SWIGTYPE_p_mpeg3_atrack_t swig_types[10]
#define SWIGTYPE_p_mpeg3_vtrack_t swig_types[11]
#define SWIGTYPE_p_short swig_types[12]
#define SWIGTYPE_p_p_char swig_types[13]
#define SWIGTYPE_p_char swig_types[14]
#define SWIGTYPE_p_p_mpeg3_title_t swig_types[15]
#define SWIGTYPE_p_mpeg3_title_t swig_types[16]
#define SWIGTYPE_p_mpeg3_css_t swig_types[17]
#define SWIGTYPE_p_long swig_types[18]
#define SWIGTYPE_p_int swig_types[19]
#define SWIGTYPE_p_mpeg3audio_t swig_types[20]
 static swig_type_info *swig_types[22];




#define SWIG_init    initlibmpeg3c

#define SWIG_name    "libmpeg3c"

#include <ctype.h>


 static swig_type_info *SWIG_POINTER_int_p = 0;
 static swig_type_info *SWIG_POINTER_short_p =0;
 static swig_type_info *SWIG_POINTER_long_p = 0;
 static swig_type_info *SWIG_POINTER_float_p = 0;
 static swig_type_info *SWIG_POINTER_double_p = 0;
 static swig_type_info *SWIG_POINTER_char_p = 0;
 static swig_type_info *SWIG_POINTER_char_pp = 0;
static PyObject *ptrvalue(PyObject *PTRVALUE, int index, char *type) {
  void      *ptr;
  char      *s;
  PyObject *obj;

  if (SWIG_ConvertPtr(PTRVALUE,&ptr,0,0)) {
    PyErr_SetString(PyExc_TypeError,"Type error in ptrvalue. Argument is not a valid
         pointer value.");
    return NULL;
  }


  if (!type) {

    if (!SWIG_ConvertPtr(PTRVALUE,&ptr,SWIG_POINTER_int_p,0)) {
      type = (char *)"int";
    } else if (!SWIG_ConvertPtr(PTRVALUE,&ptr,SWIG_POINTER_double_p,0)) {
      type = (char *)"double";
    } else if (!SWIG_ConvertPtr(PTRVALUE,&ptr,SWIG_POINTER_short_p,0)) {
      type = (char *)"short";
    } else if (!SWIG_ConvertPtr(PTRVALUE,&ptr,SWIG_POINTER_long_p,0)) {
      type = (char *)"long";
    } else if (!SWIG_ConvertPtr(PTRVALUE,&ptr,SWIG_POINTER_float_p,0)) {
      type = (char *)"float";
    } else if (!SWIG_ConvertPtr(PTRVALUE,&ptr,SWIG_POINTER_char_p,0)) {
      type = (char *)"char";
    } else if (!SWIG_ConvertPtr(PTRVALUE,&ptr,SWIG_POINTER_char_pp,0)) {
      type = (char *)"char *";
    } else {
      type = (char *)"unknown";
    }
  }
  if (!ptr) {
    PyErr_SetString(PyExc_TypeError,"Unable to dereference NULL pointer.");
    return NULL;
  }

  if (strcmp(type,"int") == 0) {
    obj = PyInt_FromLong((long) *(((int *) ptr) + index));
  } else if (strcmp(type,"double") == 0) {
    obj = PyFloat_FromDouble((double) *(((double *) ptr)+index));
  } else if (strcmp(type,"short") == 0) {
    obj = PyInt_FromLong((long) *(((short *) ptr)+index));
  } else if (strcmp(type,"long") == 0) {
    obj = PyInt_FromLong((long) *(((long *) ptr)+index));
  } else if (strcmp(type,"float") == 0) {
    obj = PyFloat_FromDouble((double) *(((float *) ptr)+index));
  } else if (strcmp(type,"char") == 0) {
    obj = PyString_FromString(((char *) ptr)+index);
  } else if (strcmp(type,"char *") == 0) {
    char *c = *(((char **) ptr)+index);
    if (c) obj = PyString_FromString(c);
    else obj = PyString_FromString("NULL");
  } else {
    PyErr_SetString(PyExc_TypeError,"Unable to dereference unsupported datatype.");
    return NULL;
  }
  return obj;
}


static PyObject *ptrcreate(char *type, PyObject *PYVALUE, int numelements) {
  void      *ptr;
```

```
PyObject *obj;
int      sz;
swig_type_info *cast;
char     temp[40];



if (strcmp(type,"int") == 0) {
  sz = sizeof(int)*numelements;
  cast = SWIG_POINTER_int_p;
} else if (strcmp(type,"short") == 0) {
  sz = sizeof(short)*numelements;
  cast = SWIG_POINTER_short_p;
} else if (strcmp(type,"long") == 0) {
  sz = sizeof(long)*numelements;
  cast = SWIG_POINTER_long_p;
} else if (strcmp(type,"double") == 0) {
  sz = sizeof(double)*numelements;
  cast = SWIG_POINTER_double_p;
} else if (strcmp(type,"float") == 0) {
  sz = sizeof(float)*numelements;
  cast = SWIG_POINTER_float_p;
} else if (strcmp(type,"char") == 0) {
  sz = sizeof(char)*numelements;
  cast = SWIG_POINTER_char_p;
} else if (strcmp(type,"char *") == 0) {
  sz = sizeof(char *)*(numelements+1);
  cast = SWIG_POINTER_char_pp;
} else {
  PyErr_SetString(PyExc_TypeError,"Unable to create unknown datatype.");
  return NULL;
}
```

```
ptr = (void *) calloc(1,sz);
if (!ptr) {
  PyErr_SetString(PyExc_MemoryError,"Out of memory in swig_create.");
  return NULL;
}



if (PYVALUE) {
  if (strcmp(type,"int") == 0) {
    int *ip,i,ivalue;
    ivalue = (int) PyInt_AsLong(PYVALUE);
    ip = (int *) ptr;
    for (i = 0; i < numelements; i++)
      ip[i] = ivalue;
  } else if (strcmp(type,"short") == 0) {
    short *ip,ivalue;
    int i;
    ivalue = (short) PyInt_AsLong(PYVALUE);
    ip = (short *) ptr;
    for (i = 0; i < numelements; i++)
      ip[i] = ivalue;
  } else if (strcmp(type,"long") == 0) {
    long *ip,ivalue;
    int i;
    ivalue = (long) PyInt_AsLong(PYVALUE);
    ip = (long *) ptr;
    for (i = 0; i < numelements; i++)
      ip[i] = ivalue;
  } else if (strcmp(type,"double") == 0) {
    double *ip,ivalue;
    int i;
    ivalue = (double) PyFloat_AsDouble(PYVALUE);
```

```
      ip = (double *) ptr;
      for (i = 0; i < numelements; i++)
        ip[i] = ivalue;
    } else if (strcmp(type,"float") == 0) {
      float *ip,ivalue;
      int i;
      ivalue = (float) PyFloat_AsDouble(PYVALUE);
      ip = (float *) ptr;
      for (i = 0; i < numelements; i++)
        ip[i] = ivalue;
    } else if (strcmp(type,"char") == 0) {
      char *ip,*ivalue;
      ivalue = (char *) PyString_AsString(PYVALUE);
      ip = (char *) ptr;
      strncpy(ip,ivalue,numelements-1);
    } else if (strcmp(type,"char *") == 0) {
      char **ip, *ivalue;
      int  i;
      ivalue = (char *) PyString_AsString(PYVALUE);
      ip = (char **) ptr;
      for (i = 0; i < numelements; i++) {
        if (ivalue) {
          ip[i] = (char *) malloc(strlen(ivalue)+1);
          strcpy(ip[i],ivalue);
        } else {
          ip[i] = 0;
        }
      }
      ip[numelements] = 0;
    }
  }


  obj = SWIG_NewPointerObj(ptr,cast,0);
  return obj;
}
```

Appendix G

Matlab Source

Listing G.1: Block operator Matlab function

```
function b=blk_mask(i_blk,msk)

a= size(i_blk,1);
b= size(i_blk,2);
if a~=b and b~=8,
    print( 'incorrect block size')
    return
end
if size(msk) ~= 8 8,
    print( 'incorrect block size')
    return
end
b=i_blk.*msk;
end
```

Listing G.2: DCT mask operator Matlab function

```
function b=dct_mskmat(sigma)
x=(0:7);
u_x=x'*x;

sigma_u_s=sigma*u_x;
b=u_x.*exp(-sigma_u_s);
```

Listing G.3: sigma masks Matlab program

```
sigma=0.1:0.1:2;
z=1:20;

msk_fun = @blck_mask;
dctfun = @dct2;
idctfun= @idct2;


I = imread('cameraman.tif');
imagesc(I), colormap('gray'), title(['sigma=',num2str(0.10)])
Mn= ones(1,20);
Mx= ones(1,20);
Mean = ones(1,20);
Std= ones(1,20);
Var = ones(1,20);
hignum = ones(1,20);
lownum = ones(1,20);

for i=1:20
    msk = dct_mskmat(sigma(i));
    J= blkproc(I,[8 8],dctfun);
    msk = msk.*-1;
    K= blkproc(J,[8 8],msk_fun,msk);
    L= blkproc(K,[8 8],idctfun);




    figure
    title(['sigma=',num2str(sigma(i))])
    subplot(1,2,1); imshow(K,[]);title(['dct after mask sigma=',num2str(sigma(i))]);
    subplot(1,2,2); imshow(L,[]); title(['idct masked sigma =',num2str(sigma(i))]);
    s_z=size(K);
    s_z(1)*s_z(2);
    nw_sz=s_z(1)*s_z(2);
    rav_K = reshape(K,nw_sz,1);
    Mn(i)= min(rav_K);
    Mx(i)= max(rav_K);
    Mean(i)=mean(rav_K);
    Var(i)=var(rav_K);
    Std(i)=std(rav_K);
    min_bound=-6*Std(i)
    max_bound=6*Std(i)
    hignum(i)= sum(rav_K>max_bound)
    lownum(i)=sum(rav_K<min_bound)

end
figure
plot(z,Mn,z,Mx,z,Mean,z,Var,z,Std,z,hignum,z,lownum);
title('sigma = 0.1:0.1:2')
legend('Min','Max','Mean','Variance','Standard Deviation','> 6*Std','<6*Std')
```

References

[1] Kjersti Aas and Line Eikvil. A survey on: Content-based access to image and video databases. Technical Report Report Number 915, Norsk Regensentral (Norwegian Computing Center), Gaustadalleen 23, Post Office Box 114 Blindern, N-0314 Oslo, Norway, 1997.

[2] Brett Adams. *Mapping the Semantic Landscape of FIlm: Computational Extraction of Indices Through Film Grammar.* PhD thesis, Curtin University of Technology, 2002.

[3] Sabzall Aghagolzadeh and Okan K. Ersoy. Transform edge detection. *SPIE Optical Engineering*, 32(5), May 1993.

[4] Philippe Aigrain and Philippe Joly. The automatic real-time analysis of film editing and transition effects and it's applications. *Comput. and Graphics*, 18(11):93–103, 1994.

[5] H. Ait-Kaci. *Warrens Abstract Machine: A Tutorial Reconstruction.* MIT Press, 1991.

[6] Hassan Ait-Kaci. *Warrens Abstract Machine: A Tutorial Reconstruction.* ISBN 0-262-51058-8. The MIT Press, 1991.

[7] O.S. Akhmanova, Z.S. Vigodskaya, T.P. Gorbunova, N.F. Rothstein, A.I. Smirnitsky, and A.M. Taube. *Russian-English Dictionary.* Soviet Encyclopaedia Publishing House, Moscow, 1969.

[8] Francesc Alted. PyTables, Processing and Analyzing Large Amounts of Data in Python, June 2003.

[9] Amphion. CS6790 MPEG-4 Video Codec Ultra Low Power.

[10] Amphion. CS6651 MPEG-2 Video Decoder MP@ML for FPGA, 2003.

[11] Sandy Anderson and Brian M. Slator. Requiem for a theory: the 'story grammar' story. *Journal of Experimental and Theoretical Artificial Intelligence*, 26:253–275, 1990.

[12] Preetha Appan and Hari Sundaram. Networked multimedia event exploration. In *Proceedings of the 12th annual ACM international conference on Multimedia*, pages 40 – 47, 2004.

[13] S. Arghagolzadeh and O.K. Ersoy. Transform domain edge detection. *Optical Engineering*, 32(5):933–943, May 1993.

[14] Daniel Arijon. *Grammar of the FIlm Language.* ISBN 1-879505-07-X. Silman James Press, 1181 Angelo Drive, Beverley Hills, Los Angeles CA 90210, 1976.

[15] Aristotle. *The Poetics of Aristotle Edited with Critical Notes and Translation.* Macmillan and Co. Ltd., London, New York, 3rd edition, 1902.

[16] F. Arman, A. Hsu, and M-Y. Chiu. Image processing on compressed data for large video databases. In *Proceedings of First ACM International Conference on Multimedia*, pages 267–272, 1993.

[17] Jurgen Assfalg, Marco Bertini, Carlo Colombo, Alberto Del Bimbo, and Walter Nunziati. Semantic annotation of soccer videos: auomatic highlights identifcation. *Computer Vision and Image Understanding*, 92(2/3), November/December 2003.

[18] Dana H. Ballard and Christopher M. Brown. *Computer Vision.* ISBN 0131653164. Prentice Hall, Englewood Cliifs, New Jersey, 1982.

[19] Roland Barthes. *Image Music Text.* ISBN 0006861350. Fontana Press, London, 1977.

[20] Roland Barthes. *L'obvie et l'obtus*, chapter Rhetorique de l'image, pages 25–42. ISBN 2020062488. Seuil, 1982.

[21] B.G. Batchelor. *Intelligent Image Processing in Prolog.* ISBN 0-540-19647-1 and ISBN 0-387-19647-1. Springer Verlag, 1991.

[22] Bruce Batchelor and Frederick Waltz. *Intelligent Machine Vision Techniques, Implementations and Applications.* ISBN 3540762248. Springer Verlag, 2001.

[23] David M. Beazley. SWIG : An Easy to Use Tool For Integrating Scripting Languages with C and C++. *Fourth Annual USENIX Tcl/Tk Workshop,*, July 1996.

[24] D.M. Beazley. *Python Essntial Reference.* ISBN 0-7357-0901-7. New Riders, 1999.

[25] D.M. Beazley and P. S. Lomdahl. Lightweight computational steering of very large scale molecular dynamics simulation. In *Supercomputing '96*, Pittsburgh, USA, 1996.

[26] D.M Beazley and P. S. Lomdahl. Feeding a large-scale physics application to python. In *6th International Python Conference*, San Jose, California, October 14-17, 1997.

[27] A.B. Benitez, S.-F. Chang, and J.R. Smith. Mpeg-7 mds content description tools and applications. In *Proceedings of the 2001 International Conference on Computer Analysis of Images and Patterns*, Warsaw Poland, September 2001.

[28] Berkeley Multimedia Research Centre. *The Berkeley Mpeg player.* Regents of the University of California, 1995.

[29] Brent Berlin and Paul Kay. *Basic Color terms: their universality and evolution.* Berkeley University of California Press, 1969.

[30] T. Berners-Lee. Conceptual graphs and the semantic web, 2001.

[31] I. Biederman. Human image understanding: Recent research and a theorey. *Computer Vision, Graphics and Image Processing*, 32(1):29–73, 1985.

[32] J.B. Black and G.H. Bower. Story understanding as problem solving. *Poetics*, 9:223–250, 1980.

[33] J.B. Black and R. Wilensky. An evaluation of story grammars. *Cognitive Science*, 3:213–230, 1979.

[34] Andrew Blake and Michael Isard. *Active Contours.* ISBN 3-540-76217-5. Springer Verlag, Berlin Heidelberg, 1998.

[35] Giles R. Bloch. *Elements d'une Machine de Montage Pour l'Audio-Visuel.* PhD thesis, Ecole Nationale Superieure Des Telecommunications, 1986.

[36] Giles R. Bloch. From concepts to film sequences. In *RIAO 88*, pages 760–767, March 21-24 1988.

[37] D. Bordwell. *Narration in the Fiction Film.* ISBN 0415018773. Methuen, Routledge, 1985, 1995.

[38] J.S. Boreczky and L.A. Rowe. Comparsion of video shot boundary detection techniques. In *Proceedings of 1ST SPIE International Symposium Electronic Imaging*, San Jose, California, 1996.

[39] N.K. Bose and P. Liang. *Neural Network Fundamentals.* ISBN 0-07-052182-4. McGraw Hill, 1996.

[40] R.J. Brachman and H.J. Levesque. *Readings in Knowledge Representation.* ISBN 0-934613-01-X. Morgan Kaufmann Publishers Inc, 1985.

[41] Ronald J. Brachman and Hector J. Levesque., editors. *Readings in knowledge representation.* ISBN 093461301X. Morgan Kaufmann Publishers, 1985.

[42] N. Bradley. *The XML Companion.* ISBN 0-201-342855. Addison Wesley, 1998.

[43] Marta Braun. *Picturing time: the work of Etienne Marey.* ISBN 022607173. University of Chicago Press, Chicago, 1992.

[44] Phil Brodatz. *Textures A Photogrpahic Album for Artisits and Designers.* ISBN 048640699-7. Dover, Mineola, 1996, 1999.

[45] R. Brunelli, O. Mich, and C.M. Modena. A survey of the automatic indexing of video data. *Journal of Computer Vision and Image Representation*, 10:78–112, 1999.

[46] Warren Buckland. *A Companion to Fim Theory*, chapter 6, pages 84–104. ISBN 0 631 20644-2. Blackwell, 108 Cowley Road Oxford OX4 1JF, 1999.

[47] S. Butler and Alan P. Parkes. Space time diagrams for film structure visualisation. *Signal Processing: Image Communications*, 8(4):269–280, 1996.

[48] S. Butler and Alan P. Parkes. Film sequence generation strategies for automatic intelligent video editing. *Applied Artifical Intelligence*, 11(4):367–388, 1997.

[49] D. M.-C. Ip C.-M. Lee. A robust approach for camera break detection in color video sequences. In *Proc. IAPR Workshop Machine Vision Applications*, pages 502–505, Kawasaki, Japan, 1994.

[50] Roberto Cabeza and Alan Kingstone. *Handbook of Functional Neuroimaging of Cognition.* ISBN 0262032805. Bradord Books MIT Press, Cambridge Massachusettes, 2001.

[51] Joseph Campbell. *The Hero with a Thousand Faces.* ISBN 0586085718. Fontana Press, London, 1949, 1993.

[52] John Canny. A computational approach to edge detection. *IEE Transactions on Pattern Analysis and Machine Inteligence*, 8(6):679–625, 1986.

[53] John M. Carroll. *Towards a Strucural Psychology of the Cinema.* ISBN 9027934479. Mouton, The Hague, Netherlands, 1980.

[54] S. Chatman. *Story and Discourse: Narrative Structure in Fiction and in Film.* ISBN 0-8014-9186-X. Cornell University, 1978, 1993.

[55] CH Chen, LF Pau, and PSP Wang. *Handbook of Pattern Recognition and Computer Vision.* ISBN 981-02-1136-8. World Scientific, Farrer Raod, Singapore, 1993.

[56] Lei Chen, Shriq J. Rizvi, and M. Tamer Oszu. Incorporating audio cues into dialog and action scene extraction. In *Proceedings of SPIE Conference on Storage and Retrieval for Media Databases*, pages 252–264, San Jose California, Jnauary 2003.

[57] B. Chipraset and K.R. Rao. Discrete cosine transform filtering. *Signal Processing*, 19:233–245, 1990.

[58] Noam Chomsky. *Syntatic Structures.* ISBN 9027933855. Mouton, The Hague, Netherlands, 1957, 1976.

[59] W.F. Clocksin and C.S. Mellish. *Programming in Prolog.* ISBN 0-387-58350-5. Springer Verlag, 1994.

[60] B.N. Colby. A partial grammar of eskimo folktales. *American Anthropologist*, pages 645–662, 1973.

[61] M.A. Conway. Sensory-perceptual episodic memory and its context: autobiographical memory. *Philosophical Transactions of the Royal Society of London*, 356:1375–1484, 2001.

[62] David Cronenberg. *Crash.* ISBN 0 571 19127 4. Faber and Faber, 1996.

[63] Chapman D. *Vision, Instruction, and Action.* PhD thesis, Massachusettets Insitute of Technology, 1990.

[64] Steven d Katz. *Film Directing Shot by Shot.* ISBN 0941188108. Michael Wise Productions, Studio CIty, 1991.

[65] Roy G. D'Andrade and A. Kimball Romney. A quantative model for transforming reflectance spectra into munsell color space using cone sensitivity functions and opponent process weights. *Proceedings of the National Academy of Sciences*, 100(10):6281–6286, May 2003.

[66] J Davidoff and E.K. Warrington. A dissociation of shape discrimination and figure-ground perceprion in a patient with normal acuity. *Neuropsychologia*, 1(31):83–93, 1993.

[67] Marc Davis. Editing out video editing. *IEEE Multimedia*, 10(2):54–56, 2003.

[68] Marc Elliot Davis. *Media Streams: Representing Video for Retreival and Repurposing.* PhD thesis, Massachusettets Insitute of Technology, February 1995.

[69] Clarisse Sieckenius de Souza. *The Semiotic Engineering of Human Computer Interaction.* ISBN 0-262-04220-7. MIT Press, 2005.

[70] Daniel Defoe. *The life and adventures of Robinson Crusoe, of York, mariner.* Gall and Inglis, 1870.

[71] John R. Deller, John H.L. Hansen, and John G. Proakis. *Dicrete Time Processing of Speech Signals.* ISBN 0-7803-5386-2. IEEE Press, Macmillan, New York, 1993 2000.

[72] S.J. DeRose and D.G. Durand. *Making Hypermedia Work: A Users Guide to HyTime.* ISBN 0-7923-9432-1. Kluwer Academic Publishers, Boston, 1994.

[73] Antoni Diller. *Z An Introduction to Formal Methods.* ISBN 047193973. Wiley Interscience Publication, 2nd edition, 1994.

[74] A. Divakaran, K. A. Peker, K.A. R. Radharkishnan, Z. Xiong, and R. Cabasson. Video summarization using mpeg-7 motion activity and audio descriptors. In *Dimacs Workshop on Video Mining.* Kluwer Academic Publishers, October 2003.

[75] Lubomir Dolezel. *Semiosis: Semiotics and the History of Culture. In Honorem Georgii Lotman*, chapter Aristotelian Poetics as a Science of Literature, pages 125–138. Michigan Slavic Contributions, 1984.

[76] Chitra Dorai. Bridging the semantic gap in content management systems: Computational media aesthetics. In *Proceedings of Cosign 2001: Computational Semiotics*, CWI, Amsterdam (The Netherlands), 10th - 12th September 2001.

[77] Dublin Core. *Dublin Core Metadata Element Set, Version 1.1: Reference Description Dublin Core Metadata Initiative*, 2003.

[78] Touradj Ebrahimi and Fernando Pereira. *The MPEG-4 Book.* ISBN 0130616214. Prentice Hall PTR, 2002.

[79] Umberto Eco. *Movies and methods: an anthology*, volume 1 of *ISBN 0520031512*, chapter Articulations of the Cinematographic Code, pages 591–606. University of California Press, 1976.

[80] Umberto Eco. *A Theory of Semiotics.* ISBN 0-253-35955. Indiana University Press, Bloomington, London, 1976.

[81] Leatrice Eisman and Lawrence Herbert. *The Pantone Book of Colour: over 1000 Colour Standards, Colour Basics and Guidelines for Design Fashion, Furnishings and More.* ISBN 0810937115. Harry N. Adams Inc., 1992.

[82] D.F. Elliot and K.R. Rao. *Fast Trasnsforms, Algorithms Analyses Applications.* ISBN 0 12 237080 5. Academic Press, 1982.

[83] Thomas Elsaesser and Adam Barker. *Early Cinema Space Frame and Narrative*, chapter Introduction to Part 3 The Continuity System Grifiith and Beyond, pages 293–313. ISBN 0-85170-245-7. British Film Institute, 21 Stephen Street, London W1P 1PL, 1990.

[84] Robert Englemore and Tony Morgan. *Blackboard Systems.* ISBN 0-201-17431-6. Addison Wesley, Wokinham, England, 1988.

[85] Francois Ennesser and Gerard Medioni. Finding waldo or focus of attnation using local color information. *IEE Transactions on Pattern Analysis and Machine Inteligence*, 17(8):805–809', August 1995.

[86] Hans-Erik Eriksson and Magnus Penker. *UML Toolkit.* ISBN 0-471-19161-2. John Wiley and Sons, Inc, 1998.

[87] Okan K. Ersoy and David Hansen. Adaptive approach to edge detection. *Optical Engineering*, 34(11):3271–3276, November 1995.

[88] Euripides. *Greek Drama*, chapter Trojan Women, pages 257–287. LCCCN 65-26668. Bantam World Drama, 1960.

[89] Christiane Fellbaum. *WordNet an Electronic Lexical Database.* ISBN 0-262-06197-X. Bradord Books MIT Press, 1998.

[90] Jaques Ferber. *Multi-Agent Systems; An introduction to distributed artificial intelligence*, volume ISBN 0-201-36048-9. Pearson Education, London, 1999.

[91] Charlotte Fiell and Peter Fiell. *1000 Chairs.* ISBN 3822857602. Taschen, August 2000.

[92] T. Finin, R.Fritzson, D. McKay, and R. McEntire. KQML as an Agent Communication Language (1994). In *Proceedings of the 3rd International Conference on Information and Knowledge Management (CIKM'94)*, Gaithersburg, Maryland, November 29 - December 2, 1994. National Institute of Standards and Technology (NIST).

[93] Peter C. Fishburn. *Interval orders and interval graphs: a study in partially ordered sets.* ISBN 0471812846. Wiley Interscience Publication, 1985.

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

[95] Monika Fludernik. *The Fictions of Language and the Languages of Fiction: The Linguistic Representation of Speech and Conciousness.* Routledge, London, 1993.

[96] Jonathan Foote, John S. Boreczky, and Lynn D. Wilcox. An intelligent media browser using multimodal analysis. In *1ST/SPIE Conference on Storage and Retrieval for Image and Video Databases*, pages 375–380, Bristol, England, 1998.

[97] Department for Education and Skills. *Further Literacy Support.* DfES 0584/2002. Department for Education and Skills, 2002.

[98] C. L. Forgy. Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence*, 19:17–37, 1982.

[99] W.T. Freeman and E.H. Adleson. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(9):891–906, 1991.

[100] O. Chevet T. Dechilly R. Ronfard G. Auffret J. Carrive and B. Bachimont. Audiovisual-based hypermedia authoring: using structured representations for efficient access to av documents. In *Proceedings of the 10th ACM conference on Hypertext and Hypermedia*, pages 169– 178, Darmstadt, Germany, February 21-25, 1999.

[101] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Software.* ISBN 0-201-63361-2. Addison Wesley Professional Computing Series, 1995.

[102] Karen M. Gardner, Alexander R. Rush, Michael Crist, Robert Konitzer, and Bobbin Teergarden. *Cognitive Patterns: Problem Solving Frameworks for Object Technology.* ISBN 0-521-64998-6. Cambridge University Press, Cambridge, 1998.

[103] U. Gargi, S. Oswald, D. Kosiba, S. Devadiga, and R. Kastur. Evaluation of video sequence indexing and hierarchical video indexing. *Proceedings of 1ST SPIE Conference on Storage and Retrieval for Image and Video Databases iii*, SPIE2430:1522–1530, 1995.

[104] A. Garnham. What's wrong with story grammars? *Cognition*, 15:145–154, 1983.

[105] M.R. Genesereth and R.E. Fikes. Knowledge interchange format, version 3.0, reference manual. Technical report, Computer Science Department, Stanford University, 1992.

[106] Joseph C. Giarrantano. *CLIPS User's Guide*, 6.2 edition, March 2002.

[107] J. Giarranto and G. Riley. *Expert Systems, Principles and Programming.* ISBN 0-534-95053-1. PWS Publishing, 1998.

[108] R Goebel, D Khorram-Sefat, L Muckli, H Hacker, and W Singer. The constructive nature of vision: direct evidence from functional magnetic resonance imaging studies of apparent motion and motion imagery. *European Journal of Neuroscience*, 10:1563–1573, 1998.

[109] Rafael C. Gonzalez and Paul Wintz. *Digital Image Processing.* ISBN 0-201-11026-1. Addison Wesley, Reading, Mass, 1987.

[110] Perry Greenfield, Todd Miller, Rick White, J.e. Hsu, Pau I Barrett, and Jochen Kupper. *Numarray.* Space Telescope Science Institute, Baltimore, 0.8 edition, January 2004.

[111] J. Griffioen, R. Mehrotra, and R. Yavatkar. A semantic data model for embedded image information. In *Proceedings of Second International Conference on Information and Knowledge Management, Washington, D.C.*, pages 393–402, Washington, November 1993.

[112] W.I. Grosky and D.V. Sreenath. Metadata mediated browsing and retrieval in semantically-rich cultural image collections. In *Proceedings of the 2001 Tokyo Symposium for Digital Silk Roads*, Tokyo, Japan, December 2001.

[113] A. Gupta and R. Jain. Visual information retreival. *Communications of the ACM*, 40(5):70–79, May 1997.

[114] Antonin Guttman. R-trees a dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD international*, pages 47–57, Boston, Massachusetts, 1984. ACM.

[115] Losang Chokyi Gyaltsan, editor. *Offering to the Spiritual Guide, A special Guru yoga practice of Je Tsongkhapa's tradition.* Tharpa Publications, Ulverston, England, Glen Spey, New York.

[116] M. Potke H. Kriegel and T. Seid. Managing intervals efficiently in object-relational databases. In *Proceedings of IEEE International Conference on Very LArge Databases*, ISBN 1-55860-715-3, Cairo, Egypt., September 10-14 2000. Morgan Kaufmann.

[117] K. Haberlandt. Story time and rerading time of story constituents. *Poetics*, 9:99–118, 1980.

[118] K. Haberlandt, C. Berian, and J. Sandson. The epsisode schema in story processing. *Journal of Verbal Learning and Verbal Behaviour*, 19:635–650, 1980.

[119] Frank Halasz and Mayer Schwartz. The Dexter hypertext reference model. *Communications of the ACM*, 37(2):30–39, 1994.

[120] P. Halley. *Little Electric Chair Paintings.* ISBN 0971168709. Stellan Holm Gallery, 2002.

[121] Arun. Hampapur, Ramesh Jain, and Terry Weymouth. Digital video segmentation. In *Proceedings of the second ACM international conference on Multimedia*, pages 357–364, San Francisco, 1994. IEEE.

[122] L. Hardman, M. Worring, and D. Bulterman. Integrating the amsterdam hypermedia model with the standard reference model for intelligent multimedia systems, 1997.

[123] Edward Hartley, Alan P. Parkes, and D. Hutchison. *A Conceptual Framework to Support Content-Based Multimedia Applications*, pages 297–315. ISSN 0302-9743 LNCS 1629. Springer, 1999.

[124] Edward Hartley and Ian Sommerville. Comp-all email discussion about furniture for info lab, 2003.

[125] C. Hartshorne and P. Weiss, editors. *Elements of Logic 2.228 in Collected Papers of Charles Sanders Pierce*. LCCN 60-9172. The Belknap Press of Harvard University Press, 1932,1960.

[126] Jeremy Hawthorne. *A Concise Glossary of Contempory Literary Theory*. ISBN 0 340 69222 7. Arnold, 1998.

[127] Gerd Hillebrand, Paris Kanellakis, and Sridhar Ramaswamy. *Advances in Object Orientated Database Systems*, chapter Functional Programming Formalisms for OODBMS Methods, pages 73–99. ISBN 3-587 57825-0. Springer Verlag, 1994.

[128] Louis Hjelmslev. *Prolegomena to a Theory of Language*. LCCN62-7095. The University of Wisconsin Press, Madison, 1963.

[129] Horn and Schunk. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[130] D.A. Huffman. A method for the construction of minimum redundancy codes. *Proceedings of IRE*, 40:1098–1101, 1951.

[131] Kim Binstead in conversation with Sue Nelson. Passions, March 22 2004.

[132] Indian Conference on Computer Vision Graphics and Image Processing. *A Fast Method for Textual Annotation of Compressed Video*, Space Applications Centre (ISRO) Ahmedabad,, India, Decemeber 16-18 2002.

[133] Intel. *Open Source Computer Vision Library Reference Manual*, -001 original issue edition, December 2000.

[134] International Telecommunication Union. *ITU-T H.261 Video Codec for Audiovisual Services at px64 Kbits*. International Telecommunication Union, 1993.

[135] International Telecommunication Union. *ITU-T H.263 Video coding for low bit rate communication*, May 1998.

[136] ISO. *ISO 8879: 1986 Information Processing - Text and office systems - Standard Generalized Markup Language (SGML)*. ISO, 1986.

[137] ISO IEC. *ISO-ISO/IEC 10744:1997 Information technology Hypermedia/Time-based Structuring Language (HyTime)*, International Standards Organisation Geneva 20, Switzerland, 1997. ISO IEC.

[138] ISO IEC SC29 WG11. *ISO/IEC 15398 part 1 Systems.* ISO IEC, 2000.

[139] ISO IEC SC29 WG11. *ISO/IEC 15398 part 5 Multimedia Description Schemes.* ISO IEC, 2000.

[140] ISO IEC SC29 WG11. *ISO/IEC 15398 part 5 Visual.* ISO IEC, 2000.

[141] ISO IEC SC29 WG11. *ISO/IEC 14496-2:2001 Visual.* ISO IEC, 2001.

[142] ISO IEC SC29 WG11. *ISO IEC mnnnn Applications Document.* ISO IEC, 2003.

[143] ISO IEC SC29 WG11. *ISO/IEC 14496-10 Advanced Video Coding.* ISO IEC, 2003.

[144] ISO/IEC SC29 WG11. *ISO/IEC 11172 part 2 Video.* ISO/IEC, 1993.

[145] ISO/IEC SC29 WG11. *ISO/IEC 13818 part 3 video.* ISO/IEC, 1998.

[146] Peter Jackson. *Introduction to Expert Systems.* ISBN 0-201-17578. Addison Wesley, 1994.

[147] Allen J.F. Maintinaing knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[148] N.S. Johnson and N.S. Mandler. A tale of two structures: underlying and surface forms in stories. *Poetics*, 9:51–86, 1980.

[149] Phillip Nicholas Johnson-Laird. *Mental Models.* ISBN 0521241235. Cambridge University Press, Cambridge Cambridgeshire, 1983.

[150] Ed. C. Jones. PyOpenCV, 1999.

[151] Norton Juster. *The Dot and the Line: A Romance in Lower Mathematics.* ISBN: 1587170663. SeaStar Books, 2000.

[152] N. Kanwisher, J. McDermott, and M.M. Chun. The fusiform face area: A module in human extrastriate cortex specialized for face perception. *Journal of Neuroscience*, 17:4302–4311, 1997.

[153] R. Katsuri and R. Jain. *Computer Vision Principles*, chapter Dynamic Vision, pages 469–480. ISBN. IEEE Press, 1991.

[154] Brian W. Kernighan and David M. Ritchie. *The C Programming Language.* ISBN 0-13-110362-8. Prentice Hall Software Press, Englewood Cliifs, New Jersey, second edition, 1988.

[155] T. Kikukawa and S. Kawafuchi. Development of an automatic summary editing system for the audio-visual resources. *Transactions on Electronics and Information J75-A*, pages 204–212, 1992.

[156] J. Kittler, M. Ballette, W.J. Christmas, and Kieron Messer. *Fusion of Multiple Cue Detectors for Sports Video Annotation*, pages 596–606. Lecture Notes in Computer Science 2396. Springer Verlag, Berlin Heidelberg, 2002.

[157] K.N.Nagan. Image dsiplay techniques using the cosine transform. *IEE Transactions on Acoustics Speech and Signal Processing*, 32:173–177, February 1984.

[158] Janet L. Kolodner. *Case Based Reasoning.* ISBN 1-55860-237-2. Morgan Kaufmann, San Mateo, Ca., 1993.

[159] Anita Komlodi and Gary Marchionini. Visual video browsing using key frames. In *CHI 98 conference summary on human factors in computing Systems*, pages 337–338, Los Angeles, California, 1998.

[160] Irena Koprinska and Sergio Carrato. Temporal video segmentation: A survey. *Signal Processing: Image Communication*, 16:477–500, 2001.

[161] Irena Koprinska and Sergio Carrato. Hybrid rule based/neural approach for segmenting mpeg compressed video. *Multimedia Tools and Applications*, 18:187–212, September 2002.

[162] N. Ahmed K.R.Rao. *Orthogonal Transforms for Digital Signal Processing.* ISBN 0387065563. Springer Verlag, 1975.

[163] Albrecht Kunst. *A Dictionary of Kinetography Laban (Labanotation)*, volume 1 Text. Macdonald and Evans, Plymouth, UK, 1979.

[164] Albrecht Kunst. *A Dictionary of Kinetography Laban (Labanotation)*, volume 2 Examples. Macdonald and Evans, Plymouth, UK, 1979.

[165] G. P. Lakoff. Structural complexity in fairy tales. *The Study of Man*, 1:128–190, 1972.

[166] S. St. Laurent and L. Biggar. *Inside XML DTDs.* ISBN 0-07-134621-X. McGraw Hill, 1999.

[167] P.A. Laven and M.R. Meyer, editors. *EBU / SMPTE task Force for Harmonized Standards for the Exchange of Programme Material as Bitstreams*, volume Special Supplement, August 1998.

[168] John Leach. *Survival Psychology.* ISBN 033351855. Macmillan, 1994.

[169] Peter M. Lee. *Bayesian Statistics and introduction.* ISBN 0340814055. Arnold, 3rd edition, 2004.

[170] Theodore W. Leung. *Professional XML development with Apache Tools : Xerces, Xalan, FOP, Cocoon, Axis, Xindice.* ISBN: 0764543555. Wrox, 2003.

[171] Michael S. Lew, editor. *Principles of Visual Information Retreival.* ISBN 1852333812. Springer Verlag, Lon, Berlin Heidleberg, 2001.

[172] Ying Li and C.C. Jay Luo. *Video Content Analysis Using Multimodal Information For Movie Content Extraction Indexing and Representation.* ISBN 1402074905. Kluwer Academic Publishers, Norwell, Mass., 2003.

[173] Rainer Lienhart. Comparison of automatic shot boundary detection algorithms. *Image Video Processing*, VII, 1999.

[174] Rainer Lienhart and Andre Zaccarin. A system for reliable dissolve detection in videos. In *Proceedings of ICIP 2001*, Thessalonika, Grrece, October 2001.

[175] Stefan Fischer Rainer Lienhart and Wolfgang Effelsberg. Automatic recognition of film genres. In *Proceedings ACM Multimedia*, pages 295–304, San Francisco CA, Nov 1995.

[176] Kjetil Nørvåg Marit Limstrand and Lene Myklebus. TeXOR: Temporal XML Database on an Object-Relational Database System. In *Proceedings of Perspectives of System Informatics (PSI'03)*, pages 520–530, Novosibirsk, Russia, July 2003. Springer Verlag.

[177] A.D. Lindsay. *Berkeley A New Theory of Vision and other writings*, pages 13–86. ISBN 0 460 01483. Dent Everymans Library, London, 1910, 1972.

[178] Lispworks. *LispWorks Personal Edition.* Xanalys, Waltham MA, 2003.

[179] Robert Lougher, David Hutchison, Alan Parkes, and Andrew Scott. A hypermedia model for the organisation and description of video. *Proceedings of the World Conference on Education Multimedia and Hypermedia and World Conference on Educational Telecommunications*, June 1997.

[180] Fredrik Lundt and Mathew Ellis. *Python Imaging Library (PIL).* Pythonware, 1.1.3 edition, March 2002.

[181] Dick C.A. Bulterman Lynda Hardman and Guido van Rossum. The amsterdam hypermedia model. *Communications of the ACM*, 37(2), 1994.

[182] John Lyons. *Introuction to Theoretical Linguistics.* SBN 521 05617 9. Cambridge University Press, London, 1968.

[183] W. Mahadi, M. Ardebilian, and L.M. Chen. Automatic video scene segmentation based on spatial-temporal clues and rythm. *Networking and Information Systems Journal*, 2(5):1–25, 2000.

[184] M.K. Mandal, F. Idris, and S. Panchanathan. A critical evaluation of image and video indexing techniques in the compressed domain. *Image and Video Computing*, 17:513–529, 1999.

[185] J. M. Mandler and N. Johnson. Rememberance of things parsed: story structure and recall. *Cognitive Psychology*, 9:111–191, 1977.

[186] Jean M. Mandler. *Stories, scripts and scenes: aspects of schema theory.* ISBN 0-89859-446-4. Lawrence Erlbaum Associates, 365 Broadway Hillsdale, New Jersey 07642, 1984.

[187] Jean M Mandler and Nancy S Johnson. On throwing out the baby with the bathwater: A reply to black and wilensky's evaluation of story grammars. *Cognitive Science*, 4:305–312, 1980.

[188] B.S. Manjunath, Philippe Salembier, and Thomas Sikora, editors. *Introduciton to MPEG-7 Multimedia Content Description Interface.* ISBN 0 471 48678 7. Wiley, 2002.

[189] Nathan Good Marc Davis, Simon King and Risto Sarvas. From context to content: Leveraging context to infer media metadata. In *Proceedings of 12th Annual ACM International Conference on Multimedia*, pages 188–195, 2004.

[190] D. Marr. *Vision.* ISBN 0-716-1567-8. W. H. Freeman and Company, New York, 1982.

[191] R. McKnee. *Story; Substance, Structure, Style and the Principles of Screenwriting.* ISBN 0 413 71560 4. Reagan Books, Methuen, London, 1988, 1999.

[192] Thomas Tamburin Mel Byars, Alexander Von Vegesack. *50 Chairs: Innovations in Design and Materials.* ISBN 0823065057. Watson-Guptill Publications, 1996.

[193] J. Meng, Y. Yuan, and S-F. Chang. Scene change detection in a mpeg compressed video sequence. *Proceedings of IST/SPIE International Symposium on Electronic Imaging*, 2417:14–25, 1995.

[194] Christian Metz. *Film Laguage A Semiotics of the Cinema.* LCCN 73-90363. Oxford University Press, 1974.

[195] Christian Metz. *Movies and methods: an anthology*, volume 1 of *ISBN 0520031512*, chapter On the notation of the Cinematographic Language, pages 582–589. University of California Press, 1976.

[196] Marvin Minsky. *A Framework for representing knowledge*, volume Readings in Knowledge Representaiton, chapter 12, pages 245–262. Morgan Kaufmann, 1985.

[197] A.H. Munsell. *A Color Notation.* Geo. H. Ellis Co, Boston, 1905.

[198] Albert H. Munsell. *Munsell Book of Color.* Munsell Color, 1976.

[199] Edweard Muyerbridge. *The Human Figure in Motion.* ISBN 0-486-20204-6. Dover, New York, 1955.

[200] F. Nack. *Auteur: The Application of Video Semantics and Theme Representation for Automated Film Editing.* PhD thesis, Lancaster University, 1996.

[201] F. Nack and Alan P. Parkes. Towards the automated editing of theme orientated video sequences. *Applied Artificial Intelligence*, 11(4):331–366, 1997.

[202] F. Nack and A.P. Parkes. The application of video semantics and theme representation in automated video editing. *Multimedia Tools and Applications*, 4(ISSN 1380-7501):57–83, 1997.

[203] Frank Nack and Craig Lindley. Production and maintenance environments for interactive audio visual stories. In *In Proceedings ACM MM 2000 Workshops - Bridging the Gap: Bringing Together New Media Artists and Multimedia Technologists*, pages 21– 24, Los Angeles, CA., October 31, 2000. ACM Multimedia.

[204] Frank Nack and Wolfgang Putz. Saying what it means. *Multimedia Tools and Applications*, 22:261–300, 2004.

[205] A Nagasaki and Y Tanaka. Automatic video indexing and full-video search for object appearances. In *IFIP Proceedings of Visual Database Systems*, volume II, pages 113–127, Amsterdam, 1992. Elesevier Science Publishers.

[206] National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign. *HDF5 Reference Manual*, October 2003 2003.

[207] Isaac Newwton. *Opticks, or, a treatise on the reflections, refractions, inflections and colours of light.* SBN 486602052. Dover, 1730, 1931, 1952.

[208] Bill Nichols. *Movies and methods: an anthology*, volume 1 of *ISBN 0520031512*, chapter Style, Grammar, and the Movies, pages 607–628. University of California Press, Berkeley, 2 edition, 1976, 1985.

[209] Helga Noice and Tony Noice. The effects of segmentation on the recall of theatrical material. *Poetics*, 22:51–67, 1993.

[210] K.M. O'Craven and N. Kanwisher. Mental imagery of faces and places activates corresponding stimulus specific brain regions. *Journal of Cognitive Neuroscience*, 12(6):1013–1023, 2002.

[211] Susan Onega and Jose Angel Garcia Landa, editors. *Narratology: an introduction to.* ISBN 0582255422. Longman, 1996.

[212] Alan P. Parkes. Towards a script-based representation language for educational films. *Programmed Learning and Educational Technology*, 3(24):234–246, 1987.

[213] Alan P. Parkes. *An Artificial Intelligence Approach to the Conceptual Description of Videodisc Images.* PhD thesis, Lancaster University, November 1988.

[214] Alan P. Parkes. The prototype cloris system describing, retreiving and disccussing videodisc stills and sequences. *Information Processing and Management*, 25(2):171–186, 1989.

[215] Alan P. Parkes. Settings and the settings structure: The description and automated propogation of networks for perusing videodisc image states. *Proceedings of the Twelth*

Annual International ACMSIGIR on Research and Development in Information Retrieval, pages 229–238, June 1989.

[216] G. Pass and R. Zabih. Comparing images using joint histograms. *Multimedia Systems*, 7(3):234–240, 1999.

[217] N.V. Patel and N.V. Sethi. Video shot detection and chrecterisation for video databases. *Pattern Recognition*, 30:583–592, 1997.

[218] Frank Nack Menzo Windhouwer Lynda Hardman Eric Pauwels and Michèle Huijberts. The role of high-level and low-level features in style-based retrieval and generation of multimedia presentations. *The New Review of Hypermedia and Multimedia*, 7(11):39–65, July The role of high-level and low-level features in style-based retrieval and generation of multimedia presentations.

[219] A. Pentland, R.W. Picard, and S. Sclaroff. Photobook: Tools for content based manipulation of image databases. *Proceedings SPIE*, 2185:33–47, Feb 1994.

[220] F. Pereira. Mpeg-7 requirements document v.18. (ISO/IEC JTC1/SC29/WG11/N6881), January 2005.

[221] N. Pitts-Moulis and C. Kirk. *XML Black Book*. ISBN 1-57610-284-X. Coriolis Technology Press, 1999.

[222] Mette T. Posamentier and Herve Abdi. Processing faces and facial expressions. *Neuropsychology Review*, 13(3), 2003.

[223] Charles Poynton. *Digital Video and HDTV Algorithms and Interfaces*. ISBN 1558607927. Morgan Kaufmann Publishers, San Francisco, 2003.

[224] Roger S. Pressman. *Software Engineering A Practioner's Approach*, volume European Adaptation of *ISBN 0-07-052182-4*. McGraw Hill, fourth edition, 1997.

[225] V. Propp. *Morphology of the Folk Tale*. LCCCN 68-65567. University of Texas Press, Austin and London, 1968 [1928].

[226] Tim Pyron, Rod Gill, Laura Stewart, Melette Pearce, Winston Meeker, Toby Brown, Ira Brown, and Jo E. Shires. *Using Microsoft Project 2000 (Special Edition)*. ISBN: 0789722534. Que, Indiapolis Indiana, 2000.

[227] Lawrence R. Rabiner and Bernard Gold. *Theory and Application of Digital Signal Processing*. ISBN 0-13-914101-4. Prentice Hall Inc., Englewood Cliifs, New Jersey, 1975.

[228] K.R. Rao and R. Yip. *Discrete Cosine Transform: Algorithms, Adantages Applications*. ISBN 0-12-580203-X. Academic Press Inc., 1990.

[229] F.H. Raven. *Automatic Control Engineering*. ISBN 0-07-051341-4. McGraw Hill, 1995.

[230] Leni Reifenstall. Hitler's Olympia: the Nazi Olympics, 1936.

[231] Peter E. Hart Richard O. Duda and David G. Stork. *Pattern Classification.* ISBN 0-471-05669-3. Wiley Interscience Publication, 2000.

[232] Phillipe Rigaux, Michel Scholl, and Agnes Voisard. *Spatial Databases.* ISBN 1-55860-588-6. Morgan Kaufmann Publishers, 2002.

[233] Azriel Roenfeld. Survey image analysis and computer vision: 1999. *Computer Vision and Image Understanding*, 76:222–302, 2000.

[234] A.J. Morgan R.S. Englemore and H.P. Nii. *The Hearsay-II Speech Understanding System: Integrating Knowledge to resolve uncertainty*, chapter 2, pages 31–86. Morgan Kaufmann, 1988.

[235] James Rumbaugh, Ivar Jacobson, and Grady Booch. *The Unified Modeling Language Reference Manual.* ISBN 0-201-30998-X. Adison Wesley, 1999.

[236] D. E. Rumelheart. On evaluating story grammars. *Cognitive Science*, 4:313–316, 1980.

[237] F. Saussure. *Course in General Linguistics.* ISBN 0070165246. McGraw Hill, 1966.

[238] Roger Schank. Conceptual dependency: A theory of natural language understanding. *Cognitive Psychology*, 3(4):552–631, 1972.

[239] Roger C. Schank. *Computer Models of Thought and Language*, chapter Indentification of Conceptualizations Underlying Natural Language, pages 187–247. ISBN 0-7167-0834-5. W H Freeman and Company, San Francisco, 1973.

[240] Roger C. Schank. *Dynamic memory : a theory of reminding and learning in computers and people.* ISBN 0521248582. Cambridge University Press, Cambridge and New York, 1982.

[241] Roger C. Schank. *Tell Me a Story: A New Look at Real and Artifical Memory.* ISBN0-604-19049-4. Scibners, New York, 1990.

[242] Roger C. Schank. *Dynamic Memory Revisited.* ISBN 0521633982. Cambridge University Press, Cambridge, 1999.

[243] Roger C. Schank and Roger Ableson. *Scripts Plans Goals and Understanding.* ISBN 0-470-99033-3. Lawrence Erlbaum Associates, New Jersey, 1977.

[244] Roger C. Schank and Charles J. Rieger III. *Readings in Knowledge Representation*, chapter Inference and the Conputer Understanding of Natural Language, pages 121–139. ISBN 0-934613-01-X. Morgan Kaufmann Publishers, Los Altos, California, 1985.

[245] B. Schiele. *Object Recognition using Multidimensional Field Histograms English translation.* PhD thesis, Institute National Polytechnic de Grenoble, 1997.

[246] Timos Sellis, Nick Roussopoulos, and Christos Faloutsos. The R+-Tree: A DYnamic Index for Multi-Dimensional Objects. In *Proceedings of the 13th VLDB Conference*, Brighton, 1987.

[247] B. Shahraray. Scene change detection and content-based sampling of video sequences. *Digital Video Compression: Algortithms and Technologies*, 2419:2–13, 1995.

[248] Gaurav Sharma and H. Joel Trussell. Digital color imaging. *IEEE Transactions on Image Processing*, 6(7):901–932, July 1997.

[249] K. Shen and E. Delp and. A fast algorithm for video parsing of mpeg compresssed sequences. In *Proceedings of International Conference of International Conference on Image Processing*, Lusanne, 1996.

[250] Eeero Simoncelli and David J. Heeger. A Model of Neronal Response in Visual Area MT. *Vision Research*, 38(5):743–761, 1998.

[251] R. Barrett A. Ramsay A. Sloman. *Pop-11 A practical Language for Artificial Intelligence.* ISBN 0-85312-924-X. Ellis Horwood, 1985.

[252] Aguierre Smith and Glorianna Davenport. The stratification system. a design environment for random access video. In *ACM workshop on Networking and Operating System Support for Digital Audio and Video*, pages 250–261, 1992.

[253] Alvy Ray Smith. Color gamut transform pairs. *Computer Graphics*, 12(3):12–19, 1978.

[254] Brian C. Smith. *Prologue to 'Reflection and Semantics in a Procedural Language'*, chapter 3, pages 31–39. Morgan Kaufmann, 1985.

[255] John R. Smith and Shih-Fu Chang. Visualseek. In *Proceedings of the fourth ACM international conference on Multimedia*, pages 87–89. ACM, 1997.

[256] J.R. Smith. *Integrated Spatial and Feature Image Systems: Retrieval, Analysis and Compression.* PhD thesis, Columbia University, 1997.

[257] SMPTE. Smpte 377m material exchange format.

[258] SMPTE. Smpte 335m-2001 television - metadata dictionary structure, 2001.

[259] SMPTE. Smpte 336m-2001 television - data encoding protocol using key-length-value, 2001.

[260] SMPTE. *Proposed SMPTE Standard 392M: Material Exchange File Format (MXF)*, June 2003.

[261] Andrew Solway. *Video Annotation the role of the specialist text.* PhD thesis, Department of Computing, University of Surrey, 1999.

[262] Ian Sommerville. *Software Engineering.* ISBN 020139815. Addison Wesley, 6 edition, August 2000.

[263] Ian Sommerville and Peter Sawyer. *Requirements Engineering: A Good Practice Guide.* ISBN 0-471-97444-7. John Wiley and Sons Ltd, 1997.

[264] Milan Sonka, Vaclav Hlavac, and Roger Boyle. *Image Processing, Analysis and Machine Vision.* ISBN 041245570. Chapman and Hall, London, 1995.

[265] J.F. Sowa. *Principles of Semantic Networks*, chapter 5 Towards the expressive power of natural language, pages 157–190. Moragan Kaufmann, 1991.

[266] J.F. Sowa. *Knowledge Representation Logical Philosophical and Computational Foundations.* Brookes/Cole Thompson Learning, 2000.

[267] John F. Sowa. *Conceptual structures : information processing in mind and machine.* ISBN 0201144727. Addison Wesley, Reading, Mass, 1984.

[268] John F Sowa, editor. *Draft Conceptual Graph Standard*, April 2001.

[269] Steve Spicklemire, Kevin Friedly, Jerry Spicklemire, and Kim Brand. *Zope Web Application Development and Content Management.* ISBN 0-7357-1110-0. New Riders, Idianapolis, Indiana, 2002.

[270] Spiridon. How Distributed is Visual Category Information in Human Occipito-Temporal Cortex? An fNMR Study. *Neuron*, 35:1157–1165, September 2002.

[271] Oswald Stack and John Halliday. *Pasolini su Pasolini conversazioni con Jon Halliday.* ISBN 88-7746-622-7. Biblioteca della Fenice, Parma Italy, 1992.

[272] Robert Stam, Robert Bourgoyne, and Sandy Flitterman Lewis. *New Vocabularies in Film Semiotics.* ISBN 0-415-06594-1. Routledge, 11 New Fetter Lane, London, 1992.

[273] Mark Steifek. *Introduction to Knowledge Systems.* ISBN 1-55860-166-X. Morgan Kaufmann, 1995.

[274] Leon Sterling and Ehud Shapiro. *The Art of Prolog.* ISBN 0-262-19338-8. MIT Press, 2nd edition, 1994.

[275] Michael Stonebraker, Paul Brown, and Dorothy Moore. *Object-relational DBMSs : tracking the next great wave.* ISBN 1558604529. Morgan Kaufmann Publishers, San Francisco, California, 2nd edition, 1999.

[276] B. Stroustrop. *The C++ Programming Language.* ISBN 0-201-53992-6. ATT Bell Telephone Laboratories Incorporated, second edition, 1991, 1993.

[277] M. J. Swain. Interactive indexing into image databases. In *Proceedings of SPIE Conference on Storage and Reteival inImage and Video Databases*, pages 173–187. SPIE, 1993.

[278] MT Swain and DH Ballard. Color indexing. *International Journal of Computer Vision (IJCV)*, 7(1):11–32, 1991.

[279] D. Swanberg, C-F. Su, and R. Jain. Knowledge guided parsing in video databases. In *Proceedings of SPIE Conference Vol 1908*, pages 13–24, 1993.

[280] C. Taskiran and E. Delp. Video scene change detection using the generalised sequence trace. In *Proceedings of IEEE International Conference on Acoustics Speech and Signal Processing*, pages 2961–2964, Seattle, Wa., 1998.

[281] Jason C. Teague and David Teague. *Final Cut Pro and the Art of Filmmaking.* ISBN 0-7821-40270. Sybex, Alameda California, 2002.

[282] Stith Thompson. The types of the Folktale: Antti Aarne's Verzeichnis der Marchentypen, Translated and Enlarged. *FF Communications*, 74, 1928.

[283] Jaime G. Carbonell Tom M. Mitchell and Ryszard S. Michalski. *Machine Learning; A guide to current research.* ISBN0-89838-214-9. Kluwer Academic Publishers, Norwell, Mass., 1986.

[284] Raymond Turner. *Logics for AI.* ISBN 0-85312-713-1. Ellis Horwood, England, 1984.

[285] University Corporation for Atmospheric Research. *NetCDF User's Guide*, May 2005.

[286] Teun Andreas van Dijk. *Some Aspects of Text Grammars: A study of theoretical linguistics and poetics.* Mouton, The Hague, Netherlands, 1972.

[287] E. van Herwijen. *Practical SGML.* ISBN 0-7923-9434-8. Kluwer Academic Publishers, 1994.

[288] Johann Wolfgang von Goethe. *Theory of colours.* ISBN 262 57021 1. MIT Press, 1840, 1970, 1973.

[289] John von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behaviour.* Princeton University Press, 3rd edition, 1974.

[290] W3C. XML eXtensible Markup Language, February 1998.

[291] W3C. Html specification 4.01, 1999.

[292] Juergen Walther. *The AI Workbench BABYLON: An open and portable development environment for expert*
systems. ISBN 0-12-174235-0. Academic Press, London, 1992.

[293] James Z. Wang, Jia Li, and Go Wiederhold. Simplicity: Semantics-sensitive integrated matching for picture libraries. *IEE Transactions on Pattern Analysis and Machine Inteligence*, 23(9), 2001.

[294] Bonnie Lynn Webber and Nils J. Nilsson. *Readings in Artifical Intelligence.* ISBN 0-935382-03-8. Tioga Publishing Company, Poalo Alto California, 1981.

[295] P.H. Winston and B.K.P. Klaus. *LISP.* ISBN 0-201-08319-1. Addison Wesley, 3rd edition, 1989.

[296] Siegfreid Wyler. *Clour and Language: colour terms in English.* ISBN 3823342193. Tubigen, 1992.

[297] W. Xiong and J.C.M. Lee. Efficient scene change detection and camera motion annotation for video classification. *Computer Vision and Image Understanding*, 71(2):166–181, 1998.

[298] Wei Xiong, John Chung-Mong Lee, and Ding-Gang Shen. Net comparison: An adaptive and effective method for scene change detection. In *Proceedings of SPIE Conference on Storage and Reteival inImage and Video Databases III Vol 2420*, pages 318–328, San Jose, California, 1995.

[299] B. Yeo and B. Liu. Rapid scene change analysis on compressed video. *IEEE Transactions on Circuits and Systems Video Tedchnology*, 5(6):533–544, 1995.

[300] H. Yu, Bozdagi G, and S. Harrington. Feature-based hierachical video segmentation. In *Proceedings of International Conference on Image Processing (ICIP-97)*, pages 498–501, Santa Barbara, 1997.

[301] S Zeki, JD Watson, CJ Lueck, KJ Friston, C Kennard, and RS Frackowiak. A direct demonstration of functional specialization in human visual cortex. *The Journal of Neuroscience*, 11:641 – 649, March 1991.

[302] H.J. Zhang, Y.C. Low, and S.W. Smolier. Video parsing using compressed data. *MultimediaTools and Applications*, 1:89–111, 1995.

[303] HongJiang Zhang, A. Kankanhalli, and Stephen W. Smolier. Automatic partitioning of full-motion. *Multimedia Systems*, 11(1):10–28, 1993.

[304] HongJiang Zhang, Chien Yong Low, Yihong Gong, and Stephen W. Smolier. Video parsing using compressed data. *Image Video Processing*, SPIE 2182(II), 1994.

[305] R. Zhao and W.I. Grosky. Negotiating the semantic gap: From feature maps to semantic landscapes. *Pattern Recognition*, 35:51–58, March 2002.

Filmography

2001: A space Odyssey, Stanley Kubrick, UK, 1968

Crash, David Cronenberg, USA, 1996

Grand Prix, John Frankenheimer, 1966, USA, 1966

Galaxy Quest, Dean Parisot, USA, 1999

High Noon, Fred Zinnemann, USA, 1952

Jaws, Stephen Spielberg, USA, 1975

Rebecca, Alfred Hitchcock, USA, 1940

Romanoff and Juliet, Sir Peter Ustinov, USA, 1961

Rope, Alfred Hitchcock, UK, 1948

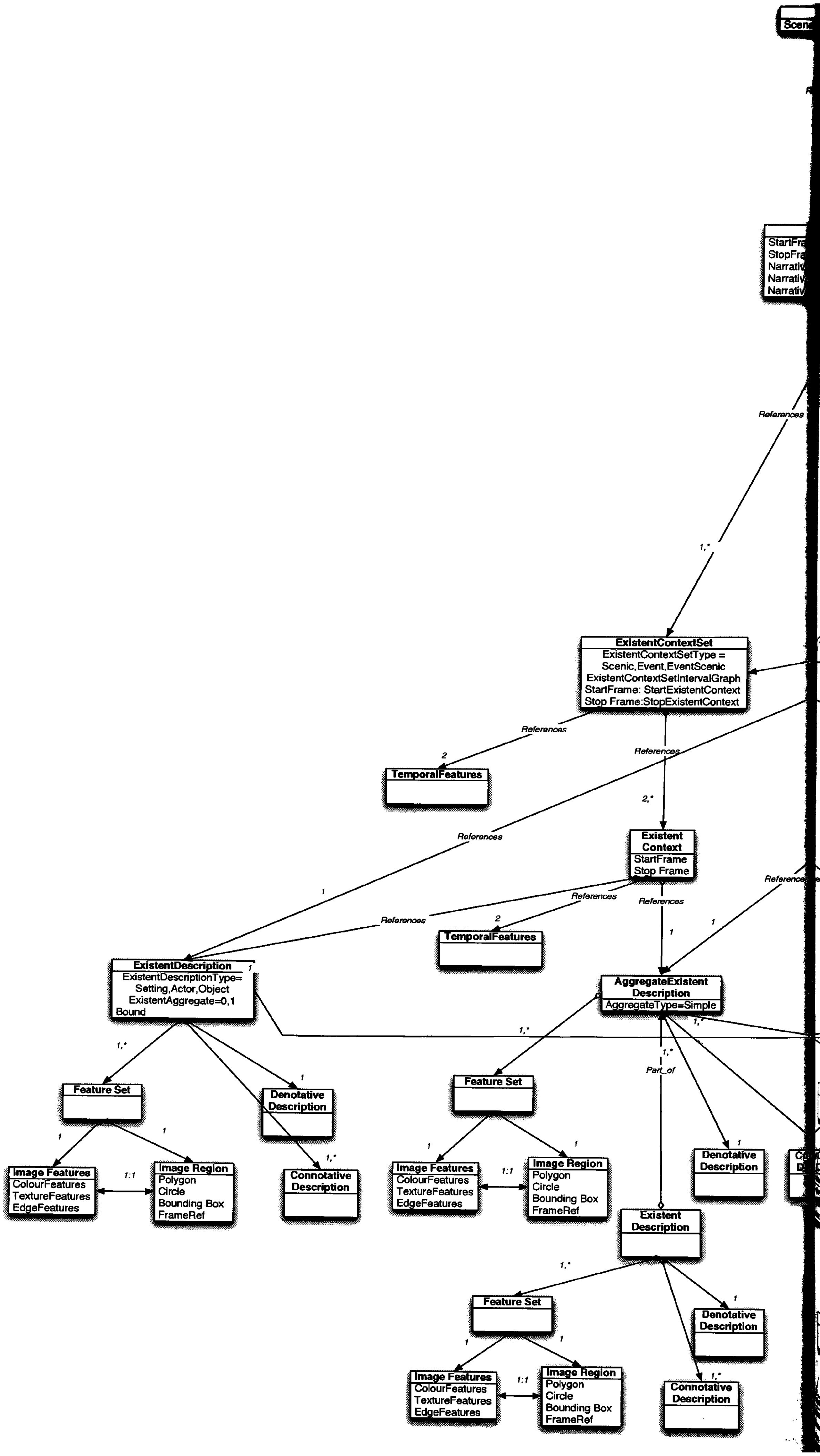
True Romance, Tony Scott, USA, 1993,

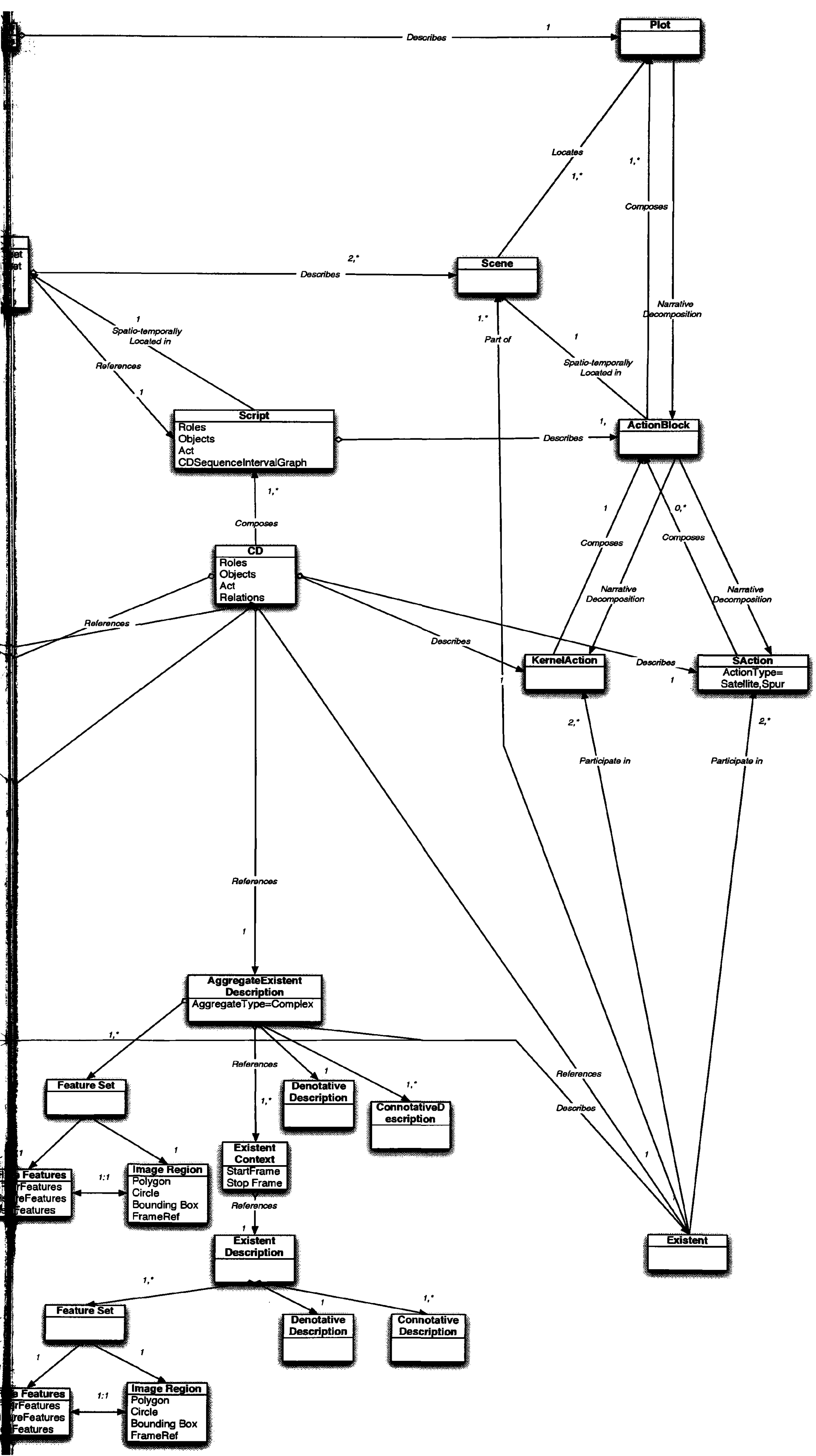
The Dot and The Line, Chuck Jones, USA,1965

The Thomas Crown Affair, Norman Jewison, USA, 1968

West Side Story,Jerome Robbins and Robert Wise, USA,1961

William Shakespeare's Romeo + Juliet, Baz Luhrmann, USA, 1996


ExistentContextSet
ExistentContextSetType = Scenic,Event,EventScenic
ExistentContextSetIntervalGraph
StartFrame: StartExistentContext
Stop Frame:StopExistentContext
References
1,*
References
2
TemporalFeatures
References
2,*
Existent Context
StartFrame
Stop Frame
References
1
References
References
2
TemporalFeatures
References
1
References
1
ExistentDescription
ExistentDescriptionType= Setting,Actor,Object
ExistentAggregate=0,1
Bound
1
1,*
AggregateExistent Description
AggregateType=Simple
1,*
1,*
Part_of
1,*
Feature Set
1
Denotative Description
1
1
Image Features
ColourFeatures
TextureFeatures
EdgeFeatures
1:1
Image Region
Polygon
Circle
Bounding Box
FrameRef
1,*
Connotative Description
Feature Set
1
1
Image Features
ColourFeatures
TextureFeatures
EdgeFeatures
1:1
Image Region
Polygon
Circle
Bounding Box
FrameRef
1
Denotative Description
Existent Description
1,*
Feature Set
1
Denotative Description
1
1
Image Features
ColourFeatures
TextureFeatures
EdgeFeatures
1:1
Image Region
Polygon
Circle
Bounding Box
FrameRef
1,*
Connotative Description


Plot
Describes
1
Locates
1,*
1,*
Composes
Scene
Describes
2,*
1.*
Part of
1
Spatio-temporally Located in
Narrative Decomposition
1
Spatio-temporally Located in
References
1
Script
Roles
Objects
Act
CDSequenceIntervalGraph
Describes
1,
ActionBlock
1,*
Composes
CD
Roles
Objects
Act
Relations
1
Composes
0,*
Composes
Narrative Decomposition
Narrative Decomposition
References
Describes
Describes
KernelAction
SAction
ActionType=Satellite,Spur
1
1
2,*
2,*
Participate in
Participate in
References
1
AggregateExistent Description
AggregateType=Complex
1,*
References
1
1,*
References
Describes
Feature Set
Denotative Description
ConnotativeDescription
1,*
1
1
Existent Context
StartFrame
Stop Frame
Features
rFeatures
reFeatures
Features
1:1
Image Region
Polygon
Circle
Bounding Box
FrameRef
References
1
Existent Description
1
Existent
1,*
Feature Set
1
Denotative Description
1,*
Connotative Description
1
1
e Features
rFeatures
reFeatures
Features
1:1
Image Region
Polygon
Circle
Bounding Box
FrameRef