



Escuela Politécnica Superior  
Departamento de Ingeniería Informática

# A SCALABLE APPROACH TO VIDEO SUMMARIZATION AND ADAPTATION

A thesis submitted in fulfilment of the requirements for the  
degree of Doctor of Philosophy

Luis Herranz Arribas  
Thesis advisor: José María Martínez Sánchez

October 2010



© Copyright by Luis Herranz Arribas 2010. All Rights Reserved

Work partly supported by the European Commission under project FP6-001765 (aceMedia) and the Ministerio de Ciencia y Tecnología of the Spanish Government under projects TIN2004-07860 (MEDUSA) and TEC2007-65400 (SemanticVideo).



Thesis committee

President:	<b>Narciso García Santos</b>	
	Universidad Politécnica de Madrid, Spain	_____
		(Narciso García Santos)
Secretary:	<b>Jesús Bescós Cano</b>	
	Universidad Autónoma de Madrid, Spain	_____
		(Jesús Bescós Cano)
Vocal:	<b>Janko Čalić</b>	
	University of Surrey, United Kingdom	_____
		(Janko Čalić)
Vocal:	<b>Fernando Jaureguizar</b>	
	Universidad Politécnica de Madrid, Spain	_____
		(Fernando Jaureguizar)
Vocal:	<b>Fernando Díaz de María</b>	
	Universidad Carlos III de Madrid, Spain	_____
		(Fernando Díaz de María)



# Abstract

Nowadays, the users of multimedia services are overwhelmed with the huge amount of video information available across different networks. Current sources of multimedia content are numerous: media networks, digital libraries, social networks, etc. This increasing amount of content and the data intensive nature of video makes the mangement and browsing of video collections, as well as their search and retrieval, increasingly difficult. Video abstractions (or summaries) make navigation easier, providing the user with a quick idea about the content. Another characteristic of current multimedia systems is that the same content can be accessed from a wide variety of terminals through different networks. Adaptation of multimedia content is a key element to provide the user with a suitable version of the content, according to the usage environment (mainly terminal and network).

In general, video summarization and adaptation are time and resource consuming tasks. In this thesis, efficient methods are proposed to generate the output bitstream with a very low delay and with low resource requirements. These methods are based on scalable approaches. A bitstream is scalable if, selecting certain packets, a basic version of the content can be obtained, while by including another set of packets an enhanced version can be obtained (e.g. higher resolution).

The thesis explores the use of the temporal scalability of H.264/AVC for summarization purposes, proposing a model to represent summaries (storyboards, fast forwards and video skims) and an efficient generation method, based on bitstream extraction. This approach is then extended to SVC, using other modes of scalability to adapt the summary to the requirements of different terminals and networks.

The idea of scalability is also integrated in the summary itself, which is represented, coded and generated in a scalable fashion. Thus, the system can generate summaries of different lengths without analyzing the content again. The analysis is performed using an incremental algorithm based on an iterative ranking. This method creates scalable storyboards and video skims in an efficient way. The idea of scalable summary is also studied in the context of comic-like summaries, which are inspired by the structure of comic strips.

Finally, several applications of the methods developed in the previous parts are proposed, such as customized summaries, storyboards that can adapt to the window size, multichannel TV summaries or composite summaries.

---

# Resumen

Actualmente, los usuarios de servicios multimedia se ven abrumados por la ingente cantidad de información presente en las diversas redes. Son numerosas las fuentes de contenido multimedia: medios de comunicación, bibliotecas digitales, redes sociales, etc. Esta creciente cantidad de contenido y la propia naturaleza del video hace difícil su gestión y la búsqueda y acceso a contenido específico. Las abstracciones de video o resúmenes facilitan la navegación permitiendo al usuario hacerse una cierta idea rápida del contenido. Otra característica de los sistemas actuales de acceso a contenido multimedia es la diversidad y heterogeneidad de las formas en que se puede acceder. Terminales de todo tipo se pueden utilizar a través de diferentes redes. La adaptación del contenido multimedia es un elemento clave para proporcionar al usuario contenido adecuado a su contexto de uso.

En general, la creación de resúmenes y la adaptación de contenido son tareas costosas y que requieren bastantes recursos para procesarlos. En esta tesis se proponen métodos eficientes que permitan generar el bitstream con el menor retardo posible y consumiendo pocos recursos. Estos métodos se basan en enfoques escalables. Un bitstream es escalable si seleccionando ciertos paquetes se puede obtener una versión básica del contenido, mientras que incluyendo otro conjunto de paquetes se puede obtener una versión mejorada (p.e. mayor resolución).

La tesis explora un uso de la escalabilidad temporal de H.264/AVC distinto al de adaptación temporal, proponiendo un modelo de representación de resúmenes (storyboards, fast forwards y video skims) y un método de generación eficiente basado en la extracción de paquetes. Este enfoque se extiende posteriormente a SVC, utilizando el resto de tipos de escalabilidad para adaptar el resumen a las necesidades de diferentes terminales y redes.

La idea de escalabilidad también se integra dentro del propio resumen, que se representa, codifica y genera de una forma escalable. Así el sistema puede generar resúmenes de diferentes longitudes sin necesidad de volver a analizar el contenido. El análisis se realiza mediante un método incremental basado en un ranking iterativo, capaz de obtener storyboards y video skims escalables de forma también eficiente. La idea de resumen escalable se estudia también en el contexto de resúmenes de tipo comic, inspirados en la estructura de las tiras cómicas.

Por último, se proponen una serie de aplicaciones de los métodos desarrollados en las partes anteriores, como resúmenes personalizados, resúmenes adaptables al tamaño de ventana, resúmenes de múltiples canales de TV o resúmenes compuestos.

---

TO MY GRANDPARENTS



# Acknowledgements

During these years, I had the chance to meet many people from the research and academic communities. I would like to acknowledge a few of them.

In first place, I would like to thank my advisor José María Martínez, for his guidance, patience and support, and for giving me the freedom to experiment with my own ideas. My gratitude is also to Jesús Bescós, who advised me during my second year, and from whom I also received support. Thanks them for giving me the opportunity to widen my research experience by attending conferences and visiting other research labs while I was a member of the VPU (former GTI).

Most memories from these years come back to the crowded lab B-408 and to the members of the VPU, with whom I have shared projects, travels, meetings, lunches, football team and many good moments. A few of them were there almost from the beginning (Víctor V., Víctor F., Fernando, Fabri), while some others joined later (Javi, Juan Carlos, Marcos, Álvaro G. (II), Álvaro B, Miguel Ángel) or were part of the group only for some time (Raúl, Jorge H., Álvaro G. (I), Luis, Virginia, and many others).

These good memories are also extended to the rest of the fellows of the B-408: Mariano, Ignacio, Daniel, José Miguel and, of course, Ana; the members of the Information Retrieval Group (former NETS): Pablo, Iván, David, Miriam and Alex; and to the frequent visitor Jorge “tío Jorge” Ruiz, who I have to thank for his invaluable advise and help in so many things.

During this time I had the fortune to visit several research labs, where I could learn from other research environments. I would like to thank Ebroul Izquierdo, for welcoming me at the MMV lab of the Queen Mary University of London, and the rest of the members of the lab. I am especially grateful to Nikola Sprljan and Marta Mrak for their help during my stay there. The seed of many ideas developed years later come from those days. I want to thank Marta Mrak, again, and Janko Calic, for making the visit to the I-lab of the University of Surrey possible, where I enjoyed many interesting and fruitful discussions. Finally, I would like to thank Miroslav Bober and Stavros Paschalakis for giving me the opportunity to work for Mitsubishi Electric R&D during the last stage of the thesis, and the rest of the members of the lab for the warm welcome.

Teaching at the university while working on a thesis has been a demanding but rewarding experience, where I had the privilege to work and share experiences with excellent researchers from other fields, such as Doroteo Torre, Jorge Ruiz and Daniel Tapias. Special thanks are given

---

to a long list of former students, from whom I learnt more than they did from me.

Many people will remember this thesis because of the numerous, and often long and tedious, subjective evaluation sessions. I am indebted with all those volunteers, because the thesis could have never been finished without their generous collaboration. I am especially grateful to Marcos, who was the first person to suffer many of them as test user, providing me with an invaluable feedback.

Last but foremost, I would like thank my parents and sister, for their unconditional support and help during this long journey.

# Contents

<b>Abstract</b>	<b>VII</b>
<b>Resumen</b>	<b>IX</b>
<b>Acknowledgements</b>	<b>XIII</b>
<b>List of Tables</b>	<b>XXI</b>
<b>List of Figures</b>	<b>XXVI</b>
<b>I Introduction and context</b>	<b>1</b>
<b>1. Introduction</b>	<b>3</b>
1.1. Motivation . . . . .	3
1.2. Objectives . . . . .	4
1.3. Structure of the thesis . . . . .	5
<b>2. Context and related work</b>	<b>9</b>
2.1. Scalability . . . . .	9
2.1.1. Time and space scalability in computational complexity . . . . .	9
2.1.2. Scalability in other fields . . . . .	10
2.1.3. Scalability in video coding . . . . .	11
2.2. Video coding . . . . .	12
2.2.1. MPEG-1, MPEG-2 and MPEG-4 . . . . .	12
2.2.2. H.264/MPEG-4 Advanced Video Coding . . . . .	13
2.2.3. Scalable extension of H.264/MPEG-4 AVC . . . . .	14
2.3. Video abstraction . . . . .	15
2.3.1. Trade-off between information and browsing time . . . . .	15
2.3.2. Modalities of video summaries . . . . .	16
2.3.3. Approaches to video summarization . . . . .	18
2.3.3.1. Evaluation of video summaries . . . . .	19
2.3.3.2. Other trends in video summarization . . . . .	21

2.4.	Video adaptation . . . . .	21
2.4.1.	Universal Multimedia Access and MPEG-21 . . . . .	21
2.4.2.	Content-blind adaptation . . . . .	23
2.4.3.	Content-based adaptation . . . . .	24
2.5.	Summary and conclusions . . . . .	24
<b>II</b>	<b>Summarization and adaptation using scalable video coding</b>	<b>25</b>
<b>3.</b>	<b>Generation of video summaries by bitstream extraction</b>	<b>27</b>
3.1.	Related work on bitstream customization . . . . .	27
3.2.	H.264/MPEG-4 AVC and hierarchical prediction structures . . . . .	28
3.3.	Summarization approach . . . . .	30
3.4.	Summarization model . . . . .	32
3.4.1.	Basic model for extraction . . . . .	32
3.4.2.	Extended model with hierarchical prediction structures . . . . .	34
3.4.3.	Other coding structures . . . . .	35
3.5.	Summary description . . . . .	36
3.5.1.	Conventional model for transcoding . . . . .	36
3.5.2.	Summarization constraint . . . . .	37
3.5.3.	Modalities of video summaries . . . . .	38
3.6.	Generation of the summary . . . . .	39
3.6.1.	Architecture based on transcoding . . . . .	39
3.6.2.	Architecture based on extraction . . . . .	41
3.6.2.1.	Extraction using IDR access units . . . . .	42
3.6.2.2.	Extraction using I access units . . . . .	42
3.6.2.3.	Presentation schemes . . . . .	45
3.7.	Experimental evaluation . . . . .	46
3.7.1.	Summarization algorithms . . . . .	46
3.7.1.1.	Image storyboard . . . . .	47
3.7.1.2.	Video skim . . . . .	48
3.7.1.3.	Fast forward . . . . .	48
3.7.2.	Subjective evaluation . . . . .	50
3.7.3.	Efficiency . . . . .	51
3.7.4.	Rate-distortion performance . . . . .	54
3.7.4.1.	Comparison with other architectures . . . . .	55
3.8.	Summary and conclusions . . . . .	58
<b>4.</b>	<b>Integrated summarization and adaptation</b>	<b>61</b>
4.1.	Motivation . . . . .	61
4.2.	Integrated summarization and adaptation framework in MPEG-4 SVC . . . . .	62
4.2.1.	Summarization units in MPEG-4 SVC . . . . .	62

4.2.2.	Extraction process in MPEG-4 SVC . . . . .	63
4.2.3.	Including summarization in the framework . . . . .	65
4.3.	Experimental evaluation . . . . .	66
4.3.1.	Test scenario . . . . .	66
4.3.2.	Efficiency . . . . .	68
4.3.3.	Rate-distortion performance . . . . .	70
4.4.	Summary and conclusions . . . . .	71
<b>III</b>	<b>Scalable summaries</b>	<b>73</b>
<b>5.</b>	<b>Scalable storyboards and video skims</b>	<b>75</b>
5.1.	Motivation . . . . .	75
5.2.	Related work . . . . .	76
5.3.	Properties of video summaries . . . . .	77
5.3.1.	Semantic coverage . . . . .	77
5.3.2.	Visual pleasantness . . . . .	77
5.3.3.	Properties in the context of storyboards and video skims . . . . .	77
5.3.4.	Influence of the length of the summary . . . . .	78
5.4.	Scalable summarization framework . . . . .	78
5.4.1.	Efficient generation of the summary . . . . .	79
5.4.2.	Scalable summaries . . . . .	79
5.4.3.	Ranked lists . . . . .	81
5.4.4.	Architecture . . . . .	81
5.4.5.	Scalable summarization by incremental growing . . . . .	81
5.5.	Keyframe and feature extraction . . . . .	82
5.5.1.	Structure and notation . . . . .	83
5.5.2.	Shot change detection . . . . .	83
5.5.3.	Feature extraction . . . . .	84
5.6.	Clustering . . . . .	85
5.6.1.	Comparison of $K$ -means and hierarchical clustering . . . . .	85
5.6.1.1.	$K$ -means . . . . .	85
5.6.1.2.	Hierarchical agglomerative clustering . . . . .	86
5.6.2.	Clustering in the proposed summarization algorithm . . . . .	86
5.7.	Iterative ranking . . . . .	88
5.7.1.	Cluster level . . . . .	88
5.7.2.	Shot level . . . . .	91
5.7.3.	Residual ranking . . . . .	93
5.8.	Generation . . . . .	93
5.8.1.	Generation via bitstream extraction . . . . .	93
5.8.2.	Generation via set of images . . . . .	94
5.9.	Efficiency and delay analysis . . . . .	95

5.9.1. Efficient and low delay summarization . . . . .	95
5.9.2. Delay analysis of the algorithm . . . . .	96
5.10. Experiments . . . . .	96
5.10.1. Shot change detection . . . . .	96
5.10.2. Subjective evaluations . . . . .	97
5.10.3. Efficiency . . . . .	100
5.11. Summary and conclusions . . . . .	103
<b>6. Scalable comic-like summaries</b>	<b>105</b>
6.1. Related work . . . . .	105
6.1.1. Advanced pictorial summaries. Beyond storyboards . . . . .	105
6.1.2. Comic-like summaries . . . . .	106
6.2. Motivation . . . . .	109
6.2.1. Proposed framework . . . . .	110
6.2.2. Keyframe extraction . . . . .	111
6.3. Estimation of frame sizes . . . . .	111
6.4. Single scale layout . . . . .	113
6.4.1. Panel template generation . . . . .	113
6.4.2. Optimal solution using full search . . . . .	113
6.4.3. Optimized full search . . . . .	115
6.4.4. Suboptimal solutions . . . . .	117
6.5. Multiscale layout . . . . .	118
6.5.1. Independent layouts . . . . .	118
6.5.2. Disturbance . . . . .	119
6.6. Heuristic approach to multiscale layout . . . . .	119
6.6.1. Anchor keyframes . . . . .	120
6.6.2. Temporally constrained sampling . . . . .	120
6.6.3. Anchor based layout . . . . .	121
6.7. User interface . . . . .	123
6.8. Experimental results . . . . .	123
6.8.1. Experimental setup . . . . .	123
6.8.2. Objective Evaluation . . . . .	125
6.8.3. User evaluation . . . . .	125
6.8.3.1. Scenario 1: Interactive summaries . . . . .	125
6.8.3.2. Scenario 2: Progressive summaries . . . . .	127
6.8.3.3. General evaluation . . . . .	127
6.9. Summary and conclusions . . . . .	128

<b>IV</b>	<b>Applications</b>	<b>131</b>
<b>7.</b>	<b>Applications of integrated summarization and adaptation</b>	<b>133</b>
7.1.	Customized summaries . . . . .	133
7.1.1.	Audio extraction and multiplexing . . . . .	134
7.1.2.	Web based demo . . . . .	136
7.2.	Local browsing . . . . .	137
7.3.	Comparison of architectures for summarization and adaptation . . . . .	138
7.3.1.	Application scenarios . . . . .	138
7.3.1.1.	Multiple summaries and Universal Multimedia Access . . . . .	138
7.3.1.2.	Web-based video library . . . . .	139
7.3.2.	Summarization-adaptation architectures . . . . .	140
7.3.2.1.	Variations . . . . .	140
7.3.2.2.	Transcoding . . . . .	141
7.3.2.3.	Extraction . . . . .	141
7.3.2.4.	Hybrid architectures . . . . .	142
7.3.3.	Comparison of architectures . . . . .	142
7.3.3.1.	Efficiency . . . . .	143
7.3.3.2.	Rate-distortion performance . . . . .	143
7.3.3.3.	Storage requirements . . . . .	144
7.3.3.4.	Coding . . . . .	144
7.3.3.5.	Flexibility . . . . .	144
7.4.	Summary and conclusions . . . . .	145
<b>8.</b>	<b>Adaptation of scalable summaries</b>	<b>147</b>
8.1.	Resizable pictorial summaries . . . . .	147
8.1.1.	Introduction . . . . .	147
8.1.2.	Resizable storyboards . . . . .	148
8.2.	Summarization of live TV streams . . . . .	148
8.2.1.	Summarization in a multichannel broadcast scenario . . . . .	149
8.2.2.	Online architecture . . . . .	149
8.2.3.	Delay analysis . . . . .	150
8.2.4.	Experimental results . . . . .	151
8.2.5.	User interface . . . . .	152
8.3.	Summary and conclusions . . . . .	152
<b>9.</b>	<b>Combined scalabilities and composite summaries</b>	<b>155</b>
9.1.	Length scalable SVC bitstreams . . . . .	155
9.1.1.	4-D scalable bitstreams . . . . .	155
9.1.2.	Coding of ranked lists . . . . .	156
9.2.	Composite summaries . . . . .	158
9.2.1.	Introduction . . . . .	158

9.2.2. Composite summaries of news video . . . . .	159
9.2.2.1. Server side . . . . .	159
9.2.2.2. Client side . . . . .	160
9.2.3. Summarization approaches . . . . .	160
9.2.3.1. Fast forwards . . . . .	161
9.2.3.2. Scalable video skims . . . . .	161
9.2.4. Experimental results . . . . .	162
9.3. Summary and conclusions . . . . .	164
<b>V Conclusions</b>	<b>165</b>
<b>10. Conclusions and future work</b>	<b>167</b>
10.1. Summary and conclusions . . . . .	167
10.2. Main results and contributions . . . . .	169
10.3. Scalability in the different proposed techniques . . . . .	170
10.4. Future work . . . . .	170
10.5. Published work . . . . .	171
<b>A. Conclusiones y trabajo futuro</b>	<b>175</b>
A.1. Resumen y conclusiones . . . . .	175
A.2. Resultados y contribuciones . . . . .	178
A.3. Trabajo futuro . . . . .	178
<b>Glossary</b>	<b>181</b>
<b>Bibliography</b>	<b>183</b>

# List of Tables

3.1. Detail of the DPB management with I access units: (a) input bitstream, (b) summary bitstream. . . . .	45
3.2. Details of the software implementations used in the experiments. . . . .	52
4.1. Settings of the layers for SVC encoding . . . . .	67
4.2. Methods and cases used in the experiments. . . . .	68
5.1. Comparison of K-means and hierarchical clustering: (a) usual scenario, (b) after data reduction. . . . .	87
5.2. Results of shot change detection experiment. FP: false positives, M: missed, R: recall, P: precision. . . . .	97
5.3. Processing time (in seconds) for the test sequences. . . . .	101
6.1. Panels templates for $h = 4$ . . . . .	114
6.2. Characteristics of the data set and summaries. . . . .	123
6.3. Subjective results for the interactive scenario. . . . .	127
6.4. Preference of the algorithms in the progressive scenario: <i>basic</i> (B), <i>anchor</i> (A) and <i>anchor</i> with highlighting (A+H). . . . .	127
7.1. Comparison of summarization-adaptation architectures. . . . .	142
8.1. Processing times and delays for the offline and online architectures (in seconds). Note: the precision to measure times was 10ms. . . . .	151
9.1. Coding results for ranked lists. S16: size of the list in KB using fixed-length integers (16 bits), EN: entropy, LH: mean length of the Huffman code, SH: size of the list in KB using Huffman. . . . .	158
9.2. Results sorted by summarization rate. IN: Informative; RT: Rhythm; OV: Overall satisfaction; GT: Generation time . . . . .	162
10.1. Use of the different scalabilities along the thesis. A: used for adaptation, S: used for summarization. . . . .	170



# List of Figures

1.1. Relationships among parts and chapters. . . . .	7
2.1. Example of scalable video stream. . . . .	11
2.2. Search results for the word <i>shuttle</i> in the Open Video interface with different layouts: (a) keyframe and description, (b) keyframe and title. ( <i>Source</i> : Reproduced with permission of the Open Video project) . . . . .	16
2.3. Example of summary (storyboard) in a digital library. ( <i>Source</i> : Reproduced with permission of the Open Video project) . . . . .	17
2.4. Example of fast forward summary and adapted versions of the content. ( <i>Source</i> : Reproduced with permission of the Open Video project) . . . . .	18
2.5. Modalities of video summaries. . . . .	18
3.1. Prediction structures: (a) MPEG-2 structure, (b) hierarchical structure in H.264/AVC using P frames (structural delay 0), (c) hierarchical structure in H.264/AVC using B frames (structural delay 8). . . . .	29
3.2. Methods for the generation of summaries: (a) conventional approach, (b) proposed approach for H.264/AVC bitstreams with online summarization analysis, (c) proposed approach for H.264/AVC driven by metadata . . . . .	30
3.3. Summarization units and embedded summaries: (a) closed GOP, (b) open GOP (overlapped SUs). . . . .	33
3.4. Examples of summarization units: (a) low delay structure with hierarchical P frames, (b) overlapped SUs using hierarchical B frames (display order). . . . .	34
3.5. Summarization units using hierarchical B frames (coding order). . . . .	35
3.6. Other coding structures: (a) summarization unit using long term prediction, (b) streaming oriented coding structure, (c) unsuitable coding structure. . . . .	36
3.7. Bitstream adaptation guided by summarization constraint. . . . .	38
3.8. Summarization curves (left) and frame selection (right) for different types of video summarization. . . . .	39
3.9. Summarization architecture using transcoding (decoder-encoder cascade). . . . .	40
3.10. Summarization architecture using extraction. . . . .	41
3.11. Low level extraction process with IDR access units (no header updating). . . . .	42

3.12. Low level extraction process with I access units (header updating). . . . .	43
3.13. Example of modification of headers in the extraction process. Left: original bitstream, right: summary bitstream . . . . .	44
3.14. Constant and variable frame rate presentation schemes. . . . .	46
3.15. Summarization constraint in the case of a 7 keyframes storyboard with a SU length of 8 frames. . . . .	47
3.16. Examples of storyboards with 4, 7 and 16 keyframes. . . . .	47
3.17. Details of the results for video skim: (a) face detection, (b) summarization constraint (16 frames per GOP) and (c) summarization constraint (2 frames per GOP). . . . .	49
3.18. Details of the results for fast forward: (a) summarization constraint (32 frames per GOP) and (b) summarization constraint (4 frames per GOP). . . . .	49
3.19. Subjective evaluation results: a) visual distortion, b) semantic distortion. . . . .	50
3.20. Comparison of the coded sequences. . . . .	52
3.21. Processing time: (a) storyboard, (b) video skim, (c) fast forward and frame dropping. . . . .	53
3.22. Comparison of average PSNR for extraction and transcoding with different values of QP and sequences: (a) <i>stefan</i> CIF, (b) <i>foreman</i> CIF, and (c) <i>foreman</i> QCIF. . . . .	55
3.23. Comparison of average PSNR for extraction and transcoding with different GOP lengths and sequences: (a) <i>stefan</i> CIF, (b) <i>foreman</i> CIF, and (c) <i>foreman</i> QCIF. . . . .	56
3.24. Comparison of average PSNR per frame of sequence <i>stefan</i> CIF. . . . .	56
3.25. Comparison of run time of extraction and transcoding: (a) <i>stefan</i> CIF (different GOP lengths), (b) <i>foreman</i> CIF (different QP), and (c) <i>foreman</i> QCIF (different QP). . . . .	57
4.1. Adaptation in the SVC framework. . . . .	62
4.2. Coding structures and summarization units in SVC. . . . .	63
4.3. Prioritization of NAL units in the JSVM extractor (adapted from [Amonou et al., 2007]). . . . .	64
4.4. Integrated summarization and adaptation of SVC. . . . .	65
4.5. Bitstream adaptation guided by summarization and environment constraint. . . . .	66
4.6. Dependency of the processing speed with the GOP length: a) 4CIF 30 Hz and, b) CIF 15 Hz. . . . .	69
4.7. Processing speed for different modalities. Note that half of the vertical scale is linear and the rest is logarithmic. . . . .	70
4.8. Comparison of average PSNR: a) 4CIF 30 Hz and, b) CIF 15 Hz. . . . .	71
5.1. Embedded (a) and scalable (b) summaries. . . . .	80
5.2. Proposed architecture . . . . .	82
5.3. Analysis structures. . . . .	83
5.4. Example of clusters after ranking ( $\alpha_{cluster} = 1$ , $W_{max} = 3$ ). . . . .	90

5.5. Cluster scores for each iteration: (a) $\alpha_{cluster} = 1$ (duration), (b) $\alpha_{cluster} = 0.0001$ (distance). Scores are normalized for each row (iteration) and sorted by order of selection. . . . .	90
5.6. Example of scales of a scalable storyboard of <i>news11</i> ( $\alpha_c = 0.5$ ). . . . .	91
5.7. Distance score using the Hausdorff distance . . . . .	93
5.8. Length of the summaries depending on the index of the ranked list: (a) whole ranking, (b) cluster level ranking. . . . .	94
5.9. Absolute evaluations (storyboards): (a) baseline, (b) scalable. . . . .	98
5.10. Absolute evaluations (skims): (a) baseline, (b) scalable. . . . .	99
5.11. Relative evaluations: (a) storyboards, (b) skims. . . . .	100
5.12. Number of keyframes for different values of $W_{max}$ : (a) <i>MRS042538</i> and (b) <i>dn2002-208</i> . . . . .	102
5.13. Clustering time for different values of $W_{max}$ : (a) <i>MRS042538</i> and (b) <i>dn2002-208</i> . . . . .	102
5.14. Clustering time for different numbers of clusters: (a) <i>MRS042538</i> and (b) <i>dn2002-208</i> . . . . .	103
5.15. Generation time for different lengths of the summary (in GOPs): (a) <i>MRS042538</i> and (b) <i>dn2002-208</i> . . . . .	104
6.1. Advanced pictorial summaries: (a) video poster (from [Yeung and Yeo, 1997]), (b) stained-glass summary (from [Chiu et al., 2004]), (c) video snapshot (from [Ma and Zhang, 2005]), (d) video-tree (from [Jansen et al., 2008]), (e) video collage (from [Mei et al., 2009]), and f) Free-eye interface (from [Ren et al., 2010]) . . . .	107
6.2. Comic-like video summary (from [Calic et al., 2007]) . . . . .	108
6.3. Trade-off between information and compactness: (a) compact summary (storyboard), (b) detailed summary with information about temporal evolution (comic-like). . . . .	109
6.4. Architecture of the proposed framework. . . . .	111
6.5. Comparison of layout methods: (a) processing time, (b) difference to the optimal layout error, and (c) number of panel evaluations. . . . .	116
6.6. Example of transition between two consecutive scales. . . . .	119
6.7. Illustration of the temporally constrained sampling algorithm. . . . .	121
6.8. Examples of comic-like summaries: (a) scale 1 ( <i>basic</i> and <i>anchor</i> ), (b) scale 8 ( <i>basic</i> ), (c) scale 8 ( <i>anchor</i> ). New panels are highlighted. . . . .	122
6.9. Span of the layout change in images (a, c, e) and panels (b, d, f). . . . .	124
6.10. Comparison of basic and anchor algorithms: (a, c, e) accumulated cost, (b, d, f) inserted and removed panels. . . . .	126
6.11. General assessments. . . . .	128
7.1. Customized summaries using extraction. . . . .	134
7.2. Audio extraction and multiplexing. . . . .	136
7.3. Processing speed of audio extraction and transcoding. . . . .	136

7.4. Demo web interface for customized summaries. . . . .	137
7.5. Architecture of a player capable of presenting a preview of the content based on summaries. . . . .	138
7.6. Access to video content from heterogeneous terminals and networks. . . . .	139
7.7. Example of web-based video library using different types summaries. . . . .	140
7.8. Architectures for summarization and adaptation: (a) variations based, (b) transcoding based, and (c) extraction based. . . . .	141
8.1. Example of window resizing using a scalable storyboard: (a) 44 images (full storyboard), (b) 18 images, (c) 32 images and (d) 4 images. . . . .	148
8.2. Architecture of the multichannel summarization system. . . . .	150
8.3. User interface with resizable windows for multichannel summarization: (a) equally sized canvases (zoom factor 2), (b) equally sized canvases (zoom factor 4), (c) resized canvases (zoom factor 2) and (d) resized canvases (different zoom factors). . . . .	153
9.1. 4-D scalable bitstreams and the adaptation process. . . . .	156
9.2. Lists and probability distributions after different transformations for <i>news12</i> : (a) and (b) original ranked list, (c) and (d) differential, (e) and (f) rank transformation, (g) and (h) rank transformation plus differential. . . . .	157
9.3. Overview of the composite summarization process. . . . .	160
9.4. Different composition layouts in the web interface. . . . .	161
9.5. Subjective evaluation results: (a) information coverage, (b) rhythm, and (c) overall satisfaction. . . . .	163
9.6. Preference between video skims and fast forwards in composite summaries. . . . .	163

## Part I

### Introduction and context



# Chapter 1

## Introduction

### 1.1. Motivation

During the last years, the amount of video content in multimedia systems has increased dramatically. In addition to a large number of commercial sources, each user has become a potential contributor, sharing content through different communication networks. Nowadays, we can find video content in digital libraries (e.g. Internet Archive<sup>1</sup>, Open Video<sup>2</sup>), personal collections, social networks (e.g. YouTube<sup>3</sup>), web sites of media networks (e.g. BBC<sup>4</sup>, CNN<sup>5</sup>, ABC<sup>6</sup>, TVE<sup>7</sup>), video on-demand services, optical storage discs (e.g. DVD, Bluray), digital television broadcasting, and an endless list of sources. Thus, users are overwhelmed with an enormous and increasing amount of video information, which often makes very difficult its management and the search and retrieval of specific content. In addition, video is, in nature, a time consuming media, as it requires some time to be visualized. For those reasons, video abstractions are essential for efficient access and navigation[Pfeiffer et al., 1996; Yeung and Yeo, 1997]. A video abstract is a compact representation that can provide the user with a coarse idea of what happens in the video sequence. In this thesis we focus on visual abstracts (i.e. non textual or audio abstracts) using the term summary for them.

Last years have also witnessed the emergence of new devices capable of playing video content, but also the convergence of services and networks. Now, the same piece of content can be accessed from a wide range of heterogeneous terminals (e.g. personal computers, netbooks, PDAs, mobile phones, interactive TVs) through a variety of networks (e.g. different types of broadband, wireless and mobile networks, TV broadcasting). The content is adapted to the specific requirements of the usage environment (e.g. terminal, network), performing adaptation operations such as spatial downsampling or bitrate reduction in order to accommodate the bitstream to the available

---

<sup>1</sup><http://www.archive.org>

<sup>2</sup><http://www.open-video.org>

<sup>3</sup><http://www.youtube.com>

<sup>4</sup><http://www.bbc.co.uk>

<sup>5</sup><http://www.cnn.com>

<sup>6</sup><http://www.abc.go.com>

<sup>7</sup><http://www.rtve.es>

screen size or network bandwidth[Vetro, 2004; Chang and Vetro, 2005]. This adaptation is often addressed using pre-stored versions of the same content with different encoding configurations.

Owing to the crucial role of both adaptation and summarization in pervasive media environments, there is an increasing interest in developing efficient and effective methods. One of the main objectives of video adaptation is the generation of the adaptation bitstream with low delay and computational cost (e.g. efficient transcoding[Xin et al., 2005]). An interesting approach is scalable video coding, especially with the publication of H.264/SVC, the last standard for scalable video coding[Schwarz et al., 2007]. Adaptation of scalable bitstreams is a simple and extremely efficient process.

Research on video summarization has primarily focused on the analysis of the content to remove semantic redundancies in order to obtain compact summaries. However, we believe that the generation of the bitstream must be also taken into consideration as part of the summarization process, as the summary is often delivered to the user in a compressed format. A long latency due to the generation of the bitstream is not desirable, as the main goal of video summaries is to make search and navigation livelier. Besides, low delay is also desirable in applications requiring interactivity, such as customized summaries. Thus, some techniques used traditionally for video adaptation, such as transcoding or scalable coding, could be also exploited for video summarization. Moreover, the generation of the bitstream of a summary and its adaptation to the usage environment are addressed usually as two independent operations. A joint approach to both problems may be worth to explore, as the objective of both is the efficient generation of the output bitstream.

While scalable approaches have been explored in video coding for almost twenty years[Ohm, 1994, 2005; Schwarz et al., 2007; Anastassiou, 1994], scalability in the context of video summarization has barely been considered. We have found very few works dealing with summaries with multiple levels of detail[Zhu et al., 2004, 2005b; Benini et al., 2006]. Most of them create hierarchical summaries, which can provide a certain scalability (few scales). Although useful for hierarchical browsing, the utility of hierarchical summaries for adaptation is limited, where a finer granularity and more scales may be desirable. Besides, the term scalable summary has been rarely used[Zhu et al., 2004]. In current multimedia systems, where a personalized presentation for each user is desirable, we believe that scalable summaries may be useful to cope with the diversity of preferences and characteristics of each user's usage context. They can also provide different levels of detail that can be easily adjusted for interactive navigation. However, a proper analysis of the requirements, advantages, limitations and potential applications is required.

## 1.2. Objectives

The main goal of this thesis is to **explore the use of scalability in the context of video summarization**. This goal can be further specified in the following general objectives, which roughly correspond to the central parts of the thesis:

- To explore the use of the adaptation techniques used in scalable video coding in the context

of video summarization, and how they can be used jointly for both summarization and adaptation. The proposed techniques and results are described mainly in Part II.

- To explore the idea of scalability as an intrinsic property of the summary. The concept of scalable summary must be developed along with the analysis of the implications of having multiple lengths in a summary. Besides, for practical applications an adequate framework is required. The methods proposed are covered mainly in Part III.
- To explore potential applications of the techniques developed previously, which are covered in Part IV.

Apart from this central goal, the techniques described in this thesis have been also designed with some additional objectives in mind:

- Both analysis and adaptation methods should be efficient. Efficient generation of adapted summaries is essential to fully benefit from scalable summaries. Besides, other applications can also benefit from efficient processing, such as low delay and interactive summarization.
- The proposed models and methods should be flexible and generic enough so they can be applied to different modalities of summaries (e.g. storyboards, video skims).

### 1.3. Structure of the thesis

The thesis has been organized in three main and two complementary parts. The three main parts contain the contributions in terms of methods, frameworks and applications.

The first part introduces and overviews the related research areas. The second and third part contain the main contributions of the thesis regarding new techniques and frameworks. The following part describes several applications that take advantage of these techniques for improved performance and enhanced functionalities. Finally, the last part draws the conclusions. These parts are further structured in chapters:

- Part I: Introduction and context
  - *Chapter 1: Introduction*
  - *Chapter 2: Context and related work.* This chapter provides the research context related with the thesis, including a brief overview of related works and research areas.
- Part II: Summarization and adaptation using scalable video coding
  - *Chapter 3: Generation of video summaries by bitstream extraction.* This chapter describes a model and an efficient framework to generate video summaries using bitstream extraction. The framework is also evaluated in terms of quality and efficiency.
  - *Chapter 4: Integrated summarization and adaptation.* This chapter extends the previous framework in order to include adaptation to the usage context using scalable video coding.

- Part III: Scalable summaries

- *Chapter 5: Scalable storyboards and video skims.* This chapter presents the concept of scalable summary and its advantages and requirements. In addition, a framework and an efficient analysis algorithm to generate scalable storyboards and video skims are described.
- *Chapter 6: Scalable comic-like summaries.* The idea of scalable summary is extended in this chapter to comic-like summaries, along with methods to generate them.

- Part IV: Applications

- *Chapter 7: Applications of integrated summarization and adaptation.* This chapter describes several scenarios and applications that can benefit from the frameworks described in the second part of the thesis, including personalization and browsing.
- *Chapter 8: Adaptation of scalable summaries.* This chapter describes the application of scalable summaries to resizable graphic interfaces and multichannel TV summarization.
- *Chapter 9: Combined scalabilities and composite summaries.* Some applications combining scalable summaries and scalable video adaptation are described in this chapter.

- Part V: Conclusions

- *Chapter 10: Conclusions and future work.* This chapter concludes the document summarizing the main results and contributions of the thesis.

The relationships among chapters and parts of the thesis are depicted in Figure 1.1.

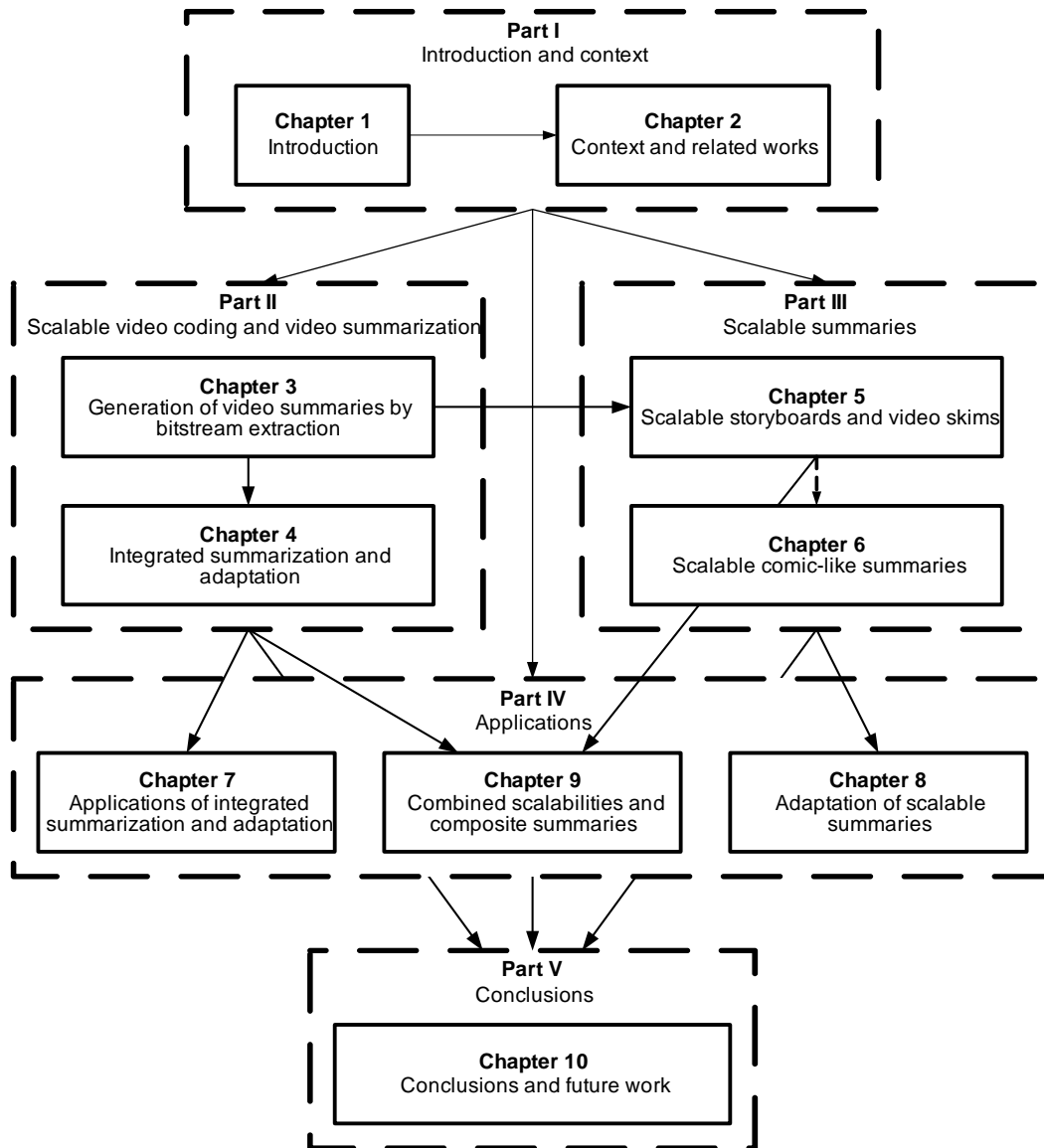


Figure 1.1: Relationships among parts and chapters.



## Chapter 2

# Context and related work

This chapter introduces the main research fields which have motivated the work described in the thesis. The chapter provides a survey of the main concepts and technologies used. In addition, some chapters include a section discussing works more closely related to that specific chapter.

### 2.1. Scalability

Unfortunately, most definitions of the term *scalability* are not generic enough and often subject to interpretations. This term has been used in many research fields with different meanings, but with the underlying idea that something that is scalable can adapt itself adequately to different working conditions. [Hill, 1990] attempted to give a rigorous definition, in the context of multiprocessor systems, but after failing in that purpose concludes his paper discouraging from the use of that term. However, the research community has continued using the term. According to [Bondi, 2000], “scalability is a desirable attribute of a network, system, or process. The concept connotes the ability of a system to accommodate an increasing number of elements or objects, to process growing volumes of work gracefully, and/or to be susceptible to enlargement”. Although not rigorous, this general definition is useful to get an idea of what is expected from something that it is claimed to be scalable.

In this thesis, we use the traditional interpretation that the audio/video coding research community has been using when refers to scalable bitstreams, but conveniently adapted to the context of video summarization.

#### 2.1.1. Time and space scalability in computational complexity

The theory of computational complexity studies the problems related to the resources required to execute some algorithm or task, such as memory requirements or execution time. Thus, the time complexity is related with the number of steps required to solve a problem, and it is usually referred to the size of the input data set. Similarly, the space complexity is related

with the amount of space (e.g. memory) required during the execution of the algorithm. These complexities are expressed using the “big O” notation which focuses on the growth rate rather than on the exact values.

As an example, let us consider the problem of sorting an array of values. A large number of methods have been proposed in order to perform that task faster and using less memory. One of such methods is the insertion algorithm, which has an average time complexity of  $O(n^2)$  and a space complexity of  $O(1)$ , meaning that the memory requirements are constant, independently of the size of the array. However, the time required to sort the array grows quadratically, which means that if the size of the array doubles, the average time required to sort it would be approximately four times longer. Another well-known sorting method is the Quicksort algorithm [Hoare, 1961; Sedgewick, 1978], which has an average time complexity of  $O(n \log n)$  and a space complexity of  $O(\log n)$ . Although both algorithms will need more time to sort a larger array, the growth rate is slower in the case of the Quicksort, which means that it scales (in time) better than the insertion method. However, the insertion algorithm scales better than the Quicksort in space, as the latter requires more memory space while the former does not.

Clustering is another problem in which scalability is essential in many applications using large data sets. Popular algorithms such as  $K$ -means [MacQueen, 1967] and hierarchical clustering [Ward, 1963; Jain et al., 1999] have different behaviours in terms of space and time complexities. Hierarchical clustering has a time complexity of  $O(N^3)$  ( $O(N^2)$  in some implementations [Theodoridis and Koutroumbas, 2006]) while  $K$ -means has a time complexity of approximately  $O(N)$ . That makes  $K$ -means more suitable when fast clustering of large data sets is required. However  $K$ -means also has several drawbacks. A number of algorithms especially designed to process large data set have been proposed, such as BIRCH [Zhang et al., 1997b] and O-cluster [Milenova and Campos, 2002].

### 2.1.2. Scalability in other fields

As the amount of information grows, the problem of efficient indexing and retrieval becomes more and more difficult. Databases have to deal with an increasing amount of transactions per second, and scalable data management approaches are required [Delis and Roussopoulos, 1992; Milliner et al., 1995]. Usually, the tables are partitioned and the workload is distributed in several database servers. Web search engines are examples of large scale systems dealing with huge amounts of information, processing millions of queries every day from millions of users. A scalable distributed architecture has a decisive role in these systems [Brin and Page, 1998].

Other scenarios in which scalability is a desirable attribute are networked systems. A scalable system should be able to cope with an increasing number of network nodes without a significant loss in performance. Examples of such systems are networked online games, in which low latency and scalable architectures are essential to provide the players with the feeling of true real time interactivity [Jiang et al., 2005]. Instant messaging and presence systems also need to be able to scale to millions of users [Schippers et al., 2010].

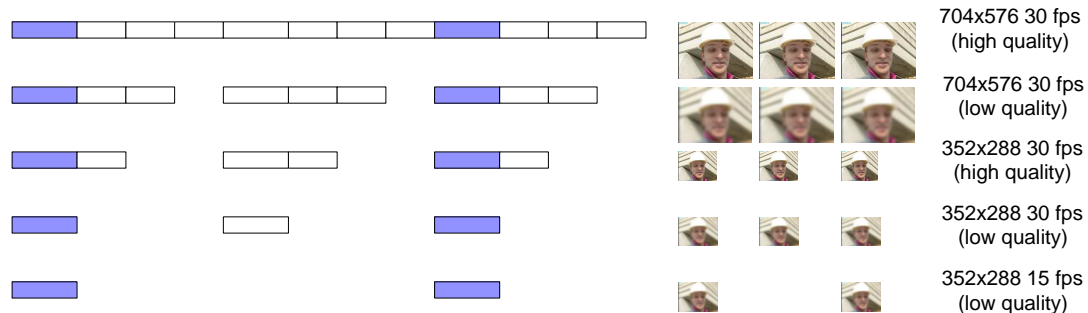


Figure 2.1: Example of scalable video stream.

### 2.1.3. Scalability in video coding

In the field of video coding, the term scalability has been used with slightly different meanings. In a similar sense to that used in computational complexity, scalable algorithms have been proposed to control the complexity of video encoding and decoding [Mietens et al., 2004; Yang et al., 2005; Tan et al., 2010]. [Mietens et al., 2004] proposes a scalable architecture able to reduce the complexity of MPEG encoding at the expense of some loss in quality. The scalability is achieved by varying the number of computed Discrete Cosine Transform (DCT) coefficients and the number of evaluated motion vectors. Reduced complexity is very useful in mobile devices in order to have an intelligent management of the limited resources by adjusting the workload according to the device or the specific usage conditions. Similarly, a complexity scalable architecture for H.264/AVC encoding is proposed in [Tan et al., 2010].

However, scalability has been also used in a different sense as a property of video (or audio [Brandenb, 1994; Homayounfar, 2003; Creusere, 2005; Kandadai and Creusere, 2008; Hansen et al., 2009]) bitstreams. In that sense, scalability refers to “the removal of parts of the video bitstream in order to adapt it to the various needs or preferences of end users as well as to varying terminal capabilities or network conditions” [Schwarz et al., 2007]. Figure 2.1 shows an example of a scalable bitstream. The bitstream contains a high resolution and high quality version of a video sequence. The bitstream is structured into packets, which can be discarded leading to different versions of the same sequence but with different resolution, quality or frame rate. Scalable video coding has applications in fields such as video adaptation and surveillance.

Although the popularity of scalable coding is relatively recent, some of the former video coding standards, such as MPEG-2 [ITU-T and ISO/IEC, 1994], H.263 [ITU-T, 2000] and MPEG-4 [ITU-T and ISO/IEC, 1999], already included some coding tools for scalable coding. MPEG-2 already used the concept of *layered coding*. It provided tools for temporal, spatial and quality (or SNR) scalability. MPEG-4 increases the number of tools for scalability, in addition to temporal, spatial and quality. It includes fine granularity scalability (FGS), which provides a wide range of possible bitrates and qualities. FGS is achieved through bit-plane coding of the transform coefficients [Li, 2001]. H.264/AVC [Wiegand et al., 2003] supports, in principle, temporal scalability, due to its flexibility to use any frame as reference, so scalable dependencies can be defined in the coding prediction structure of frames.

However, until the development of the scalable extension of H.264/AVC (also known as SVC)[Wiegand et al., 2007], the scalable profiles of previous standards were rarely used in practice. Some reasons can be found in the characteristics of traditional video systems, the loss in coding efficiency of some tools, such as those used for spatial and quality scalability, and the availability of competing alternatives such as transcoding[Schwarz et al., 2007].

Wavelets have been also used extensively in image and video coding[Ohm, 1994; Hsiang and Woods, 2001; Ohm et al., 2004; Ohm, 2005; Adami et al., 2007]. Wavelet transforms are usually performed using subband decompositions, which make them very suitable for scalable coding. The standard JPEG 2000[Christopoulos et al., 2000; ISO/IEC, 2004] for image coding is based on wavelets providing functionalities such as spatial and quality (or SNR) scalability, and an improved compression performance compared to the JPEG standard. For video coding, inter-frame wavelet codecs [Hsiang and Woods, 2001; Ohm et al., 2004; Ohm, 2005] use hierarchical subband decompositions to provide an embedded representation for spatial scalability. Temporal scalability is achieved using a hierarchical subband decomposition of frames within each Group of Pictures (GOP). This decomposition is combined with motion compensation[Ohm, 1994] in order to improve the coding efficiency. Depending on the order in which the temporal and spatial transforms are combined, different architectures have been proposed[Adami et al., 2007]. During the exploratory activity carried out by MPEG prior to the actual development of the SVC standard, a large number of proposals were based on wavelets. However, the proposal adopted as base model was an extension of H.264/AVC which showed better performance, in addition to the advantage of being backward compatible with H.264/AVC.

## 2.2. Video coding

### 2.2.1. MPEG-1, MPEG-2 and MPEG-4

In the late 1980s, with the progressive digitization of communication technologies and the maturity of compression techniques of digital image and video signals, international organizations such as the ITU-T (with the name CCITT prior to 1993) and the ISO/IEC showed an interest on the development of standards using those techniques. Following the success of the Joint Photographic Experts Group (JPEG), which addressed coding of still images, the ISO/IEC created in 1988 the Moving Pictures Experts Group (MPEG) to standarize the coding of digital video. These standards define the coded bitstream syntax and the decoding process, without specifying any encoding functionality (e.g. motion estimation algorithms, rate control).

The first international standards for video coding were H.261[CCITT, 1992] of the CCITT and MPEG-1[ITU-T and ISO/IEC, 1992] of the ISO/IEC, which use a hybrid coding scheme in which motion compensation and transform coding are combined. This coding scheme, with modifications, is still used in most video coding standards. In MPEG-1, the DCT is applied independently to disjoint blocks of 8x8 pixels. The coefficients are then quantized and encoded using variable length codes (VLC). This method is valid for the so called intra pictures or I pictures. The standard also uses the so called P and B pictures, which are encoded predictively

using a previous picture as reference. A prediction of the new picture is obtained from the reference picture using blocks (macroblocks) of 16x16 pixels that are displaced according to motion vectors (also coded in the bitstream). The difference between the actual picture and its prediction is also coded and transmitted. MPEG-1 was designed for storage of digital media in CD-ROM at a rate of about 1.5 Mbps. Functionalities such as random access and fast forward/reverse were already provided. Nowadays, MPEG-1 coded media can still be found in many video repositories. Apart from video, MPEG-1 also standardized the coding of digital audio[ITU-T and ISO/IEC, 1992]. MPEG-1 layer III (also known as MP3[Musmann, 2006]) is probably the most successful audio coding format to date.

The MPEG-2 standard[ITU-T and ISO/IEC, 1994] was developed jointly by the ITU-T and ISO/IEC, and can be seen as an extension of MPEG-1 to a much broader range of applications. Based on the same coding algorithms of MPEG-1, MPEG-2 introduces new coding tools for improved efficiency, interlaced video and higher bitrates and resolutions than those MPEG-1 was designed for. It also includes tools for spatial, temporal and quality scalable coding. MPEG-2 introduced the concept of profiles, which specify different sets of tools (usually including those specified in lower profiles), and levels, which specifies the range of parameters supported by that implementation (e.g. frame size, frame rate)[Sikora, 1997]. The conformance of an encoder or decoder implementation is specified with the profile and level (e.g. MPEG-2 MP@ML, i.e. Main Profile at Main Level). MPEG-2 is probably the most successful video coding standard to date (maybe only comparable to the more recent H.264/AVC). MPEG-2 is widely used in digital television and adopted by the corresponding broadcasting standards (e.g. DVB, ATSC). It is also widely used in the distribution and visualization of video content in optical devices since the DVD standard adopted MPEG-2 as main video coding format. Other consumer electronics, such as digital video cameras, also use MPEG-2.

The next standardization effort was MPEG-4[Schafer, 1998; Battista et al., 1999, 2000], which is a complex standard to code and represent audiovisual data for interactive applications and services. The basic unit of MPEG-4 are the audiovisual objects, which are multiplexed and combined into a scene. Each of these objects can be natural or synthetic audio or video. MPEG-4 includes a large number of coding tools for natural video, natural audio, synthetic images and video, synthetic audio, 3D meshes, etc. However, many tools remain almost unused due to the lack of interest from industry or because the technologies to create the content are not mature enough (e.g. reliable and accurate object segmentation). Regarding video coding, MPEG-4 Video[ITU-T and ISO/IEC, 1999] is similar to MPEG-2 including additional tools for improved efficiency and new tools for scalable coding[Li, 2001], and a bitstream syntax that supports both rectangular and arbitrary shaped video objects.

### 2.2.2. H.264/MPEG-4 Advanced Video Coding

As most of the preceding standards, H.264/AVC[ITU-T and ISO/IEC, 2003a; Wiegand et al., 2003; Sullivan and Wiegand, 2005] (released under the name H.264 by the ITU-T and MPEG-4

Advanced Video Coding by the ISO/IEC) is also based on temporal prediction with block-based transform coding. It uses the traditional types of slices, I, P and B, with intraframe and interframe prediction. However, it includes major changes compared to previous standards and new sets of coding tools which help to better exploit the redundancies in the sequence and to significantly improve the coding efficiency.

The recommendation specifies two different layers: a video coding layer (VCL) which deals with the efficient representation of the samples and video content, and a network abstraction layer (NAL) which deals with the format and header information in a suitable manner to be used by a variety of network environments and storage media. The bitstream is composed of a succession of NAL units, each of them containing payload and header sections with several syntax elements. An access unit (AU) is a set of consecutive NAL units which results in exactly one decoded picture.

H.264/AVC targets a broad range of applications[Wiegand and Sullivan, 2007] including cable, satellite and terrestrial broadcast, storage and distribution of high definition video, interactive video applications, conversational video services and video distribution through wireless and mobile networks. The industry has also shown a notable interest in the standard, including it in a large variety of consumer electronic devices. Most digital television broadcast standards, such as DVB, ATSC and ISDB have been updated to support H.264/AVC, and several countries are already using this coding format for terrestrial digital television services. Other important applications include storage and distribution of video over online video repositories (e.g. YouTube), in magnetic and optical devices (e.g. Blu-Ray disc system[Kelly et al., 2003; Kozuka, 2004], TV video recorders) and mobile devices (e.g. iPod, iPhone). The recommendation has been extended recently with additional functionalities to cover new applications, such as multiview video coding[Smolic et al., 2007] and scalable video coding.

### 2.2.3. Scalable extension of H.264/MPEG-4 AVC

The recent SVC standard[Wiegand et al., 2007; Schwarz et al., 2007; Schwarz and Wien, 2008] is built as an extension of H.264/AVC, including new coding tools for the generation of scalable bitstreams. SVC is based on a layered scheme, in which the bitstream is encoded into a base layer, H.264/AVC compliant, and one or more enhancement layers. Each enhancement layer improves the video sequence in one or more of the scalability modes (mainly temporal, spatial and quality).

Spatial scalability is achieved by using interlayer prediction from a lower spatial layer, in addition to intralayer prediction mechanisms such as motion compensated prediction and intra prediction. The same mechanism of interlayer prediction for spatial scalability can provide also coarse grain scalability (CGS) for quality scalability. Quality scalability can be also achieved using medium grain scalability (MGS), which provides quality refinements inside the same spatial or CGS layer. Temporal scalability in SVC is provided using hierarchical prediction structures, already present in H.264/AVC. Each temporal enhancement layer increases the frame rate of the decoded sequence.

In SVC, versions at different spatial and quality resolutions for a given instant form an AU, and it can contain both base layer and enhancement layer NAL units. Each NAL unit belongs to a specific spatial, temporal and quality layer. This information is stored in the header of the NAL unit in the syntax elements *dependency\_id*, *temporal\_id* and *quality\_id*. The length of the NAL unit header in H.264/AVC is extended to include this information. In SVC, the base layer is always H.264/AVC compatible. However, the extended NAL unit header would make the bitstream non compliant with H.264/AVC. For these reason, NAL units at the base layer have non extended headers, but they are preceded by additional NAL units containing only the SVC related information. These units are called prefix NAL units. If the stream is processed by a H.264/AVC decoder, these prefix NAL units and the other enhancement layer NAL units are simply ignored, and the base layer can still be decoded.

Many scenarios can benefit from the scalable properties of SVC, including video conferencing, IPTV[Schierl et al., 2007], adaptive streaming[Wien et al., 2007], efficient adaptation to heterogeneous terminals and networks[Schierl et al., 2007] and erosion storage of video surveillance sequences (i.e. a lower quality/resolution version is kept for long-term storage after some legal period in which the full quality and resolution version must be available)[Amon et al., 2007].

## 2.3. Video abstraction

Video is perhaps the type of content that requires more time to be consumed (i.e. visualized). The duration of a clip may range from minutes to hours, and the only way that a user can access to all the semantic information that the video is conveying is by its complete visualization. However, in most applications this is not possible or extremely inefficient. For that reason, a surrogate or abstract is often used instead of the actual content. Notable examples of systems using video abstracts are digital video libraries, such as YouTube, the Internet Archive or the OpenVideo project[Marchionini et al., 2006]. Television networks often publish content in their websites, such as the BBC, CNN, ABC, TVE. Search and browsing are much easier and efficient using abstracts than browsing actual video sequences. Usually, a single key image, the title and a short description are used to represent a specific piece of content.

### 2.3.1. Trade-off between information and browsing time

When an abstract is used as a surrogate of the content, the semantic information is dramatically reduced. However the amount of time required to its visualization is dramatically reduced too. In systems involving a large number of videos, it is very useful to present several abstracts simultaneously so the user can quickly browse them. Usually, the less time required to visualize an abstract, the less information or detail that it can convey. A reasonable trade-off between these two factors is desirable in a good abstract.

Some systems provide different levels of abstraction so the user can access interactively to more detailed abstracts. In order to illustrate this idea, we searched for the word *shuttle* in the search interface of the Open Video project. Figure 2.2 shows the results using two different



Figure 2.2: Search results for the word *shuttle* in the Open Video interface with different layouts: (a) keyframe and description, (b) keyframe and title. (*Source*: Reproduced with permission of the Open Video project)

layouts. Figure 2.2a shows abstracts using mainly keyframe, title and description. The layout shown in Figure 2.2b shows less detailed abstracts based on smaller keyframes and titles. The first layout presents more information about each result. However, in approximately the same area the first layout shows ten results, while the second one presents the 51 results. Thus, using the second layout, the browsing of the results is faster, although with the first layout the user has more information about each result.

In general, the trade-off between information and browsing time will be also present in any type of visual summary.

### 2.3.2. Modalities of video summaries

Although a keyframe with a title and an optional description is the most extended representation (see Figure 2.2), often a detailed abstraction is more convenient to represent the complexity of video content, especially in the case of long videos. For that reason, other modalities of visual abstractions have been proposed, in order to include more (audio)visual information. Although the terms abstract and summary are often used interchangeably, in this thesis we will prefer the term summary as a specific audiovisual abstract without any other associated information, such as textual data.

A widely used representation is the image *storyboard*, which abstracts the content into a set

of key images that are presented simultaneously. Figure 2.3 shows another example of the web interface of the Open Video project. It depicts a storyboard summary in addition to a textual description of the sequence. Compared to a single keyframe summary, a storyboard also shows the temporal nature of video, providing some visual information about the events taking place in the video sequence. Thus, the user can obtain a more detailed approximation of the underlying content. However, following the discussion of the previous section, the storyboard is a more detailed abstraction, but requires more browsing time and layout area than a single keyframe.

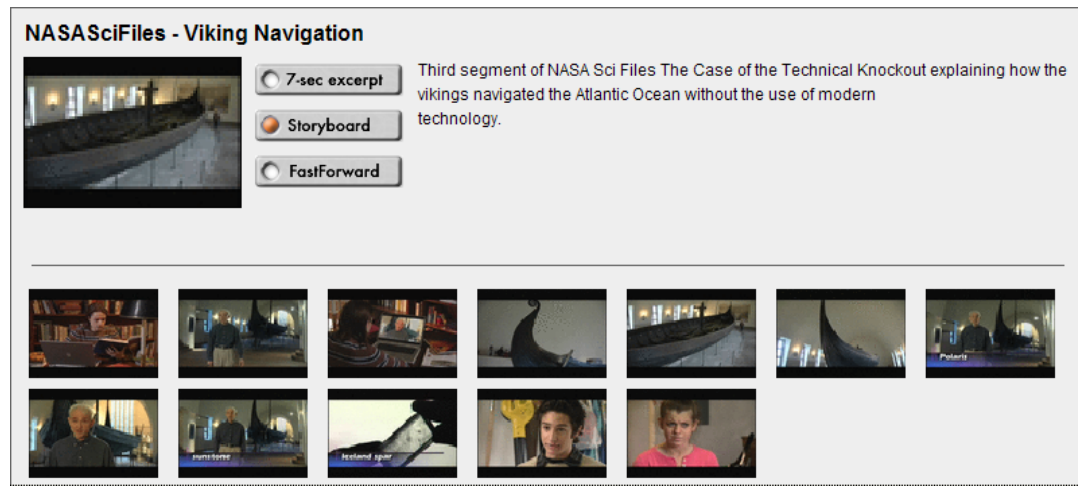


Figure 2.3: Example of summary (storyboard) in a digital library. (*Source*: Reproduced with permission of the Open Video project)

When dealing with video content, often it is more useful and meaningful to present the summary as a short video sequence, instead of independent frames. Segments provide dynamic information about the events and actions in the video sequence, that isolated images cannot provide. This representation, usually known as *video skim*, is obtained by selecting certain segments of the original sequence. An additional advantage of video skims is that they can include audio.

Between selecting single frames and selecting whole segments, there is still the possibility of selecting a variable amount of frames per segment. A *fast forward* is obtained by accelerating the sequence at a constant rate, which is useful to browse the content in a shorter time [Wildemuth et al., 2003]. Figure 2.4 shows an example of how a fast forward summary is integrated in a browsing interface.

Depending on how the summary is presented, the modalities are often classified in two groups: sequence-based summaries and pictorial summaries. The former includes video skims and fast forwards, and they are visualized as video sequences, requiring video playing capabilities in the browsing device. The latter includes keyframes, storyboards and other representations such as comic-like summaries [Calic et al., 2007] and video collages [Mei et al., 2009]. These summaries consist of a set of representative frames which are combined in some spatial layout and presented in a still format. Figure 2.5 shows how the source video sequence is transformed into video

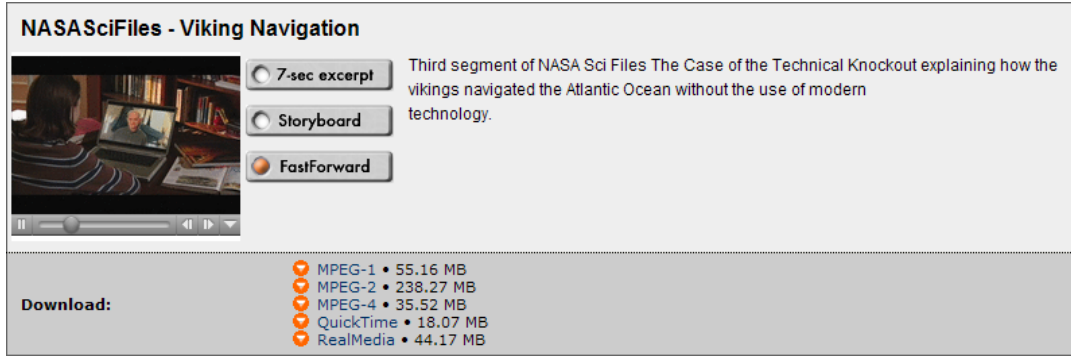


Figure 2.4: Example of fast forward summary and adapted versions of the content. (*Source*: Reproduced with permission of the Open Video project)

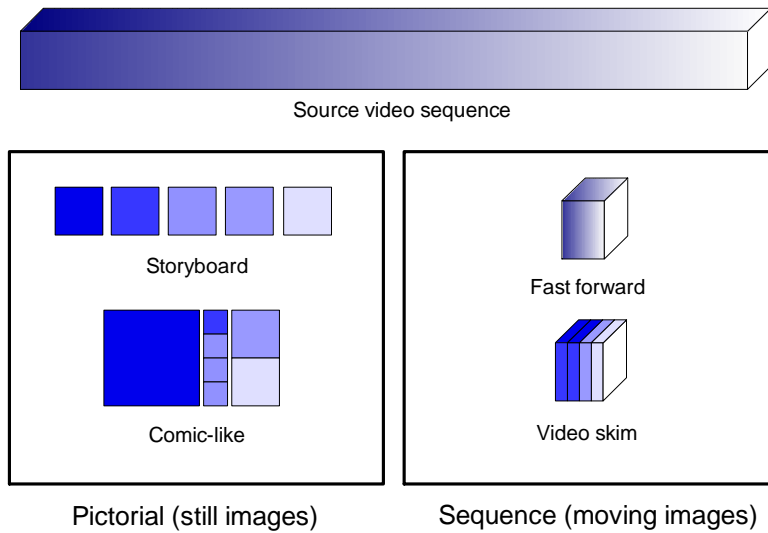


Figure 2.5: Modalities of video summaries.

summaries. It also depicts the different modalities used in this thesis.

### 2.3.3. Approaches to video summarization

In general, a video summary is built from the source sequence selecting frames according to some kind of semantic analysis of the content. Many algorithms have been proposed for keyframe selection and video summarization, using different criteria and abstraction levels. Recent surveys[Kang, 2002; Truong and Venkatesh, 2007; Money and Agius, 2008b] provide comprehensive classifications and reviews of summarization techniques.

At a low level, keyframe selection has been formulated as an optimization problem from both the set theory [Chang et al., 1999] and the rate-distortion[Li et al., 2005] points of view. At a higher semantic level, the sequence can be structured as a collection of shots, grouped into more abstract units (e.g. scenes, chapters), obtaining a hierarchical representation of the content[Zhu et al., 2003]. Many approaches use clustering algorithms to remove redundancy and select few

representative keyframes according to the resulting clusters[Zhuang et al., 1998; Hanjalic and Zhang, 1999; Gong and Liu, 2000; Mundur et al., 2006]. Usually, the set of keyframes is presented to the user as a storyboard.

The dynamic nature of video skims often requires more complex analysis, and the use of additional temporal features such as motion, audio or speech. If audio is included in the summary, special care has to be taken in the boundaries. Several approaches have been used in video skimming, including visual attention[Ma et al., 2005], image and audio analysis[Smith and Kanade, 1998; Li et al., 2006], highlight detection[Ekin et al., 2003] and high level semantics[Pfeiffer et al., 1996].

Fast forwards can be obtained easily just speeding up the sequence at a constant rate, without any content analysis. Although easy and effective, constant fast forwards are limited and therefore they are sometimes improved using a content-based approach. Often, there are parts that can be sped up, because they do not convey relevant information, while more significant parts can be played at normal rate. Thus, a content-based fast forwarding can be obtained by selecting frames based on some semantic clue. Motion activity and camera motion have been used as clues to drive the selection of frames[Peker et al., 2001; Bescós et al., 2007]. [Peker et al., 2006] proposes the use of face tracks as semantic clues.

In order to obtain better results, the domain of the content can be exploited by the summarization method. For instance, sports video summarization tries to use prior knowledge, such as the structure and characteristics of a specific sport game[Li and Ibrahim Sezan, 2001; Ekin et al., 2003; Tjondronegoro et al., 2004; Zhao et al., 2006; Babaguchi et al., 2007]. Usually, these approaches are based on the detection of some important events that must be included in the summary (e.g. goals, end of game). Other typical scenarios are news[Zhang et al., 1997a; Maybury et al., 2004; Lie and Lai, 2005; Peker et al., 2006; Damnjanovic et al., 2007], which is a highly structured and edited video content, surveillance[Damnjanovic et al., 2007] and home videos[Peng et al., 2008]. Additionally, metadata or auxiliar information can be provided for higher level understanding of the content[Smith and Kanade, 1998; Fonseca and Pereira, 2004].

Recently, an intense research in rushes summarization has been motivated by the TRECVID rushes summarization task[Over et al., 2007, 2008]. This type of content is significantly different from other video sources, as rushes are unedited footage containing retakes, being much more redundant than other types of video content. This content also contains undesirable junk segments such as blank frames, clapboards, etc. The participants in the task adapted their systems to cope with the specific characteristics of this content[Dumont and Merialdo, 2007; Valdés and Martínez, 2008; Ren and Jiang, 2009].

#### **2.3.3.1. Evaluation of video summaries**

Perhaps the most debatable aspect in video summarization is the evaluation of the results. A consistent and widely accepted evaluation framework is still unavailable, and the evaluation methodology varies from publication to publication. Not only the metrics used differ, but also the evaluation criteria.

The most accepted methodology is the subjective evaluation or user study. A number of subjects are required to visualize a number of video summaries and then they have to answer some questions. These questions try to assess the agreement of the subject according to some criteria, such as informativeness, pleasantness, coherence, conciseness or ease of view[Ngo et al., 2003; Zhu et al., 2004; Jansen et al., 2008]. Another variation is to pose some specific questions about the objects and events in the video to estimate how well the summary covers the semantic information in the video[Santini, 2007]. Sometimes the evaluation is more application-oriented, and the subject is required to perform some specific task (e.g. search for a specific event or object), while some parameters are measured (e.g. time required to perform the task, number of clicks on the interface).

The main problem with subjective evaluations is that they must involve a large number of human individuals and visualization tests to be statistically significant, which leads to a very time consuming task. Sometimes the users are required to view the original sequence before, which also increases notably the time cost of the evaluation. Another major problem is that, once the evaluation is performed, it is not replicable nor reusable. If the summarization algorithm is modified or changed, the evaluation has to be repeated. Subjective evaluations are also influenced by other human factors which may not be easy to conceal, such as different profiles of users, interface issues, fatigue in long sessions, etc.

Several objective metrics have been proposed for video summary evaluation. For storyboard evaluation, a fidelity (or error) metric can be computed from the set of keyframes and the original sequence[Liu et al., 2004]. For video skims, the inclusion of relevant segments can be measured (i.e. precision and recall), according to a manual annotated ground truth[Chang et al., 2002; Ariki et al., 2003]. In order to avoid subjective biases in the ground truth, some works use highlights or replays, produced by third parties, as reference. However, there is no clear evidence that these metrics map well to the goodness of a summary from a subjective point of view[Truong and Venkatesh, 2007].

The dataset used for the evaluation has a crucial impact on the results. The access to suitable content to evaluate video summaries is often difficult due to legal restrictions. Some commonly used datasets are the rushes from the TRECVID evaluation. However, although valuable information can be obtained, the main drawback is that rushes are much more redundant than the video content used in most applications, which makes the results difficult to extrapolate to other domains.

Another problem of the evaluation of the TRECVID rushes summarization task is that it cannot be replicated. It is not useful to evaluate and compare with other approaches. For that reason, several approaches to automatic assessment of video summaries have been proposed recently[Huang et al., 2004; Ren et al., 2008; Dumont and Merialdo, 2010; Valdés, 2010]. However, these approaches are still only valid to compare approaches in the context of the previous dataset (i.e. rushes).

### 2.3.3.2. Other trends in video summarization

Most research in video summarization has been focused on the analysis of the video sequence, with techniques trying to get some insight about the underlying content. However, current research in video summarization also addresses new functionalities and enhanced applications. Some examples are:

- Customization and personalization. Different users have different preferences. Personalized summaries are more effective, as they can provide each user with more interesting information according to a personal profile[Tseng et al., 2004]. A customized summary can be also generated from a query formulated by the user with the requirements for the summary[Fonseca and Pereira, 2004].
- Online summarization. As the processing delay of most summarization algorithms is very large, summaries are usually generated offline. The objective of online summarization is to process the data as it arrives, with no need for future data, generating the output summaries with a minimal delay[Valdés, 2010]. In principle, some methods can be considered online, such as those based on highlights[Ekin et al., 2003]. Camera information or motion activity can be also used to generate summaries online, as they can be obtained without future information[Bescós et al., 2007].
- Hierarchical summaries. These approaches analyze the sequence and create a hierarchical abstract, structured with different levels of detail, such as frames, shots, groups and scenes[Zhu et al., 2003; Meessen et al., 2006]. Users can interact with the summary and navigate through different abstraction levels.
- New types of presentation. Other appealing and intuitive formats have also been explored, such as comic-like summaries[Yeung and Yeo, 1997; Uchihashi et al., 1999; Calic et al., 2007], video booklets[Zhu et al., 2005a], video collages[Mei et al., 2009] and video trees[Jansen et al., 2008].
- Multi-view and multi-document summarization. In multi-view video systems, many cameras record the same scene simultaneously. Multi-view video summarization[Fu et al., 2010] exploits the correlation among the different views and presents a single summary with the important events combining information from the different views. A similar approach is multi-document summarization[Wang and Merialdo, 2009], which tries to avoid presenting redundant information of related video documents using a single multi-document summary instead of several single-document summaries.

## 2.4. Video adaptation

### 2.4.1. Universal Multimedia Access and MPEG-21

Nowadays, multimedia information can be accessed from a diverse set of devices using heterogeneous networks. In this scenario, the concept of Universal Multimedia Access (UMA)[Vetro

et al., 2003a; Vetro, 2004] proposes the access to multimedia information from any device and independently from the usage conditions. However, this information must be provided in a suitable format according to the usage environment (e.g. terminal, network, preferences). Content adaptation is a main requirement to effectively bring the content from service providers to the actual users, handling the enormous variability of resource constraints such as bandwidth, display capabilities or processing power[Chang and Vetro, 2005]. Especially important is the case of mobile devices, such as PDAs and mobile phones, where other issues such as limited computational resources and low power consumption requirements become very important.

The MPEG-21 standard[ITU-T and ISO/IEC, 2001b; Bormans et al., 2003; Tseng et al., 2004] aims at developing a normative open framework for multimedia delivery and consumption, based on the concepts of Digital Item (DI) as basic unit of transaction, and Users as entities that interact with DIs. The objective is to enable a transparent and augmented use of multimedia data across a wide range of networks and devices. The description of the usage environment in which the multimedia content is consumed is essential to be able to adapt the content to each case in the UMA paradigm. The Usage Environment Description (UED) tools of MPEG-21 DIA[ITU-T and ISO/IEC, 2003b; Vetro, 2004; Vetro and Timmerer, 2005] can be used to describe, among others, the terminal capabilities, network characteristics and user characteristics with a standardized specification. The following example shows how some basic, but important, characteristics of the terminal and the network can be described using the *TerminalCapability* and *NetworkCharacteristics* elements. It describes the context of a user who accesses multimedia content using a PDA (with a resolution of 480x320 pixels) through a 384 kbps network.

```
<DIA>
  <Description xsi:type="UsageEnvironmentPropertyType">
    <!-- Network description -->
    <UsageEnvironmentProperty xsi:type="NetworksType">
      <Network xsi:type="NetworkType">
        <NetworkCharacteristic xsi:type="NetworkConditionType"
          maxCapacity="384000"/>
      </Network>
    </UsageEnvironmentProperty>
    <!-- Terminal description -->
    <UsageEnvironmentProperty xsi:type="TerminalsType">
      <Terminal id="pda">
        <TerminalCapability xsi:type="DisplaysType">
          <Display>
            <DisplayCapability xsi:type="DisplayCapabilityType">
              <Mode>
                <Resolution horizontal="480" vertical="320"/>
              </Mode>
            </DisplayCapability>
          </Display>
        </TerminalCapability>
      </Terminal>
    </UsageEnvironmentProperty>
  </Description>
</DIA>
```

```
</Terminal>
</UsageEnvironmentProperty>
</Description>
</DIA>
```

### 2.4.2. Content-blind adaptation

Dealing with different terminals and networks, it becomes evident that in constrained environments a high quality version is not suitable. Quite likely, such a version would not be delivered properly (e.g. the network cannot fulfill the bitrate requirements, the terminal cannot decode high quality video) or the quality of the version is somehow wasted (e.g. the terminal has a small display so the resolution of the video is reduced significantly). Delivering a lower bitrate version with a suitable resolution and frame rate is more useful and makes better use of the limited resources. A first approach is content-blind adaptation, which deals with the adaptation of the audiovisual signal (e.g. resolution downsampling, bitrate adaptation), but does not take into account the content itself.

Considering a source bitstream and its adaptation and delivery as a modified bitstream, the whole process often implies decoding, adaptation to the target usage environment and encoding of the adapted content. This adaptation method is known as transcoding[Ahmad et al., 2005], and it can be computationally very demanding. Simplified architectures have been proposed in which encoding and decoding are not performed completely up to the pixel domain, reusing part of the information (e.g. motion vectors, macroblock coding modes, DCT coefficients) in order to avoid complex processing[Acharya and Smith, 1998; Vetro et al., 2003b; Ahmad et al., 2005; Xin et al., 2005; Lefol et al., 2006]. This architectures introduce some quality loss compared to the full decoding-encoding cascade, although it can be controlled. As performing a transcoding every time a user requests a video can be extremely demanding, an alternative is the preencoding of several versions (i.e. variations), so the user (or the adaptation engine) can select only among the available versions. Figure 2.4 shows an example of adapted versions available as offline variations (e.g. MPEG-1, MPEG-2, etc.). The user then can decide which version is the most suitable according to codec capabilities, display resolution, network capacity or storage requirements. However, if the adaptation engine has knowledge about the terminal and network (e.g. using the associated UED), it can deliver the most appropriate version, transparently to the user.

Another approach to content-blind adaptation is scalable coding. With scalable bitstreams the problem of adaptation is addressed at the encoding stage, in a way that simplifies the adaptation process. A scalable video stream contains embedded versions of the source content that can be decoded at different resolutions, frame rates and qualities, simply selecting the required parts of the bitstream. Thus, scalable video coding enables a very simple, fast and flexible adaptation framework to a variety of terminals and networks, with different capabilities and characteristics.

### 2.4.3. Content-based adaptation

In contrast to content-blind adaptation, content-based adaptation takes advantage of a certain knowledge of what is happening in the content (i.e. semantics) to perform a better adaptation. For example, in a video surveillance application, if the adaptation engine detects which segments do not contain objects of interest (e.g. people, vehicles), it can remove them from the adapted version in order to save resources. Note that what the adaptation engine is actually doing is discarding useless information, creating thus a summary. Indeed, in [Chang and Vetro, 2005], video summarization is considered a special type of structural adaptation, in which the summary is an adapted version of the original content.

Content-based adaptation, often also known as semantic adaptation, includes personalization[Tseng et al., 2004; Maybury et al., 2004], video foveation[Lee and Bovik, 1999, 2000; Itti, 2004], region of interest[Bae et al., 2006; De Schrijver et al., 2007] and object-based adaptation[Cavallaro et al., 2005; Cheng et al., 2007]. The knowledge about the content can be extracted automatically or provided as metadata[van Beek et al., 2003; Magalhaes and Pereira, 2004] from previous automatic analysis or manual annotation. This knowledge ranges from very low level (e.g. shot changes, color and motion features) to high level (e.g. events, objects, actions).

## 2.5. Summary and conclusions

This thesis involves frameworks and applications integrating several research fields, namely video summarization, adaptation, scalable approaches and video coding. In this chapter we have provided a brief overview of these related technologies, along with some review of recent works and trends. We have focused especially on the concept of scalability and its use and interpretation across different fields. Scalable approaches in other fields inspired many of the ideas developed in this thesis.

## Part II

# Summarization and adaptation using scalable video coding



## Chapter 3

# Generation of video summaries by bitstream extraction

One objective of this thesis is to develop new applications of video summarization where the generation and delivery of the summary must be fast. Most works on video summarization do not consider the problem of the generation of the bitstream as an integral part of the summarization process, assuming that it is a non-critical offline process. Here we tackle the problem of efficiency in the generation of the bitstream as an integral part of the summarization process.

Alternatively to conventional generation based on transcoding, this chapter presents a different approach based on bitstream extraction. The bitstream extraction framework is used extensively along the rest of this thesis as the last stage of the summarization process. It is based on processing directly the packets of the bitstream. Both transcoding and extraction architectures are compared in terms of efficiency and rate-distortion performance.

Part of this chapter is based on the publications: [Herranz and Martínez, 2009c, 2010b].

### 3.1. Related work on bitstream customization

Adaptation in the bitstream domain has been tackled and standardized at some extent in the Bitstream Syntax Description (BSD) tools [Devillers et al., 2005] of MPEG-21 Digital Item Adaptation (DIA), aimed to generic adaptation of coded sequences directly operating with the bitstream. Particularly, the adaptation of H.264/AVC along the temporal axis using MPEG-21 BSDL is detailed in [De Schrijver et al., 2006]. [Gang et al., 2004] describes a system using frame dropping based on the perceived motion energy. However, most of these works are used from a content-blind adaptation point of view, where the bitstream extraction is guided by constraints in the usage environment. If we consider video summaries as semantic constraints to be applied to the source sequence, we can use the same framework for summarization. Following a similar approach, a semantic adaptation framework is described in [De Bruyne et al., 2007], combining BSD tools and semantic metadata to perform shot-based adaptation.

In the context of a wavelet-based scalable video codec, [Herranz, 2007] uses an activity measure to change dynamically the frame rate in order to obtain a content-driven fast forward of the input sequence. The adaptation is performed efficiently using bitstream extraction.

### 3.2. H.264/MPEG-4 AVC and hierarchical prediction structures

An H.264/AVC bitstream is composed by a succession of NAL units, each of them containing a syntax structure. Some of such structures are the parameter sets. In particular, the Sequence Parameter Set (SPS) and the Picture Parameter Set (PPS) are essential for the decoder in order to be able to reconstruct the sequence. They convey important header information such as frame resolution, profile, frame rate, etc. The information present in these parameter sets is used by the decoder to decode all the pictures following them, although it is not fixed and can be modified during encoding if necessary, sending new parameter sets.

In H.264/AVC, a picture (frame or field) is divided into slices. Each slice contains a number of macroblocks and it is usually packed into a NAL unit. An Access Unit (AU) is a set of consecutive NAL units which results in exactly one decoded picture. For the sake of simplicity, we will assume frame coding and one slice per frame, using interchangeably frame, picture, slice and access unit. Frames are also structured in Groups of Pictures (GOPs) related by temporal prediction, though frames from different GOPs can be also related by prediction.

One of the key features of H.264/AVC to increase the compression performance is the possibility of specifying much more flexible prediction structures. In prior standards, there is a strict dependency between the ordering of frames for motion compensation prediction (coding order) and the ordering for presentation. For instance, in MPEG-1/2/4, P frames are predicted only from the preceding I or P frame, and B frames are not used as references and are predicted only from the preceding and succeeding I or P frames (see Figure 3.1a). In contrast, in H.264/AVC any frame can be marked as reference and used for prediction of subsequent frames. One of such family of prediction structures are hierarchical prediction structures, where a set of frames is coded at a base level (level 0) and at each step a new set of frames is coded using previously coded frames. The process is repeated with additional sets of frames adding new levels to the hierarchy. In [Schwarz et al., 2006], the impact of hierarchical prediction in the coding efficiency is studied and experimental results showed that these structures have a good coding efficiency, which usually improves when the length of the structure is increased. Hierarchical prediction implicitly generates temporal scalable bitstreams. Each set of frames from each temporal level forms a new enhancement layer. Thus, selecting only the part of the bitstream corresponding to the frames of the base layer, a low frame rate version of the bitstream can be decoded, and can be refined adding enhancement layers. Some typical hierarchical structures are those using dyadic decompositions, where the number of frames is doubled with each enhancement layer. Figures 3.1b and 3.1c show two typical hierarchical structures with 4 dyadic stages (3 enhancement layers):

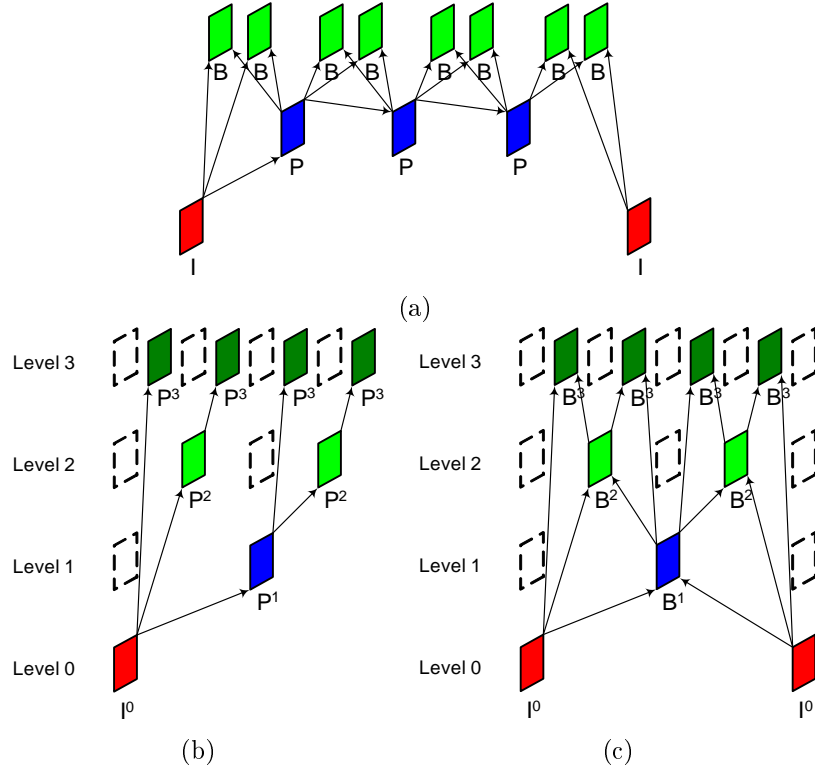


Figure 3.1: Prediction structures: (a) MPEG-2 structure, (b) hierarchical structure in H.264/AVC using P frames (structural delay 0), (c) hierarchical structure in H.264/AVC using B frames (structural delay 8).

- $I^0P^3P^2P^3P^1P^3P^2P^3$  (see Figure 3.1b). This structure does not use backward prediction, being compliant with the H.264 baseline profile, as only I and P frames are necessary. All the predictions are from past frames in display order, and thus the structural delay (as the maximum difference between the presentation and decoding indexes of a frame) is 0 frames.
- $I^0B^3B^2B^3B^1B^3B^2B^3(I^0)$  (see Figure 3.1c). This structure uses also backward prediction and B frames are necessary. In this case, the structural delay is 8 frames but the coding efficiency is higher than in the previous structure.

In addition to arbitrary frame referencing, H.264/AVC also supports multi-frame motion compensation, which means that several prior coded frames can be used as reference. In order to handle the complexity of these new prediction structures, H.264/AVC specifies the operation of the decoded picture buffer (DPB), which is a buffer containing previously decoded frames which can be used as references. Frames used as reference are signalled according to the current state of the buffer and the order in which decoded frames are stored in the buffer. The DPB of the decoder must replicate the status of the multiframe buffer of the encoder, according to the memory management control operations (MMCO) included in the bitstream.

An instantaneous decoding refresh (IDR) access unit is a special type of access unit containing

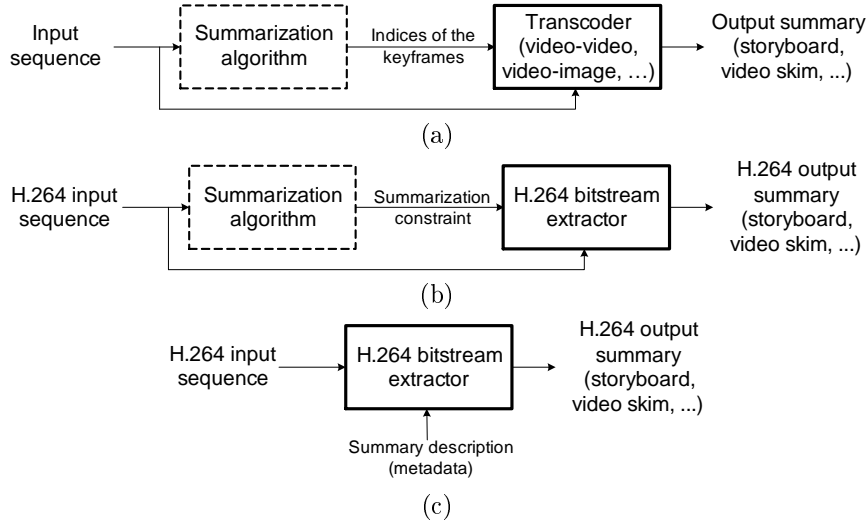


Figure 3.2: Methods for the generation of summaries: (a) conventional approach, (b) proposed approach for H.264/AVC bitstreams with online summarization analysis, (c) proposed approach for H.264/AVC driven by metadata

an intracoded frame, but also signalling that the decoding of the subsequent frames does not require any reference frame prior to that intracoded frame. Thus, the DPB can be flushed and the decoding process of subsequent frames is independent from previous ones. IDR access units prevent from propagating errors due to the use of incorrectly decoded frames as reference. In H.264/AVC, conventional I frames do not prevent completely from error propagation as erroneous frames may remain in the DPB after an I frame, and eventually used as reference. IDR access units are also important to provide random access points.

### 3.3. Summarization approach

Every summarization system has two different stages: analysis and generation. The analysis stage, using the term *analysis* in a wide sense, includes all the processes addressed to characterize and to represent the content in order to remove semantic redundancies, and the selection of the frames to be included in the summary. Feature extraction, shot boundary detection, high level structuring, keyframe selection, personalization, clustering or optimization algorithms are examples of operations that can be included in the analysis stage. Most works in video summarization [Truong and Venkatesh, 2007] deal only with this stage, but the generation stage is barely studied. However, in many scenarios, the generation of the bitstream is critical for efficient summarization. The generation stage obtains the coded bitstream of the summary from the input bitstream, once the sequence has been analyzed and the frames to be included in the summary have been determined. The analysis and generation stages are connected by some kind of summary description with the frames to be included in the summary. Both analysis and generation are not required to be done at the same time, as the summary description can be stored as metadata.

In general, the generation of the output summary requires the decoding of the frames to be included and the subsequent encoding stage (see Figure 3.2a). Most of the frames are encoded predictively from previous frames, and all the referenced frames must be also decoded prior to decode a given frame. The whole transcoding process may have an important computational cost, especially when the summary is in the form of a video sequence (e.g. video skim).

If the bitstream is encoded in such a way that the transcoding process can be replaced by a simple selection of parts of the input bitstream, the generation of the summarized bitstream will be much more efficient. In that case, we use the term *embedded summary* to point out the fact that the summary is already available in the input bitstream. An example of embedded summaries is the case of uncompressed video (e.g. in YUV format), in which it is possible to select each frame independently and build a new sequence just concatenating the values of the samples of each selected frame. Another example is the case in which all the frames are intracoded (e.g. MJPEG), as they can be decoded independently and easily concatenated operating directly over the compressed format, perhaps with some minor header updating. In all of those cases, the summary can still be described with the indexes of the frames of the source sequence that must be included, and the frame is still the basic unit for summarization.

However, in most video coding formats, frames are coded in groups rather than individually, in order to exploit temporal redundancy. For that reason, it is more convenient to refer the output of the analysis stage to these groups rather than to single frames. The frame-based model to describe summaries can still be used, but a number of constraints must be applied depending on each case and the coding structure. For convenience, we introduce a different model to describe summaries, based on coding units as basic units for summarization - we use the term *summarization unit* (SU) -. Particularly, we use temporal scales rather than individual frames, which is specially suitable for H.264/AVC with hierarchical prediction structures.

The model relies on the assumption that the only allowed selection of frames in each SU is the selection of those frames belonging to the same temporal level, and the selection of all the frames in that level. The summary is described using a function called *summarization constraint* (defined in the next section), which is the only information that the bitstream extractor needs to generate the summary. The architecture with the main modules is shown in Figure 3.2b. The analysis for summarization is completely detached from the generation, and it would even be possible an architecture where analysis is performed previously (e.g. at encoding time) and the description (i.e. the summarization constraint) is stored as metadata (see Figure 3.2c).

Selecting subsets of frames from the SUs rather than individual frames has the drawback of losing the exact location of each frame in time. Therefore, in general, it is not possible to select a given frame of the bitstream, but a neighbouring frame within the SU. However, as frames are very similar to their neighbours (except when a shot change occurs), if the length of the SU is small enough and the analysis stage is designed carefully to prevent from including problematic units, there should not be any noticeable difference.

### 3.4. Summarization model

In the following, we introduce the basic definitions and concepts of the extraction-oriented summarization model, in the context of H.264/AVC with hierarchical prediction structures, but that can be easily extended to other coding formats and coding structures.

#### 3.4.1. Basic model for extraction

In H.264/AVC each frame is coded in an integer number of NAL units. For simplicity, we will consider that each frame is coded into one slice, which in turn is a single NAL unit, and it also corresponds to a single AU. However, building the summary as a concatenation of those NAL units containing the frames of the summary will probably lead to a non-decodable bitstream, as most of them are encoded predictively with respect to previous frames in the source bitstream.

The source sequence  $V$  with  $N$  frames can be described as a sequence of consecutive AUs (each AU representing also a frame)

$$\mathbf{V} = (AU_0, AU_1, \dots, AU_n, \dots, AU_{N-1}) \quad (3.1)$$

where  $n \in I_{\mathbf{V}} = \{0, 1, \dots, N-1\}$  is the frame index.

We define the *summarization unit* (SU)  $\mathbf{U}_m = (AU_n, AU_{n+1}, \dots, AU_{n+L_{\mathbf{U}_m}-1})$  (of length  $L_{\mathbf{U}_m}$ ) as a set of consecutive AUs related by the prediction coding structure without references to other AUs not belonging to the summarization unit. Thus, the source sequence  $\mathbf{V}$ , can be structured into  $M$  summarization units, denoting the set of indexes of the SUs of the sequence as  $I_{\mathbf{V}} = \{0, \dots, M-1\}$ .

We define an *embedded summary*  $\mathbf{S} \subseteq \mathbf{V}$  as a subset of the sequence  $\mathbf{V}$  in which all the AUs (once assembled into the bitstream) are decodable. It implies that any AU used as reference to decode any other AU in  $\mathbf{S}$  must also belong to  $\mathbf{S}$ . The summary is then perfectly described by the set  $I_{\mathbf{S}} \subseteq I_{\mathbf{V}}$ , which contains the indexes of the AUs belonging to the summary.

Alternatively, an embedded summary can also be obtained from a subset of SUs, which ultimately is another subset of AUs. As it is more convenient, we will use this approach to analyze and generate the summary. In this case, the summary is described by the set  $I_{\mathbf{S}} \subseteq I_{\mathbf{V}}$  with the indexes of the SUs belonging to the summary.

Figure 3.3 illustrates the concept of summarization units and embedded summaries. As it can be seen, intracoded frames can be included directly in the summary, which is especially useful in modalities based on isolated and separated images (e.g. storyboards). A closed GOP is a valid summarization unit (see Figure 3.3). As shown in Figure 3.4a, for a given index  $m$ , multiple SUs can be extracted. For convenience, we consider only two possibilities: the intracoded AU and the whole unit. These two possibilities within the same coding unit can be considered implicitly as two temporal scales or levels (note that with P and B frames is even possible to define up to three temporal levels without using hierarchical structures; see Figure 3.1a).

We say that two summarization units  $\mathbf{U}_i$  and  $\mathbf{U}_j$  are *overlapped* if they share one or more AUs, i.e.  $\mathbf{U}_i \cap \mathbf{U}_j \neq \emptyset, i \neq j$ . Figure 3.3b shows an example of overlapped SUs in which the last

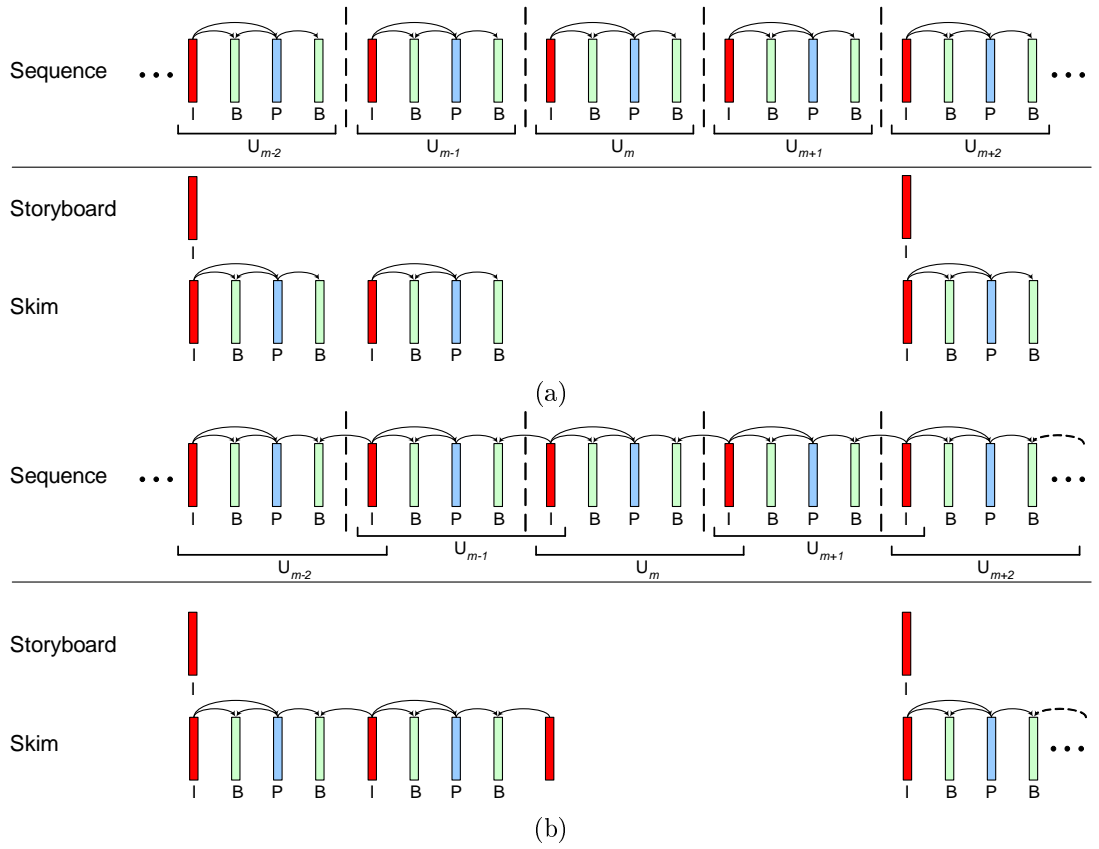


Figure 3.3: Summarization units and embedded summaries: (a) closed GOP, (b) open GOP (overlapped SUs).

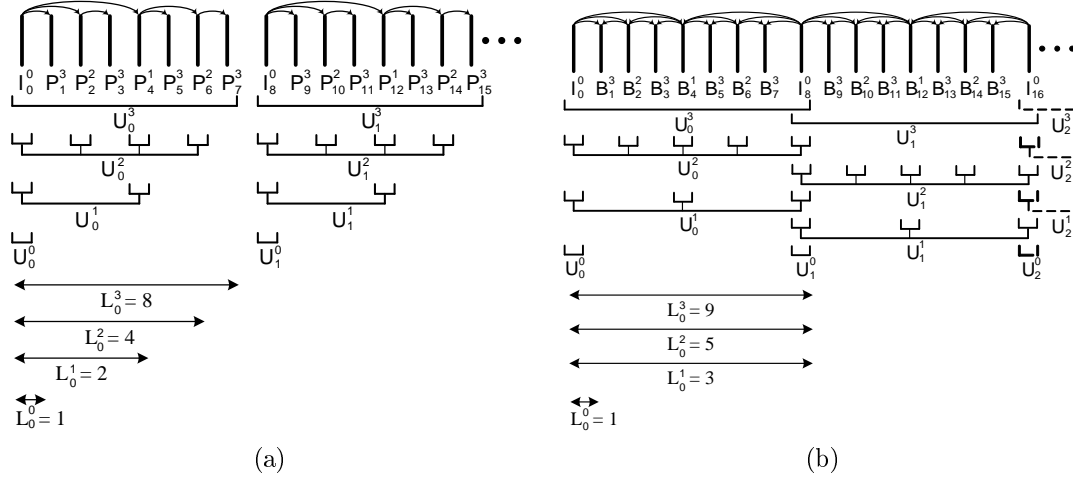


Figure 3.4: Examples of summarization units: (a) low delay structure with hierarchical P frames, (b) overlapped SUs using hierarchical B frames (display order).

B frame of each GOP is predicted from the I frame of the next GOP. In that case, that I frame must be included in order to be able to decode the B frame.

This model, although described in the context of H.264/AVC, is also valid for other coding formats using conventional I, P and B frames, such as MPEG-1, MPEG-2 and MPEG-4. In these coding formats, the I frame is a random access point where the decoder can resume the decoding.

### 3.4.2. Extended model with hierarchical prediction structures

In the case of hierarchical coding structures, the previous model can be extended to an arbitrary number of temporal levels. If the sequence  $\mathbf{V}$  is encoded using hierarchical structures with  $T$  temporal decompositions (which means  $T+1$  temporal levels), an AU can be also denoted as  $AU_n^t$ , where  $t \in \{0, 1, \dots, T\}$  is the temporal level. For simplicity, we assume that level 0 is composed only of intracoded AUs.

For each temporal level, a subsampled version can be decoded, as there are no breaks in the prediction chain (as shown in Figure 3.1). For this reason, there are several valid SUs for each index  $m$ , depending on the temporal level.  $U_m^t$  denotes the summarization unit at temporal level  $t$  and index  $m$ . The SUs satisfy

$$U_m^0 \subset U_m^1 \subset \dots \subset U_m^t \subset \dots \subset U_m^T \quad (3.2)$$

An example of SUs obtained from a low delay structure, such as the structure  $I^0P^3P^2P^3P^1P^3P^2P^3$ , is shown in Figure 3.4a. As the figure shows, the length of the SU depends on the temporal level. The most important characteristic of this structure is that it does not require the use of B frames. As a first advantage, it can be decoded by a baseline profile decoder. A second advantage is that this structure only uses forward prediction, so it is suitable for low delay applications, having a structural delay of 0 frames. And regarding the proposed

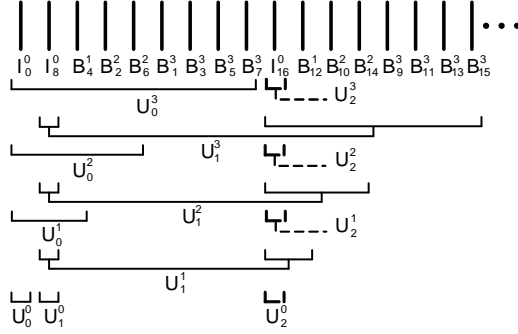


Figure 3.5: Summarization units using hierarchical B frames (coding order).

model of SUs, it enables the structuring of the bitstream in non-overlapped SUs, which is also very desirable. However, the main drawback of this structure is the lower compression efficiency compared to structures using B frames. Another drawback is the appearing of temporal artifacts due to the different prediction paths followed to code each frame of the structure [Schwarz et al., 2007], accumulating prediction errors in a non-equal manner.

On the other hand, adding backward and bidirectional prediction to the coding structure increases notably the compression efficiency, and reduces significantly temporal artifacts. It is achieved using B frames in the prediction structure. A typical example of this structure providing the same functionality as Figure 3.4a in terms of temporal scalability is  $I^0B^3B^2B^3B^1B^3B^2B^3(I^0)$  (see Figure 3.4b), where the length of the GOP is also of 8 frames, but also needs an additional I frame from an adjacent GOP. Adding bidirectional prediction has effects on the coding order, increasing the structural delay: frames used as references for backward prediction must be coded before frames referencing them. Figure 3.5 shows the coding order of the frames from the previous example. The two I frames must be coded before all the frames between them, leading to a structural delay of 8 frames. SUs using B frames can be overlapped when they share references from different GOPs. Most of the resulting SUs from the previous example are always overlapped, except for temporal level 0, where no prediction is used.

### 3.4.3. Other coding structures

Although the coding structures described in the previous subsections are the most used in practice, H.264/AVC is flexible enough to allow many other possible structures. In contrast to previous coding formats, H.264/AVC includes tools such as arbitrary referencing, multiframe motion compensation and long term prediction that overcome some of the limitations of previous coding formats and improve the coding efficiency.

Compression efficiency can be further improved using long term prediction in the base layer instead of coding the frames only as I slices. The structure of Figure 3.6a is the result of replacing a number of I frames of the base layer of Figure 3.4b by P frames, which are predicted from the previous I or P frame in the base layer. Note that it will increase coding efficiency, but it will also increase the length of the SU. A larger SU brings a decreasing of the precision in selecting

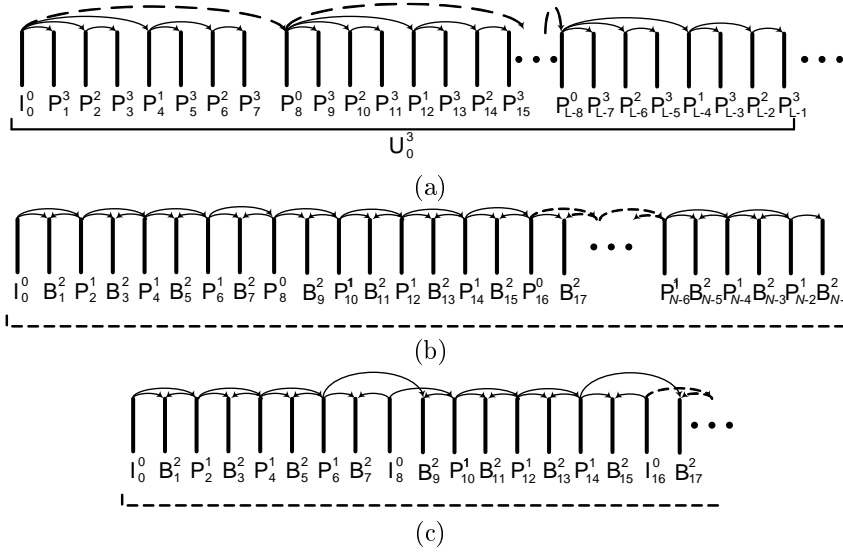


Figure 3.6: Other coding structures: (a) summarization unit using long term prediction, (b) streaming oriented coding structure, (c) unsuitable coding structure.

single frames at a given instant. There is always a trade-off between coding efficiency with long SUs and precision in selecting frames using this model. Anyhow, the tolerable maximum length of the SUs depends on the context and the application.

In some cases, the complexity and variety of tools make difficult to define appropriate summarization units, mainly because temporal prediction dependencies extend along the entire bitstream. In those cases, the proposed model is not applicable and partial or full transcoding would be necessary to generate the summary. Figure 3.6b shows a coding structure with a single intracoded frame at the beginning of the bitstream. It is impossible to decode any of the last frames without having decoded the chain of I and P frames used as reference, only being able to define a single SU covering the entire bitstream. Such structure is not suitable for the proposed summarization approach. Although that is the extreme case, very long coding structures are sometimes used in streaming or broadcasting applications because the bitstream is not going to be browsed. Few random access points (e.g. IDR access units) are provided only for error and network resilience, but the SUs may be too long to build suitable summaries. Even if regular intracoded frames are provided, in H.264/AVC it is possible to use coding structures with frames bypassing I frames and using previous frames as references (I frames do not flush the decoding frame buffer), as shown in Figure 3.6c. Such coding structures are not suitable either.

## 3.5. Summary description

### 3.5.1. Conventional model for transcoding

In order to emphasize the difference between transcoding and extraction, we first introduce a simple model to describe summaries and to guide a transcoding-based generation stage.

Using a transcoder in the generation stage, the result of the analysis stage is a description of the summary in terms of the input frames. For a sequence with  $N$  frames, any summary can be described with the following binary function:

$$include(n) = \begin{cases} 1 & n \in \text{summary} \\ 0 & \text{otherwise} \end{cases} \quad n = 0, 1, \dots, N-1 \quad (3.3)$$

The transcoder decodes all the frames and for each of them decides either to include or to discard it according to  $include(n)$ . Every frame is independent of other frames so the resulting sequence of frames is always valid and it can be encoded using a suitable format, even different from the input format.

### 3.5.2. Summarization constraint

Besides the concept of summarization unit, we also define the *summarization constraint* as a function  $tlevel(m) : I_V \rightarrow \{-1, 0, \dots, T\}$  with  $m \in I_V$ . The summarization constraint is the description of the summary in this model, guiding the extraction process. For each index  $m$ , the constraint indicates the scale of  $U_m$  that must be included in the summary, or if  $U_m$  must not be included at all (when the value is -1).

Finally, we define (*bitstream*) *extraction*  $\mathcal{E}(V; I_{S'})$  as the operation in which the summarization units from the input sequence  $V$  are selected and combined according to the AU indexes  $I_{S'}$  to form the summary  $S$ . For convenience, bitstream extraction can be reformulated using SUs and the summarization constraint  $tlevel(m)$  as

$$S = \mathcal{E}(V; tlevel(m)) = (\tilde{U}_0, \tilde{U}_1 \setminus \tilde{U}_0, \dots, \tilde{U}_m \setminus \tilde{U}_{m-1}, \dots, \tilde{U}_{M-1} \setminus \tilde{U}_{M-2}) \quad (3.4)$$

where  $\tilde{U}_m$  is the adapted SU and  $\setminus$  is the set difference operation. The notation  $\tilde{U}_m \setminus \tilde{U}_{m-1}$  means that for the index  $m$ , all the AUs in  $\tilde{U}_m$  must be included in the summary except those that were included previously in  $\tilde{U}_{m-1}$ , in order to avoid duplicated AUs in overlapped SUs. The adapted summarization unit  $\tilde{U}_m$  is obtained as

$$\tilde{U}_m = \tilde{U}_m(tlevel(m); U_m) = \begin{cases} U_m^{tlevel(m)} & tlevel(m) \geq 0 \\ \emptyset & tlevel(m) = -1 \end{cases} \quad (3.5)$$

Note that (3.4) provides a method to generate summaries, taking advantage of the hierarchical arrangement into temporal levels. The process of extraction guided by the summarization constraint is depicted in Figure 3.7. The potential summaries that can be generated with this approach are a subset of the potential summaries generated using arbitrary frames, as the former are constrained by the coding structure. Nevertheless, this constraint is not very important in practical applications using common hierarchical structures with reasonable SU length and the analysis designed conveniently. Designing the analysis stage properly and adapted to this model helps to avoid possible artifacts. For instance, the analysis stage can detect shot boundaries in order to avoid the inclusion of those SUs having frames from different shots, which can lead to

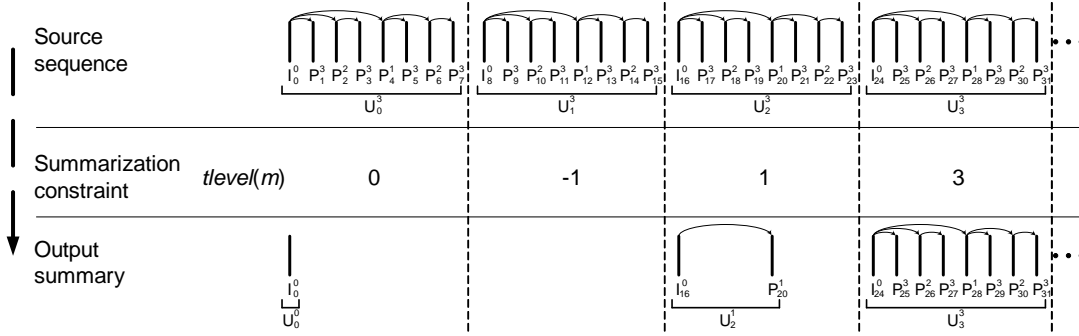


Figure 3.7: Bitstream adaptation guided by summarization constraint.

artifacts in some cases.

In the case of non scalable structures, a virtual hierarchy can be defined using the intracoded frames as level 0 and the whole coding unit as level 1. In the case of joint use of I, P and B frames, a three level hierarchy can be defined, using the P frames as intermediate level (as in Figure 3.1a).

### 3.5.3. Modalities of video summaries

There are different video summarization modalities that can be easily adapted to the proposed model. Depending on the values that  $tlevel(m)$  takes for the SUs, we distinguish several modalities of video summaries (see Figure 3.8):

- *Storyboard*: built by selecting a few independent and separated frames to represent the content in few images. Within the proposed model, for convenience, we restrict the potential selected frames to be I frames (belonging to the lowest temporal level). We also assume that the lower temporal resolution has only one I frame. There is no noticeable difference in practical applications, and actually most storyboard summarization algorithms use temporal subsampling to speed up the analysis. With this assumptions, the storyboard is characterized as follows

$$tlevel(m) = \begin{cases} 0 & \text{keyframe in } U_m \\ -1 & \text{otherwise} \end{cases} \quad (3.6)$$

- *Video skim*: the adapted sequence is shorter than the input sequence, obtained by selecting certain segments of the input sequence. In this case, the valid options for each SU are either not constraining its temporal level or skipping it. Thus, if the maximum temporal level is  $T$  the video skim can be characterized as follows

$$tlevel(m) = \begin{cases} T & U_m \in skim \\ -1 & \text{otherwise} \end{cases} \quad (3.7)$$

- *Fast forward*: this modality is based on the acceleration and deceleration of the sequence,

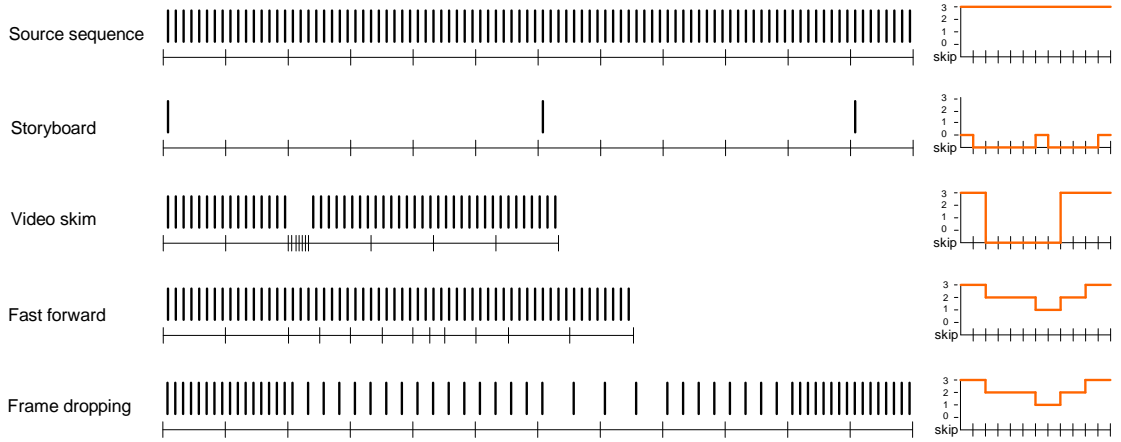


Figure 3.8: Summarization curves (left) and frame selection (right) for different types of video summarization.

driven by a certain content based criterion, in order to visualize it in a shorter time. In this case, the number of frames of each SU is variable depending on the required frame rate at each SU. Thus, there are no constraints on the values of  $tlevel(m)$ . Frames are presented with a constant frame rate, with a constant interval between frames  $\tau_m = \tau_{forward}$ , usually the same as in the source sequence.

- *Frame dropping*: similarly to the previous modality, the number of frames of each SU varies according to some analysis of the semantic or perceptual relevance of the frames, dropping those considered less relevant than the others, in order to accommodate the bitstream to constrained environments. There are no constraints on the values of  $tlevel(m)$ , but the duration of the sequence is preserved, following the timing of the source sequence. The interval between frames varies with the temporal level  $tlevel(m)$  selected, keeping the duration of each SU constant. In the common case of dyadic structures the interval between frames can be computed as

$$\tau_m = \tau_{source} 2^{T-t} \quad (3.8)$$

Note that the last one is not strictly a summary, as the duration of the sequence is preserved. However, content based frame dropping is used frequently for adaptation purposes and can be included easily in the proposed model, as a fast playback with modified timing.

## 3.6. Generation of the summary

### 3.6.1. Architecture based on transcoding

Transcoding is frequently used in (non content-based) bitstream adaptation to constrained bitrate conditions. Figure 3.9 shows a conventional summarization architecture using a transcoder for the generation stage. A transcoder can be easily obtained from the cascade of a decoder and an encoder. This is the approach used in most of video summarization systems, because

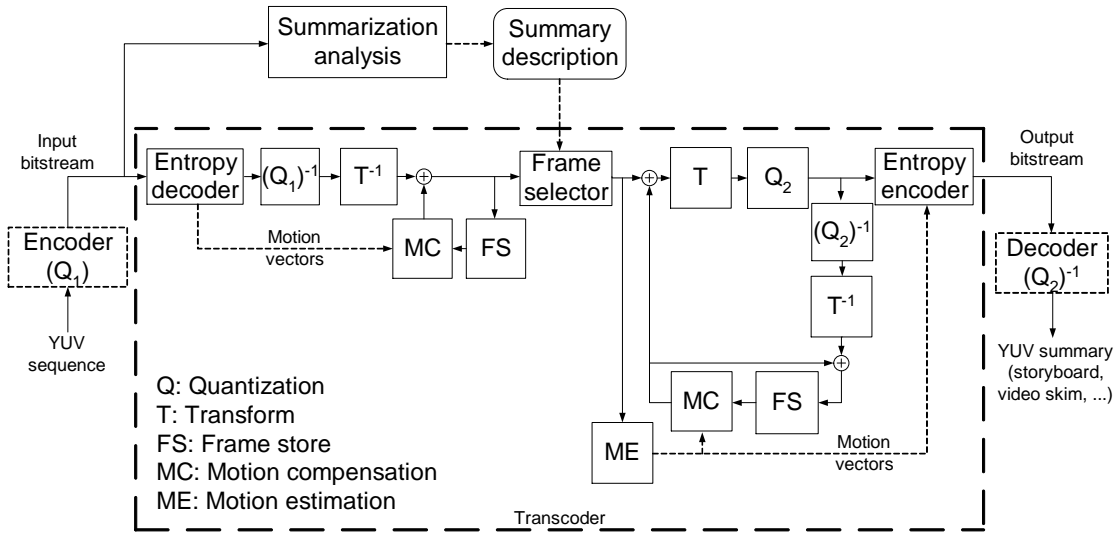


Figure 3.9: Summarization architecture using transcoding (decoder-encoder cascade).

of its straightforward implementation from general-purpose decoders and encoders. Besides, it is simple to separate the analysis from the generation, as the analysis stage usually creates a summary description referred to uncompressed frames as basic units. However, the information in the coded bitstream (e.g. motion vectors, coding modes, ...) could be still exploited for faster analysis.

The transcoder shown in Figure 3.9 has an architecture with all the stages of a conventional decoder (entropy decoding, dequantization, inverse transform and motion compensation), and a conventional closed-loop encoder (motion estimation and compensation, transform, quantization and entropy coding). The link between them is the frame selector, which is also the entry point for summarization. After a frame with index  $n$  is decoded, the frame selector discards it if  $include(n) = 0$  according to the summary description. Thus, only frames belonging to the summary are encoded into the summary bitstream.

Summary generation using transcoding is very inefficient, specially for long summaries, such as video skims. The complexity of the coding format also influences the complexity of the generation (e.g. H.264/AVC is usually much more complex than MPEG-2). The generation delay depends on these factors and the number of frames to be included in the summary. The most demanding part of the whole process is motion estimation. However, limited search ranges or simplified search algorithms, used to speed up encoding, lead to a degradation of the rate-distortion performance. Additionally, many transcoding architectures have been proposed in this context[Xin et al., 2005; Lefol et al., 2007; De Cock et al., 2007], trading off rate-distortion performance and efficiency.

Besides inefficiency, transcoding suffers from an inherent drawback related to the additional quantization ( $Q_2$ ) introduced by the transcoder. A first loss of information occurred before the transcoding, when the input sequence was lossy encoded with a first quantization ( $Q_1$ ). When comparing transcoding architectures, the decoder-encoder cascade with full range search is the

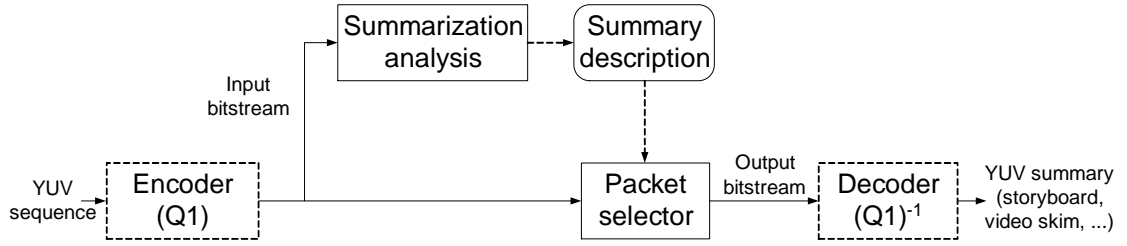


Figure 3.10: Summarization architecture using extraction.

optimal architecture which gives the best end to end rate-distortion performance, and it is used as reference in most transcoding comparisons[Xin et al., 2005; De Cock et al., 2007; Lefol et al., 2007].

### 3.6.2. Architecture based on extraction

If the coding format is the same for both input and output bitstreams, an alternative approach for the generation of the bitstream of the summary is bitstream extraction (see Figure 3.10). Similar to the extraction approach used in scalable bitstream adaptation guided by context constraints, the extraction for summarization is guided by the summary description. The whole transcoder of Figure 3.9 is replaced by an extractor which basically consists of a packet selector. The packet selector selects only those packets containing the required frames, i.e. AUs in H.264/AVC, and discards those not required.

This approach has two inherent advantages. Extraction is a very simple operation which requires few resources and that can be done very efficiently. Besides, the frames themselves are not modified, so the quality of each frame is the same as in the input bitstream. Particularly, compared to transcoding, there is no quality degradation due to an additional quantization stage.

However, as discussed in Section 3.4, the use of extraction is not always possible if the coding structure does not satisfy some requirements. If extraction is not possible, transcoding is required to generate the summary.

Bitstream extraction consists basically of the copy of chunks of the input bitstream to the output bitstream. However, the output bitstream may be non-decodable by a compliant decoder, due to mismatches between the expected and the actual decoding status. It must be emphasized that the original bitstream was encoded using prediction dependencies according to the frames encoded previously. When the bitstream of the summary is decoded, all the syntax elements are referred to the decoding status of the original bitstream, which is different from the actual decoding status, as some parts were removed. For this reason, the extractor must also ensure that the decoding status, and particularly the DPB, is valid, fixing headers as appropriate, so the bitstream can be decoded correctly. In the following, we describe two mechanisms to obtain a decodable bitstream, in the context of H.264/AVC.



Figure 3.11: Low level extraction process with IDR access units (no header updating).

### 3.6.2.1. Extraction using IDR access units

In H.264/AVC, the DPB stores decoded frames that may be used as references by following AUs. The indexes of the references signalled in the header are referred to the current status of the DPB, according to the decoding process, that includes and discards frames from the buffer dynamically. In general, discarding some SUs introduces discontinuities in the status of the DPB.

Using an IDR access unit as the first access unit of a SU is the simplest mechanism to obtain a valid bitstream that can be decoded correctly. The IDR access unit flushes the DPB so the decoding of each SU begins with an empty DPB. The numbering of frames for referencing purposes (e.g. syntax element `frame_num`) must be consistent with the new sequence of AUs. With this mechanism, the numbering is also reset at the beginning of each SU, so the sequence can be decoded properly starting from any SU. Thus, the status of the decoder for a given AU is the same for both the source and summary bitstreams. The use of IDR access units in SUs leads to non-overlapped SU.

The process of extraction in this case is depicted in Figure 3.11. It shows an example of IDR based SUs of length 4, with AUs shown in coding order. Firstly, the parameter sets are copied without any modification to the output bitstream. After that, NAL units encoding AUs belonging to the summary are included, while the rest are discarded. In the example, the summary begins with  $IDR_{64}^0$ , which is the first AU included. After that AU, the rest of the AUs are included, conforming a valid bitstream. Note that no modification is done to any NAL unit, and thus the extraction process is extremely simple in this case.

### 3.6.2.2. Extraction using I access units

In contrast to the previous case, I access units do not reset the status of the decoder. The DPB still contains previous frames, some of them marked as reference, and the numbering is not reset. For this reason, I access units do not ensure that the status of the decoder at a given AU be the same for both the source and summary bitstreams. Therefore, headers must be checked and updated in order to correct the discontinuities in the decoding status due to those AUs removed from the bitstream.

Whenever a gap is found in the decoding process, in order to preserve a valid status of the DPB, the extractor updates the header of NAL units (see Figure 3.12 for an example of the

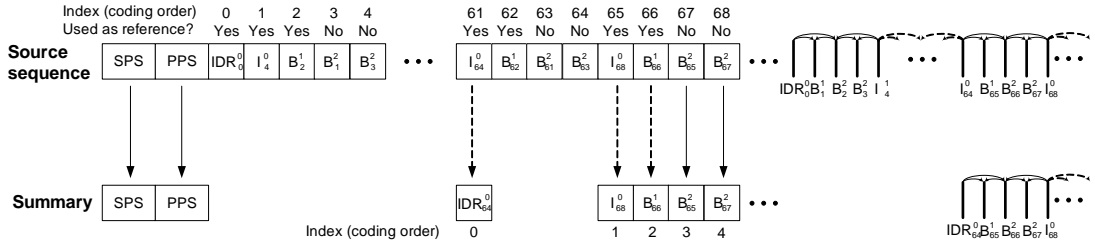


Figure 3.12: Low level extraction process with I access units (header updating).

extraction process and Figure 3.13 for header modifications) in two cases:

- Transformation of I access units to IDR access units. In H.264/AVC, the first AU must be an IDR access unit. If the first AU is removed from the bitstream, it will not be compliant anymore. Therefore, the very first I access unit must be converted to an IDR access unit, which implies a major update of the header, changing the values of `nal_reference_idc`, `nal_unit_type`, and removing some syntax elements while including others (see Figure 3.13). The value of `frame_num` may be also required to be updated, according to the numbering mechanism specified in the standard [ITU-T and ISO/IEC, 2003a]. Apart from the first frame, whenever a new gap appears, the first AU after the gap must be also converted to an IDR access unit.
- Updating of MMCO commands. MMCO commands control how the references are managed in the multiframe buffer. These commands are coded in the header and are referred to the current status of the buffer. A reference frame in the multiframe buffer is identified by the value of `picNumX`, which is derived differentially from the current frame (picture) number as [ITU-T and ISO/IEC, 2003a]:

$$picNumX = CurrPicNum - (difference\_of\_pic\_nums\_minus1 + 1) \quad (3.9)$$

where `difference_of_pic_nums_minus1` is the value of the syntax element `difference_of_pic_nums_minus1` of the MMCO command and `CurrPicNum` is the value of the syntax element `frame_num` of the current frame. Both MMCO commands and frame numbers must be fixed, updating the value of `memory_management_control_operation` and the value of `difference_of_pic_nums_minus1`.

In the example shown in Figure 3.12, the SU consists of a four frame dyadic coding unit  $I^0 B^2 B^1 B^2 (I^0)$ , which is an open GOP (overlapped SUs). Both  $I^0$  and  $B^1$  are used as references, while the two  $B^2$  are not. According to the results of the analysis stage, the first frame of the summary (a video skim in this example) is  $I_{64}^0$ . As the SU is overlapped, it requires also  $I_{68}^0$  to decode the rest of the frames, so these two frames must be included first. As shown in Figure 3.13, the header of  $I_{64}^0$  is modified to convert the frame to an IDR access unit,  $IDR_{64}^0$ . As the frame  $B_{62}^1$ , used as reference in the source bitstream, is not included in the summary bitstream, the value

## CHAPTER 3. GENERATION OF VIDEO SUMMARIES BY BITSTREAM EXTRACTION

NALU len 19, nal_ref_idc 3, nal_unit_type 7 (SPS)	NALU len 19, nal_ref_idc 3, nal_unit_type 7 (SPS)
NALU len 4, nal_ref_idc 3, nal_unit_type 8 (PPS)	NALU len 4, nal_ref_idc 3, nal_unit_type 8 (PPS)
NALU len 1024, nal_ref_idc 3, nal_unit_type 5 (IDR)	
NALU len 1109, nal_ref_idc 2, nal_unit_type 1 (I)	
.....	
NALU len 29985, nal_ref_idc 2, nal_unit_type 1 (I)	NALU len 29984, nal_ref_idc 3, nal_unit_type 5 (IDR)
first_mb_in_slice 0	first_mb_in_slice 0
slice_type 7	slice_type 7
pic_parameter_set_id 0	pic_parameter_set_id 0
frame_num 31	<b>frame_num 32</b>
	<b>idr_pic_id 0</b>
pic_order_cnt_lsb 128	pic_order_cnt_lsb 128
	<b>no_output_of_prior_pics_flag 1</b>
	<b>long_term_reference_flag 0</b>
<del>adaptive_ref_pic_buffering_flag 1</del>	slice_qp_delta 2
<del>memory_management_control_operation 1</del>	
<del>difference_of_pic_nums_minus1 0</del>	
<del>memory_management_control_operation 0</del>	
slice_qp_delta 2	
NALU len 7429, nal_ref_idc 2, nal_unit_type 1 (B)	
NALU len 6600, nal_ref_idc 0, nal_unit_type 1 (B)	
NALU len 7607, nal_ref_idc 0, nal_unit_type 1 (B)	
NALU len 28168, nal_ref_idc 2, nal_unit_type 1 (I)	NALU len 28169, nal_ref_idc 2, nal_unit_type 1 (I)
first_mb_in_slice 0	first_mb_in_slice 0
slice_type 7	slice_type 7
pic_parameter_set_id 0	pic_parameter_set_id 0
frame_num 33	frame_num 33
pic_order_cnt_lsb 136	pic_order_cnt_lsb 136
adaptive_ref_pic_buffering_flag 1	<b>adaptive_ref_pic_buffering_flag 0</b>
<del>memory_management_control_operation 1</del>	
<del>difference_of_pic_nums_minus1 0</del>	
<del>memory_management_control_operation 0</del>	
slice_qp_delta 2	slice_qp_delta 2
NALU len 9238, nal_ref_idc 2, nal_unit_type 1 (B)	NALU len 9239, nal_ref_idc 2, nal_unit_type 1 (B)
first_mb_in_slice 0	first_mb_in_slice 0
slice_type 6	slice_type 6
pic_parameter_set_id 0	pic_parameter_set_id 0
frame_num 34	frame_num 34
pic_order_cnt_lsb 132	pic_order_cnt_lsb 132
direct_spatial_mv_pred_flag 1	direct_spatial_mv_pred_flag 1
num_ref_idx_override_flag 1	num_ref_idx_override_flag 1
num_ref_idx_l0_active_minus1 0	num_ref_idx_l0_active_minus1 0
num_ref_idx_l1_active_minus1 0	num_ref_idx_l1_active_minus1 0
ref_pic_list_reordering_flag_l0 0	ref_pic_list_reordering_flag_l0 0
ref_pic_list_reordering_flag_l1 0	ref_pic_list_reordering_flag_l1 0
adaptive_ref_pic_buffering_flag 1	adaptive_ref_pic_buffering_flag 1
memory_management_control_operation 1	memory_management_control_operation 1
difference_of_pic_nums_minus1 2	<b>difference_of_pic_nums_minus1 1</b>
memory_management_control_operation 0	memory_management_control_operation 0
slice_qp_delta 3	slice_qp_delta 3
NALU len 10217, nal_ref_idc 0, nal_unit_type 1 (B)	NALU len 10217, nal_ref_idc 0, nal_unit_type 1 (B)
NALU len 4691, nal_ref_idc 0, nal_unit_type 1 (B)	NALU len 4691, nal_ref_idc 0, nal_unit_type 1 (B)
NALU len 28697, nal_ref_idc 2, nal_unit_type 1 (I)	NALU len 28697, nal_ref_idc 2, nal_unit_type 1 (I)
.....	.....

Figure 3.13: Example of modification of headers in the extraction process. Left: original bitstream, right: summary bitstream

of  $\text{frame\_num}$  of  $\text{IDR}_{64}^0$  must be updated from 31 to 32 to preserve the continuity of the numbering of references. The MMCO command is also removed from  $\text{I}_{68}^0$ , as it is no longer required because the frame it is referred to is no longer in the buffer. The header of  $\text{B}_{66}^1$  is also updated to obtain the correct  $\text{picNumX}$  according to (3.9) ( $\text{difference\_of\_pic\_nums\_minus1}$  is changed from 2 to 1). The status of the frames used as references in the multiframe buffer for both the source and summary bitstreams are shown in Table 3.1, along with the MMCO commands used. Note the differences between the status of the decoder for each AU in both cases. While containing the same visual data, headers must be referred to the each particular decoding status.

The resulting bitstream is correctly decoded by the JM reference decoder [Tourapis et al., 2007]. However, headers may be further modified if required, for example, to number summary frames starting from 0, or to change dynamically the frame rate.

Frame (frame_num)	Reference	diff	MMCO command	References in DPB
$B_{59}^2(31)$	No			$I_{60}^0(29) B_{58}^1(30)$
$I_{64}^0(31)$	Yes	0	$B_{58}^1(30=31-(0+1))$ marked as “unused for ref”	$I_{60}^0(29) I_{64}^0(31)$
$B_{62}^1(32)$	Yes	2	$I_{60}^0(29=32-(2+1))$ marked as “unused for ref”	$I_{64}^0(31) B_{62}^1(32)$
$B_{61}^2(33)$	No			$I_{64}^0(31) B_{62}^1(32)$
$B_{63}^2(33)$	No			$I_{64}^0(31) B_{62}^1(32)$
$I_{68}^0(33)$	Yes	0	$B_{62}^1(32=33-(0+1))$ marked as “unused for ref”	$I_{64}^0(31) I_{68}^0(33)$
$B_{66}^1(34)$	Yes	2	$I_{64}^0(31=34-(2+1))$ marked as “unused for ref”	$I_{68}^0(33) B_{66}^1(34)$
$B_{65}^2(35)$	No			$I_{68}^0(33) B_{66}^1(34)$
$I_{72}^0(35)$	Yes	0	$B_{58}^1(34=35-(0+1))$ marked as “unused for ref”	$I_{68}^0(33) I_{72}^0(35)$
$B_{70}^1(36)$	Yes	2	$I_{68}^0(33=36-(2+1))$ marked as “unused for ref”	$I_{72}^0(35) B_{70}^1(36)$

(a)

Frame (frame_num)	Reference	diff	MMCO command	References in DPB
$IDR_{64}^0(32)$	Yes		(Flush buffer)	$IDR_{64}^0(32)$
$I_{68}^0(33)$	Yes			$IDR_{64}^0(32) I_{68}^0(33)$
$B_{66}^1(34)$	Yes	1	$IDR_{64}^0(32=34-(1+1))$ marked as “unused for ref”	$I_{68}^0(33) B_{66}^1(34)$
$B_{65}^2(35)$	No			$I_{68}^0(33) B_{66}^1(34)$
$B_{67}^2(35)$	No			$I_{68}^0(33) B_{66}^1(34)$
$I_{72}^0(35)$	Yes	0	$B_{58}^1(34=35-(0+1))$ marked as “unused for ref”	$I_{68}^0(33) I_{72}^0(35)$
$B_{70}^1(36)$	Yes	2	$I_{68}^0(33=36-(2+1))$ marked as “unused for ref”	$I_{72}^0(35) B_{70}^1(36)$

(b)

Table 3.1: Detail of the DPB management with I access units: (a) input bitstream, (b) summary bitstream.

### 3.6.2.3. Presentation schemes

The way the frames in a summary are presented in the terminal can be modified in order to obtain a different effect. For instance, it can be used to control the delay between slides if the set of frames in the summary is presented as a slideshow. The difference between a content-based fast forward and content-based frame dropping is also the presentation scheme. Thus, the possibility of adjusting the presentation of the frames is also a desirable feature and it should be also considered in the generation of the summary.

We consider two presentation schemes: constant frame rate and variable frame rate. In a constant frame rate presentation, each frame is presented after the previous one with the same constant delay. On the other hand, if the frame rate varies throughout the sequence, the delay between frames also varies, and must be signalled in the bitstream. Constant frame rate does not require any special processing, unless specifying a different frame rate, if necessary, at the beginning of the summarized bitstream.

Although variable frame rate can be managed at the system layer, we propose two methods to achieve such variable frame rate behaviour at the coding layer (see Figure 3.14):

1. Signalling the changes of frame rate. A SPS is required to be sent before each frame with a frame rate different from the previous frame. The value of `num_units_in_tick` and `time_scale` are set according to the frame rate. Note that only a single SPS, modified dynamically, is necessary and the identifier `seq_parameter_set_id` in the slice header is not changed. This method can fit to a larger variation of frame rates.
2. Using a predefined set of SPSs. If the possible frame rates are restricted to a fixed number of possibilities  $K$ , it would be more useful to define  $K$  SPSs with different

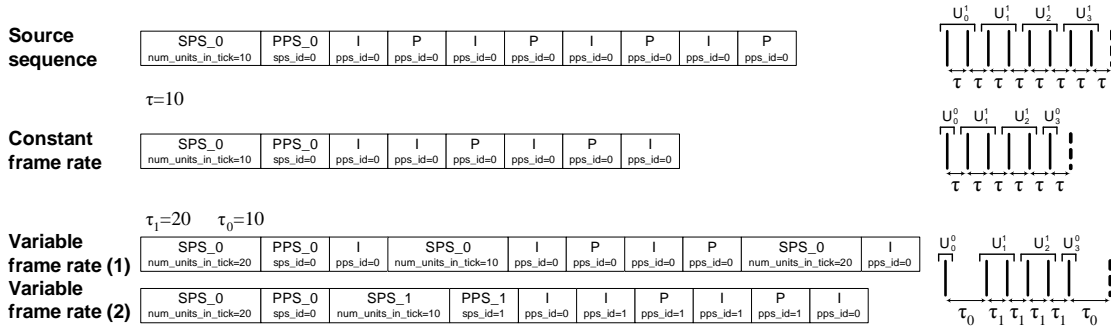


Figure 3.14: Constant and variable frame rate presentation schemes.

values of `num_units_in_tick` and `time_scale` and include them at the beginning of the bitstream, with `seq_parameter_set_id` varying from 0 to  $K - 1$ . It will be necessary to send  $K$  PPSs, each of them referring to one of the SPSs using `pic_parameter_set_id=seq_parameter_set_id` from 0 to  $K-1$ . In each header slice, the value of `pic_parameter_set_id` needs to be modified to use the corresponding SPS via the PPS.

### 3.7. Experimental evaluation

The proposed generation framework was tested in several experiments [Herranz and Martínez, 2009c, 2010b], including efficiency, subjective evaluations and rate-distortion performance.

#### 3.7.1. Summarization algorithms

In order to study properly the generation framework, we used some algorithms as analysis stage for testing purposes. We describe some simple and widely used summarization techniques, which have been adapted to fit into the proposed framework and representation model. It must be noted that the analysis itself is outside of the scope of this chapter, which is focused on the representation and generation of the summaries. More complex and specific content-based analysis algorithms could be used to obtain better summaries in terms of semantic coverage.

The algorithms were tested with the sequence *Sun-Earth Connection*, a video downloaded from the Open Video Project's repository [Marchionini et al., 2006] in MPEG-2 format, and transcoded to H.264/AVC using the JM 12.4 reference implementation [Tourapis et al., 2007]. The sequence has 11593 frames of 720x480 pixels at 30 frames per second. In order to study the influence of the GOP length and frame type, the sequence was encoded with different hierarchical structures, which differ in the GOP length (from 1 to 32 frames) and in the use of either P or B access units. In the experiments we assume a base layer with only IDR access units, so each GOP corresponds to a SU. In the case of B access units, I access units are used, in order to have overlapped SUs.

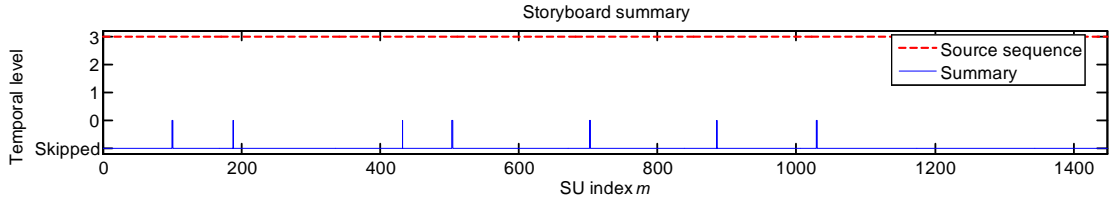


Figure 3.15: Summarization constraint in the case of a 7 keyframes storyboard with a SU length of 8 frames.



Figure 3.16: Examples of storyboards with 4, 7 and 16 keyframes.

#### 3.7.1.1. Image storyboard

Storyboards are represented by few independent frames, trying to cover the semantics in the sequence. For this reason, a widely used criterion is the distance between frames in some feature space. Clustering algorithms have been successfully used to group similar frames into clusters. Then, a few of them are selected as representative of the clusters [Mundur et al., 2006; Zhuang et al., 1998] to build the storyboard summary. In order to test the proposed framework in this case, a simple clustering approach is used, based on the  $K$ -means algorithm. Each frame is decoded in the YUV color space and divided into  $2 \times 2$  subimages in order to have some information about the spatial distribution of colours what can not be done with only one global feature vector. Each subimage is represented with a 32-bin histogram for the Y component and two 8-bin histograms for the U and V components. Each frame is then represented by a 192-D feature vector. The feature vectors from all the frames are clustered using the  $K$ -means algorithm with the Euclidean distance, resulting in  $K$  centroids. For each centroid, the closest frame from its cluster is selected as keyframe. Temporal subsampling is often used in summarization to reduce the computational burden without degrading the quality of the summaries [Mundur et al., 2006], due to the redundancies between consecutive frames. This temporal subsampling can be achieved in the summarization model processing only frames at the lowest temporal resolution of the bitstream ( $\mathbf{U}_m^0$ ). Besides, selecting the same temporal level in analysis and in adaptation prevents from problems derived from the lack of exact temporal localization in the model (e.g. when shot changes occur in a SU). The output of the algorithm for 10 clusters corresponds to the summarization constraint shown in Figure 3.15.

The resulting summaries for different values of  $K$  are shown in Figure 3.16.

### 3.7.1.2. Video skim

One of the most interesting summarization applications is the generation of video skims, simply selecting video segments according to some semantic criteria. Depending on the application, the selection technique can vary, for instance, from video trailers with the most active parts of the sequence to video skims with the parts with a given person. In this experiment, we assume that the segments with more semantic relevance are those with a person speaking. This criterion may be useful in many domains, such as broadcast video programs[Peker et al., 2006].

The face detection method of Viola and Jones[Viola and Jones, 2001] is used to mark the frames with at least one face detected (see Figure 3.17a). The minimum size of the face is set to 88x72 pixels. If the number of frames with a face detected in a SU is greater than the number of frames without any detection, the SU is marked to be included in the video skim. In order to cope with false detection and to reduce undesirable short segments or gaps, a moving median filter is used to obtain a smoother curve. In the experiments we used a window of 11 frames for the median filter. Then, the highest level of each selected SU is included in the skim. Figure 3.17b represents the summarization constraint obtained for 16 frames per GOP. As we use a dyadic decomposition, it is possible to build a summary with the same frames using a submultiple of 32 as GOP length, which is useful to compare the same summary generated with different GOP lengths. For example, Figure 3.17c shows the equivalent summarization curve of Figure 3.17b for 2 frames per GOP.

### 3.7.1.3. Fast forward

The test algorithm for semantic fast forwards is based on the method proposed in [Herranz, 2007]. In this method, activity is used as semantic clue guiding the playback of the sequence. The temporal levels are selected according to this assumption in order to approach to the target frame rate. Skipping is also used to achieve a lower virtual frame rate along several SUs.

A widely used measure of activity is the MPEG-7 intensity of motion activity descriptor[Jeannin and Divakaran, 2001], which has been successfully used in indexing, fast browsing[Peker et al., 2001] and video rate control[Lotfallah et al., 2006]. The basic assumption in [Herranz, 2007] is that the instantaneous frame rate in the output sequence should be proportional to the measure of activity. A similar approach is used in other works in the context of scalable video coding[Bescós et al., 2007; Mrak et al., 2009]. We use a variant of the MPEG-7 intensity of motion activity without quantization (in order to have the flexibility provided by the use of a continuous range), and adapted to the variable block size motion vectors of H.264/AVC. An advantage is that it can be computed directly in the compressed domain with minimum cost, avoiding most of the decoding process. Besides, as the activity is computed in a SU basis, we use only frames of the first enhancement level ( $P^1$  or  $B^1$ , depending on the coding structure), without having to process the rest of the frames of the SU. Figure 3.17 shows some of the summarization constraints for this case.

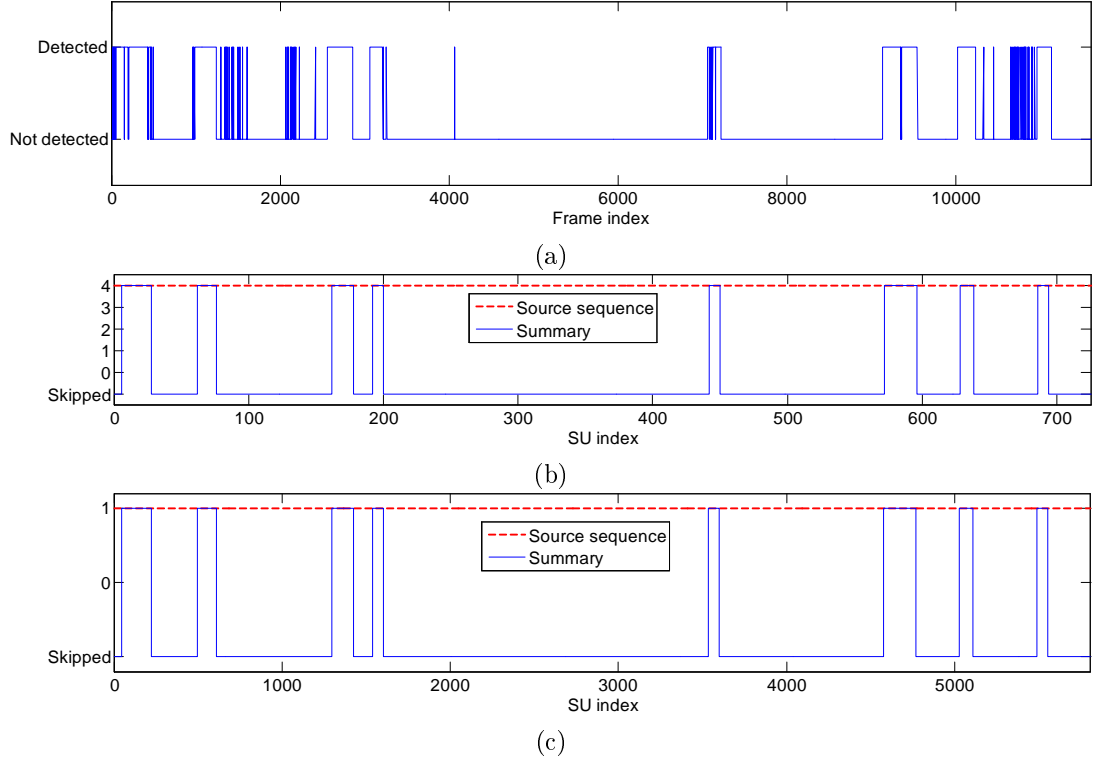


Figure 3.17: Details of the results for video skim: (a) face detection, (b) summarization constraint (16 frames per GOP) and (c) summarization constraint (2 frames per GOP).

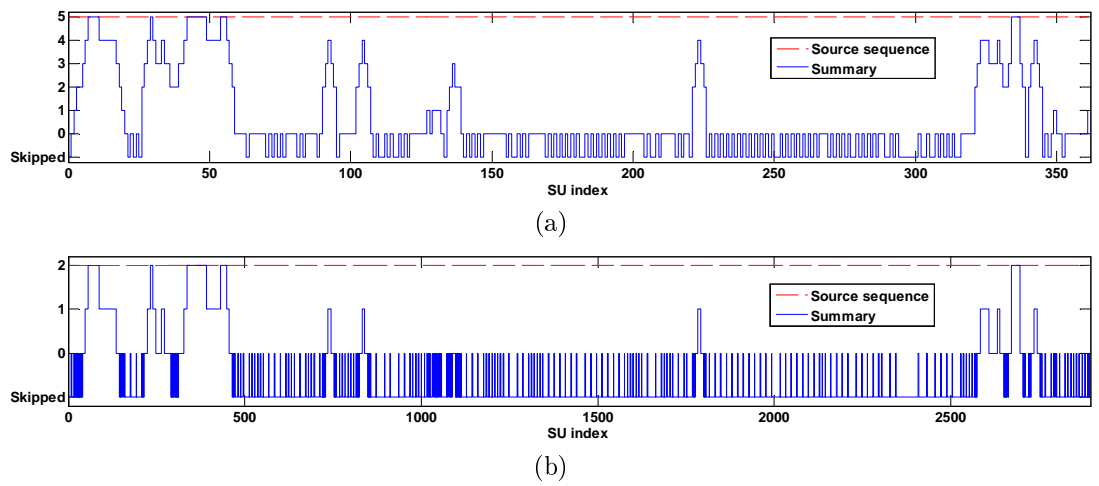


Figure 3.18: Details of the results for fast forward: (a) summarization constraint (32 frames per GOP) and (b) summarization constraint (4 frames per GOP).

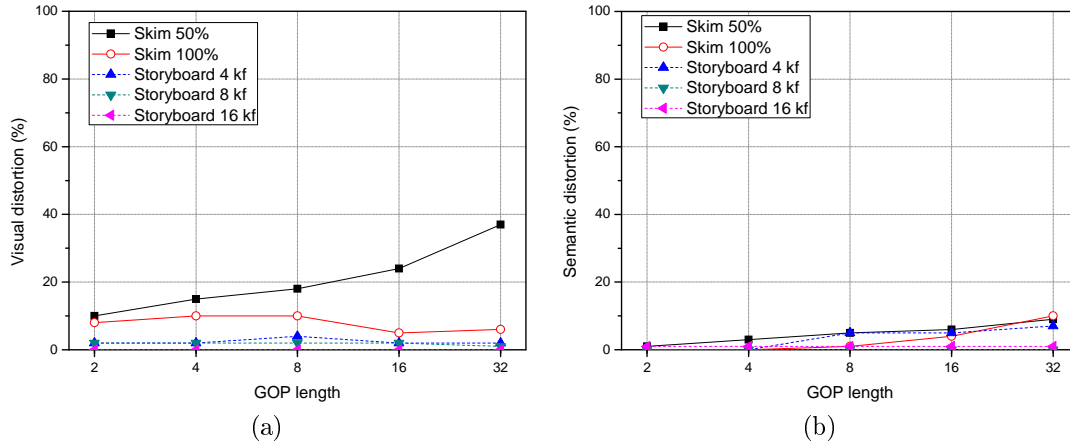


Figure 3.19: Subjective evaluation results: a) visual distortion, b) semantic distortion.

### 3.7.2. Subjective evaluation

In the proposed model, frames are selected in groups rather than individually, which leads to a lack of accuracy in the selection of arbitrary frames when the length of the SU increases. In this first experiment we studied this effect and how it is perceived. The experiment tries to quantify the effect over the skims and storyboards obtained from the test sequence. Firstly, the sequence is analyzed with frame precision and a set of frames to be included is obtained at the finest scale (i.e. GOP length of 1 frame). For each GOP length, a different summarization constraint is obtained by subsampling the previous one. For storyboards, the closest I frame of the corresponding SU at a given scale is selected. For skims, in a first approach (skim 50% in Figure 3.19), a SU is included at a given scale if half of the frames belonging to it are included in the summary at the finest scale.

Ten people were asked to assess the distortion that they perceived for the sequence *Sun-Earth Connection*. The summary obtained with frame precision is used as reference (no distortion). Two criteria were used: visual distortion, which measures if the subsampled summary is visually similar to the reference one and if annoying artifacts are included in it; and semantic distortion, which measures if the subsampled summary is equivalent to the reference one and if important information is lost by the effect of the subsampling.

In the case of storyboard, almost every assessor agrees that no significant distortion is perceived for both visual and semantic points of view. In the case of skims, there is no semantic distortion perceived, as most of the information of the summary is still present in the summaries with coarser scales. However, visual distortion increases with GOP length, being important at 32 frames per GOP. The approach used for subsampling has the drawback of including new frames at the boundaries of previous segments, which leads to temporal artifacts, mainly when some frames from adjacent shots are included. However, this problem can be lessened if the approach used in analysis is designed carefully with the generation model in mind, in order to avoid in advance the inclusion of problematic SU (for example, SU including shot boundaries). In the experiment we also used a slightly different approach for subsampling, selecting a SU at a scale

only if all the frames are included at the finest scale (skim 100% in Figure 3.19). Now the problem is that some frames can be lost at the boundaries, but we avoided the problem of including new frames at the boundaries. Figure 3.19 shows that this alternative subsampling approach can reduce significantly the visual distortion, and only a small semantic distortion is perceived (note that audio is not considered, and a different approach would be probably necessary in that case).

### 3.7.3. Efficiency

The main advantage of the proposed approach is its efficiency, which depends on the coding structure, and particularly on the GOP length. A longer GOP may help to improve coding efficiency, and the size of the bitstream may be reduced. That has impact on the performance of the generation of the summary. This section provides some experimental measures in order to assess the performance of the system in terms of processing time.

In order to have a comparison with a coding scheme not using the hierarchical prediction structures of H.264/AVC, we have also implemented the generation of video skims and storyboards for MPEG-2 coded sequences. For storyboards only the selected I frames are preserved in the output bitstream, and for video skims complete GOPs are preserved. In order to compare the systems under the same conditions, the test sequence was reencoded in MPEG-2 with the same GOP lengths as for H.264/AVC and for two GOP structures: one with only P frames (IPPP...) and another with P and B frames (IBPB...). The summaries were generated using the same summarization constraints as those used for H.264/AVC.

For each coding format, the test sequence was encoded with the same configuration in both cases (quantization parameters, motion estimation parameters, etc.), except that one structure uses P frames and the other uses B frames. Figure 3.20 shows the mean bitrate of the sequence coded using different GOP lengths and compared to the same sequences coded with MPEG-2. As expected, the size of the bitstream drops as the GOP length increases, although, for this sequence, it does not decrease for GOPs larger than 8 frames with H.264/AVC. It must be noted that the original sequence was already lossy encoded in MPEG-2 when obtained, so it is not very appropriate for rate-distortion comparisons (e.g. PSNR).

In the experiment, the time required for the generation of the bitstream is compared to that required using a conventional approach with a transcoder generating the same frames<sup>1</sup>. The transcoders used in these experiments consist of simple cascades of decoder, extractor of frames (in uncompressed YUV) and encoder. The encoder uses the same configuration used for the encoding of the input bitstream. However, a comparison of different approaches in terms of processing time is highly dependent on the specific implementation of a codec and its degree of optimization. Table 3.2 provides complementary information to the experimental curves, describing briefly some key aspects of the implementation of each module used in the experiments.

In contrast to the previous subjective experiments, we must only compare the generation of

<sup>1</sup>Experiments performed in an Intel Core 2 at 2.83 Ghz (2 GB of RAM)

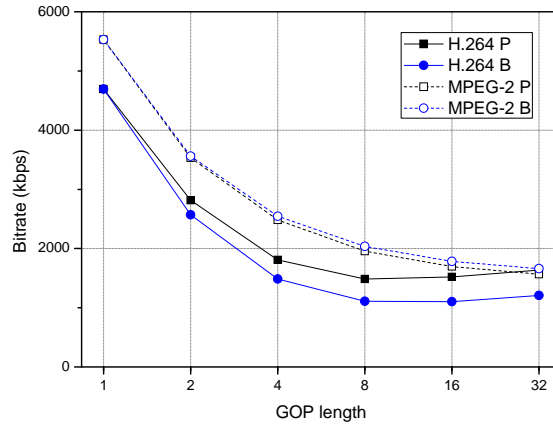


Figure 3.20: Comparison of the coded sequences.

Module	Based on	Optimization	Uses bitstream description
Extractor H.264	Some parts of JM 12.4	Medium	Yes
Transcoder H.264	Decod+Encod JM 12.4	Very low	-
Extractor MPEG-2	-	Medium	No
Transcoder MPEG-2	FFmpeg	Medium-high	-

Table 3.2: Details of the software implementations used in the experiments.

the summarized bitstream, independently of the analysis and the GOP length. We performed the analysis at the coarsest scale (32 frames per GOP) and then reconstructed the equivalent summarization constraints to scales with higher accuracy (see Figure 3.15b and c and Figure 3.17b and c). Thus, the summary is fixed and it includes the same frames in all cases. Figure 3.21 shows the results for video skim, storyboard, fast playback and frame dropping. In all of them, bitstream extraction performs significantly faster than transcoding for both H.264/AVC and MPEG-2, with a factor between 50 and 1000 times. The use of P or B frames does not affect significantly the performance in the tests.

As expected, summaries with more frames, such as skims, require more extraction time than storyboards, with fewer frames. In the case of storyboards, H.264/AVC extraction performs better than MPEG-2 extraction. This fact is due to the use of bitstream descriptions by the H.264/AVC extractor, which provides effective information about the localization and boundaries of the packets in the input bitstream. The bitstream description is generated by the encoder and stored along with the bitstream. Thus, most of the header parsing is avoided. In contrast, the MPEG-2 extractor does not use any extra information so it needs to parse each header in order to detect the boundaries of each packet.

Although the generation process is specified for each SU (or GOP), the basic unit of the bitstream is the NAL unit. In this sense, the generation time is approximately independent of the GOP length, as the number of NAL units does not vary significantly (although the relative amount of NAL units with I slices and NAL units with P/B slices does). However, the curves in Figure 3.21b show that the processing time decreases as the GOP length increases. This fact

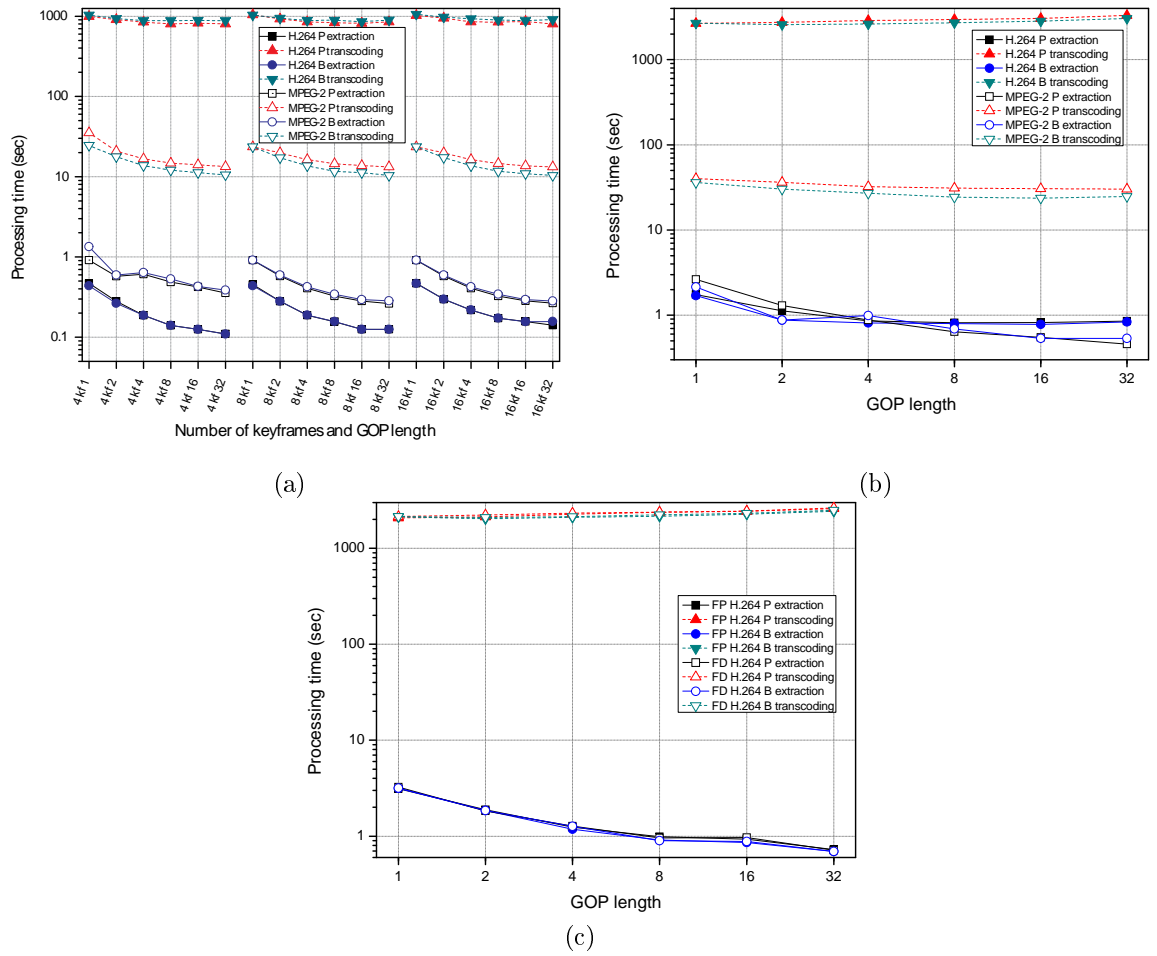


Figure 3.21: Processing time: (a) storyboard, (b) video skim, (c) fast forward and frame dropping.

is explained considering that the storage and parsing of the summarization constraint by the extractors were not optimized for these experiments and parsing consisted of reading plain text files. Note that for shorter GOPs these text files have more values than for longer GOPs, and thus with this implementation the parsing time of these files becomes more dominant as the GOP length decreases. In the case of fast forward and frame dropping (see Figure 3.21c) there are no significant differences, as both lead to almost the same bitstream, differing only in the few additional packets of frame dropping.

The performance of the implementations used in these tests can be further improved for both transcoding and extraction (especially for H.264/AVC transcoding), and thus the processing time can be further reduced. However, bitstream extraction provides a simple way of adaptation which is intrinsically faster than transcoding, and in the same conditions, should outperform transcoding in terms of processing time.

#### 3.7.4. Rate-distortion performance

The second main advantage of the extraction framework compared to transcoding is the absence of requantization. For this reason, a better rate-distortion performance is expected. In this set of experiments, we compared experimentally the rate-distortion performance of transcoding and extraction approaches, in the context of H.264/AVC with hierarchical B-frames.

The optimal transcoding architecture in terms of rate-distortion performance is the cascade of decoder and decoder with full range search. However, that is computationally very intensive in practice. With this architecture, quality and efficiency can be traded off via the motion estimation strategy. Five variations were tested, depending on the algorithm (full search or EPZS) and search window size (64, 8 or 0 pixels): FULL64, FULL8, EPZS64, EPZS8 and ZERO (only zero vectors are evaluated). Due to the large number of possible summaries that can be obtained from a given sequence, for these experiments we consider only, without any loss of generality for rate distortion measures, the generation of a summary including all the frames, which is equivalent to the original sequence.

Rate-distortion must be measured using appropriate test sequences. We encoded the sequence *stefan* (300 frames of 352x288 pixels -CIF-) and the sequence *foreman* (300 frames of 352x288 pixels -CIF- and 176x144 pixels -QCIF-) with the JM 12.4 encoder with full search (64 pixel window size) and different values of GOP length and quantization parameter. The coding structure was a dyadic structure with B frames and IDR access units. For each test, the transcoder uses the same GOP length and the same quantization parameter as the encoder, as the purpose of transcoding in this work is the generation of the summary and not bitrate adaptation.

Figure 3.22 and Figure 3.23 show the rate-distortion curves for all the approaches and the impact of quantization parameter and GOP length. As expected, extraction outperforms transcoding, as the quality is not degraded by an additional quantization stage. Besides, as the GOP length increases, the degradation of the transcoding approach is more significant, suggesting that requantization affects more to motion predicted frames, and the quantization error is propagated

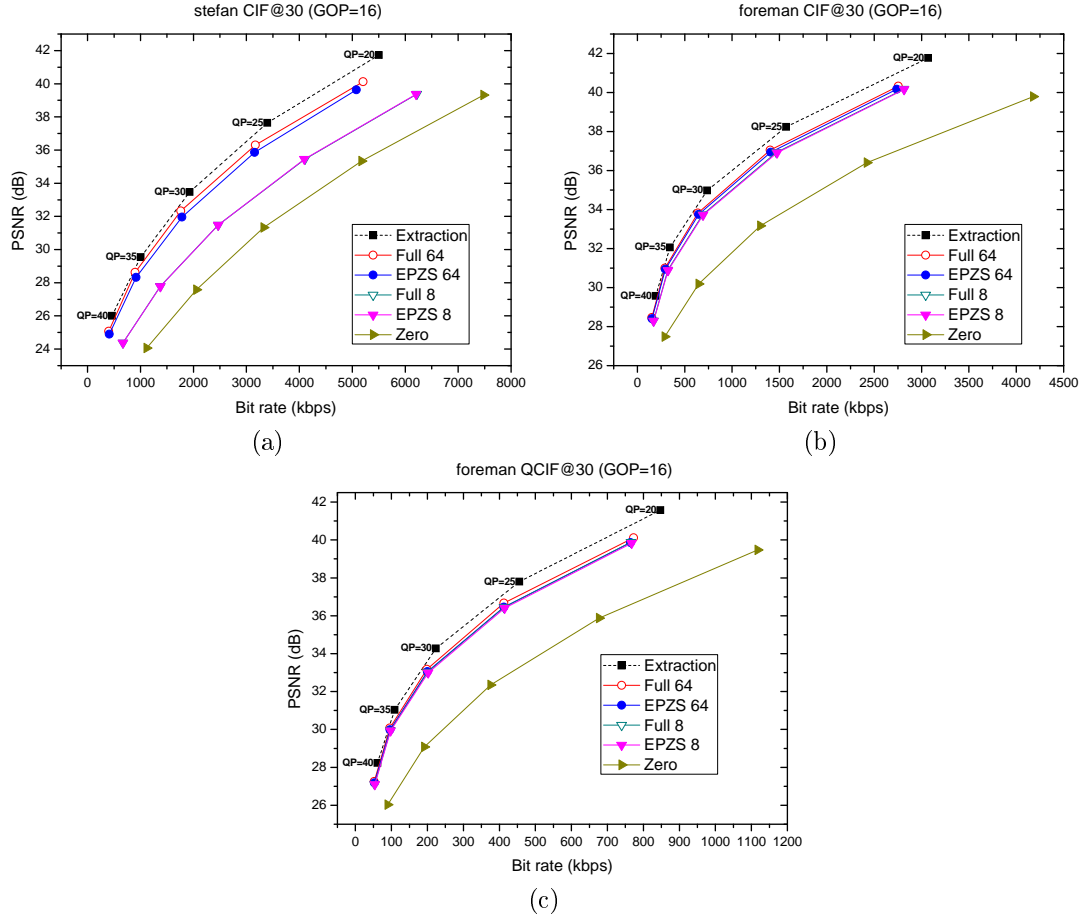


Figure 3.22: Comparison of average PSNR for extraction and transcoding with different values of QP and sequences: (a) *stefan* CIF, (b) *foreman* CIF, and (c) *foreman* QCIF.

and accumulated in other intercoded frames. In the transcoding experiments, FULL64 has the best quality, with EPZS64 close to it. The other configurations degrade very fast as the GOP length increases. However, even using only intracoded frames (i.e. GOP=1 in Figure 3.23), in which no motion estimation is used, the quality is still notably better in the extraction approach.

The effect of motion estimation is better shown in Figure 3.24. Although the transcoder uses a closed-loop drift-free architecture, a progressive loss of quality within the GOPs is evident in the plots. In this example the degradation propagates backwards (the intracoded frame is the last frame in the GOP) until a new intracoded frame is found. Obviously, this degradation is higher for longer GOPs, and the average PSNR decreases, as shown in Figure 3.23. For extraction, the absence of requantization avoids this problem, and the quality loss is due only to the source encoder.

#### 3.7.4.1. Comparison with other architectures

In general, efficiency and quality are traded off in transcoding architectures[Xin et al., 2005; Lefol et al., 2007; De Cock et al., 2007]. In order to have a better comparison, we also measured

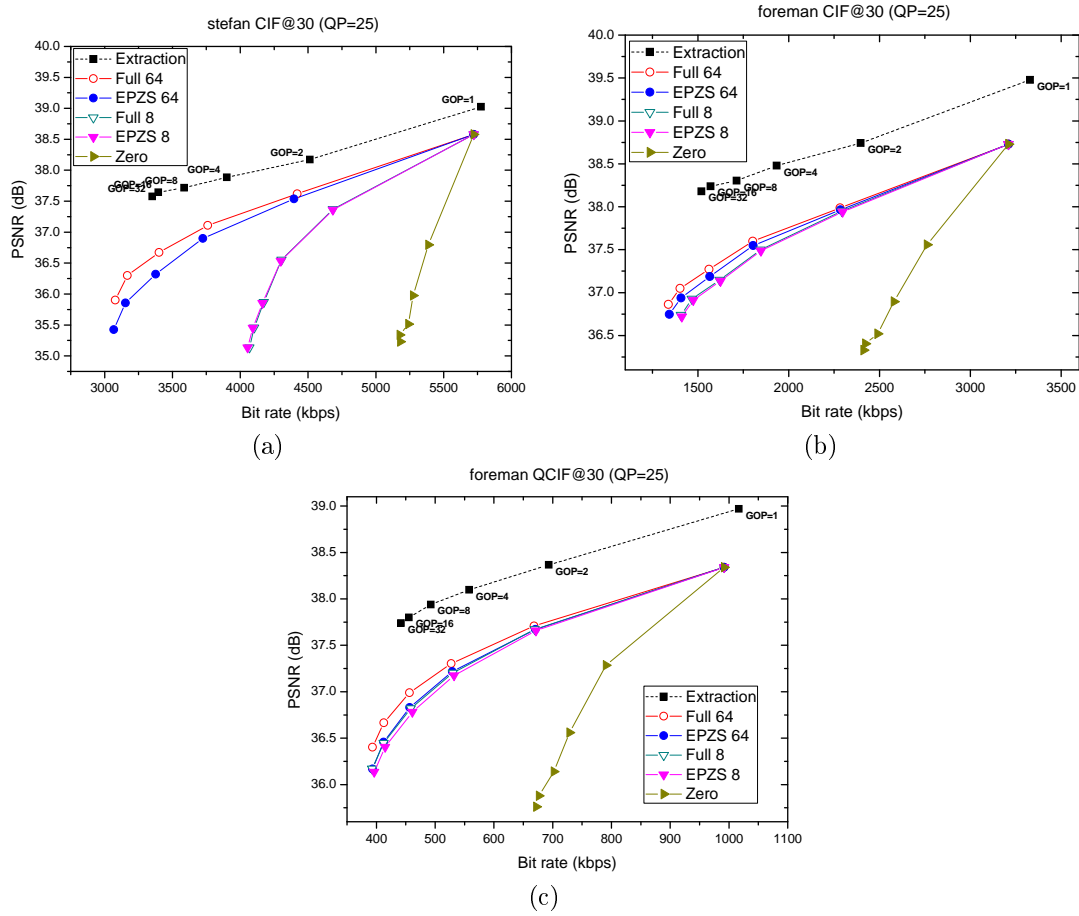


Figure 3.23: Comparison of average PSNR for extraction and transcoding with different GOP lengths and sequences: (a) *stefan* CIF, (b) *foreman* CIF, and (c) *foreman* QCIF.

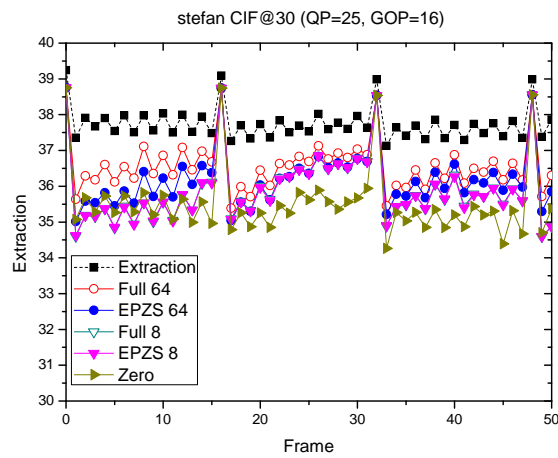


Figure 3.24: Comparison of average PSNR per frame of sequence *stefan* CIF.

the efficiency of each of the experiments. These measures are complementary to those shown in Section 3.7.3. Figure 3.25 shows the average frames per second obtained in the case of the processing of the whole sequences. Although simplified architectures and optimized implementations can greatly improve the performance of transcoding, extraction seems to remain as the best option when high efficiency in the generation is required.

Extraction works mainly as a selective packet forwarding operation, and the only factor having some noticeable influence in the performance is the bitrate. The larger the bitstream, the more time required to copy the packets. An inverse linear trend is observed, with slower processing as packets become larger.

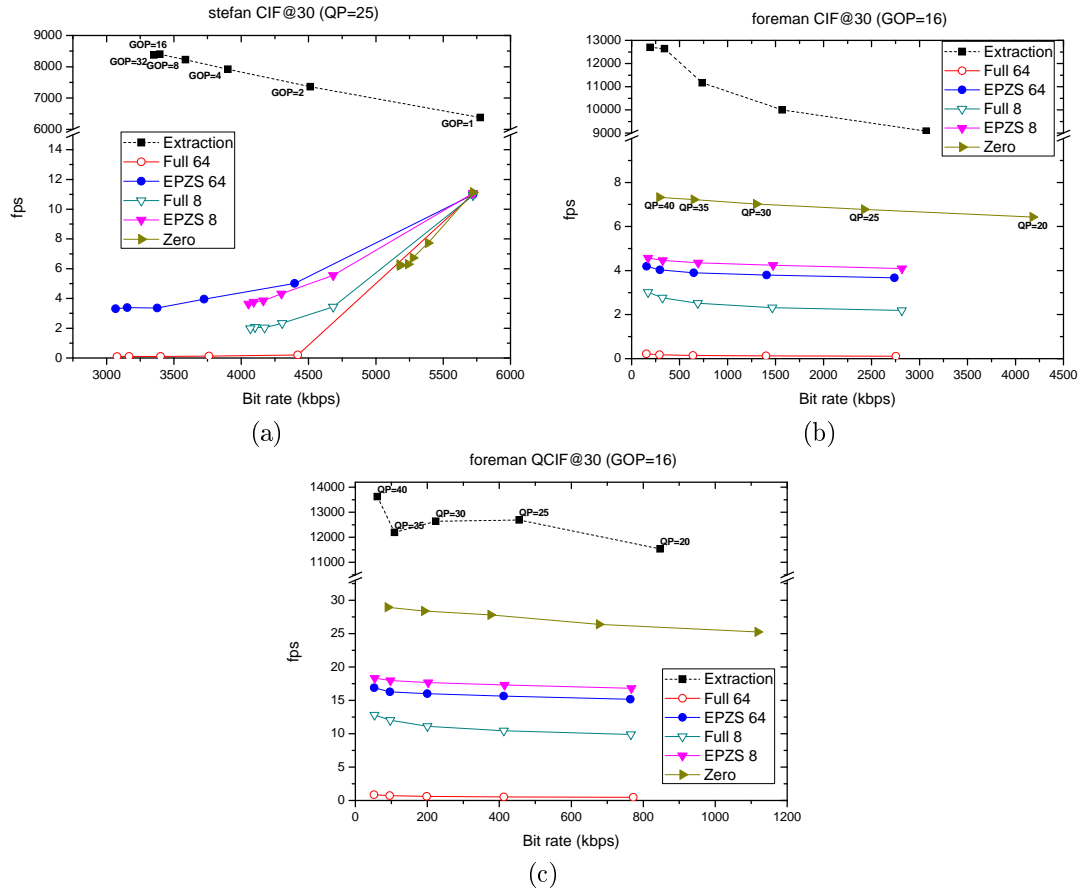


Figure 3.25: Comparison of run time of extraction and transcoding: (a) *stefan* CIF (different GOP lengths), (b) *foreman* CIF (different QP), and (c) *foreman* QCIF (different QP).

The efficiency of transcoding is highly related to the motion estimation strategy, ranging from 15% of the total transcoding time with ZERO to 98% with FULL64, in the worst cases. EPZS and small search areas speed up the transcoding, although still below real time processing. In contrast to extraction, transcoding is faster for shorter GOPs, as motion estimation is used in fewer frames, and particularly fast when only intracoding is used.

Experiments show better results for extraction than transcoding using a decoder-encoder cascade. Although this transcoder is not the most suitable for many applications due to its

high complexity, it is the most used because its straightforward implementation, and it also provides a useful reference to compare other architectures with. Particularly, the cascade with full search range provides the optimal rate-distortion performance for transcoding. For the sequence used in the experiments, FULL64 uses a search range large enough to be considered a close approximation.

Open-loop architectures such as requantization[Xin et al., 2005; De Cock et al., 2007] are computationally efficient, since they operate directly on the transformed coefficients, requantizing them with a different quantization parameter. However, they suffer from a progressive quality degradation (i.e. drift), due to the mismatch between the predictions used in decoding and encoding, which cannot be rectified without a closed loop architecture. A cascaded pixel-domain transcoder (CPDT) consists of a concatenation of a decoder and a simplified decoder, similar to Figure 3.9, which reuses motion vectors and other information extracted from the input bitstream. Avoiding motion estimation, CPDT is more efficient than the decoder-encoder cascade, with an efficiency comparable to ZERO (slightly better, as in ZERO motion estimation still consumes the 15% of processing time in the worst case) but with a quality significantly lower than the decoder-encoder cascade for H.264/AVC bitstreams[Lefol et al., 2007], due the large number of new coding tools introduced by this standard. Recently, in the context of H.264/AVC coding, [Lefol et al., 2006] proposed a Mixed Requantization Architecture (MRA) which performs 35% faster than CPDT but with a PSNR 3 dB lower than CPDT. [De Cock et al., 2007] proposed another requantization architecture, improving about 2 dB the MRA.

### 3.8. Summary and conclusions

If the video coding structure satisfies certain conditions for random access, extraction is an interesting alternative to transcoding for the generation of the bitstream in video summarization. In this chapter we have studied the use of bitstream extraction for video summarization. We have introduced some concepts, such as summarization units, and a model for representing video summarization results taking advantage from coding structures and particularly from the hierarchical prediction structures of H.264/AVC. This representation enables the generation of the bitstream using a bitstream extraction framework. This approach has two inherent advantages: efficiency, derived from the simplicity of the adaptation method, and quality preservation, due to the quantization-free architecture in contrast to transcoding, which introduces an additional loss of quality.

The framework was tested with some application examples and used to simulate summaries with some simple analysis algorithms. The proposed model is generic enough and independent of the analysis stage, so it can take advantage from more complex analysis algorithms (other analysis algorithms using this model are described in next chapters).

We compared experimentally both transcoding and extraction frameworks. Different encoding configurations at the transcoder are studied for faster transcoding but with an additional loss in rate-distortion performance. The size of the summarization unit (the GOP in the experiments) is an important parameter that has effect not only on the precision of the summarization

analysis, but also on the rate-distortion performance and efficiency of the generation process.



## Chapter 4

# Integrated summarization and adaptation

This chapter extends the bitstream extraction framework described in the preceding chapter to include adaptation to the usage environment. This adaptation is performed using scalable video coding (specifically MPEG-4 SVC). The summarization-adaptation framework also uses metadata tools from the MPEG-21 standard to describe the characteristics of the terminal and network. The main advantage is the simplicity and efficiency of the process, especially when it is compared to conventional approaches such as transcoding. The framework is evaluated experimentally in terms of efficiency and rate-distortion performance.

The framework described in this chapter uses three types of scalability: spatial and quality for context adaptation, and temporal for both context and semantic adaptation. Most of this chapter is based on the publications: [Herranz, 2007; Herranz and Martínez, 2008b, 2009b].

### 4.1. Motivation

Most works in video adaptation only address the adaptation of the original pieces of content. However, video summaries are also pieces of content (e.g. images, video sequences) that are consumed in the same specific usage conditions (e.g. terminal, network), and should be also adapted to them. Additionally, video summarization has been described as a specific type of adaptation, in which the structure of the content is modified [Chang and Vetro, 2005]. The conventional approach to the generation summaries adapted to the usage environment is to consider summarization and adaptation as two independent stages, in which the summaries are subsequently adapted (e.g. transcoded).

Adaptation of scalable bitstreams are based on lightweight techniques, such as bitstream extraction [Devillers et al., 2005; Panis et al., 2003; Sprljan et al., 2005; Thang et al., 2006; Paridaens et al., 2007], which enable fast and efficient adaptation. As we have shown in the previous chapter, bitstream extraction can be also used for the generation of video summaries.

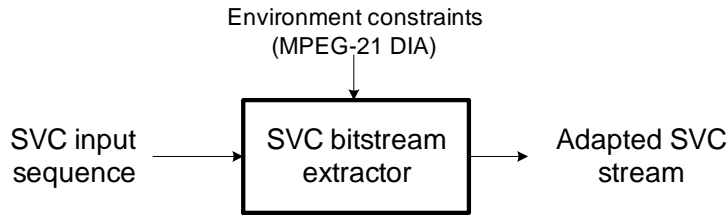


Figure 4.1: Adaptation in the SVC framework.

Combining both, this chapter describes an integrated framework using only bitstream extraction to generate adapted summaries.

## 4.2. Integrated summarization and adaptation framework in MPEG-4 SVC

The advantage of SVC relies on its efficient adaptation scheme. Using SVC, the adaptation engine is a simple module (i.e. bitstream extractor) which modifies the bitstream selecting only the parts required according to some constraints (see Figure 4.1). The constraints (resolution, bitrate, etc.) are imposed by the usage environment. The extractor selects the appropriate layers of the input bitstream satisfying the constraints. The output bitstream is also compliant with the SVC standard so it can be decoded with a suitable SVC decoder.

In the proposed framework, each user is linked at least to one UED description. Each user may use different terminals or networks depending on the situation. The summarization and adaptation engine must know this information in order to deliver an appropriate version of the sequence or the summary.

### 4.2.1. Summarization units in MPEG-4 SVC

In SVC, versions at different spatial and quality resolutions, for a given instant, form AUs. An AU can contain NAL units from both the base and enhancement layers. Each NAL unit belongs to a specific spatial, temporal and quality layer. This information is stored in the header of the NAL unit in the syntax elements *dependency\_id*, *temporal\_id* and *quality\_id*. The length of the NAL unit header in H.264/AVC is extended to include this information. In SVC, the base layer is always H.264/AVC compatible. However, the extended NAL unit header would make the bitstream non compliant with H.264/AVC. For this reason, each base layer NAL unit has a non extended header, but it is preceded by an additional NAL unit containing the SVC related information. These units are called prefix NAL units. If the stream is processed by an H.264/AVC decoder, these prefix NAL units and the other enhancement layer NAL units are simply ignored, and the base layer can still be decoded.

In SVC, the concept of SU, introduced in the previous chapter, can be extended, in order to include the additional versions given by spatial and quality scalabilities. Thus, it is possible to define more SUs, only with NAL units from the base layer, or including NAL units from

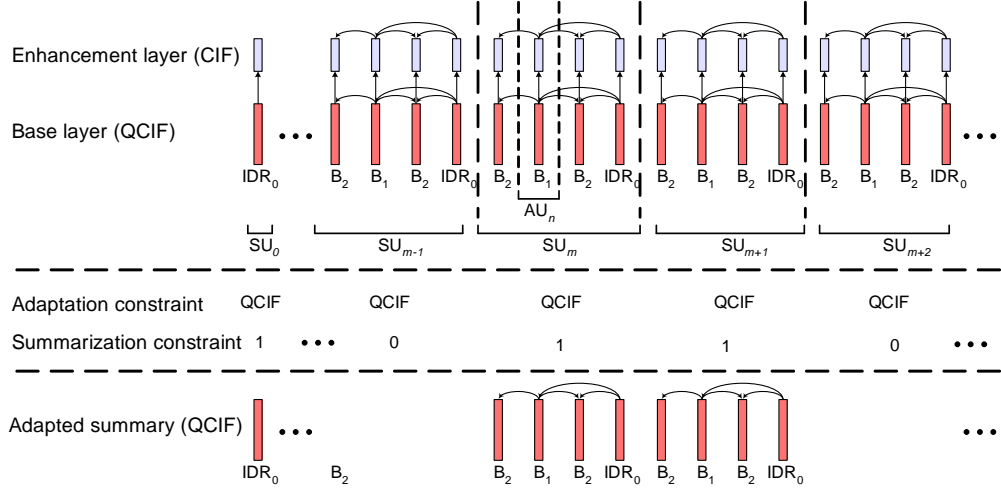


Figure 4.2: Coding structures and summarization units in SVC.

enhancement layers. Thus, versions of each SU are obtained for different spatial resolutions and qualities. Figure 4.2 shows an example of coding structures and SUs in SVC. Discarding the enhancement layer, it is still possible to find more SUs in the base layer.

#### 4.2.2. Extraction process in MPEG-4 SVC

The extraction process in SVC is non-normative, with the only constraint that the output bitstream, obtained from discarding enhancement layers, must be compliant with the SVC standard. The JVT provides the Joint Scalable Video Model (JSVM), including a software implementation of SVC. In this section we briefly describe the basic extraction process in the JSVM.

The extractor processes NAL units using the syntax elements *dependency\_id*, *temporal\_id* and *quality\_id* to decide which ones must be included in the output bitstream. Each adaptation decision is taken for each access unit  $AU_n$ , where  $n$  is the temporal instant. Each layer (base or enhancement) in  $AU_n$  can be denoted as  $L(d, t, q; n)$ . An operation point  $OP_n = (d_n, t_n, q_n)$  is a specific coordinate  $(d, t, q)$  at temporal instant  $n$ , representing a particular resolution (spatial and temporal) and quality, related, respectively, to the syntax elements *dependency\_id*, *temporal\_id* and *quality\_id*. If we denote the extraction process as  $\mathcal{E}(OP, AU)$ , the result of adapting an access unit  $AU_n$  with a particular operation point  $OP_n$  can be defined as the adapted access unit  $\tilde{AU}_n = \mathcal{E}(OP_n, AU_n)$ , containing all the layers and data necessary to decode the sequence at this particular resolution and quality. For each  $AU_n$ , the extractor must find the operation point  $OP_n$  satisfying the constraints and maximizing the utility of the adaptation. In a typical adaptation scenario, the terminal and the network impose constraints that can be fixed (*display\_width*, *display\_height* and *display\_supported\_rate*) or variable (*available\_bits(n)* related to the available network capacity). Thus, the adaptation via bitstream extraction can be formulated as an optimization problem:

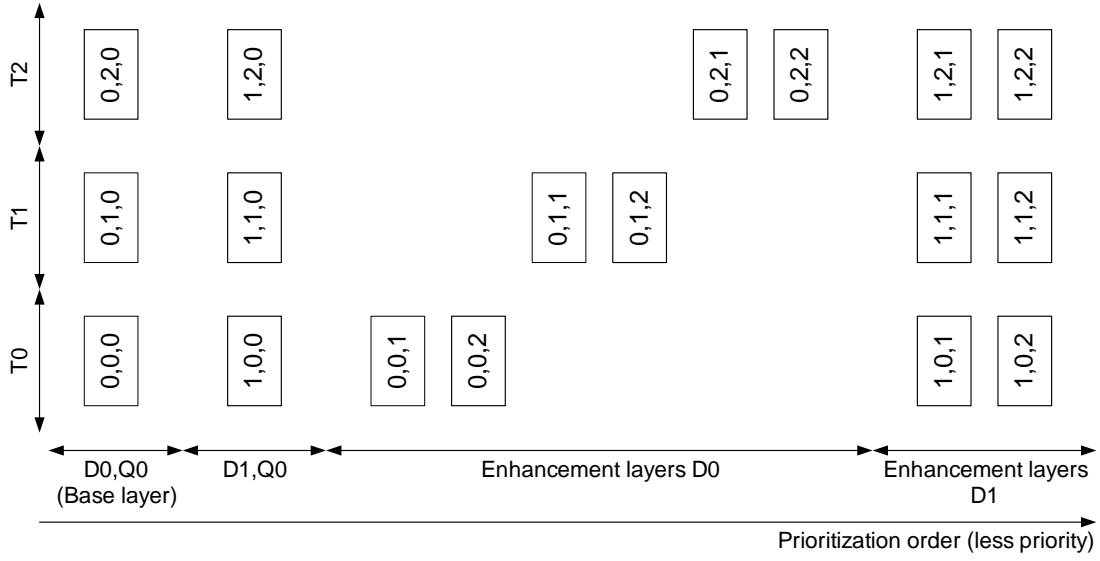


Figure 4.3: Prioritization of NAL units in the JSVM extractor (adapted from [Amonou et al., 2007]).

for each instant  $n$  find  $OP_n^* = (d_n^*, t_n^*, q_n^*)$  maximizing  $utility(A\tilde{U}_n)$   
 subject to

$$\begin{aligned} frame\_width(d_n) &\leq display\_width \\ frame\_height(d_n) &\leq display\_height \\ frame\_rate(t_n) &\leq display\_frame\_rate \\ bitsize(A\tilde{U}_n) &\leq available\_bits(n) \end{aligned}$$

In this formulation,  $utility(A\tilde{U}_n)$  is a generic measure of utility or quality of the resulting adaptation. It should be computed or estimated for all the possible adapted AUs, in order to select the most appropriate. The actual values of resolution and frame rate can be obtained indirectly from  $d$  and  $t$ , and the size of any AU can be obtained just parsing the bitstream.

The JSVM extractor solves the problem using a prioritization approach. The NAL units in an AU are ordered in a predefined order and selected in this order until the target bitrate or size is achieved. In Figure 4.3 each block represents a NAL unit containing a layer  $L(d, t, q; n)$ . The base quality layer ( $q = 0$ ) of each spatial and temporal level are placed first in the priority order. Then, NAL units including quality refinements are placed in increasing order of their temporal level. Spatial enhancement layers are placed next. The extractor just drops those NAL units with a priority lower than the required one.

However, this prioritization scheme does not ensure the optimality of the extraction path in terms of utility. For this reason, besides the basic extraction method, SVC provides additional tools for improved extraction, namely the optional syntax element *priority\_id*, which signals explicitly the priority of each NAL unit, based on any other (non-normative) criteria[Amonou et al., 2007].

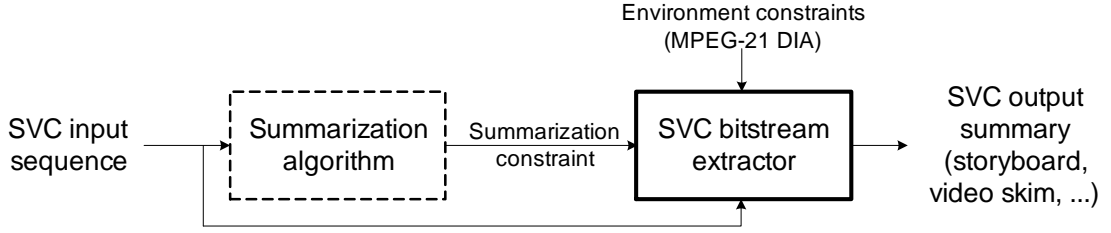


Figure 4.4: Integrated summarization and adaptation of SVC.

### 4.2.3. Including summarization in the framework

The constraints imposed to the adaptation engine are external, due to the presence of a constrained usage environment (*environment constraints*). Adaptation modifies the resolution and quality of the bitstream, but the information in the content itself does not change. However, there is no restriction on the nature of the constraints. As discussed in the previous chapter, summarization can be seen as a modification of the structure of the bitstream, in order to remove semantic redundancies in the temporal axis, in a constrained situation where the number of frames must be reduced considerably. For this reason, we use the model to describe summaries introduced in the preceding chapter. The summarization constraint can modify the value of the temporal resolution. If both environment and summarization constraints are used together in the extraction, the result is an integrated summarization and adaptation engine which can generate summaries adapted to the usage environment using only SVC tools (see Figure 4.4).

The adaptation process, as described previously, is performed on an AU basis. However, in the proposed summarization model, the summaries are referred to the SU index with the summarization constraint  $tlevel(m)$ , so it must be harmonized with the adaptation process. When a sequence is partitioned into SUs, each of them contains one or more AUs and, for simplicity, we assume that each AU belongs only to a single SU. Then we define a new summarization constraint  $\widetilde{tlevel}(n)$  for each  $AU_n$  associated to a certain  $U_m$ :

$$\widetilde{tlevel}(n) \equiv tlevel(m), AU_n \in U_m, \forall n \in \{0, \dots, N-1\} \quad (4.1)$$

The problem of adaptation in the extractor, including the new summarization constraint, can be now expressed as

for each instant  $n$  find  $OP_n^* = (d_n^*, t_n^*, q_n^*)$  maximizing  $utility(\mathcal{E}(OP_n, AU_n))$   
subject to

$$\begin{aligned} frame\_width(d_n) &\leq display\_width \\ frame\_height(d_n) &\leq display\_height \\ frame\_rate(t_n) &\leq display\_frame\_rate \\ bitsize(\mathcal{E}(OP_n, AU_n)) &\leq available\_bits(n) \\ t &\leq \widetilde{tlevel}(n) \end{aligned}$$

The last constraint makes the extraction process content-based, constraining directly the temporal level. The problem can be solved using the same tools described in the previous

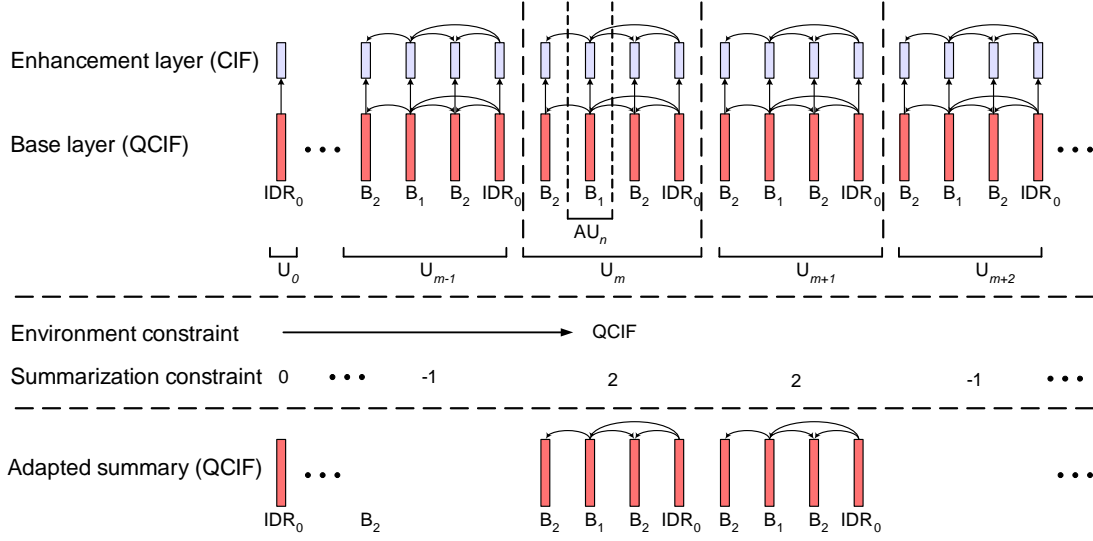


Figure 4.5: Bitstream adaptation guided by summarization and environment constraint.

section, including the prioritization scheme of the JSVM. Implicitly  $d$ ,  $t$  and  $q$  are assumed to be positive (or zero). Thus, if  $\widetilde{tleve}(n)$  takes a negative value for a certain  $n$ , the problem has no solution, as the new summarization constraint cannot be satisfied. In that case, we assume that the extractor will skip that AU not including any of its NAL units in the output bitstream. The summarization algorithm can take advantage of this to signal when a certain SU must not appear in the output bitstream.

As in the model for H.264/AVC, all the SUs must be independently decodable for all the possible adapted versions. Again, the simplest solution is the use of IDR Access Units. In SVC, IDR Access Units only provide random access points for a specific dependency layer. For this reason, enhancement layers must also have an IDR Access Unit at the beginning of each SU, in order to guarantee the independence of the SUs for all layers.

### 4.3. Experimental evaluation

This section describes some experiments to evaluate the main advantage of the framework, which is the efficient generation of the bitstream of adapted summaries. For comparison, we also provide experimental results with an alternative approach based on transcoding.

#### 4.3.1. Test scenario

For these experiments we assume a test scenario with users accessing content via two types of terminals capable of decoding H.264/AVC and SVC: a terminal with a high resolution display in a broadband network, such as a PC or a TV, and a terminal with a medium resolution display in a medium-low capacity network, such as a PDA or mobile phone.

The experiments target both efficiency and rate-distortion measures. However, it is difficult

Layer Number	Spatial resolution	Temporal resolution	Quality resolution (QP)
0	CIF	30 Hz	39
1	CIF	30 Hz	30
2	4CIF	30 Hz	40
3	4CIF	30 Hz	29

Table 4.1: Settings of the layers for SVC encoding

to find test sequences suitable for both purposes simultaneously. On the one hand, video summarization itself and the measure of processing time require sequences with a certain length, in order to create meaningful summaries. On the other hand, evaluation of rate-distortion performance in video coding requires test sequences available in uncompressed formats, such as YUV. These sequences are usually very short sequences with a single shot, being not suitable as test sequences for video summarization. For these reasons, we created a longer test sequence using six commonly used YUV sequences (*city*, *crew*, *harbour*, *ice*, *soccer* and *foreman*) concatenated in a single YUV sequence.

We used the reference software JSVM 9.18 in the simulations. The test sequence (1729 frames at 4CIF and 30 frames per second) was encoded in SVC with 2 spatial levels and 2 quality levels, using MGS for quality scalability. The details of these layers are shown in Table 4.1. Dyadic hierarchical structures were used for temporal scalability with GOP lengths from 1 to 32 frames (1 to 6 temporal levels). In order to compare the approach with a non scalable approach, two additional versions were also encoded in H.264/AVC with the settings of layer 1 (CIF) and layer 3 (4CIF) in Table 4.1.

Given the test scenario, we considered two target conditions to test the performance of the framework:

- *4CIF@30*. Both spatial and temporal resolutions do not change with respect to the original bitstream. Therefore, neither spatial nor temporal adaptation will be required, and only efficiency in the generation of summaries is studied. This is the adaptation path for the PC or TV case.
- *CIF@15*. In this scenario there is adaptation in both spatial and temporal resolutions. Both generation of the summary and adaptation to the target conditions are studied. This is the adaptation path for the PDA or mobile phone case.

The summaries were generated and adapted to the test conditions with the following methods (see Table 4.2):

- *AVC transcoding*. The sequence is first decoded to YUV format. The summary is generated and adapted (if required) into another YUV sequence, which is finally encoded to H.264/AVC. For the case CIF@15 there are two possibilities, depending on which H.264/AVC version is used as input bitstream (4CIF or CIF).
- *SVC extraction*. It uses the SVC bitstream extractor to select the required packets and to generate the adapted summary.

Method (resolution)	Spatial resolution (input/output)	Temporal resolution (input/output)
Adaptation to 4CIF@30		
AVC transcoding (4CIF)	4CIF/4CIF	30/30 Hz
AVC extraction (4CIF)	4CIF/4CIF	30/30 Hz
SVC extraction (4CIF)	4CIF/4CIF	30/30 Hz
SVC extraction (4CIF low)	4CIF/4CIF	30/30 Hz
Adaptation to CIF@15		
AVC transcoding (4CIF)	4CIF/CIF	30/15 Hz
AVC transcoding (CIF)	CIF/CIF	30/15 Hz
AVC extraction (CIF)	CIF/CIF	30/15 Hz
AVC hybrid	4CIF/CIF	30/15 Hz
SVC extraction (CIF)	4CIF/CIF	30/15 Hz
SVC extraction (CIF low)	4CIF/CIF	30/15 Hz

Table 4.2: Methods and cases used in the experiments.

- *AVC extraction.* The same bitstream extractor is used in this case (either from 4CIF version or CIF version). This method can be used only when neither spatial nor quality adaptation are required.
- *AVC hybrid.* This method complements the previous one, as the summary is first generated using extraction from the 4CIF H.264/AVC bitstream, and then it is transcoded to the adapted version of the summary in CIF. Note that, compared to transcoding from the 4CIF version, only a few frames (depending on the length of the summary) are processed, as most of them were discarded during extraction.

For AVC transcoding and AVC hybrid methods, the settings of the encoder were modified to reduce significantly the computational burden due to encoding. Thus, a fast search method was used with a smaller search range (8 pixels).

For the purpose of these experiments, the summarization algorithm itself is out of the scope, and it could be any algorithm. In this case we used a simple method consisting of sampling the sequence at constant intervals, selecting single frames for storyboards, and segments of 64 frames (2 seconds) for video skims. Although extremely simple, this method is very suitable for the test sequence used in the experiment, as the shots have similar lengths and they are distributed regularly in the sequence, so it is very likely that the algorithm creates summaries covering most of the shots.

### 4.3.2. Efficiency

As in the previous chapter, we compared transcoding and extraction for different SU lengths (GOP length in the experiments). Figure 4.6 shows the results for a video skim (20% of the total length) with the different methods tested. In this case we used the processing speed (as the number of frames of the input sequence divided by the processing time), measured in frames per second<sup>1</sup>. As expected, transcoding is much slower than methods based on extraction. Both

<sup>1</sup>Experiments performed in an Intel Xeon at 2.83 Ghz (24 GB of RAM)

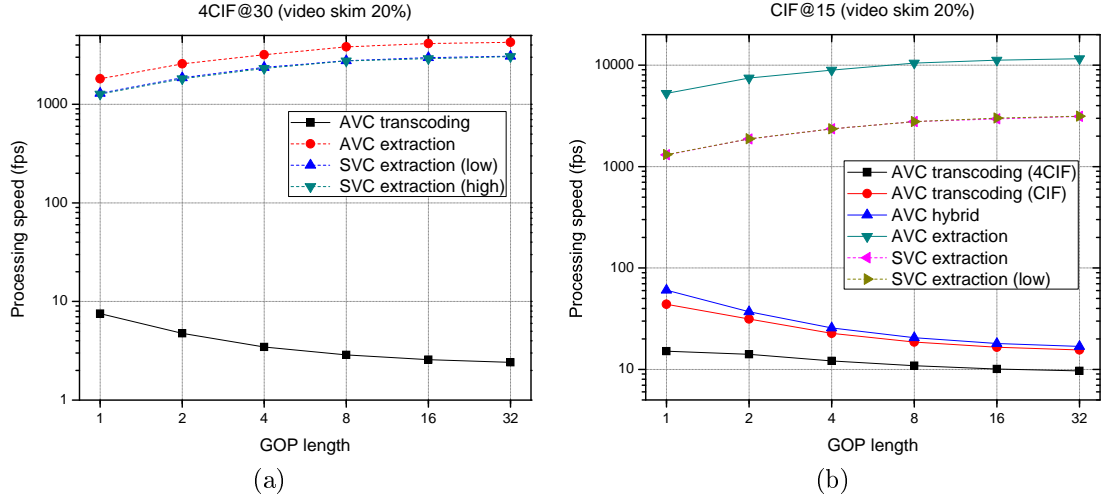


Figure 4.6: Dependency of the processing speed with the GOP length: a) 4CIF 30 Hz and, b) CIF 15 Hz.

SVC extraction and AVC extraction have very good performance, over 1000 frames per second. The latter is faster for both 4CIF and CIF as it needs to parse a lower number of NAL units, due to the absence of enhancement layers and the smaller size of the bitstream.

The length of the coding unit has different effects on transcoding and extraction. In the case of extraction, longer GOPs result on smaller bitstreams, which are processed faster, as extraction basically is a selective packet forwarding operation. On the contrary, encoding using longer GOPs requires more computational effort on motion estimation. Thus, the efficiency of transcoding decreases as the GOP length increases.

In the case of transcoding to CIF there are three possibilities. Transcoding from the CIF version is faster than transcoding from the 4CIF version, due to the faster decoding of lower resolution sequences. The hybrid method combining extraction and transcoding is also faster than pure transcoding from 4CIF. However, their performance is still quite far from that of extraction approaches.

The different methods were also compared for several modalities and summary lengths, ranging from the empty to the whole sequence, using a GOP length of 8 frames. Methods based on extraction also have an almost constant performance for all the summary lengths, slightly degraded for long summaries. Methods based on transcoding are more sensitive to the length of the summary. For short summaries (e.g. storyboards), most of the processing time in transcoding is due to decoding, as encoding complexity was reduced and only a few frames are encoded in contrast to the decoding of all frames. However, a significant increment of the processing time can be observed for longer summaries. The hybrid method reduces the number of frames to be decoded, increasing the performance dramatically for short summaries, although it is still degraded for long summaries due to transcoding.

The generation of a summary with the 0% of the frames in the sequence (an empty summary) is very useful to have a reference of the time used in initialisation and other processes independent

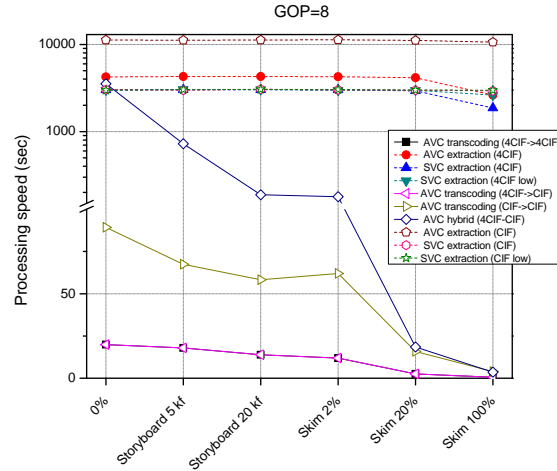


Figure 4.7: Processing speed for different modalities. Note that half of the vertical scale is linear and the rest is logarithmic.

of the length of the summary. In transcoding, this time is due to the decoding of all frames, and it is the most important contribution to the overall processing time. In the case of extraction, the JSVM extractor performs the extraction in two passes. In the first pass, all the NAL headers are parsed in order to obtain a description of the bitstream, which is then used to perform the actual extraction. As it can be seen in the figure, most of the extraction time is used in this first pass. The use of bitstream descriptions (as those used in the AVC extractor described in Chapter 3) reduces significantly the time required for this first pass, which in that case would consist of parsing the bitstream description instead of parsing the whole bitstream.

As experiments showed, a simple solution based on extraction has better performance than others based on transcoding, for the purpose of video summarization and adaptation. A hybrid solution based on both extraction and transcoding can also be useful when no spatial nor quality scalability are available, especially for short summaries such as storyboards.

### 4.3.3. Rate-distortion performance

As shown in the preceding chapter, AVC extraction outperforms AVC transcoding in rate-distortion performance. In general, for a single layer it is always true, as transcoding implies an additional quantization stage. However the multilayered approach of SVC has a penalty in coding efficiency compared to a single layer version. In the case of the experiment, the 4CIF SVC version of the test sequence is encoded predictively from the other 3 versions while the 4CIF AVC version is encoded directly in a single layer, which is more optimal in terms of rate-distortion performance. Although the penalty due to scalability is small in MPEG-4 SVC, each additional layer increments the overall penalty compared to a single layer version. Thus, although SVC extraction avoids the additional quantization stage, its performance was degraded previously compared to the AVC version by the use of multiple scales.

We studied the rate-distortion performance of a video skim (20%) extracted from the original

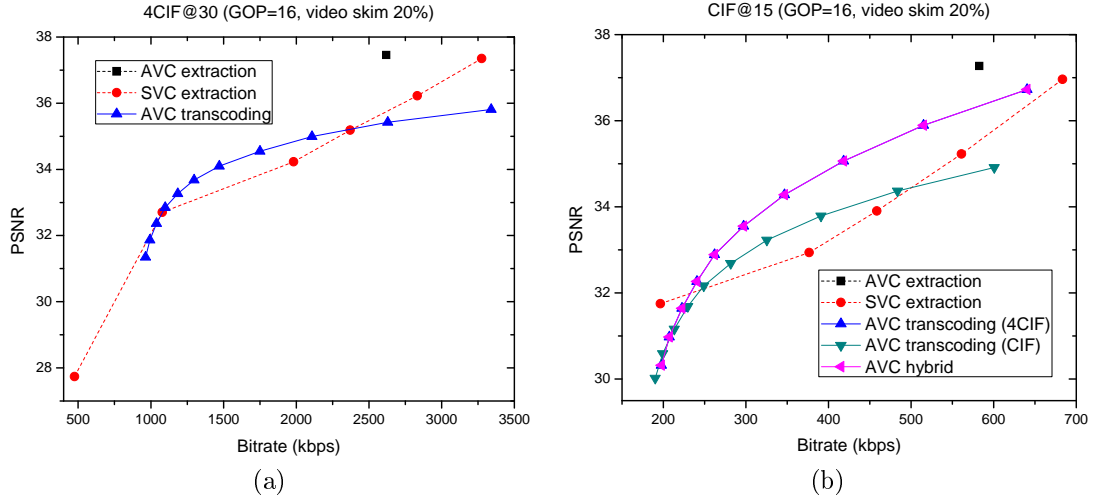


Figure 4.8: Comparison of average PSNR: a) 4CIF 30 Hz and, b) CIF 15 Hz.

bitstream coded with a GOP length of 8 frames. Figure 4.6 shows the rate-distortion curves obtained for the two test scenarios. In both cases AVC extraction has the best results, as it is a single layer and does not need to re-encode the sequence.

Transcoding curves were obtained varying the quantization parameter. It outperforms SVC extraction in the middle of the bitrate range. Figure 4.6b also shows that transcoding from a higher resolution version (4CIF) works better.

SVC extraction works better at the low and high ends of the bitrate range, as they correspond to the operation points represented in Table 4.1, while the intermediate points are obtained by discarding transform coefficients. Rate-distortion performance at these points is significantly worse.

However, both AVC transcoding and SVC extraction performances can be improved using different configurations. Transcoding can be improved using a larger search range for the motion estimation algorithm, at the cost of less efficient processing, as shown in the preceding chapter. Using fewer enhancement layers (e.g. removing layers 0 and 2 from Table 4.1 to remove quality scalability) also improves the rate-distortion performance of the remaining operation points. Alternatively, quality scalability in SVC bitstreams can be slightly optimized using rate-distortion analysis and priority identifiers[Amonou et al., 2007].

## 4.4. Summary and conclusions

In this chapter we have extended the summarization model and framework described in the preceding chapter to include adaptation using the layered approach of MPEG-4 SVC. We also compared experimentally several summarization-adaptation frameworks, based on transcoding and extraction, for different types of summaries and adaptation scenarios. As in the single layer case (preceding chapter), extraction approaches are significantly faster than other approaches. However, rate-distortion performance is not necessarily better in extraction than in transcoding,

as spatial and quality layers introduce a penalty in coding efficiency which is comparable to that due to requantization in transcoding.

## Part III

# Scalable summaries



## Chapter 5

# Scalable storyboards and video skims

In this chapter we deal with summaries with a specific functionality: they are scalable. However, the concept of scalability is used in the context of video summarization, but in a different way to that used in the previous chapters. In this case, the scalability is not a property of the bitstream which is exploited for efficient generation or adaptation, but an intrinsic property of the summary.

In contrast to most algorithms, designed to generate a single summary with a specific length, the creation of scalable summaries involves multiple target lengths. In this chapter we discuss the concept of scalable summaries, their requirements and we propose an adequate framework. The framework is also designed for efficient processing, which makes it very suitable for applications requiring efficient and low delay summarization.

Part of this chapter is based on the publications: [Herranz and Martínez, 2008a, 2009a, 2010a]

### 5.1. Motivation

Scalable approaches have been very useful in many contexts and particularly in video coding. In that context, scalability allows to remove parts of the bitstream while the remaining bitstream is still valid, containing a completely decodable version of the same video, but with lower resolution, quality or frame rate. In scalable coding, encoding is performed once, while many versions can be extracted from the bitstream, according to the specific needs of each case.

In previous chapters we have used the properties of scalable video bitstreams in the context of video summarization for fast generation and adaptation of summaries. However, the concept of scalability can be also used in video summarization in a completely different sense, as a new property of the summaries themselves[Zhu et al., 2004]. In this case, the scale is related to the length of the summary (e.g. duration, number of images). Depending on the case, a summary of a suitable length can be obtained without any further analysis. As in video coding, we can find

many applications in adaptation and personalization. For instance, depending on the terminal capabilities (e.g. display size), the length of the summary can be easily adjusted. Video retrieval systems can also benefit from this scalability. For example, a search interface using storyboards provides the user with a number of search results, with a storyboard representing each item. However, for constant display area, the more images that each individual summary has, the fewer summaries that can be browsed. In this interface there is also a trade-off between the length of the summary and the time spent in visualizing it. A longer summary means more information, but it also requires more time to be visualized. With scalable summaries, the user can easily decide a suitable length for each case.

## 5.2. Related work

Video summarization has been addressed by many researchers with multiple approaches [Truong and Venkatesh, 2007; Money and Agius, 2008b]. However, most methods follow a single scale approach, that is, the output is always a single summary. Realizing that sometimes a single scale may be insufficient, hierarchical summarization approaches [Zhu et al., 2003, 2004; Benini et al., 2006; Bescós et al., 2007] exploit the narrative structure of video sequences to provide the users with a set of summaries with different levels of detail, according to a narrative hierarchy (e.g. chapters, scenes, shots, frames). Each level of this hierarchy is in fact a different scale, with summaries with increasing length across the scales, although the summaries are not scalable within each level. These scales provide a very coarse grain scalability, which is exploited in hierarchical browsing applications, where different levels of detail can be selected in these parts that the user is more interested in.

A common strategy in summarization is the use of clustering algorithms to group frames, shots or other units into similar clusters. Then each cluster is represented by a single image in the summary. Hierarchical clustering has been used also in summarization [Hasebe et al., 2005; Benini et al., 2006] as it can generate clusters at different levels leading to summaries with different scales, not necessary related with narrative structures.

In general, we use the term *scalable summaries* for the case of summaries with a length that can be adjusted with some accuracy without running again the summarization algorithm. Similarly to scalable coding (*encode once, decode many versions*), the objective is to process the sequence once and generate different summaries depending on the length constraints (*analyze once, generate many versions*). Although hierarchical summarization creates scalable summaries, it targets hierarchical browsing and summaries with few scales corresponding to different narrative structures, while, in general, scalable summarization could target a larger number of scales to adapt the summary in constrained situations in which the length of the summary must be limited to a specific value. Nevertheless, [Zhu et al., 2004] introduces the idea of scalable summaries in a hierarchical summarization system. The system obtains a hierarchical representation of the sequence, and the summaries at the highest levels are based on this hierarchy. However, at the lowest levels, the number of frames of the summaries can be adjusted dynamically.

Apart from those based on hierarchical approaches, very few techniques create scalable descriptions of summaries. [Albanese et al., 2006] describes a representation of video sequences based on a priority curve. When this curve is computed, a summary of any desired length can be created easily. However, the main disadvantage of this method is that it needs a prior manual annotation stage of the sequence.

### 5.3. Properties of video summaries

In this section we describe two desirable properties of a good summary: *semantic coverage* and *visual pleasantness*. We also discuss the effect of the length of the summary on storyboards and video skims. These considerations motivate the proposed scalable summarization methodology.

#### 5.3.1. Semantic coverage

Summaries are compact representations of a given content that can be visualized in a much lower amount of time than the content itself. However, they should preserve as much semantic information as possible even though their length has been reduced. A good summarization algorithm should aim at both preserving as much representative information as possible while discarding as much redundant information as possible.

#### 5.3.2. Visual pleasantness

A summary must be not only informative but also comfortable and pleasant for the user when he or she visualizes it. Often, the summary is a result of an editing process of the source sequence, process that may introduce undesirable effects. But a summary is not useful if it contains artifacts which may annoy or even stress the user. As an example, let us imagine a summary made by replacing each shot with a single frame, and that is presented as another sequence played at normal playback speed. The result is a highly condensed summary and its semantic coverage is very high. However, it will be also very unpleasant because this editing operation generates temporal artifacts due to the fast changes between shots. Besides, the user will not be able to retain almost any of the semantic information conveyed by the summary.

A better approach is to use a summary less informative but easier to view for the user. Thus, a good summarization approach should avoid to include frames or segments containing artifacts, but it should also avoid to create new artifacts resulting from editing operations.

#### 5.3.3. Properties in the context of storyboards and video skims

The approach described in this chapter deals with both storyboards and summaries, which are the most used modalities of video abstracts. For storyboards, it is usually enough to focus on optimizing the semantic coverage. This representation does not give many chances to include unpleasant effects. However, some frames belonging to transitions (e.g. fades, wipes, dissolves)

are usually unsuitable as they contain mixed and incomplete information from two shots, so it could be better to avoid including them in the storyboard. For the same reason, blurred images could be also considered unsuitable.

However, the editing operation involved in video skims may lead to artifacts, so both semantic coverage and visual pleasantness must be balanced in order to obtain informative summaries while avoiding annoying artifacts. Short segments belonging to shot changes are examples of unsuitable segments for video skims.

#### 5.3.4. Influence of the length of the summary

Video summarization is motivated by the fact that video visualization is a time consuming task, due to the length (i.e. duration) of video sequences. For this reason, length plays an essential role in summarization. It must be significantly reduced but the summary must preserve as much information as possible. However this is not an easy task, because in general, the longer the summary is the more information it can convey, and thus, the better semantic coverage it can have. But if the length is severely constrained, the summarization algorithm should try to preserve relevant information and discard redundant information.

As described before, the length also affects the visual aspect of the summary, especially in video skims. If the skim is very short, but trying to include too much information, it will become annoying and quite probably useless.

Depending on the application, there may be also constraints in the length. For instance, a search result page in a digital library retrieval interface may contain several but short storyboards. However, if the user requests more information about one result, a longer storyboard can be presented. Most algorithms are designed for specific lengths, but in some cases the performance might be degraded when the required length is not in the expected range. For instance, the TRECVID 2008 rushes summarization task[Over et al., 2008] specifies summaries with a duration of 2% of the original, so most algorithms were designed and tuned for this target length.

Although primarily discussed for storyboards and video skims, other modalities of summaries are inevitably influenced by its length or duration, as information and length must be traded off (e.g. comic-like summaries; see next chapter).

### 5.4. Scalable summarization framework

Based on the analysis of the requirements of scalable summaries, we designed a generic framework for scalable summarization. This framework includes an efficient generation stage, a scalable representation and an analysis methodology to generate scalable summaries, based on the idea of incremental growing. In order to differentiate this scalability from other video scalabilities (i.e. spatial, temporal and quality), we will use the term *length scalability* when required.

### 5.4.1. Efficient generation of the summary

As discussed in Chapter 3, we can identify two stages in a summarization system: analysis and generation. Although most research efforts are devoted to the analysis stage, in some applications, such as highly scalable summarization, the generation stage is critical.

A scalable summary contains implicitly multiple single-scale summaries. As any digital content, each of these summaries is usually delivered in a compressed format. When the amount of scales is low, each summary could be stored independently, and just selected and delivered when required. However, if the number of versions is large, storing every scale independently is not possible. In that case, the summary must be generated from the original content on demand.

Using a transcoder or the bitstream extraction approach described in Chapter 3 are the two main options to generate summaries on demand. However, in order to take full advantage of a scalable representation of a summary, an efficient generation stage is critical. If the bottleneck to summarize a content is the generation stage, there is no point in having a scalable representation, as the analysis stage could be run every time a summary is requested with less cost than the generation stage. In those cases, the delay would be probably unacceptably high. Thus, an efficient generation stage is key for low delay on demand summarization.

We use bitstream extraction in our framework. Note that variations or transcoding could still be reasonable solutions for short summaries such as storyboards, as the delay required to transcode few images could be acceptable (see Figure 3.21). But for long summaries, such as video skims, the generation delay using transcoding is too high. Bitstream extraction is a more suitable solution in this case, having little delay in general.

### 5.4.2. Scalable summaries

As in the framework described in Chapter 3, we use summarization units as basic units for any kind of processing. For simplicity, we assume that the summarization unit is the GOP, which is also the basic unit for analysis and representation of the summary, and for that reason we use the term GOP instead of summarization unit for both analysis and generation in this chapter.

Thus, the source sequence  $\mathbf{V}$  is coded in  $M$  GOPs. Let  $f_m$  denote the I frame belonging to the GOP  $\mathbf{U}_m$ . This frame can be decoded independently of the other frames of the sequence, so it constitutes another summarization unit, used in keyframe based summaries. In this context, let us recall the concept of *embedded summary* from Section 3.4, which is a sequence of arbitrary GOPs (see Figure 5.1a). The bitstream of the embedded summary  $\mathbf{S}$  is obtained from the input sequence  $\mathbf{V}$  using the *extraction* operation, which combines the summarization units into a valid bitstream.

Different lengths can be addressed using different embedded summaries. However, the concept of scalability can be also used. Thus, we introduce scalable summarization as a special case of embedded summarization (see Figure 5.1b) with an important additional restriction. A *scalable summary* is a set of embedded summaries  $\mathbf{SS} = \{\mathbf{S}^{(1)}, \dots, \mathbf{S}^{(q)}, \dots, \mathbf{S}^{(Q)}\}$ , with  $q \in \mathbb{N}$  denoting the summarization scale and  $Q$  denoting the number of scales, and with each embedded

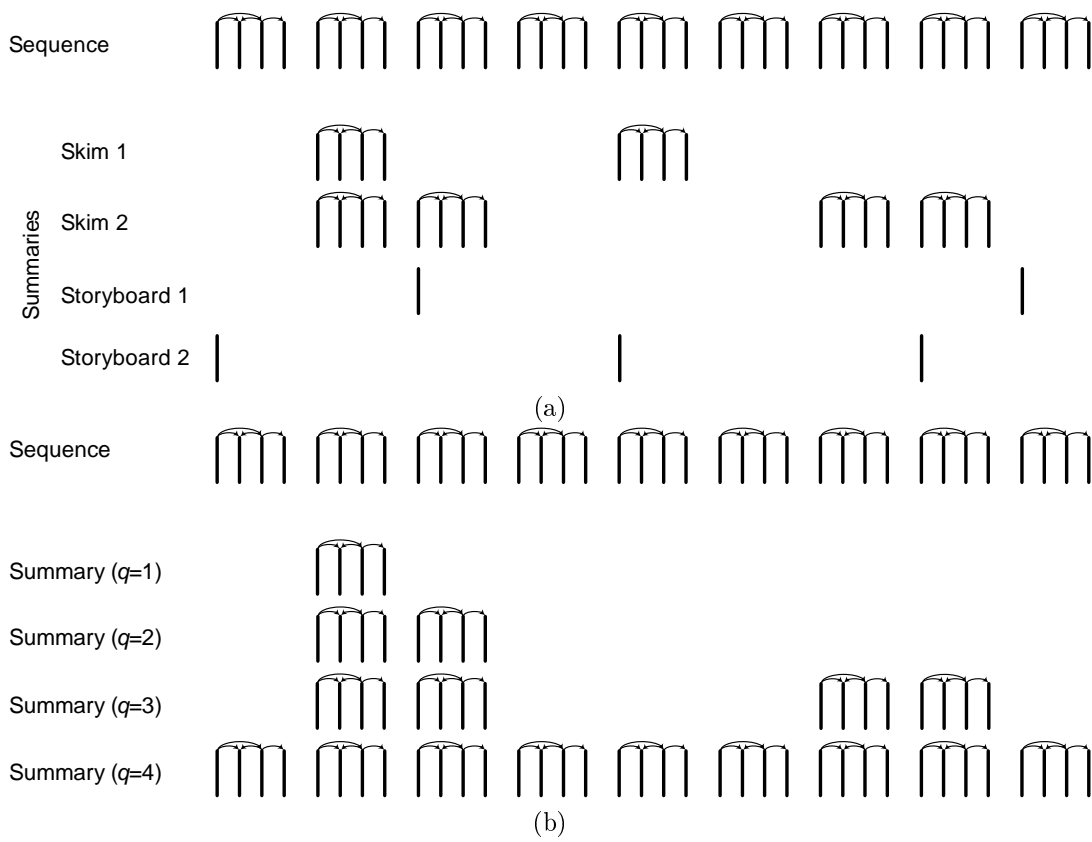


Figure 5.1: Embedded (a) and scalable (b) summaries.

summary  $\mathbf{S}^{(q)}$  satisfying

$$\mathbf{S}^{(1)} \subset \mathbf{S}^{(2)} \subset \dots \mathbf{S}^{(q)} \subset \dots \subset \mathbf{S}^{(Q)} \subseteq \mathbf{V} \quad (5.1)$$

In embedded summarization, summaries are described by a set of summarization units (GOPs in this chapter). However, for scalable summarization we introduce the ranked list as a different tool to describe the whole set of summaries.

### 5.4.3. Ranked lists

A *ranked list*  $\mathbf{list}_{ss}$  is a sequence with the indexes of the GOPs sorted by their relevance for summarization representing the scalable summary  $\mathbf{SS}$ . A ranked list  $\mathbf{list}_{ss}$  of length  $M' \leq M$  satisfies

$$\mathbf{list}_{ss} = (m_0, \dots, m_i, \dots, m_{M'-1} | m_i \in I_{\mathbf{V}}, m_i \neq m_j, \forall i \neq j) \quad (5.2)$$

where  $I_{\mathbf{V}}$  is the set of indexes of the GOPs in the sequence  $\mathbf{V}$ , and  $m$  is the index of the GOP  $U_m$ .

A summary of length  $M_S$  GOPs, embedded in  $\mathbf{SS}$  contains the GOPs with the first  $M_S$  indexes in  $\mathbf{list}_{ss}$ . Note that for each embedded summary a different set must be specified, but for scalable summaries, the ranked list contains all the summaries in a single compact representation. Thus, in scalable summarization, the objective of the analysis algorithm is to determine the ranked list.

### 5.4.4. Architecture

As in scalable coding frameworks, the analysis stage is detached from the lightweight generation/adaptation stage. The whole framework is shown in Figure 5.2. The analysis stage is performed once for every sequence, and consists of a cascade of feature extraction, clustering and ranking algorithms, as described further on. The result of the analysis is a scalable representation of the sequence as a ranked list. Whenever a summary of the sequence is requested with a specific length, the generation stage parses the ranked list and determines which GOPs must be included in the summary. Then, the bitstream extractor processes the bitstream of the sequence to generate the bitstream of the summary.

### 5.4.5. Scalable summarization by incremental growing

For scalable summarization we target a single representation with satisfactory results in a wide range of lengths. The length of the summary is selected on demand for each specific case, but no extra analysis is required, as the scalable representation has enough information to determine which packets of the source bitstream are the appropriate to build the required summary. To address the problem of different lengths we propose a generic incremental approach, which systematically creates longer summaries based on a shorter one of the same scalable summary:

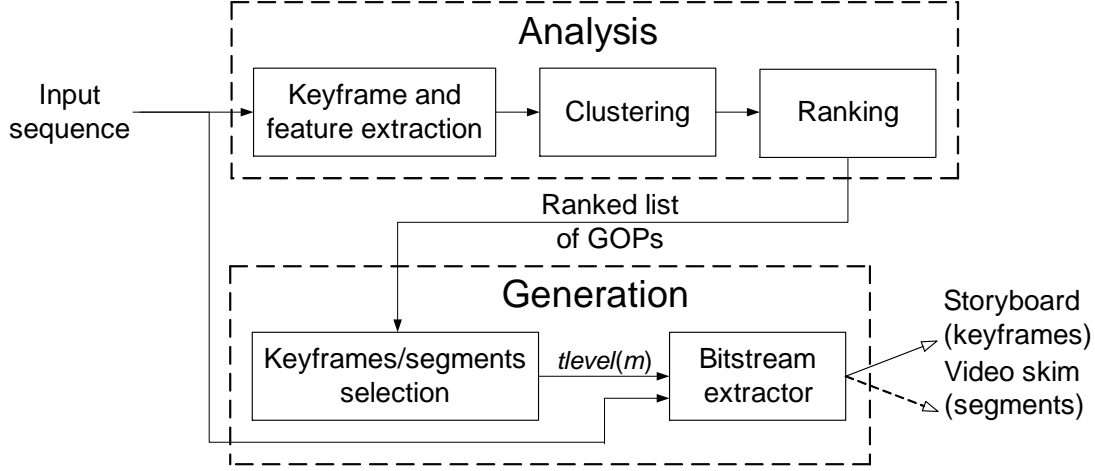


Figure 5.2: Proposed architecture

1. Set scale  $q = 0$ . Set  $\mathcal{S}^{(0)} = \emptyset$ .
2. Set  $q = q + 1$ .
  - a) Compute the score  $score^{(q)}(U_m | \mathcal{S}^{(q-1)})$  for every GOP  $U_m \notin \mathcal{S}^{(q-1)}$ .
  - b) Get the index  $m^*$  with maximum  $score^{(q)}(U_m | \mathcal{S}^{(q-1)})$  and set  $\mathcal{S}^{(q)} = \mathcal{S}^{(q-1)} \boxplus U_{m^*}$  ( $\boxplus$  denotes the operation of inserting an element in the corresponding temporal order).
3. Go to step 2 and repeat until the desired length or scale is achieved.

This approach follows an incremental procedure which improves at each step the previous summary trying to select the GOP  $U_{m^*}$  that improves more the previous summary. With this procedure, it is straightforward to obtain a representation of the scalable summary as a ranked list. Note that the score is computed *a posteriori* given the current summary. Approaches related to the priority curve formulation [Albanese et al., 2006; Truong and Venkatesh, 2007] also give a score to each summarization unit, although this score is computed *a priori*. The score can be obtained from some kind of feature analysis, such as activity [Divakaran et al., 2002], face detection [Peker et al., 2006] or attention [Ma et al., 2005], but it is independent from the rest of the summarization analysis.

## 5.5. Keyframe and feature extraction

In our system, the analysis module parses the input bitstream and structures the sequence into suitable units. The subsequent ranking algorithm processes these units to obtain the ranked list. Compressed domain data is used for efficient feature extraction and processing. The first stage preprocesses the sequence in order to structure and represent it in a set of keyframes and their feature vectors.

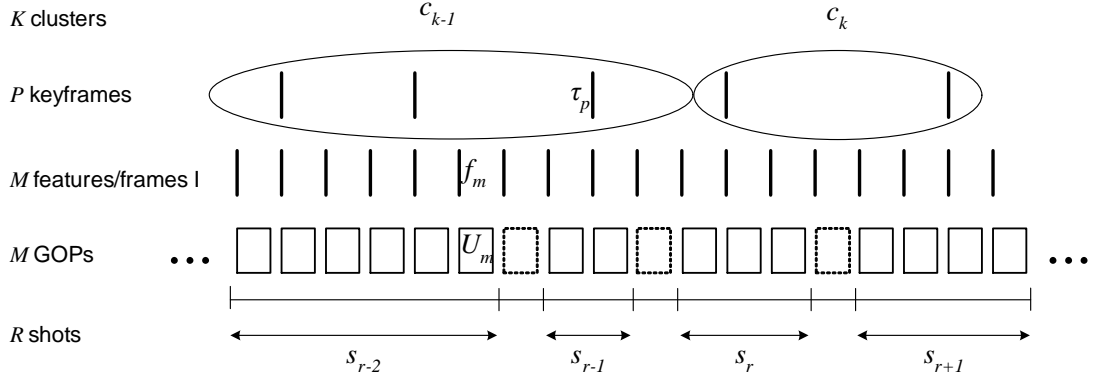


Figure 5.3: Analysis structures.

### 5.5.1. Structure and notation

As described previously, the original sequence is composed by frames, but the main summarization unit is the GOP. We assume that the original sequence is composed by  $M$  GOPs. Each GOP  $U_m$  has one I frame  $f_m$ , which is used for feature extraction in order to characterize the GOP  $U_m$ . The sequence is partitioned into  $R$  shots  $s_r$  (this partition may be incomplete as some GOPs may be discarded by analysis), which are further represented by  $P$  keyframes  $t_p$ , selected among the I frames. These keyframes are clustered into  $K$  clusters  $c_k$ , and through them, shots are also linked with clusters. These analysis structures are shown in Figure 5.3.

### 5.5.2. Shot change detection

The purpose of the shot change detection algorithm is both to structure the sequence into shots and to discard GOPs with transitions, which may lead to unpleasant effects in the summary. In this framework, which is GOP based, GOP precision is usually enough rather than frame precision. For this reason and in order to make the algorithm faster we implemented a simple cut detector based loosely on the thresholding approach described in [Nakajima et al., 1999], but applied in a GOP basis by processing the DC image [Yeo and Liu, 1995] of each I frame  $f_m$ .

A luminance histogram  $H_m^Y(y)$  is computed from the DC image of each frame  $f_m$ . The interframe luminance distance  $D_m^Y$  is computed using the histogram intersection distance as follows

$$D_m^Y = \sum_y \min(H_m^Y(y), H_{m-1}^Y(y)) \quad (5.3)$$

A two-dimensional chrominance histogram  $H_m^{UV}(u, v)$  is also computed from the DC image of each frame  $f_m$ . The interframe chrominance distance  $D_m^{UV}$  is computed using the same histogram intersection distance as follows

$$D_m^{UV} = \sum_{u,v} \min(H_m^{UV}(u, v), H_{m-1}^{UV}(u, v)) \quad (5.4)$$

The luminance distance must be below a first threshold to be considered a candidate for a shot change

$$D_m^Y < th\_min \quad (5.5)$$

Most cuts appear as sudden peaks in the interframe distance. For this reason, the interframe distances must also satisfy the following conditions[Nakajima et al., 1999]:

$$\begin{aligned} \gamma_{SC} D_m^Y &< D_{m-1}^Y \\ \gamma_{SC} D_m^Y &< D_{m+1}^Y \\ D_m^{UV} &< D_{m-1}^{UV} \\ D_m^{UV} &< D_{m+1}^{UV} \end{aligned} \quad (5.6)$$

In the previous conditions,  $\gamma_{SC}$  is a factor to guarantee that small peaks due to noise are filtered. In the experiments, after some tests, we set the value of  $\gamma_{SC}$  to 0.6. If all the conditions are satisfied, a shot change is declared between  $U_m$  and  $U_{m-1}$ . For simplicity, we assume that the I frame is the first frame of each GOP. In that case, the declared transition is contained in the GOP  $U_{m-1}$ . Analogously, if the I frame were the last frame of the GOP, the shot change should be declared in  $U_m$ .

Although no detector is implemented for gradual transitions, as the temporal distance between I frames increases, the effective length of the shot change decreases, and thus some of the short gradual transitions become cuts. On the other hand, as the GOP length increases, the feature distance between frames belonging to the same shot increases while the distance between frames belonging to different shots does not change significantly. For this reason, the detection of cuts becomes difficult for long GOPs. In this case, a different shot change detection algorithm processing all the frames may be necessary to achieve a better detection rate, in exchange of some computational cost. There are many algorithms for efficient shot change detection in the compressed domain of MPEG-1/MPEG-2[Nakajima et al., 1999; Bescós, 2004] and H.264/AVC[Zeng and Gao, 2005; De Bruyne et al., 2006].

After a shot change is declared, if the duration of the candidate shot is below a minimum, the GOPs inside are marked as belonging to a too short shot, and discarded for subsequent analysis (e.g. with a threshold of 3 GOPs,  $s_{r-1}$  would be discarded in Figure 5.3). If the candidate is not discarded, the GOPs inside form a new shot.

### 5.5.3. Feature extraction

Although all the GOPs can be processed by the next stages, for long sequences the amount of data would be very high. Sampling properly the sequence can reduce dramatically the amount of data without significant impact on the results. For this reason, each shot is represented with few GOPs (selected by regular sampling), and their I frames are used as keyframes. The number of keyframes may vary depending on the length of the shot, up to a fixed maximum of  $W_{max}$  keyframes per shot. For sequences with low activity and static segments, a low  $W_{max}$  is enough to represent the shot, and the amount of data is notably reduced. For sequences

with more activity, a higher  $W_{max}$  would be necessary. For each keyframe  $t_p$  a feature vector is then computed. In our experiments, we used the MPEG-7 colour layout, along with a suitable distance[Kasutani and Yamada, 2001].

## 5.6. Clustering

At this point, the sequence is represented by  $P$  keyframes. The next step consists of removing semantic redundancies between keyframes, grouping those with similar features into clusters. The output of the clustering stage will be a set of  $K$  clusters  $\{c_1, c_2, \dots, c_K\}$ . Any clustering algorithm can be used and the clustering processing delay will also benefit from the important reduction of the data set, although low complexity in clustering is still desirable. Besides, some algorithms may be more appropriate than others due to the fact that the data set is not dense anymore after the previous stage. The number of underlying clusters may be high compared to the size of the data set, and some algorithms are more sensitive to this fact than others. In order to study the implications of the data reduction in the choice of the clustering algorithm we compare the characteristics of the well known  $K$ -means and hierarchical clustering algorithms[Jain et al., 1999; Theodoridis and Koutroumbas, 2006].

### 5.6.1. Comparison of $K$ -means and hierarchical clustering

#### 5.6.1.1. $K$ -means

A widely used algorithm for clustering in video summarization is  $K$ -means[MacQueen, 1967] because of its simplicity which makes it very attractive for large data sets. In the basic algorithm, an initial partition of  $K$  clusters is improved iteratively minimizing the total intracluster variance. Each cluster is represented by a characteristic point called centroid, and each data point is assigned to the cluster with the closest centroid. After each iteration, the centroid is recomputed as the mean point of all the data points of the cluster, and then the points are reallocated according to the new centroids. This scheme is iterated until the stopping criteria are reached (e.g. no changes in the cluster membership). The initial partition is set randomly.

The complexity of  $K$ -means is  $O(NKq)$  where  $q$  is the number of iterations. It is usually assumed that in practice  $K$  and  $q$  are much lower than  $N$ , so the complexity is approximately lineal for large data sets. This is not the case of this work, as  $K$  is comparable to  $N$ , and the complexity can be comparable to  $O(N^2)$ .

However  $K$ -means has many drawbacks. The initialization is a critical step, as different initial partitions may lead to very different final clusterings[Peña et al., 1999]. Many heuristics and stochastic approaches have been proposed, but with a significant cost in computational complexity. A widely used heuristic is to run several times the algorithm with different initial partitions (picked randomly), and then keep the solution with minimum intracluster variance. Another drawback is the sensitivity of the algorithm to outliers and noise.

In this algorithm, the number of clusters  $K$  must be specified as an input parameter, and it has a decisive influence in the results. A wrong value may lead to poor clusterings. Thus, the

estimation of the number of clusters is critical. Several methods have been proposed, but again with some penalty in the complexity.

Other partitional algorithms used in summarization, such as spectral clustering[Ng et al., 2001; Odobez et al., 2003; Filippone et al., 2008], can estimate the number of clusters, although instead of the explicit parameter  $K$  it is usually necessary to specify other model parameters which are still difficult to estimate and the results are very sensitive to small variations of their values. They usually need a dense data set to be able to recover the underlying structure.

#### 5.6.1.2. Hierarchical agglomerative clustering

Hierarchical clustering algorithms[Ward, 1963; Jain et al., 1999] generate a hierarchy of nested clusterings rather than a single clustering. The most used are the agglomerative algorithms, which start with an initial clustering in which every data point is a cluster. At each step the distance between all the possible pairs of clusters is computed and the two clusters with less distance are merged into one. The procedure continues until the final clustering, which contains a single cluster. Different distances between clusters lead to different algorithms. The most used algorithms of this family are the single linkage, complete linkage and average linkage clusterings[Theodoridis and Koutroumbas, 2006].

The final clustering can be obtained by either specifying the number of clusters, or either specifying a cut-off distance in the accumulated cluster distance through the hierarchy. The latter is a much more suitable criterion, as a threshold distance can be set independently of the length of the sequence, and the problem of the estimation of the number of clusters is avoided.

The complexity of the agglomerative scheme is  $O(N^3)$  although there are some efficient implementations that only require  $O(N^2)$ [Theodoridis and Koutroumbas, 2006]. For large data sets, this algorithm is usually unfeasible and lower complexity algorithms such as  $K$ -means are preferable. However, in the case of the proposed analysis algorithm, the data was previously reduced in order to avoid high complexity and the subsequent high processing time. As discussed before, in this case the complexity is comparable to  $K$ -means. Besides, in contrast to partitional algorithms, the agglomerative approach obtains good clusterings in the case of the non dense data set obtained in feature extraction.

The characteristics of both clustering algorithms are summarized in Table 5.1, comparing the advantages and drawbacks for both the common scenario (see Table 5.1a) and the specific scenario of the proposed summarization algorithm, after the amount of data has been dramatically reduced in the previous stage (see Table 5.1b).

#### 5.6.2. Clustering in the proposed summarization algorithm

According to the previous analysis and comparison between both algorithms, we use the hierarchical clustering algorithm. Particularly, the algorithm used is the agglomerative algorithm with average linkage[Theodoridis and Koutroumbas, 2006], as it can recover clusters with non-spherical shapes (in contrast to other algorithms such as  $K$ -means).

	$K$ -means	Hierarchical (agglomerative)
Complexity	$O(NKq) \approx O(N)$ as usually $K \ll N, q \ll N$ ( $q$ : number of iterations)	$O(N^3)$ $O(N^2)$ (efficient implementation)
Output	$K$ clusters	Hierarchy of clusters
Advantages	- Simple. - Fast. - Scalable to large data sets	- Able to recover complex-shaped clusters. - No need to specify the number of clusters. Dendrogram analysis can be used instead. - Hierarchical representation.
Drawbacks	- $K$ must be specified. How to estimate it? - Results very dependent on the (random) initializations. Different results from run to run (non-deterministic). - Only (hyper)spherical-shaped clusters.	- More complex. - Slow with large data sets.
Comments	- Heuristic and statistical approaches can help to overcome these limitations, although the algorithm becomes notably slower.	

(a)

	$K$ -means	Hierarchical (agglomerative)
Complexity	$O(NKq) \approx O(KN)$ if $K \approx N$ then $O(NKq) \approx O(N^2)$	$O(N^2)$ (efficient implementation)
Comments	- Non-dense data set. $K$ -means performs worse with non-dense data sets. - No so fast now.	- Complexity relies on the computation of the distance matrix.

(b)

Table 5.1: Comparison of K-means and hierarchical clustering: (a) usual scenario, (b) after data reduction.

The final clustering is obtained by specifying a cut-off linkage distance in the accumulated cluster distance through the hierarchy. For each cluster  $c_k$ , the keyframe with its feature vector closest to the centroid is selected as the representative keyframe  $\bar{t}_k \in c_k$ .

Note that we do not use the hierarchy generated by the algorithm for scalability, although it could be used, as it is a scalable representation itself. Instead of that, we use a different approach based on iterative ranking to generate the scalable representation.

## 5.7. Iterative ranking

The ranking stage is motivated by the need to address the problem of generating suitable summaries for a wide range of potential lengths, but created in a single process. The objective is to obtain a compact scalable representation of storyboards and video skims. It follows the incremental growing approach described in Section 5.4.5, returning two ranked lists: **list<sub>sb</sub>** for storyboards and **list<sub>vs</sub>** for video skims. Note that, in contrast to conventional approaches, this approach creates a set (with a high number of embedded summaries) instead of a single summary. The objective is to find a good set of embedded summaries satisfying (5.1) and not a single optimal summary.

In order to balance properly the coverage and pleasantness for short and long summaries, the ranking is divided into two different stages: cluster level ranking and shot level ranking.

### 5.7.1. Cluster level

When the length of the summary is very constrained (usually in storyboards and short skims), the algorithm focuses on covering the basic semantics of the sequence with as few GOPs as possible. We assume that clusters are a reasonably good representation of these semantics, and that each cluster can be represented by a keyframe (for storyboards) and by a short excerpt of  $N_{exc}$  consecutive GOPs (for video skims). Thus, finding a representative keyframe or segment for each cluster should be enough to have a suitable summary in these limited conditions. If the length of the summary is even more constrained, some keyframes or excerpts must be discarded, with an associated loss in coverage.

To provide the best set of keyframes for each summary length, clusters are ordered by their relevance, using the following iterative ranking procedure in which the clusters are ranked and selected incrementally:

1. Set scale  $q = 0$ . Set  $\mathbf{S}^{(q)} = \mathbf{list}_{sb} = \mathbf{list}_{vs} = \emptyset$ .
2. Compute the score  $score^{(q)}(c_k)$  for every cluster  $c_k$ . Select the cluster  $c^*$  with maximum score. Mark  $c^*$  as selected including it in  $\mathbf{S}^{(q)}$ . Grow previous summaries as follows
  - a) Include  $m^*$  in **list<sub>sb</sub>**, where  $m^*$  is the index of the GOP  $U_{m^*}$  containing  $\bar{t}_k$  (keyframe representative of  $c^*$ ).
  - b) Include an excerpt **b** in **list<sub>vs</sub>** centered at the GOP with index  $m^*$ . The excerpt of fixed length  $N_{exc}$  GOPs (with  $N_{exc}$  even) is defined as

$$\mathbf{b} = \left\{ m^* - \frac{N_{exc}}{2}, \dots, m^*, \dots, m^* + \frac{N_{exc}}{2} - 1 \right\} \quad (5.7)$$

3. Set  $q = q + 1$ . Set  $\mathbf{S}^{(q)} = \mathbf{S}^{(q-1)}$ . Go to step 2 and repeat until all clusters are selected.

The score at the scale  $q$  of each cluster is then given by

$$score^{(q)}(c_k) = \begin{cases} (1 - \alpha_c) \frac{score_{dist}^{(q)}(c_k)}{\max_j score_{dist}^{(q)}(c_j)} + \alpha_c \frac{score_{dur}^{(q)}(c_k)}{\max_j score_{dur}^{(q)}(c_j)} & k \notin \mathbf{S}^{(q-1)} \\ 0 & k \in \mathbf{S}^{(q-1)} \end{cases} \quad (5.8)$$

Note that clusters selected in previous iterations are no longer considered. Note also that scores must be recalculated for each iteration, as there is not a global *a priori* score for each cluster (as in other summarization algorithms [Truong and Venkatesh, 2007; Albanese et al., 2006]), but a local *a posteriori* score for each iteration, conditioned by the summary obtained in the previous iteration.

The scores are computed based on two criteria: distance and duration, combined in a weighted sum. The duration score favors the selection of clusters with more contribution in terms of duration of the sequence, assuming that longer clusters should be included at lower scales. The score  $score_{dur}^{(q)}(c_k)$  is computed as

$$score_{dur}^{(q)}(c_k) = L(c_k) \quad (5.9)$$

where the duration of  $c_k$  is defined as  $L(c_k) = \sum_p L(s_p)$ ,  $\forall s_p \in c_k$ , and  $L(s_p)$  is the length of  $s_p$  in number of GOPs. A shot  $s_p$  belongs to  $c_k$  if any of its representative keyframes is member of  $c_k$ . The distance scores favours the selection of more dissimilar clusters than those already selected in previous iterations. The distance  $d(c_i, c_j)$  between clusters is computed as the distance between their representative keyframes (i.e.,  $d(c_i, c_j) = d(\bar{t}_i, \bar{t}_j)$ ). The score  $score_{dist}^{(q)}(c_k)$  is computed as

$$score_{dist}^{(q)}(c_k) = \begin{cases} 0 & q = 0 \text{ or } k \in \mathbf{S}^{(q-1)} \\ \min_{j \in \mathbf{S}^{(q-1)}} d(c_k, c_j) & q > 0, k \notin \mathbf{S}^{(q-1)} \end{cases} \quad (5.10)$$

Figure 5.4 shows an example of clustering (each row representing a cluster), after ranking (with  $\alpha_{cluster} = 1$ , i.e. ranked by duration). The input sequence was an excerpt of a news video. Similar shots are clustered in the same cluster and clusters with longer duration are placed above shorter ones. Note that, in some cases, due to the limitation to  $W_{max} = 3$  keyframes per shot, clusters with a longer duration may have fewer keyframes than other clusters with a shorter aggregated duration, but with more shots.

The scoring and selection mechanisms are illustrated in Figure 5.5. It shows the scores of clusters across the different iterations, when they are ranked either by duration or by distance. For better visualization, the scores are shown normalized and ordered in such a way that the scores of the cluster selected at a given iteration  $q$  are placed in the column  $q$ . Thus, the main diagonal shows the maximum score of each iteration.

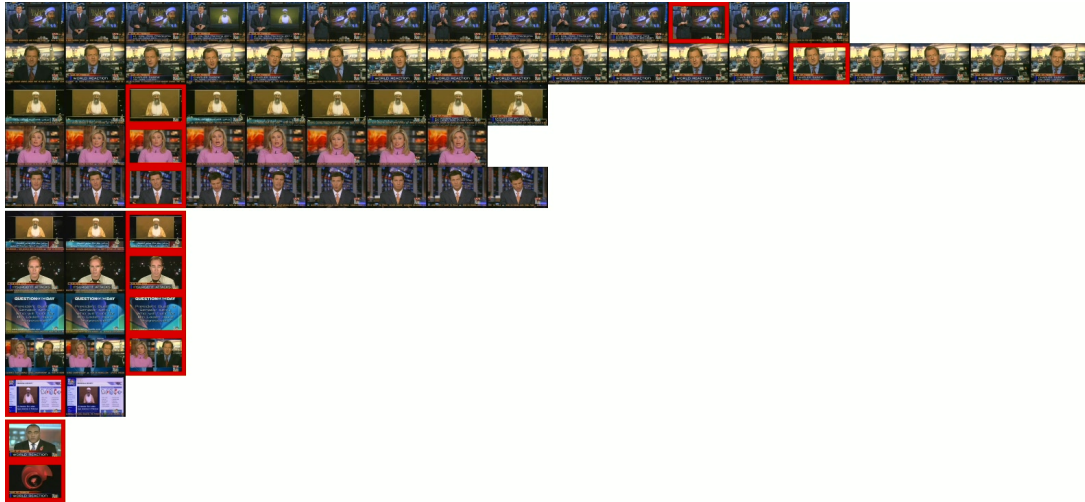


Figure 5.4: Example of clusters after ranking ( $\alpha_{cluster} = 1$ ,  $W_{max} = 3$ ).

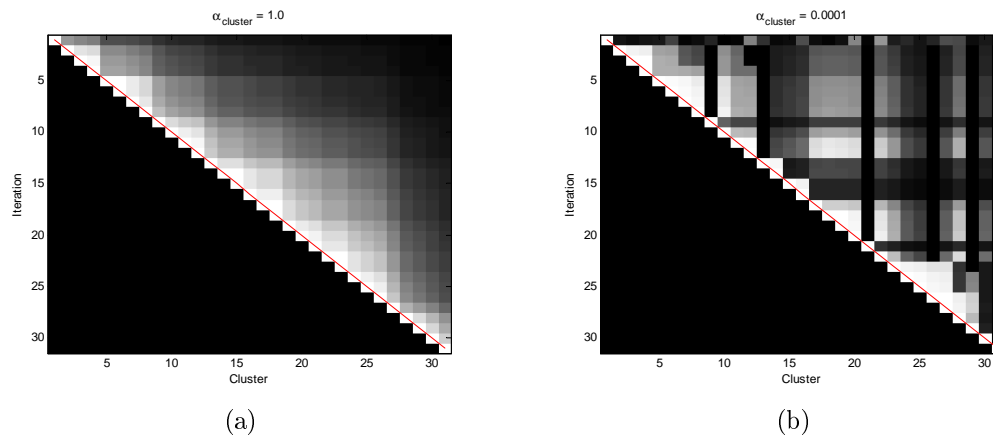


Figure 5.5: Cluster scores for each iteration: (a)  $\alpha_{cluster} = 1$  (duration), (b)  $\alpha_{cluster} = 0.0001$  (distance). Scores are normalized for each row (iteration) and sorted by order of selection.

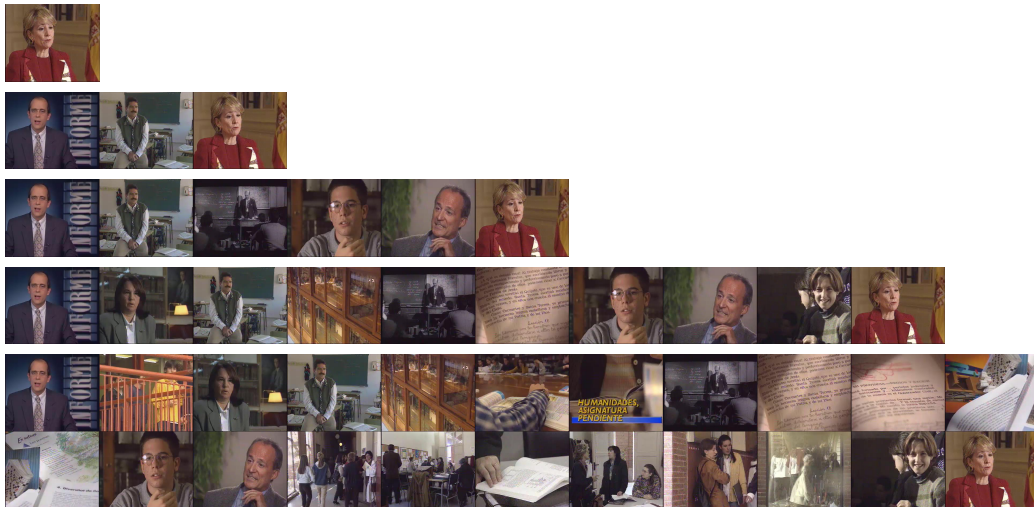


Figure 5.6: Example of scales of a scalable storyboard of *news11* ( $\alpha_c = 0.5$ ).

The iterative ranking procedure computes a different score for each iteration. This score is influenced by the summary resulting from the previous scale. The difference between scores computed *a priori* and *a posteriori* (i.e. conditioned by the previous summary) can be also observed in Figure 5.5. Although the duration score is computed for each iteration, it is based on a feature (i.e. duration) that is independent of the other clusters. For this reason, the scores in each column and row are smooth and monotonous in each row and column (see Figure 5.5a). On the contrary, the distance score depends on the other clusters, and it is influenced by the selection of specific clusters. When two clusters are very close in the feature space, if one of them is selected, the distance score of the other will decrease significantly in the subsequent iterations, and the scores of other clusters will be boosted. It stimulates the inclusion of novelty in the summary instead of redundancy. This effect can be seen in Figure 5.5b as sudden changes in the scores motivated by the selection of another cluster in the previous iteration.

Figure 5.6 shows an example of the different scales in a scalable storyboard. As it can be observed, shorter summaries are included into longer ones. Also, clusters with long shots (head shots of interviewees) are included at earlier stages, due to the contribution of the duration score to the overall score.

### 5.7.2. Shot level

For longer skims, once a minimum semantic coverage is achieved and the length of the summary increases, summaries can be improved not only from the semantic coverage point of view, but also emphasizing other aspects. In this sense, segments with different lengths can be allowed in the summary, trying to create a more natural summary. At this level, the ranking is performed iteratively for each GOP, computing scores for each shot. For each iteration, there are two possible actions: including an additional excerpt from a new shot or growing an excerpt included previously. The ranking algorithm continues after cluster ranking with the following

steps:

1. Compute  $score^{(q)}(s_r)$  for each shot  $s_r$ . Select the shot  $s^*$  with maximum score. Grow summaries as:
  - a) If  $s^*$  was not selected previously, select the keyframe  $t^*$  closer to the middle of the shot  $s^*$  and include an excerpt  $\mathbf{b}$ , centered at the GOP containing  $t^*$ , in  $\mathbf{list}_{vs}$ . Mark  $s^*$  as selected and include it in  $\mathbf{S}^{(q)}$ .
  - b) If  $s^*$  was already selected, grow the selected excerpt with an additional GOP of index  $m^*$  (alternatively from left and from right bounds of the excerpt, until shot bounds are found). Update  $\mathbf{list}_{vs}$  including  $m^*$ .
2. Set  $q = q + 1$ . Set  $\mathbf{S}^{(q)} = \mathbf{S}^{(q-1)}$ . Go to step 2 and repeat until all the GOPs in all shots are selected.

As in cluster ranking, the score  $score^{(q)}(s_r)$  weights two different criteria with  $\alpha_s$  controlling the trade-off.

$$score^{(q)}(s_r) = \begin{cases} (1 - \alpha_s) \frac{score_{dist}^{(q)}(s_r)}{\max_j score_{dist}^{(q)}(s_j)} + \alpha_s \frac{score_{dur}^{(q)}(s_r)}{\max_j score_{dur}^{(q)}(s_j)} & r \notin \mathbf{S}^{(q-1)} \\ 0 & r \in \mathbf{S}^{(q-1)} \end{cases} \quad (5.11)$$

The score  $score_{dist}^{(q)}(s_r)$  favours the inclusion of shots which are not well represented by the current summary, and it is based on the Hausdorff distance from the unselected keyframes ( $\tilde{t}^{(q)}$  represents the set of selected keyframes at scale  $q$ ) of the shot  $s_r$  to the current summary. If all the keyframes belonging to the shot  $s_r$  were included previously, then the score is zero. Thus, the score  $score_{dist}^{(q)}(s_r)$  is calculated as

$$score_{dist}^{(q)}(s_r) = \begin{cases} \max_{t_j \in s_r} \left\{ \min_{t_j \in \tilde{t}^{(q-1)}} d(t_j, t_k) \right\} & \text{if } \exists t_j \in s_r | t_j \notin \tilde{t}^{(q-1)} \\ 0 & \text{otherwise} \end{cases} \quad (5.12)$$

The score  $score_{dur}^{(q)}(s_r)$  is calculated as

$$score_{dur}^{(q)}(s_r) = e^{-\left( \frac{\rho(s_r)}{\max_{t_j, t_k \in s_r} d(t_j, t_k)} \right)^\lambda} \quad (5.13)$$

The duration criterion is based on the assumption that longer shots should be represented with longer excerpts in the summary. This ad-hoc expression measures the relevance of the shot according to  $\rho(s_r) = 1 - \frac{L(s_r \cap \mathbf{list}_{vs}^{(q)})}{L(s_r)}$ , which is the ratio between the duration of the shot not yet selected at the scale  $q$  and the total duration of the shot. To avoid an excessive prominence of the longest shots, we use an exponential function (in the experiments we used  $\lambda = 4$ ). When  $\rho(s_r)$  reaches a sufficient value, the shot is considered well represented and  $score_{dur}^{(q)}(s_r)$  decreases rapidly. The ratio  $\rho(s_r)$  is normalized by the distance between keyframes in the shot, promoting the selection of more GOPs in shots with more internal variation.

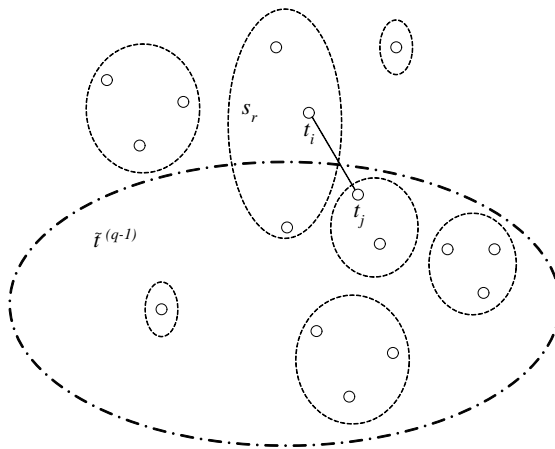


Figure 5.7: Distance score using the Hausdorff distance

### 5.7.3. Residual ranking

Finally, those GOPs initially discarded in the feature extraction stage can be included in the ranked list if a complete scalable representation of the sequence is required. First, the GOPs belonging to short shots are selected iteratively, from longer to shorter duration. After that, the GOPs belonging to shot changes are finally selected.

Note that, for most practical applications, this last ranking stage can be avoided, as it would only have effect on summaries with length close to that of the full sequence (almost no summarization at all). However, a complete scalable representation of the sequence, which could be useful in some applications, requires all the GOPs to be included in the ranked list.

Figure 5.8a shows an example of ranked list and the length of the summary obtained at each index. The cluster level ranking covers the shortest summaries, in a small range from the beginning of the list. This range is the most useful, as summaries are usually required to be short. The shot level ranking covers almost the rest of the range. As it can be observed, the granularity is very fine, being able to obtain a summary with a given target length almost with GOP precision. At cluster level, the granularity is slightly coarser, limited by the value of  $N_{exc}$ , which usually is not too large (see Figure 5.8b).

## 5.8. Generation

### 5.8.1. Generation via bitstream extraction

Once the scalable summary is represented by the ranked list **list**, any of the specific summaries at any scale can be extracted from the bitstream of the original sequence. The extracted summary will depend on the constraints imposed by the user or the context. Such constraints must be related to a characteristic function  $h(m')$  that depends on the index of the GOP in the ranked list. In general we can assume that  $h(m')$  is monotonically increasing with the index  $m'$  (and also with the scale  $q$ ). Examples of such functions are the number of images of a storyboard,

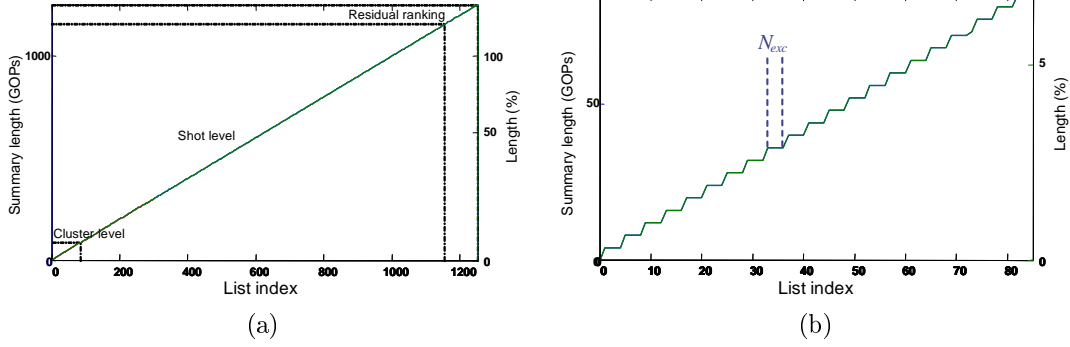


Figure 5.8: Length of the summaries depending on the index of the ranked list: (a) whole ranking, (b) cluster level ranking.

the duration in seconds of a video skim or even the file length.

The first step is the selection of the indexes satisfying a given constraint  $h(m') \leq m'_{max}$ . The cut-off index in the ranked list is

$$m'_{max} = \arg \max_{\substack{h(m') \leq h_{max} \\ 0 \leq m' \leq |\text{list}| - 1}} h(m') \quad (5.14)$$

The set  $I_S$  with the indexes of the summary is obtained as the first  $m'_{max}$  indexes

$$I_S = \{m_0, \dots, m_{m'_{max}-1} | m_{i-1} = \text{list}(m'), 0 \leq m' \leq m'_{max}\} \quad (5.15)$$

Once the set with the indexes of the summary  $I_S$  is determined, the actual bitstream of the summary can be generated. The extractor implements the bitstream extraction operations  $\mathbf{S} = \mathcal{E}_{skim}(\mathbf{V}; I_S)$  for video skims and  $\mathbf{S} = \mathcal{E}_{stb}(\mathbf{V}; I_S)$  for storyboards and extracts the bitstreams as described in Chapter 3.

The output bitstream consist of the selection of the adequate packets from the input bitstream  $\mathbf{V}$ . To generate a video skim, the packets of the whole GOP must be included, while for a stroyboard only the packets of the I frame. Note that both summaries have the form of a compressed video sequence. An appropriate video decoder is required in the client. For storyboards, an intra decoder (for I frames) is necessary, and the decoded frames need to be presented as a collection of images instead of being presented as a sequence.

### 5.8.2. Generation via set of images

Due to compatibility issues, the client may not be capable of decoding video or extracting the frames presenting them as single images. If that happens, the browser will be unable to display not only video skims, but also storyboards. An alternative to bitstream extraction is the use of sets of images. In this case, at the analysis stage, each of the frames belonging to the

storyboard is decoded and stored in a separate file using a suitable format (e.g. JPEG). Note that fast transcoding from MPEG-1/2 intra frames to some image codecs such as JPEG can be achieved processing the bitstream directly in the compressed domain[Acharya and Smith, 1998].

An index corresponding to the GOP number is associated with each image (e.g. in the file name). Then, the generation consists of the selection of the appropriate image files. No video sequence is involved in this case. Thus, legacy decoders and conventional web browsers can be used without any additional capability.

## 5.9. Efficiency and delay analysis

The whole summarization framework, and particularly the analysis stage were carefully designed to be efficient. For a full benefit of scalable representations, efficiency in the generation stage is required, as discussed in Section 5.4.1. However, other characteristic of the algorithm, such as compressed domain analysis, sampling, data reduction and fast clustering algorithm, also lead to efficient analysis. Efficiency in the analysis stage is not required, in principle, as it is performed only once and the results (ranked lists) can be stored as metadata. However, in some scenarios (e.g. broadcasting, live content), analysis cannot be done in advance, and efficient analysis is required for a number of applications, such as low delay summarization.

### 5.9.1. Efficient and low delay summarization

Efficient summarization of video content is a very difficult task, due to the huge amount of data that video sequences contain, compared to other media such as image, audio or text. One of the benchmarks provided by the TRECVID BBC rushes summarization task[Over et al., 2007, 2008] is the creation time. Although the implementations were not optimized and the comparison of creation times was only orientative, this benchmark shows that the majority of the approaches were very inefficient.

Efficient summarization requires both efficient analysis and efficient generation of the bitstream. As discussed in Chapter 3, exploiting coding structure and processing the bitstream in the compressed domain can increase significantly the efficiency of the generation stage. Regarding analysis, some efficient approaches are based on clustering methods[Ferman and Tekalp, 1998]. They provide compact summaries by grouping similar frames or shots into clusters. However, some clustering algorithms may become very slow with long video sequences (e.g. movies, TV news bulletins).

Low delay summarization is a challenging problem, and extremely efficient algorithms are required to keep reasonable the delay between the request and the delivery of the summary itself[Valdés and Martínez, 2010]. For long sequences, much faster processing than real time processing is required. For instance, considering a 10 minutes length video clip at 30 frames per second, if the summary is required to be delivered with a maximum delay of 10 seconds, the system must process the input sequence at 1800 frames per second (30 times real time). Longer videos would require more processing capability for the same delay.

An important limitation of video summarization is the inherent requirement to process all the data first, in order to be able to generate a summary (e.g. a human can hardly summarize a movie if he or she has not watched it completely). The online approach described in [Valdés and Martínez, 2008] relaxes this requirement in order to create summaries with limited delay, generating a video skim at the same time the video is being processed, being suitable for low delay summarization. This algorithm also reported the lowest creation time (excluding the baseline algorithm) in the TRECVID 2008 evaluation, processing about 400 frames per second. The baseline algorithm used in the TRECVID 2008 evaluation, with a trivial analysis stage (the video is simply presented at 50 times normal speed), reported a processing capability of 2300 frames per second.

### 5.9.2. Delay analysis of the algorithm

In the proposed framework, analysis and generation stages are chained in a sequential order, so the total processing delay required is the sum of the delay of both stages:

$$t_{total} = t_{analysis} + t_{generation} \quad (5.16)$$

Note that if the results of the analysis have been computed previously (and stored as meta-data) the processing delay is simply  $t_{generation}$ , as the analysis stage is not required anymore.

The analysis stage is composed by other three stages. Thus, the analysis delay  $t_{analysis}$  is further decomposed in its stages:

$$t_{analysis} = t_{features} + t_{clustering} + t_{ranking} \quad (5.17)$$

The first stage includes bitstream parsing, shot change detection, keyframe sampling and feature extraction.

## 5.10. Experiments

This section presents a series of experimental evaluations of the proposed framework, including objective and subjective tests. For these experiments we implemented the framework for MPEG-1/MPEG-2 coding formats. We have used sequences from different data sets according to the specific requirements of each evaluation. The MPEG-7 color layout was used as low level feature, along with the distance proposed in [Kasutani and Yamada, 2001]. We used  $\alpha_C = 0.2$  and  $\alpha_S = 0.2$ , after testing some values.

### 5.10.1. Shot change detection

In order to evaluate how the GOP length can influence the shot change detection, we tested the proposed method with the sequence *NASASF-TheTechnicalKnockout* from the TRECVID 2005 shot boundary detection corpus [Over et al., 2007]. This sequence has 105661 frames that were encoded using different GOP lengths. The original sequence has 604 cuts and 95 gradual

Sequence		Abrupt					Gradual	Total		
GOP length	#GOPs	#	FP	M	R	P	#	#	R	P
1	105660	604	33	35	0.942	0.945	95	699	0.814	0.945
2	52829	607	45	36	0.940	0.926	92	699	0.816	0.926
4	26414	622	34	47	0.924	0.944	77	699	0.822	0.944
8	13206	629	36	52	0.917	0.941	68	697	0.827	0.941
16	6602	660	48	99	0.850	0.921	33	693	0.809	0.921

Table 5.2: Results of shot change detection experiment. FP: false positives, M: missed, R: recall, P: precision.

transitions. The results are shown in Table 5.2 and were obtained using the ground truth and the evaluation tool provided by TRECVID.

The algorithm has very good results for cuts, up to GOP lengths of 8 frames. For 16 frames the recall drops, but still to an acceptable rate. The decrease in the recall with the GOP length is due to the increase of the temporal distance between consecutive I frames, because the interframe distance become noisier and then it becomes more difficult to detect cuts. No gradual shot change detector was implemented, but, as gradual transitions are less frequent than cuts (of course, it depends on the sequence), the overall performance is acceptable. Another effect of the use of longer GOPs is the shifting of detected transitions from gradual to cuts. It helps to keep the overall precision and recall in similar rates, as some of these new cuts come from gradual transitions not detected at shorter GOP lengths.

The results show a reasonable performance for all the GOP lengths studied in the experiment. However, for longer sizes, GOP processing may not be enough for an effective detection of cuts, and the decoding of the rest of frames (or DC images) would be necessary. If a better performance is required, more sophisticated algorithms would be necessary in order to detect abrupt and gradual transitions with frame precision, at the cost of some processing efficiency.

### 5.10.2. Subjective evaluations

Quality assesment of summaries is a major issue in video summarization, usually requiring long evaluation sessions involving a number of human assessors. It becomes even worse in scalable summarization, as every summary must be evaluated at a number of scales. To evaluate our approach, we conducted an experiment designed to evaluate summaries at several scales. We used the subjective approach followed in many previous works[Zhu et al., 2004; Over et al., 2007, 2008; Ngo et al., 2003], in which some assessors are asked for visualizing the summaries and then asked for answering questions according to some evaluation criteria. Previously, the assessors were asked for visualizing the original sequence. The experiment includes a set of absolute evaluations and another set of relative evaluations, in which two summaries are presented to be compared.

In order to have a reference, we compared the scalable method with a baseline method consisting of sampling the sequence at uniform intervals, selecting, at each sampling point, a frame for storyboards or a short segment for video skims (the same method used in Chapter 4).

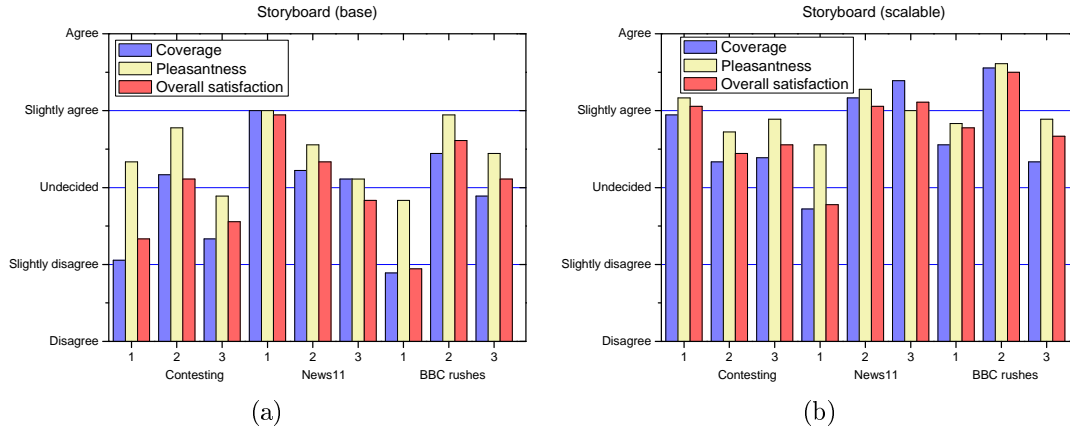


Figure 5.9: Absolute evaluations (storyboards): (a) baseline, (b) scalable.

Although the method is very simple, the summaries are satisfactory enough for many practical applications, particularly for edited content.

The evaluation data set consisted of three segments of 10 minutes each, selected from different corpora, in order to evaluate the method in different contexts. The sequences *contesting* and *news11* belong to the MPEG-7 Content Set[ITU-T and ISO/IEC, 1998]. The sequence *contesting* contains a typical question-answer TV show. The sequence *news11* is a typical news programme, with several news stories alternating with an anchorperson. The last sequence is a segment of the BBC rushes from the TRECVID 2007 corpus[Over et al., 2007] (specifically from the sequence MRS042538). In contrast to the other sequences, rushes consist of unedited footage with different takes of the same scene, so this type of content is much more redundant than edited content.

A total of 36 absolute evaluations results from the combination of all the possible summaries in a four dimensional evaluation space. These four dimensions correspond to the sequence (*contesting*, *news11*, *BBC rushes*), the modality (storyboard or skim), the scale (three scales) and the method (baseline or scalable). The scales for storyboards are 5, 10 and 30 images for *contesting* and *news11*, and 3, 6 and 10 images for *BBC rushes* due to its higher redundancy. The scales for video skims are 6, 11 and 30 seconds for *contesting* and *news11*, and 6, 11 and 25 seconds for *BBC rushes*.

The experiment involves a large number of evaluations and the evaluation of summaries of the same sequence at different scales. In order to minimize the effects of the presentation order and fatigue, we used an experimental design based on a 6x6 Latin square[Bailey, 1996], covering the six possible presentation orders for the three scales and the two summarization methods. Evaluations for each of the three sequences are interleaved in order to minimize the effect of visualizing summaries of the same sequence in a row. Storyboards and skims are also interleaved, so the summary of a given sequence with the given modality is not repeated in six consecutive evaluations. According to this design, 18 volunteers were involved in the evaluation.

For each summary three questions were posed to the assessors. The questions aimed at measuring the quality of the summary according to different criteria, and were formulated as statements. The response scale had five possible values in a typical Likert scale[Likert, 1932],

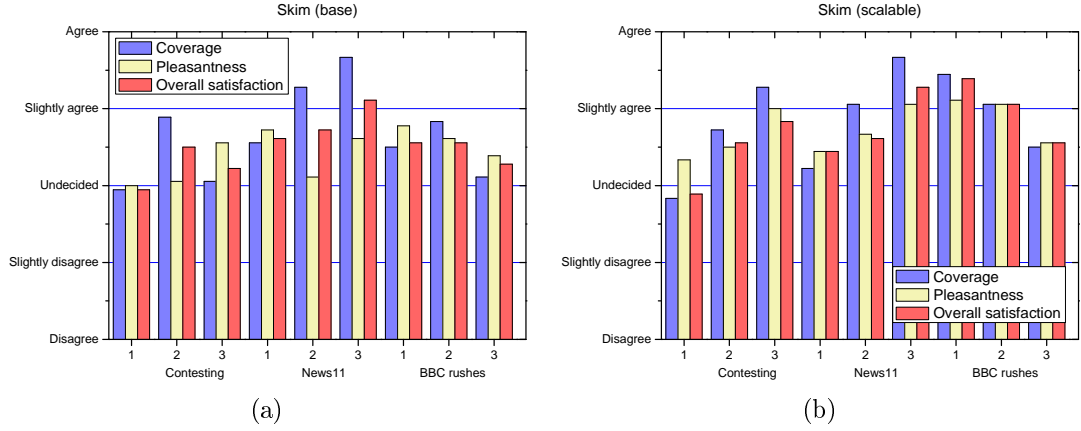


Figure 5.10: Absolute evaluations (skims): (a) baseline, (b) scalable.

ranging from agreement to disagreement with the posed statement. The first question was related to the semantic coverage, the second to the visual pleasantness and the third assessed the overall satisfaction. In contrast to most summarization approaches, the length of a scalable summary is variable and ultimately adjusted by the user, so each summary must be evaluated according to its length. During the briefing session we emphasized this issue to the assessors, that were asked to kept that in mind during the evaluation process. For instance, for information coverage, the assessor is asked if he or she thinks that the information of the sequence is reasonably well covered by the summary, provided that, for example, only the three frames shown are available.

In addition to the previous experiment, another 18 relative evaluations were carried out. A similar experimental design with Latin squares and interleaved sequences was used. The assessor had to visualize two summaries of the same sequence with the same modality and scale, generated by the baseline method and the scalable method. For storyboards, both summaries are displayed simultaneously. For video skims, they were displayed one after the other, not simultaneously, although the assessor had the chance to visualize them again if necessary. Both the layout order and the display order were random so the assessor could not know which one was generated by each method. The items were reformulated to evaluate which of the summaries is preferred according to the same criteria.

The results of the absolute evaluations (see Figure 5.9 and Figure 5.10) show that, in general, the scalable approach generates more satisfactory summaries than the baseline for both semantic and visual pleasantness criteria and also for the overall satisfaction. The results obtained for the scalable method were positive for both storyboards and skims, with different levels. The results for the baseline method were satisfactory for skims, but not for storyboards. The difference between both methods is notable for storyboards while the results for skims are similar, at least in Figure 5.10. However, the relative evaluations (see Figure 5.11) confirm a preference, in general, for the scalable method, except for the sequence *news11*, for which the baseline method is preferred at low scales.

As we can also observe, the results vary depending on the sequence. For the sequence *news11* the results are different because news content is highly edited and structured, with news

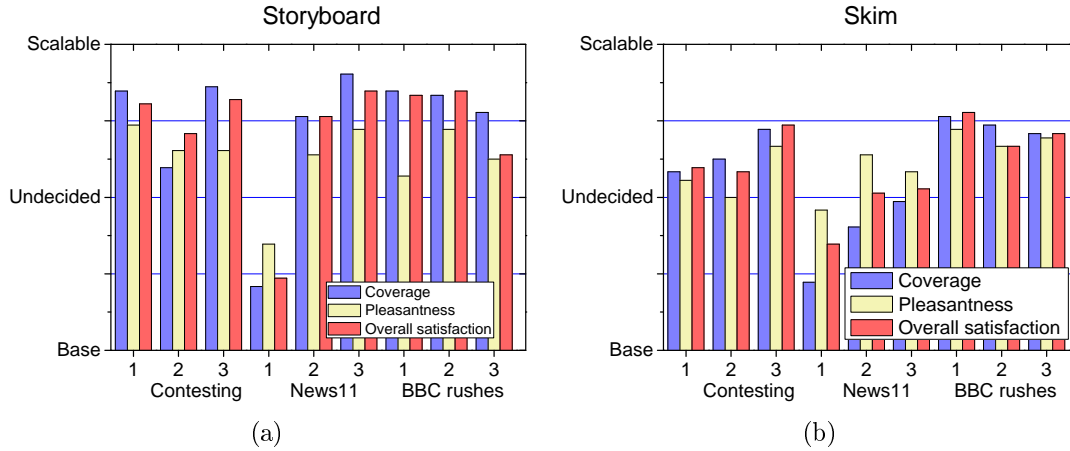


Figure 5.11: Relative evaluations: (a) storyboards, (b) skims.

stories distributed regularly along the sequence. In fact, the baseline method generates good summaries for this kind of structured video, as sampling at regular intervals ensures that the selected frames are not similar and they belong to different news stories, which is especially important at low scales. So implicitly, the baseline algorithm exploits the regular pace and distribution of information within news content. The result is a good summary covering several news stories with few images or video excerpts. For the other videos, representing unedited content or edited but with a less regular structure and more redundancy than news content, the scalable method generates better summaries.

### 5.10.3. Efficiency

We study the efficiency of the algorithm with the sequences used in the previous experiment (full length sequences, ranging from 15 to 32 minutes). In order to study the performance in a high resolution scenario (e.g. digital television, DVD) we also included a high resolution sequence (*dn2002-0228*, obtained from the Internet Archive, 720x480 pixels). The sequences were encoded in MPEG-1 (*dn2002-0228* in MPEG-2) with a GOP length of 13 frames. The sequence was summarized with the proposed algorithm using  $W_{max} = 3$  and  $N_{exc} = 4$ , hierarchical clustering with average linkage and a cut-off linkage distance of 0.18. We also compared hierarchical clustering and  $K$ -means clustering.

Table 5.3 shows the processing times for each of the test sequences for different summaries and lengths<sup>1</sup>. In general, the processing time with this framework is very low, being suitable for low delay summarization, even for the high resolution sequence *dn2002-0228*, which is notably more demanding. Compared to the creation times reported in TRECVID 2008, this framework, using compressed domain, fast analysis and bitstream extraction, can generate summaries significantly faster (in TRECVID 2008 typical sequences have a resolution of 352x240 pixels and 29.97 frames per second and were encoded in MPEG-1, and the target summarization rate was 2%; for an approximate comparison, compare results for skims 1% and 5% in Table 5.3).

<sup>1</sup>Experiments performed in a Intel Pentium M 1.86 Ghz processor (2 GB of RAM)

Sequence		Analysis				Generation			Total			
Name (data set)	Resolution@fps (format)	Duration	$t_{features}$	$t_{clustering}$	$t_{ranking}$	$t_{analysis}$ (fps)	Clusters	Summary	#frames	#GOPs	$t_{generation}$ (fps)	$t_{total}$ (fps)
<i>MSNBC/NEWS13</i> (TRECvid 2005 search)	352x240@29.97 (MPEG-1)	00:27:19.97	4.02	0.03	0.102	4.152 (11838)	146	Storyboard 5	5	5	1.564 (31426)	5.716 (8599)
								Storyboard 30	30	30	1.446 (33991)	5.598 (8780)
								Skim 1%	494	38	1.466 (33527)	5.618 (8749)
								Skim 5%	2457	189	1.566 (31386)	5.718 (8596)
								Skim 20%	9828	756	5.962 (8244)	10.114 (4860)
<i>news12</i> (MPEG-7 content set)	352x288@25 (MPEG-1)	00:15:01.90	2.704	0.002	0.042	2.748 (10076)	79	Skim 100%	49151	3781	44.594 (1102)	48.746 (1008)
								Storyboard 5	5	5	1.228 (22548)	3.976 (6964)
								Storyboard 30	30	30	1.15 (24077)	3.898 (7103)
								Skim 1%	273	21	1.174 (23585)	3.922 (7060)
								Skim 5%	1378	106	1.236 (22402)	3.984 (6950)
<i>contesting</i> (MPEG-7 content set)	352x288@25 (MPEG-1)	00:15:01.90	2.433	0.004	0.035	2.472 (9119)	40	Skim 20%	5538	426	3.92 (7064)	6.668 (4153)
								Skim 100%	27689	2130	36.256 (764)	39.004 (710)
								Storyboard 5	5	5	1.019 (22132)	3.491 (6458)
								Storyboard 30	30	30	0.951 (23702)	3.424 (6585)
								Skim 1%	221	17	0.986 (22861)	3.459 (6519)
<i>MRS012538</i> (TRECvid 2007 rushes)	352x288@25 (MPEG-1)	00:31:54.68	2.652	0.002	0.082	2.735 (10124)	18	Skim 5%	1131	87	1.04 (21680)	3.512 (6419)
								Skim 20%	4511	347	3.83 (5887)	6.302 (3577)
								Skim 100%	22547	1735	27.31 (826)	29.782 (757)
								Storyboard 5	5	5	0.888 (31170)	3.623 (7642)
								Storyboard 30	18	18	0.8 (34611)	3.535 (7833)
<i>dn2002-0228</i> (Internet Archive)	720x480@29.97 (MPEG-2)	00:30:00	10.74	0.008	0.12	10.868 (4964)	69	Skim 1%	481	37	0.813 (34044)	3.548 (7803)
								Skim 5%	2392	184	0.848 (32639)	3.583 (7727)
								Skim 20%	9581	737	3.715 (7453)	6.45 (4293)
								Skim 100%	47866	3683	26.683 (1038)	29.418 (941)
								Storyboard 5	5	5	3.24 (16652)	14.108 (3824)
								Storyboard 30	30	30	3.26 (16540)	14.13 (3818)
								Skim 1%	546	42	3.372 (16000)	14.24 (3789)
								Skim 5%	2704	208	4.758 (11339)	15.626 (3453)
								Skim 20%	10790	830	27.984 (1928)	38.852 (1389)
								Skim 100%	53953	4151	277.208 (104.63)	288.076 (187)

Table 5.3: Processing time (in seconds) for the test sequences.

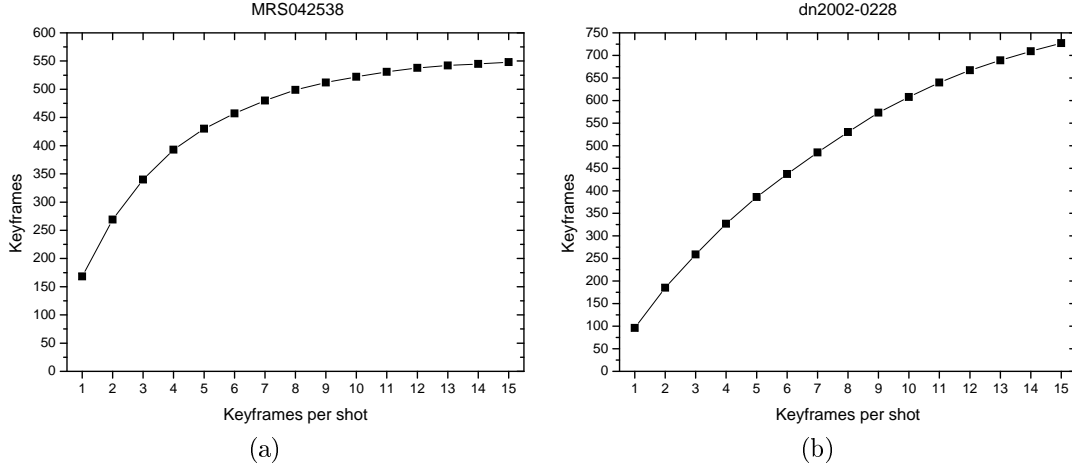


Figure 5.12: Number of keyframes for different values of  $W_{max}$ : (a) *MRS042538* and (b) *dn2002-0228*.

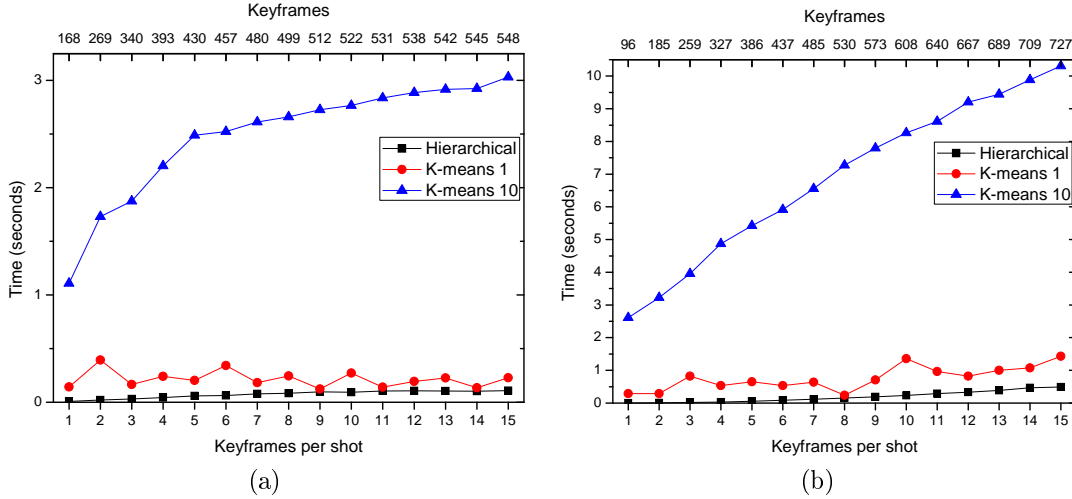


Figure 5.13: Clustering time for different values of  $W_{max}$ : (a) *MRS042538* and (b) *dn2002-0228*.

Both analysis and generation are very efficient. Analysis delay is mostly due to the feature extraction stage, more than 96% of analysis effort. Experiment also shows that clustering and ranking are extremely fast when the data set is reduced properly. It must be noted that the parsing and decoding steps are codec-dependent. DC images can be extracted in coding formats such as MPEG-1 and MPEG-2. Other complex coding formats, such as H.264/AVC need probably complete decoding of frames, which may also lead to less efficient results.

The efficiency of the clustering and ranking stages depends mainly on the amount of data to be processed. Feature extraction effectively reduces the set of data allowing fast clustering and ranking. An important parameter in such data reduction is the number of keyframes per shot  $W_{max}$ . Figure 5.12 shows the relation between the number of total keyframes and the number of keyframes per shot. Figure 5.13 shows how this number of keyframes has effect on the clustering time, for both hierarchical and *K*-means clustering. In order to lessen the

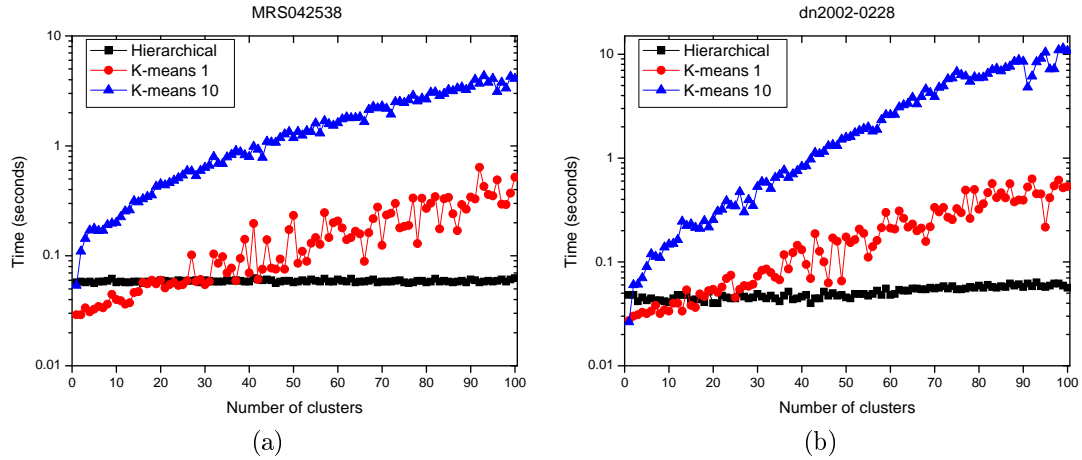


Figure 5.14: Clustering time for different numbers of clusters: (a) *MRS042538* and (b) *dn2002-208*.

dependency on the random initialization, we included in the comparison a variation which runs  $K$ -means ten times and selects the solution with lower clustering error. That results in better and more stable clusterings, although it also implies an important increase in the processing time. Processing time increases approximately linearly, although  $K$ -means grows faster than hierarchical clustering. The number of clusters is another important parameter with effect on the clustering efficiency. Figure 5.14 shows the clustering time for different number of clusters. With hierarchical clustering, processing time remains almost constant, independently of the number of clusters, while with  $K$ -means it increases, being significantly slower when the number of clusters is high. In general, hierarchical clustering is more suitable in this case.

Finally, Figure 5.15 shows how the generation time increases for two sequences. The results are similar to those presented in previous chapters. High resolution sequences usually are encoded in larger bitstreams, requiring more time to process the packets, which results in a higher generation delay (see Figure 5.15b). However, summaries are generated very fast, especially compared to the duration of the sequences (about half an hour in both cases).

## 5.11. Summary and conclusions

This chapter has introduced the concept of scalable summaries, in which the length of the summary can be accommodated to certain specific constraints. We have discussed the role of length in video summaries, as well as its effect over some properties such as semantic coverage and visual pleasantness. The problem of scalable summaries is addressed using an incremental growing approach, in which the summary at each scale is obtained by improving the previous one. This incremental growing approach is implemented using hierarchical clustering and iterative ranking. Addressing also efficient processing, compressed domain analysis is used, in addition to a bitstream extraction framework for MPEG-1/2. Fast generation of the bitstream is crucial to fully benefit from the advantages of scalable representation.

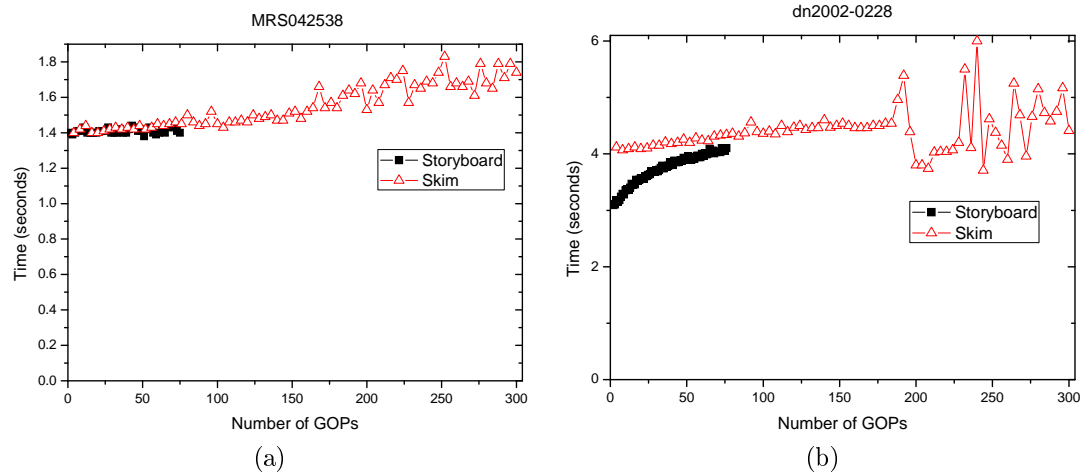


Figure 5.15: Generation time for different lengths of the summary (in GOPs): (a) *MRS042538* and (b) *dn2002-0228*.

Subjective evaluations showed reasonably good results at different scales and for different types of sequences. We also studied the efficiency of both analysis and generation, with encouraging results for low delay summarization, in which conventional summarization techniques cannot be applied due to their high processing burden.

## Chapter 6

# Scalable comic-like summaries

In this chapter we extend the idea of scalable summaries, developed for storyboards and video skims, to a more complex modality: comic-like summaries. This type of summary is based on the popular format of comic strips, which can be exploited to create more intuitive and easily readable summaries. The presence of a non-trivial layout in this modality (which changes with scales) makes the layout algorithm the most critical part of the whole system. We also explore some undesirable effects that appear due to the variable size nature of this modality, and some techniques to make the summaries more pleasant.

### 6.1. Related work

Although traditionally research in summarization is related to storyboards and video skims[Truong and Venkatesh, 2007; Money and Agius, 2008b], there are some recent works proposing new types of presentation in order to create more intuitive summaries. We review some of these works, focusing especially on comic-like summaries.

#### 6.1.1. Advanced pictorial summaries. Beyond storyboards

In contrast to sequence-based summaries such as video skims and fast forwards, pictorial summaries represent video sequences in a static visual abstraction, in which several images are presented simultaneously in a spatial layout according to certain storyline. Among pictorial summaries, the most simple but also most representative is the storyboard, which is basically built with some fixed-size keyframes displayed in a trivial layout. However, other approaches have been proposed to enrich this pictorial information with more effective and visually appealing formats. Composition techniques can be used to create more compact representations which enhance the information of interest. In this section we review some of these approaches (see Figure 6.1).

One of the characteristics of storyboards is the fixed size of the images which suggests that each one conveys equally relevant information. Keeping the size constant, video snapshots[Ma

and Zhang, 2005] display the keyframes in a grid layout, which is decomposed into several groups containing related keyframes. Thus, the user has a compact and structured representation of the video sequence. However, the use of different frame sizes have been studied[Yeung and Yeo, 1997; Uchihashi et al., 1999; Calic et al., 2007; Jansen et al., 2008; Ren and Calic, 2009] to present the salient information in larger images. In [Yeung and Yeo, 1997], the video sequence is summarized as a set of posters, each of them containing a relevant event represented in a variable size template with few images. A similar approach, but containing the whole sequence in a single representation, is the comic-like summary[Uchihashi et al., 1999; Girgensohn, 2003; Calic et al., 2007]. Representation of large sets of keyframes (from long sequences) in a single representation.

Related approaches using variable frame size images[Huynh et al., 2005; Ren and Calic, 2009] have been used in the context of image and video search and browsing (see Figure 6.1f for one example). After selecting a set of keyframes and extracting some video structure, most of these methods can be also used for video abstraction and browsing. Video browsing is closely related to video abstraction, but usually includes some degree of interactivity to move and zoom into some specific parts of the sequence. However, specific image-based methods to browse video content have been developed, such as video trees[Jansen et al., 2008], which summarize the video sequence into a hierarchical layout where few important large images are displayed on the top and which is filled in a top-down fashion with progressively downsized images, temporally related to their neighbors (see Figure 6.1d). The user can zoom into one of the smaller images to obtain another local video tree with detailed information.

Another trend in summarization is the use of regions of interests (ROIs) to obtain more compact summaries[Chiu et al., 2004; Pritch et al., 2008; Mei et al., 2009]. In these methods, salient regions are detected and extracted from the keyframes. The ROIs are combined into a single pictorial representation containing all the salient information from the sequence. The final image can be obtained by fitting the irregular shapes of the ROIs into a layout, as in the stained-glass approach[Chiu et al., 2004], shown in Figure 6.1b. ROIs can be combined in a more appealing way by blending their borders, as in the video collage approach[Mei et al., 2009], shown in Figure 6.1e. ROI-based summarization can be also used in sequence-based summaries to create very compact summaries, as in video synopsis[Pritch et al., 2008], which combines several temporally separated moving ROIs (spatiotemporal tubes) in a single and very compact sequence.

### 6.1.2. Comic-like summaries

Exploiting the narrative structure of comics, comic-like summaries have been proposed as a user-friendly and easily-readable representation of video summaries[Calic et al., 2007]. Being defined as “spatially juxtaposed images in deliberate sequence intended to convey information”[McCloud, 1994], comics are able to use spatial relations of their imagery to convey the notion of time. In contrast to conventional storyboards, the narrative structure of comic-like video summaries is more complex and utilizes images of different sizes, laid out so that the

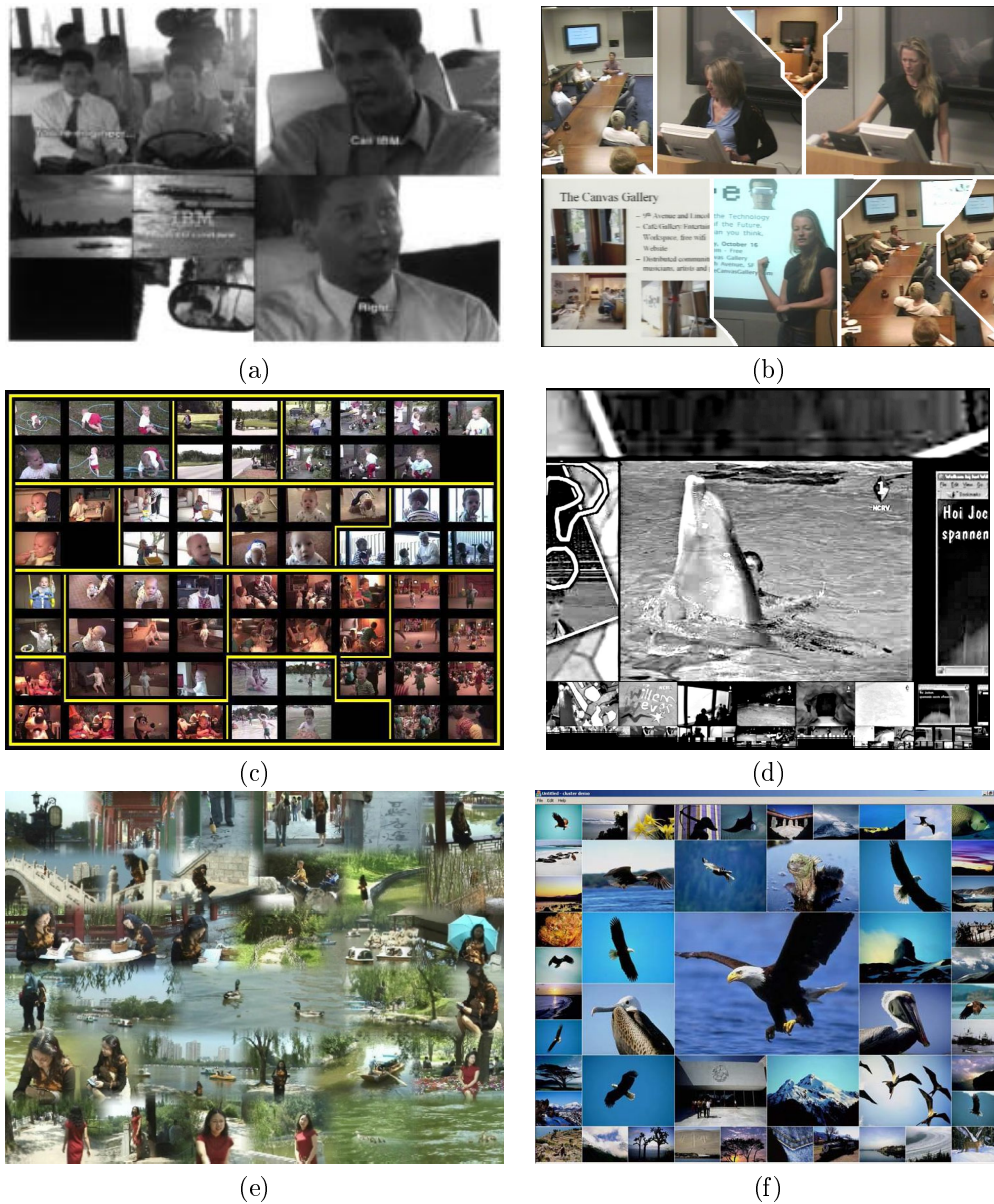


Figure 6.1: Advanced pictorial summaries: (a) video poster (from [Yeung and Yeo, 1997]), (b) stained-glass summary (from [Chiu et al., 2004]), (c) video snapshot (from [Ma and Zhang, 2005]), (d) video-tree (from [Jansen et al., 2008]), (e) video collage (from [Mei et al., 2009]), and (f) Free-eye interface (from [Ren et al., 2010])



Figure 6.2: Comic-like video summary (from [Calic et al., 2007])

position and scale relate to an estimated frame importance.

The first related summarization approach was the video poster[Yeung and Yeo, 1997], proposed as a pictorial representation (with variable image sizes) summarizing the most dramatic incident taking place in a meaningful segment of the video (scene, dialog, action). Video posters, however, do not follow necessarily the temporal structure of the video segment. The pattern of the video poster is selected among a few predefined patterns, according to a set of predefined rules. In addition, each video poster is limited to a maximum of 16 images (as reported in [Yeung and Yeo, 1997]).

Addressing this limitation, a number of methods[Uchihashi et al., 1999; Girgensohn, 2003; Calic et al., 2007] proposed more efficient algorithms capable of generating larger layouts. The problem of optimal image layout in a comic-like visual structure is usually posed as a combinatorial optimization problem. Full search methods can be used to find the optimal solution, but they become impractical when the number of images increases[Uchihashi et al., 1999]. One way of addressing this problem is to apply suboptimal algorithms based on heuristic simplifications[Girgensohn, 2003]. However, as proposed in [Calic et al., 2007], nearly optimal performance can be achieved by utilizing a fast suboptimal algorithm suitable for large layouts due to its linear complexity. Figure 6.2 shows an example of large comic-like summary.



Figure 6.3: Trade-off between information and compactness: (a) compact summary (storyboard), (b) detailed summary with information about temporal evolution (comic-like).

## 6.2. Motivation

As described previously, a storyboard is a sequence of images of the same size, displayed in a temporally ordered manner by their spatial layout (typically from left to right and from top to bottom). The majority of storyboarding algorithms attempt to select as few images as possible, while covering most of the information present in the video sequence. This results in the removal of redundancy by minimizing repetition of similar images.

However, repetition of similar images, specially if frames are sampled regularly, can provide extra information such as the duration or activity of a specific event (e.g. a shot is long if many images from that shot are shown, even if they are very similar). In some cases, this extra information is very useful and it is preferred to a more compact summary, providing more intuitive coverage of every part of the video sequence. In this context, comic-like summaries are very useful, as they can adapt the size of the displayed images according to their relevance. Thus, a salient image (e.g. a keyframe) representing a shot may be surrounded by other smaller auxiliar images which provide additional information about the temporal evolution of that shot.

On the one hand, the storyboard summary provides a compact representation of the dominant content while the repetitive comic-like summaries, on the other hand, can generate overwhelmingly detailed summaries of the video sequence. Figure 6.3 illustrates both cases. The two summarization approaches are therefore complementary. Here, we propose an approach that utilizes scalable comic-like summaries, providing arbitrary levels of detail and length, so that the users can select their desired scale.

We distinguish between two types of applications of scalable comic-like summaries. The first type is the adaptation to some specific constraints, mainly the length of the summary. These constraints may result from different user's preferences or usage contexts. Each user

only visualizes one scale of the summary. The other type includes progressive visualization and interactive navigation. In these applications users visualize multiple scales in a progressive basis, usually from coarse to finer scales. The transition between two scales may result disturbing and uncomfortable if the delay between them is not enough to follow the changes in the layout.

### 6.2.1. Proposed framework

In the proposed video summarisation framework, we conceive comic-like summaries as an extension of conventional storyboards. The coarsest scale of the summary, with the lowest level of detail (few images), is a conventional storyboard, which can be seen as a special case of comic-like summaries with a constant size and a trivial layout. The most important images are selected to form this storyboard. As new images are included in subsequent scales, the summary is enriched with new details completing the flow of temporal events. These images are conveniently scaled according to their importance and laid out into a spatial structure, which becomes more complex as the scale increases.

We define a comic-like summary  $\mathcal{C} = \{\mathbf{Y}, \mathbf{V}\}$  as a pair of layout  $\mathbf{Y}$  and keyframes  $\mathbf{V} = (f_1, f_2, \dots, f_N)$ . For convenience, we introduce  $I_{\mathbf{V}} = \{1, 2, \dots, N\}$  as the set of indexes of  $\mathbf{V}$ . The layout  $\mathbf{Y}$  will be defined further on as a sequence of indexes of panels. Similarly, each scale  $q$  of a scalable comic-like summary consists of a comic-like summary  $\mathcal{C}^{(q)} = \{\mathbf{Y}^{(q)}, \mathbf{V}^{(q)}\}$  with a specific layout  $\mathbf{Y}^{(q)}$  and keyframes  $\mathbf{V}^{(q)}$ , with the corresponding set of indexes  $I_{\mathbf{V}^{(q)}} \subseteq I_{\mathbf{V}}$ , arranged in temporal order. In order to use the same set of keyframes  $\mathbf{V}$  for all the scales, in the mathematical expressions we will use  $I_{\mathbf{V}^{(q)}}$  rather than  $\mathbf{V}^{(q)}$ .

A scalable comic-like summary is a set of comic-like summaries

$$\mathcal{CC} = \{\mathcal{C}^{(1)}, \dots, \mathcal{C}^{(q)}, \dots, \mathcal{C}^{(Q)}\} \quad (6.1)$$

The scalable comic-like summary can be alternatively described as a pair  $\mathcal{CC} = \{\mathbf{YY}, \mathbf{VV}\}$  of a scalable layout

$$\mathbf{YY} = \{\mathbf{Y}^{(1)}, \dots, \mathbf{Y}^{(q)}, \dots, \mathbf{Y}^{(Q)}\} \quad (6.2)$$

and the corresponding scalable set of keyframes

$$\mathbf{VV} = \{\mathbf{V}^{(1)}, \dots, \mathbf{V}^{(q)}, \dots, \mathbf{V}^{(Q)}\} \quad (6.3)$$

with the sets with the indexes of the frames selected for each scale. Keyframes and indexes satisfy

$$\mathbf{V}^{(1)} \subset \mathbf{V}^{(2)} \subset \dots \mathbf{V}^{(q)} \subset \dots \subset \mathbf{V}^{(Q)} \subseteq \mathbf{V} \quad (6.4)$$

$$I_{\mathbf{V}^{(1)}} \subset I_{\mathbf{V}^{(2)}} \subset \dots I_{\mathbf{V}^{(q)}} \subset \dots \subset I_{\mathbf{V}^{(Q)}} \subseteq I_{\mathbf{V}} \quad (6.5)$$

The scheme of the proposed scalable comic-like summarization framework is shown in Figure 6.4. The images of the summary are first extracted as the set of keyframes  $\mathbf{V}$  from the input

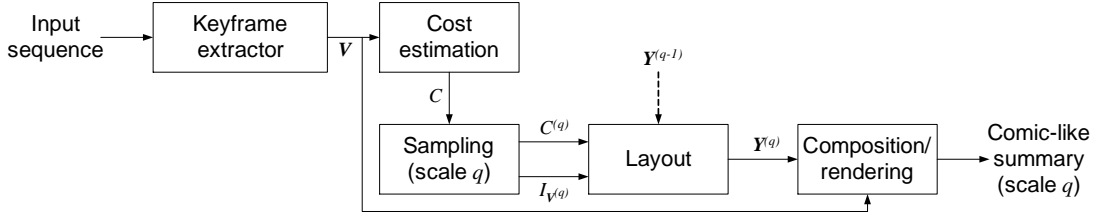


Figure 6.4: Architecture of the proposed framework.

video sequence. The importance of each keyframe is estimated and stored in the cost function  $C$ . The cost must be proportional to the expected area covered by the keyframe in the summary. Based on the cost function and the scale  $q$ , the sequences of keyframes and cost values are sampled into the subsets  $\mathbf{V}^{(q)}$  and  $C^{(q)}$ . The layout algorithm then computes the layout  $\mathbf{Y}^{(q)}$  for that scale. Finally, with this information, the summary can be composed and rendered.

### 6.2.2. Keyframe extraction

The objective of the initial keyframe extractor is to obtain the set of images to be used in the rest of the system, as comic-like summaries are created from a set of images. Although keyframe extraction for summarization has been extensively studied [Truong and Venkatesh, 2007], we assume a uniform sampling of the input sequence, as we are not interested on removing semantic redundancies at this stage, while we are interested on covering regularly all the sequence. This eventual oversampling is not important as redundant details (frames) will be removed by the subsequent stages, and included progressively as the scale increases.

## 6.3. Estimation of frame sizes

In comic-like summaries, the area covered by a keyframe in the layout represents its summarization significance, which is estimated and stored in the cost function, defined as

$$C = (C_n | n \in I_{\mathbf{V}}) \quad (6.6)$$

where  $C_n \in [0, 1]$ . The cost is estimated using the method proposed in [Calic and Campbell, 2007].

In order to evaluate the cost function in a way that will support the user's visual experience of the final layout, a clustering algorithm based on perceptual similarity is used. Although we could have used any of the clustering algorithms used in the preceding chapter ( $K$ -means or hierarchical clustering), we used the same clustering algorithm as [Calic and Campbell, 2007], which was already used in the context of comic-like summaries with satisfactory results. The algorithm is an efficient graph based method described in [Felzenszwalb and Huttenlocher, 2004], initially proposed for image segmentation. This approach enables unsupervised analysis of the inherent structure of the keyframes and it copes well with nonlinearity of cluster shapes. The clustering algorithm represents the set of keyframes as a graph  $G = (\mathbf{V}, \mathbf{E})$ , in which the vertices

$v \in V$  are the keyframes, while the keyframe differences are assigned to edges  $(v_i, v_j) \in E$  as weights  $w(v_i, v_j)$ . Here, we calculate the keyframe difference as the chi-square difference of  $18 \times 3 \times 3$  HSV colour histograms, though the algorithm is invariant to other difference metrics.

An important characteristic of the method is its ability to preserve detail in low variability clusters while ignoring detail in high variability sets. This algorithm runs in time nearly linear to the number of graph edges, and though we have taken into consideration a fully connected keyframe set, due to the relatively small number of keyframes, the processing is very fast. The algorithm starts with all vertices comprising their own cluster component  $a_i$ , iterating over all the edges (sorted in non-decreasing order), and evaluating if the components linked by a given edge can be merged. This is decided by comparing the inter and intra cluster differences. Once there is no merging of the components between consecutive iterations, the resulting components serve as data clusters.

When evaluating an edge  $(v_i, v_j)$ , the condition to merge two components  $a_k$  and  $a_m$ ,  $a_k$  containing  $v_i$  and  $a_m$  containing  $v_j$ , is that they were not merged before (i.e.  $a_k \neq a_m$ ), and that they satisfy  $w(v_i, v_j) \leq \Delta_{min}(a_k, a_m)$ . The minimum internal difference  $\Delta_{min}(a_k, a_m)$  is defined as

$$\Delta_{min}(a_k, a_m) = \min(\Delta(a_k) + \tau(a_k), \Delta(a_m) + \tau(a_m)) \quad (6.7)$$

where  $\Delta(a_k)$  is the internal difference of the component  $a_k$ , defined as the largest weight in the minimum spanning tree of the component. The threshold function  $\tau(a_k) = \frac{\gamma}{|a_k|}$ , where  $\gamma$  is a constant parameter and  $|a_k|$  denotes the size of  $a_k$ , controls the degree to which the difference between the two components must be greater than their internal differences. For more details about the clustering algorithm, the reader is referred to [Felzenszwalb and Huttenlocher, 2004].

In order to visualize the dominant content of the selected section of video, each cluster is represented with the frame closest to its centroid. Therefore the highest cost function  $C_n = 1$  is assigned to  $d_n = 0$ , where  $d_n$  is the distance of the keyframe closest to its centroid and  $\sigma_n$  is the cluster variance of the  $n$ th keyframe. Other members of the cluster are given values

$$C_n = \alpha \left( 1 - e^{-\frac{d_n^2}{2\sigma_n^2}} \right) \quad (6.8)$$

By doing this, cluster outliers (i.e. cutaways, establishing shots, etc.) are presented as more important and attract more attention of the user than keyframes concentrated around the cluster centre. The value of  $\alpha$  controls the balance between the importance of the cluster centre and the outliers. In our experiments we chose  $\alpha = 0.7$ , after some preliminary tests.

This grouping around cluster centroids is due to common repetitions of similar content in video sequences, often adjacent in time. To avoid the repetition of content in the final summary, a set of similar frames is represented by a larger representative, while the others are assigned a lower cost function value. It fits very well into the approach of comic-like summaries as enhanced storyboards (in this case composed of the representatives of each cluster, i.e. frames with cost  $C_n = 1$ ).

## 6.4. Single scale layout

Following the model of a typical narrative structure of a comic strip, the time flow of the video sequence is reflected by ordering the keyframes in a left-to-right and top-to-bottom fashion. For simplicity, we consider only single row layouts. If the summary becomes too long for a single row, the browsing device splits it into several rows. In this section, some basic units and notations are introduced, along with the methods addressing the problem of layout generation given a cost function.

### 6.4.1. Panel template generation

Following the definition of comics as a sequential art where space has the same role as time has for film [McCloud, 1994], this work intuitively transforms the temporal dimension of videos into the spatial dimension of the final summary by following the rules of comic narrative structure.

The panel is the basic spatial unit of comics, and therefore, of comic-like summaries. It distinguishes an ordered pictorial sequence conveying information from a set of images. The summary is composed by laying out the images following a sequence of panels, each of them based on a panel template. Panels in the summary layout also need to follow the basic rules of comic narrative structure (e.g. time flows from left to right, and from top to bottom).

Let denote a panel as a pair  $P = \{p, I_P\}$ , where  $p$  indicates the number of panel template and  $I_P$  the sequence of indexes of the keyframes in the panel. The panel template is a sequence of frame sizes  $\mathbf{T}_p = (\Omega_1, \Omega_2, \dots)$ , where  $\Omega_n \in \{1, 2, \dots, h\}$  is the (normalized) size of the  $n$ th keyframe of the panel, and  $h$  is the height of the panel. Let  $|\mathbf{T}_p|$  denote the length of the panel template  $p$ , and  $|\mathbf{T}|$  the number of available panels. The set of panel templates for a given height  $h$  is finite, and must be computed prior to the layout [Calic and Campbell, 2007]. Table 6.1 shows all the possible panel templates of height  $h = 4$ .

### 6.4.2. Optimal solution using full search









The main task of the layout algorithm is to find a layout that optimally follows the values of the cost function using only sizes available in panel templates. Each panel template generates a vector of frame sizes that approximates the cost function values of corresponding frames.

A layout is a sequence of  $M$  panel templates  $\mathbf{Y} = (p_1, p_2, \dots, p_M)$  following a temporal structure, which is read according to comic narrative rules. Unfolding the layout, the sequence  $\Omega$  of frame sizes is obtained as

$$\Omega = \Omega(\mathbf{Y}) = \left( \overbrace{\Omega_1, \Omega_2, \dots, \Omega_{|\mathbf{T}_{p_1}|}}^{\mathbf{T}_{p_1}}, \overbrace{\Omega_{|\mathbf{T}_{p_1}|+1}, \dots, \Omega_n}^{\mathbf{T}_{p_2}}, \dots, \overbrace{\Omega_n, \dots, \Omega_N}^{\mathbf{T}_{p_M}} \right) \quad (6.9)$$

fitting the  $N$  frames of the summary. The indexes of the keyframes of each panel  $I_{P_m}$  are also

Table 6.1: Panels templates for  $h = 4$ 

Template $T_p$	$\Omega$	Length	Width	Area	Layout
$T_1$	(1, 1, 1, 1)	4	1	4	
$T_2$	(2, 2)	2	2	8	
$T_3$	(2, 1, 1, 1, 1)	5	2	8	
$T_4$	(1, 1, 2, 1, 1)	5	2	8	
$T_5$	(1, 1, 1, 1, 2)	5	2	8	
$T_6$	(3, 1, 1, 1)	4	3	12	
$T_7$	(1, 1, 1, 3)	4	3	12	
$T_8$	(4)	1	4	16	

selected as a partition of the initial set of indexes  $I_V$  according to panel lengths

$$I_V = \left( \overbrace{1, 2, \dots, |T_{p_1}|}^{I_{P_1}}, \overbrace{|T_{p_1}| + 1, \dots, n}^{I_{P_2}}, \overbrace{n, \dots, N}^{I_{P_M}} \right) \quad (6.10)$$

The layout optimization problem consists of finding a layout minimizing the layout error for a given a cost function  $C_n$

$$\mathbf{Y}^* = \arg \min_{\mathbf{Y}} \varepsilon(\mathbf{Y}) \quad (6.11)$$

The (normalized) panel error is computed as

$$\varepsilon(P) = \sum_{i=1}^{|T_p|} \left( C_{n_0+i-1} - \frac{\Omega_i^2}{h^2} \right)^2 \quad (6.12)$$

where  $n_0 = \min_{n \in I_P} n$  is the index of the first frame of the panel.

The layout error is then computed from individual panel errors as

$$\varepsilon(\mathbf{Y}) = \sum_{m=1}^M \varepsilon(P_m) \quad (6.13)$$

The full search algorithm explores all the possible combinations of panels making up valid layouts. Note that the number of panels  $M$  is unknown until the optimal solution is found. The basic algorithm comprises the following steps:

1. Generate all possible panel templates of height  $h$ .
2. Set current keyframe  $n = 1$ , current layout  $\mathbf{Y}_0 = \emptyset$ , accumulated error  $\varepsilon_0(\mathbf{Y}) = 0$  and

minimum error  $\varepsilon_{min} = \infty$

3. For every available template  $\mathbf{T}_k$ ,  $k = 1, \dots, |\mathbf{T}|$  do
  - a) If  $n > N$  discard this solution and stop the recursion for this branch.
  - b) Compute the error  $\varepsilon_m(\mathbf{Y}) = \varepsilon_{m-1}(\mathbf{Y}) + \varepsilon(P)$ , using the template  $\mathbf{T}_k$  and  $n$  as the first keyframe of the panel.
  - c) Store the candidate layout  $\mathbf{Y}_m = (\mathbf{Y}_{m-1}, k)$
  - d) Set  $n = n + |\mathbf{T}_k|$
  - e) If  $n = N + 1$  (all keyframes selected).
    - 1) If  $\varepsilon_m(\mathbf{Y}) < \varepsilon_{min}$ , then set  $\varepsilon_{min} = \varepsilon_m(\mathbf{Y})$  and set  $\mathbf{Y}_m$  as the current solution.
    - 2) Stop the recursion for this branch.
  - f) Repeat recursively the loop of step 3.

The search space is a tree in which every node has  $|\mathbf{T}|$  child nodes. The main problem of full search is that, as  $N$  increases, the number of potential layouts grows exponentially, becoming impractical for more than few dozens of frames.

### 6.4.3. Optimized full search

Search algorithms in tree structures can be optimized using some simple strategies. Here, we present two optimized versions of the search algorithm.

The first version uses branch pruning, evaluating  $\varepsilon_m(\mathbf{Y})$  at each node (not only at leaf nodes). If  $\varepsilon_m(\mathbf{Y}) \geq \varepsilon_{min}$  then the recursion is stopped for that branch. In the worst case, the number of combinations does not change, but in most cases the recursion is stopped early, reducing dramatically the number of tested layouts.

The second version uses also a look-up-table (LUT), in order to avoid most of the operations in computing  $\varepsilon(P)$ . The panel error for each combination of keyframe  $f_n$  and template  $\mathbf{T}_k$  is precomputed as

$$\varepsilon(n, k) = \begin{cases} \sum_{i=1}^{|\mathbf{T}_k|} \left( C_{n+i-1} - \frac{\Omega_i^2}{h^2} \right)^2, & n + |\mathbf{T}_k| \leq N \\ \infty, & \text{otherwise} \end{cases} \quad (6.14)$$

with  $n = 1, \dots, N$  and  $k = 1, \dots, |\mathbf{T}|$  (e.g.  $|\mathbf{T}| = 8$  in the case of  $h = 4$ ). Thus, instead of using (6.12), the error of each panel can be evaluated directly as  $\varepsilon(P) = \varepsilon(n_0, p)$ . The number of the evaluations of the panel error in (6.12) becomes linear  $N_{LUT} = |\mathbf{T}|N$ . For large layouts, where  $N_{full} \gg N_{LUT}$ , LUT can reduce significantly the amount of operations in panel error evaluations.

Unfortunately, the number of potential layouts still grows exponentially, even using pruning and LUT, so this optimized algorithm still becomes impractical for a large number of frames. Figure 6.5 shows the result of an experimental comparison of the analyzed layout methods<sup>1</sup>.

<sup>1</sup>Experiment performed on an Intel Pentium M at 1.8 Ghz (2 GB of RAM).

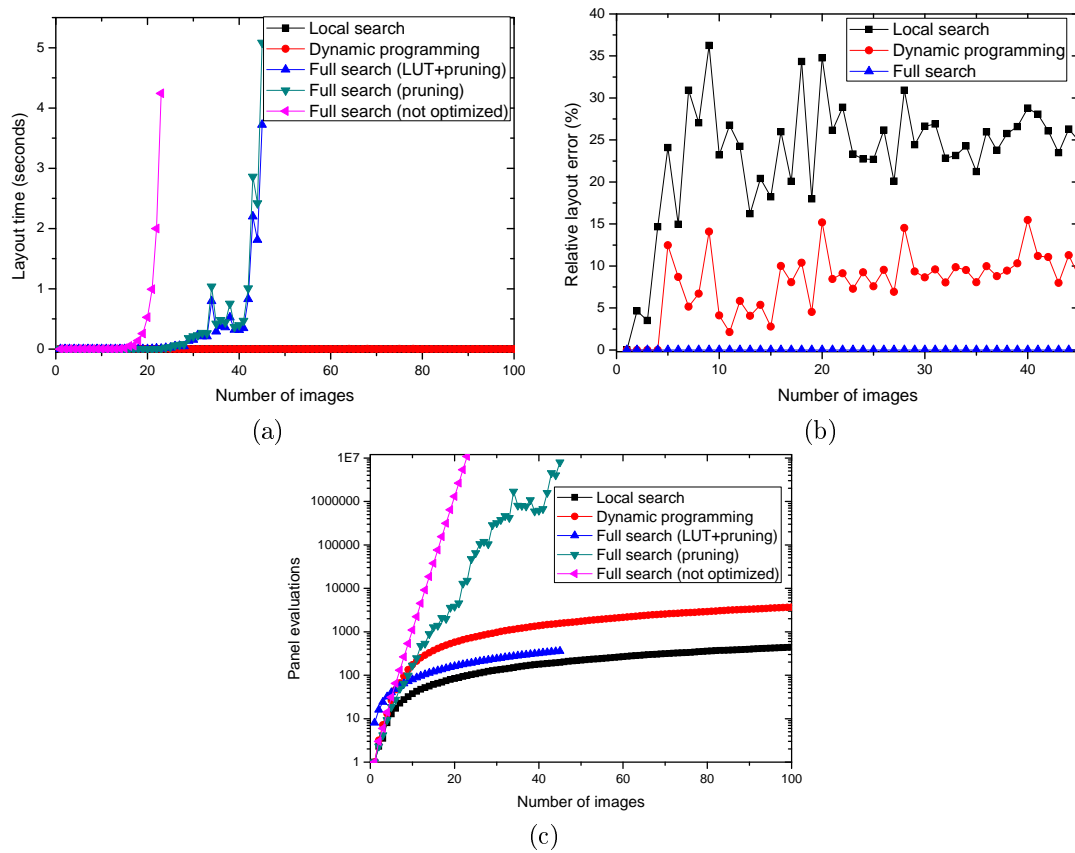


Figure 6.5: Comparison of layout methods: (a) processing time, (b) difference to the optimal layout error, and (c) number of panel evaluations.

The complexity of the different full search methods can be estimated empirically by exponential regression from Figure 6.5a, with full search, full search with pruning and full search with pruning and LUT complexities being  $O(2.14^n)$ ,  $O(1.3034^n)$  and  $O(1.3031^n)$ , respectively.

#### 6.4.4. Suboptimal solutions

Many problems in combinatorial optimization become intractable when the size of the data set increases. The majority of algorithms proposed to solve these problems are based on heuristics and do not offer optimal solutions, yet offering satisfactory results in practice. Therefore, we describe two suboptimal algorithms based on taking early decisions and we will show that the deviation of the achieved results from the optimal solution is negligible. However, with these suboptimal algorithms, the solution can be computed in linear time instead of exponential time.

The local search algorithm selects each panel looking for the local minimum of the panel error for each tree level. The algorithm comprises the following steps:

1. Generate all possible panel templates of height  $h$ .
2. Set current layout as  $\mathbf{Y}_0 = \emptyset$ ,  $m = 1$ , and  $n = 1$
3. While  $n \leq N$  do
  - a) Set  $k^* = \arg \min_k \varepsilon(P_m)$ , testing all the available templates  $\mathbf{T}_k$  and using  $n$  as the first keyframe of the panel.
  - b) Store the solution  $\mathbf{Y}_m = (\mathbf{Y}_{m-1}, k^*)$
  - c) Set  $n = n + |\mathbf{T}_{k^*}|$
  - d) Set  $m = m + 1$

In contrast to full search, the decision about a candidate solution is not global and taken at leaf nodes, but local and taken at each tree level. The main drawback is that panels are considered independently.

Taking into account adjacent panels, the suboptimal solution approximates better the optimal solution, while still keeping the algorithm very efficient. Thus, the second suboptimal method extends the search space from one to two adjacent panels. This method is equivalent to the dynamic programming method described in [Calic et al., 2007]. To implement this method, step 3a in the previous algorithm must be replaced by

- (a) Set  $k^* = \arg \min_k [\varepsilon(P_m) + \varepsilon(P_{m+1})]$ , testing all the combinations of available templates and using  $n$  as the first keyframe of the panel.

In this method, all the possible combinations of two panels  $P_m$  and  $P_{m+1}$  are tested to decide which is the optimal (i.e. lowest error), and the panel template of the first panel of the optimal combination is selected as local solution for that level. For instance, for  $h = 4$ , the local search method tests 8 possible panels, while the dynamic programming method tests 64 combinations.

As Figure 6.5b shows, suboptimal methods have a penalty in terms of layout error compared to the optimal method. However, both methods perform reasonably well, and the layout error differs by around 25% for the local search method. For the dynamic programming method this difference is reduced to around 7%, and in practice, the layout is very similar to the optimal. In contrast, as shown in Figure 6.5a, both suboptimal methods are extremely fast and grow in linear time, so they can handle large input sets. Thus, dynamic programming provides a very fast layout algorithm with little penalty in the layout error.

## 6.5. Multiscale layout

### 6.5.1. Independent layouts

The simplest approach to generate scalable summaries is to assume that each scale is independent from the rest. The only constraint we assume is that the images in a given scale  $q$  are present in the following scales  $q' > q$ . In general, the scalable layout algorithm has two stages: *keyframe sampling* and *layout computation*. In this case, the layout algorithm is directly the single scale layout algorithm, applied independently over each scale.

The keyframe sampling algorithm selects a subset of indexes from  $I_{\mathbf{V}}$  and their cost values according to some sampling strategy. The cost function is also sampled to match  $I_{\mathbf{V}^{(q)}}$ , i.e.,  $C^{(q)} = (C_n | n \in I_{\mathbf{V}^{(q)}})$ . Then, the layout algorithm is applied in this subset to obtain the layout  $\mathbf{Y}^{(q)}$ . We use a simple cost-based sampling strategy: those indexes with the  $N^{(q)}$  highest cost values are selected, where  $N^{(q)}$  is the number of images in the scale  $q$ .

At this point, an example may be useful to illustrate the different elements of a multiscale comic-like summary. Let us consider an initial set of keyframes with 20 images and the two scales represented in Figure 6.6. The scale  $q$  is described by the layout  $\mathbf{Y}^{(q)} = (T_2, T_8, T_2, T_8)$ , with frame sizes  $\Omega^{(q)} = \Omega(\mathbf{Y}^{(q)}) = (2, 2, 4, 2, 2, 4)$ , and indexes of keyframes  $I_{\mathbf{V}^{(q)}} = (2, 3, 8, 12, 15, 19)$ . In the next scale, the values are  $\Omega^{(q+1)} = \Omega(\mathbf{Y}^{(q+1)}) = (2, 2, 2, 2, 4, 1, 1, 1, 1, 4)$ ,  $\mathbf{Y}^{(q+1)} = (T_2, T_2, T_8, T_1, T_8)$  and  $I_{\mathbf{V}^{(q+1)}} = (2, 3, 8, 11, 12, 15, 16, 17, 18, 19)$ .

The layout problem is analogous to the single scale case, but using the data resulting from keyframe sampling ( $I_{\mathbf{V}^{(q)}}$  and  $C^{(q)}$ ) instead of the whole set.

$$\mathbf{Y}^{(q)} = \arg \min_{\mathbf{Y}} \varepsilon(\mathbf{Y}) \quad (6.15)$$

$$\varepsilon(\mathbf{Y}) = \sum_{m=1}^M \varepsilon(P_m^{(q)}) \quad (6.16)$$

$$\varepsilon(P^{(q)}) = \sum_{i=1}^{|T_{P^{(q)}}|} \left( C_{n_0+i-1}^{(q)} - \frac{\Omega_i^2}{h^2} \right)^2 \quad (6.17)$$

with  $n_0 = \min_{n \in I_{P^{(q)}}} n$ .

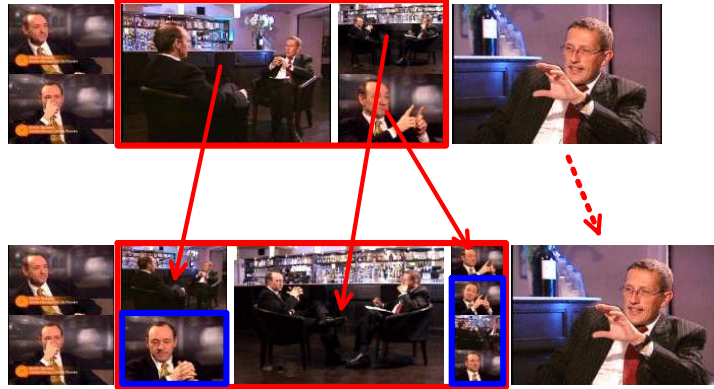


Figure 6.6: Example of transition between two consecutive scales.

Note that the layout problem must be solved for every scale, so the algorithm has to compute  $Q$  layouts. For this reason, an efficient method to solve each individual problem would be desirable. In order to balance both efficiency and limited layout error we use the dynamic programming search method described in Section 6.4.4[Calic et al., 2007].

This first approach to scalable layouts is enough for a number of scenarios in which the user interacts with a single scale of the summary. Summary adaptation is an example of application using independent scales, as the user gets a scaled version of the summary according to user's preferences or constraints in the usage environment (e.g. limited display area in the screen).

### 6.5.2. Disturbance

In some applications (e.g. progressive visualization or interactive browsing across scales), users have to visualize several scales in short time intervals. We observed that the main problem in these applications was to follow the changes in transitions between scales. Transitions between consecutive scales might become disturbing and uncomfortable, as some images may change their position and size in the new layout, and new panels may appear or disappear (see the example in Figure 6.6). Even if some panels are not modified, they can be pushed by others so they suffer a displacement, which may be also unpleasant if it is large or involves row changes. If these changes are scattered all over the summary and the delay between scales is too short, it becomes difficult to follow them. These undesirable effects may also distract users from tracking the new information (new images) added in the new scale, which should be the main objective. We call this annoying effect *layout disturbance*.

## 6.6. Heuristic approach to multiscale layout

As discussed before, layout disturbance is a major problem in applications requiring transitions between scales, and minimizing it is key to benefit from scalable comic-like summaries in these applications. In this section we propose a heuristic algorithm, based on the concept of anchor keyframes, which can create more pleasant transitions between scales.

### 6.6.1. Anchor keyframes

This heuristic algorithm is based on the idea of comic-like summaries as enhanced storyboards. From that point of view, the main summary is the storyboard, covering the main semantics in few images. The rest of the images are complementary, adding more information about the temporal evolution of the sequence and the duration of events. We call the keyframes belonging to the original storyboard (i.e. those with cost  $C_n = 1$ ) *anchor keyframes*.

In the heuristic algorithm, we use this idea to add some conditions to the sampling and layout algorithms:

- Anchor keyframes are considered especially relevant and must not change their size across scales, being always  $h$  and thus presented in a single-image panel.
- The layout algorithm is not applied to the whole sequence of keyframes, but only to segments between those anchor keyframes with new keyframes in-between.

Note that these conditions also help to limit the disturbance, as the number of changes from one scale to the next are restricted by design to a part of the layout.

### 6.6.2. Temporally constrained sampling

The problem of cost-based sampling is that when a number of keyframes are sampled for a new scale, they can be located at any position in the sequence, and consequently new images (and panels) can appear at any place in the layout, far from each other. That is the main source of disturbance, as it is more difficult to follow changes in the layout when they are spread all over. A sampling strategy that takes into account the temporal order of keyframes is more suitable to avoid disturbance. The objective of the temporally constrained sampling is to include new batches of keyframes not only based on the cost function but also on a temporal neighborhood. Thus, changes can be localized in a temporal window which results in a small spatial area in the summary.

At each scale  $q$ ,  $M^{(q)}$  new keyframes are sampled and included in the new summary

$$M^{(q)} = \begin{cases} N^{(q)} & q = 1 \\ N^{(q)} - N^{(q-1)} & q \neq 1 \end{cases} \quad (6.18)$$

The sampling algorithm tries to select a batch of  $M^{(q)}$  keyframes in a relatively short temporal interval, but all of them having a reasonably high cost. Anchor keyframes are the boundaries of these intervals. The first scale always returns the set of anchor keyframes. For the subsequent scales, the set of indexes  $I_V$  is divided into  $L$  intervals, and a bin  $H_k$  is assigned to each interval  $k$ . Figure 6.7 depicts the sampling strategy, which comprises the following steps:

1. Initialize histogram as  $H_k = \emptyset$ ,  $k = 1, \dots, L$ .
2. Sort the set of unselected keyframes at scale  $q$  by cost and let  $I_{V^*}$  be the sequence of their indexes in decreasing cost order.

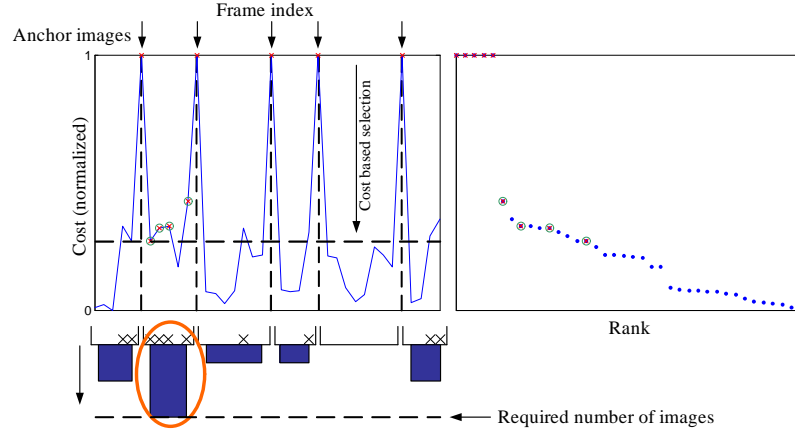


Figure 6.7: Illustration of the temporally constrained sampling algorithm.

3. Loop over  $I_{V^*}$  until no more indexes are available. Let  $n^*$  be the first index in  $I_{V^*}$ 
  - a) Find the interval  $k^*$  corresponding to  $n^*$  and set  $H_{k^*} = H_{k^*} \cup n^*$
  - b) If  $|H_{k^*}| = M^{(q)}$  then go to step 6.
  - c) If any available index in  $I_{V^*}$ , let  $n^*$  be the next index in  $I_{V^*}$  and continue to step 3a.
4. Combine pairs of consecutive intervals so the histogram bins are  $H'_k = H_k + H_{k+1}$ 
  - a) If any  $|H'_k| \geq M^{(q)}$  then  $k^* = \arg \min_k |H'_k|$  and go to step 6.
5. If there is no interval satisfying  $|H'_k| \geq M^{(q)}$ , then continue combining intervals in increasing number (three intervals, then four, etc.)
6. Set  $I_{V^{(q)}} = I_{V^{(q-1)}} \cup H_{k^*}$  and  $C^{(q)} = (C_n | n \in I_{V^{(q)}})$

Intuitively, the algorithm selects keyframes according to their cost in descending order, and tracks the number of keyframes selected from every interval. If one of the intervals reaches the number of required keyframes, the keyframes sampled in that interval are selected. If, after that first loop, there is not any interval with enough keyframes, adjacent intervals are combined and checked again.

### 6.6.3. Anchor based layout

As the changes in the layout only happen in a segment bounded by two consecutive anchor keyframes, the layout algorithm is run only in that segment. Thus, the rest of the layout remains unchanged, and the only change the user may perceive is the possible displacement due to other panels pushing them. Without loss of generality, the layout  $\mathbf{Y}^{(q-1)}$  can be expressed as

$$\mathbf{Y}^{(q-1)} = \left( \mathbf{Y}_l^{(q-1)}, T_{anchor}, \mathbf{Y}_m^{(q-1)}, T_{anchor}, \mathbf{Y}_r^{(q-1)} \right) \quad (6.19)$$

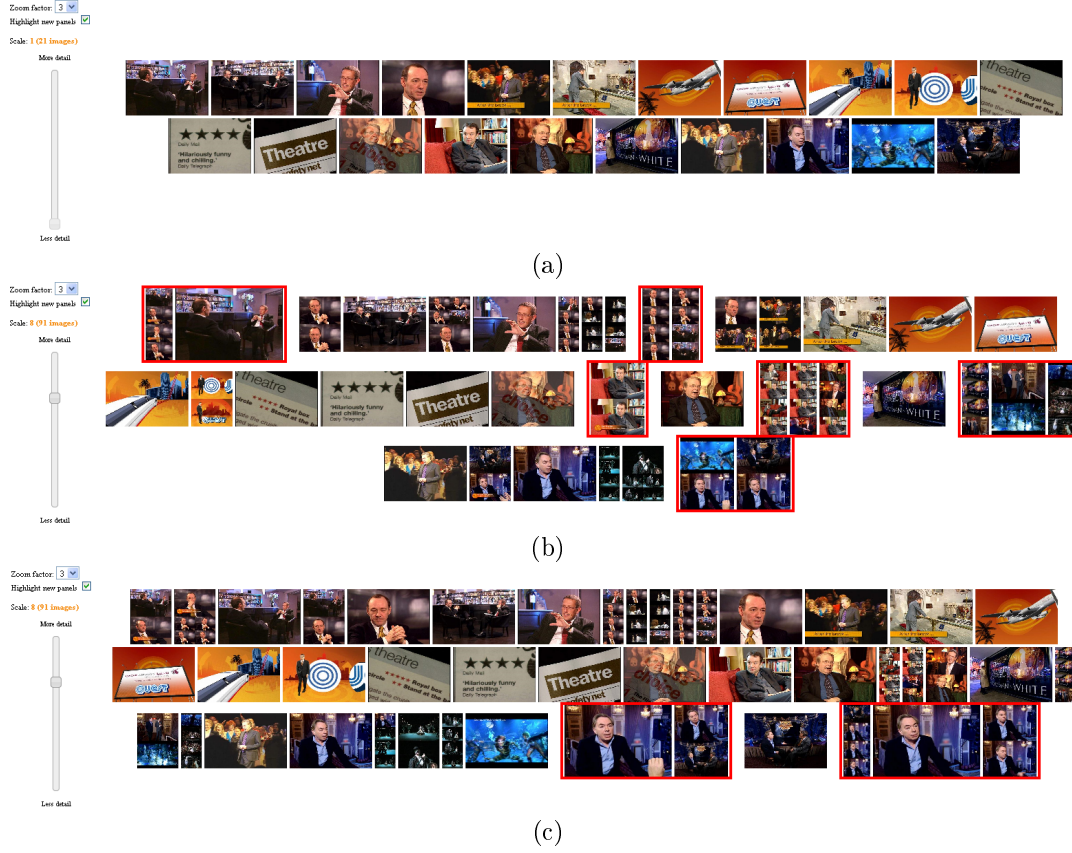


Figure 6.8: Examples of comic-like summaries: (a) scale 1 (*basic* and *anchor*), (b) scale 8 (*basic*), (c) scale 8 (*anchor*). New panels are highlighted.

where  $\mathbf{Y}_l^{(q-1)}$ ,  $\mathbf{Y}_m^{(q-1)}$  and  $\mathbf{Y}_r^{(q-1)}$  are the partial layouts at left, in-between and right of the anchor keyframes bounding the segment with new keyframes sampled at current scale  $q$ . These partial layouts are separated by two panels  $\mathbf{T}_{anchor}$ , which represents the panel template of height  $h$  (e.g. for  $h = 4$ ,  $\mathbf{T}_{anchor} = \mathbf{T}_8$  from Table 6.1), containing the anchor keyframes. If  $\tilde{C}$  denotes the associated cost function to the keyframes between both anchor keyframes, and  $\mathbf{Y} = \text{layout}(C)$  denotes the operation of computing the layout for a cost function  $C$ , the new layout at scale  $q$  is composed as

$$\mathbf{Y}^{(q)} = \left( \mathbf{Y}_l^{(q-1)}, \mathbf{T}_{anchor}, \mathbf{Y}_m^{(q)}, \mathbf{T}_{anchor}, \mathbf{Y}_r^{(q-1)} \right) \quad (6.20)$$

where  $\mathbf{Y}_m^{(q)} = \text{layout}(\tilde{C})$ . Thus, a significant part of the summary is reused in the transition between scales  $q - 1$  and  $q$ .

The previous formulation is only valid in the case of a single segment bounded by two consecutive anchor keyframes. If changes are spread in several segments, the layout algorithm is run independently for each of the segments bounded by consecutive anchor keyframes.

Table 6.2: Characteristics of the data set and summaries.

Sequence			Summary			
Name	Duration	Redundancy	#kf	#scales	Step	#clusters
<i>Trec</i>	34m17s	High	74	8	10	5
<i>Franc</i>	34m54s	Medium	255	13	20	10
<i>Quest</i>	20m14s	Low-medium	270	12	20	50

## 6.7. User interface

For the presentation and browsing of the summaries, we developed a prototype of interface based on web technologies. Instead of using images as main units for composition, the interface uses panels. Thus, it is easy to compose the summary and render it laying out the panels from left to right and top to bottom (as a storyboard of panels). Besides, it is more flexible when the window of the browser is resized than a fixed structure (e.g. using tables for the layout).

Each panel is composed previously and stored as a single image using its file name as a unique identifier. The file name contains the identifier of the panel (i.e.  $k$  from template  $\mathbf{T}_k$ ), and the indexes of the keyframes. Thus, each comic-like summary can be represented as a sequence of these unique identifiers (e.g. *panel\_2\_kf\_002003*, *panel\_8\_kf\_008*, *panel\_2\_kf\_012015*, *panel\_8\_kf\_019* represents the first layout of Figure 6.6). The inclusion of new keyframes (e.g. change of scale) implies changes in the panelled layout, with some of the panels removed from the layout and other new ones included, as shown in Figure 6.6.

The user interface should be simple and intuitive. Some preliminary tests with users showed that a suitable user interface was critical for the success of the proposed abstraction approach. In order to make the changes of panels easier to follow, we included an option in the interface which highlights those panels which were not present in the layout displayed before a transition. When that option is enabled, new panels are highlighted by enlarging them (to 130% in our system) and by emphasizing its boundaries with red frames (see Figure 6.8).

## 6.8. Experimental results

### 6.8.1. Experimental setup

We tested the proposed approach for both single scale (independent summaries) and multiscale (dependent summaries) scenarios, using the two algorithms described in this chapter: basic algorithm (*basic*) and anchor based algorithm (*anchor*) with  $h = 4$ . The experiments were conducted over three sequences extracted from different video data sets (*Trec*: high redundancy clip from TRECVID BBC rushes corpus, *Franc*: medium redundancy clip from TRECVID BBC rushes corpus [Kraaij et al., 2006] and *Quest*: low redundancy clip from TURNER Broadcasting corpus [Calic et al., 2007]), in order to cover different levels of semantic redundancy and different number of keyframes. Table 6.2 shows the characteristics of the test sequences and the scalable summaries.

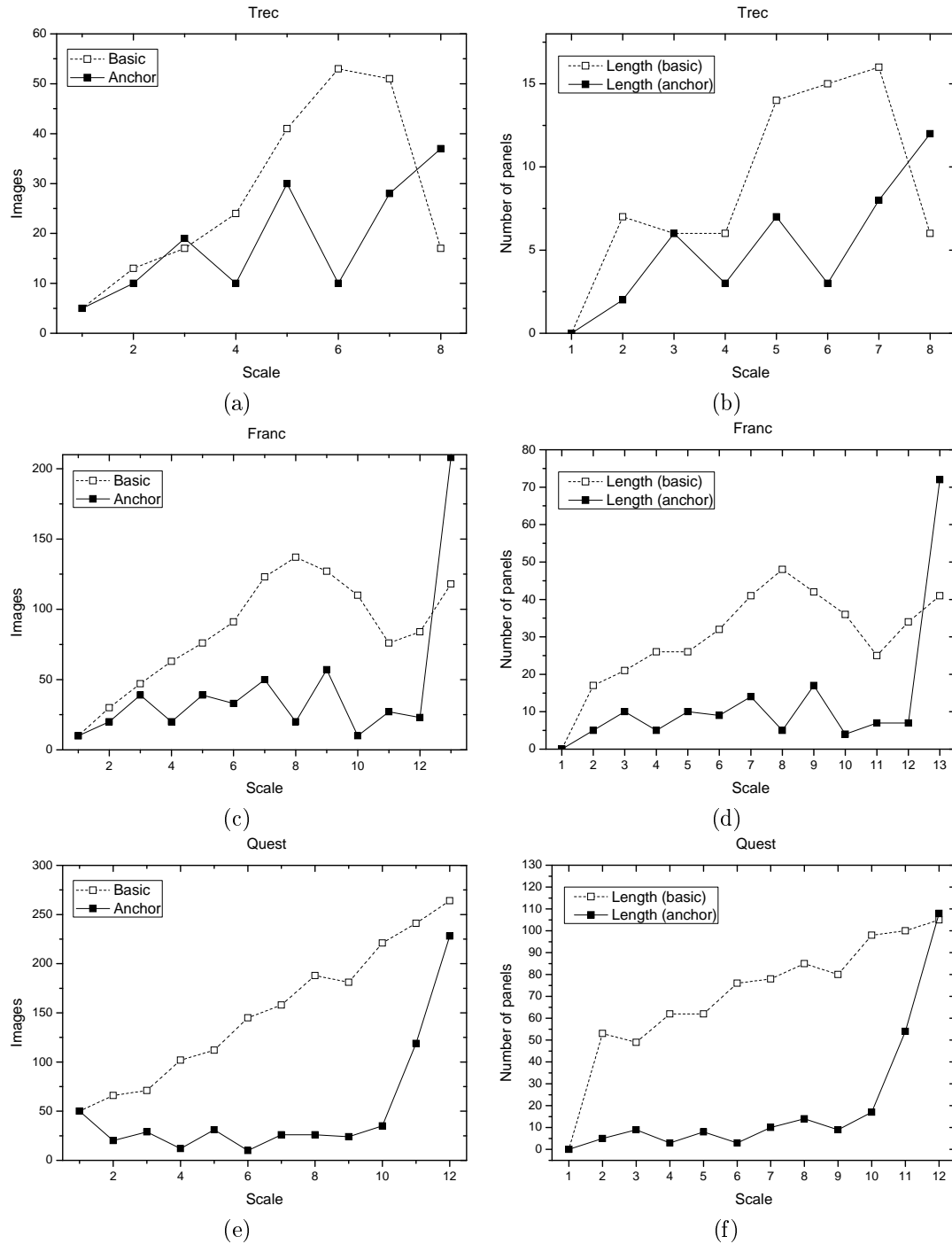


Figure 6.9: Span of the layout change in images (a, c, e) and panels (b, d, f).

### 6.8.2. Objective Evaluation

As we discussed previously, the main motivation of the *anchor* algorithm was to reduce the effect of the disturbance, mainly due to uncontrolled changes between scales. We computed some measures related to how changes from consecutive scales are distributed over the layout. Figures 6.9a, c and e show the span of the change in terms of distance between the first and the last image added in the new scale. Clearly, the temporally constrained sampling approach used in the *anchor* method helps to reduce this span for most scales. The last scale includes all the remaining keyframes, so the span is considerably larger. After computing the layout, this span can be also measured in panels. Figures 6.9b, d and f show the span of the change in panels, which show a similar trend.

The temporally constrained sampling approach cedes some of the keyframes with high cost in exchange for a more compact temporal distribution of sampled keyframes. Figures 6.10a, c and e show the accumulated cost of the sampled keyframes at every scale. It shows that most of the cost is usually covered by the first scale (initial storyboard), and that the penalty in the accumulated cost is small in *Trec* and *Quest*, while it is more notable in *Franc*.

Finally, we compared the number of inserted and removed panels for every scale. In the *basic* algorithm, the inclusion of new images (even only one) may cause changes in all the subsequent panels. However, the *anchor* algorithm restricts this effect only to a part of the layout, with a smaller number of new and removed panels, as shown in Figures 6.10b, d and f. However, the number of total panels in the layout is very similar for both methods and scales.

### 6.8.3. User evaluation

The summaries were also evaluated by a total of 14 assessors according to some subjective criteria, in two different scenarios. For the evaluation, we used the web interface shown in Figure 6.8. The summaries were displayed on a large screen (1920x1200 pixels).

#### 6.8.3.1. Scenario 1: Interactive summaries

In the first scenario, the assessors were free to interact with the interface and navigate across the scales of the summaries. In order to avoid biases, the name of the algorithm was hidden and the order of evaluation randomized. Results are shown in Table 6.3. The satisfaction criterion was posed as an affirmative statement (“In general, the summary represents adequately the original content.”) and evaluated using a typical Likert scale (1: Strongly disagree; 3: Nor agree nor disagree; 5: Strongly agree)[Likert, 1932]. In general, the results were very similar for both algorithms, and users were satisfied with the summaries. The assessors were also asked for their preference between both algorithms, with no clear preference except for a very slight preference for the *basic* algorithm in *Trec* and *Franc* sequences. Finally, the assessors were also asked about the utility of the interface (“The user interface helps to follow the changes across scales.”), with a positive evaluation.

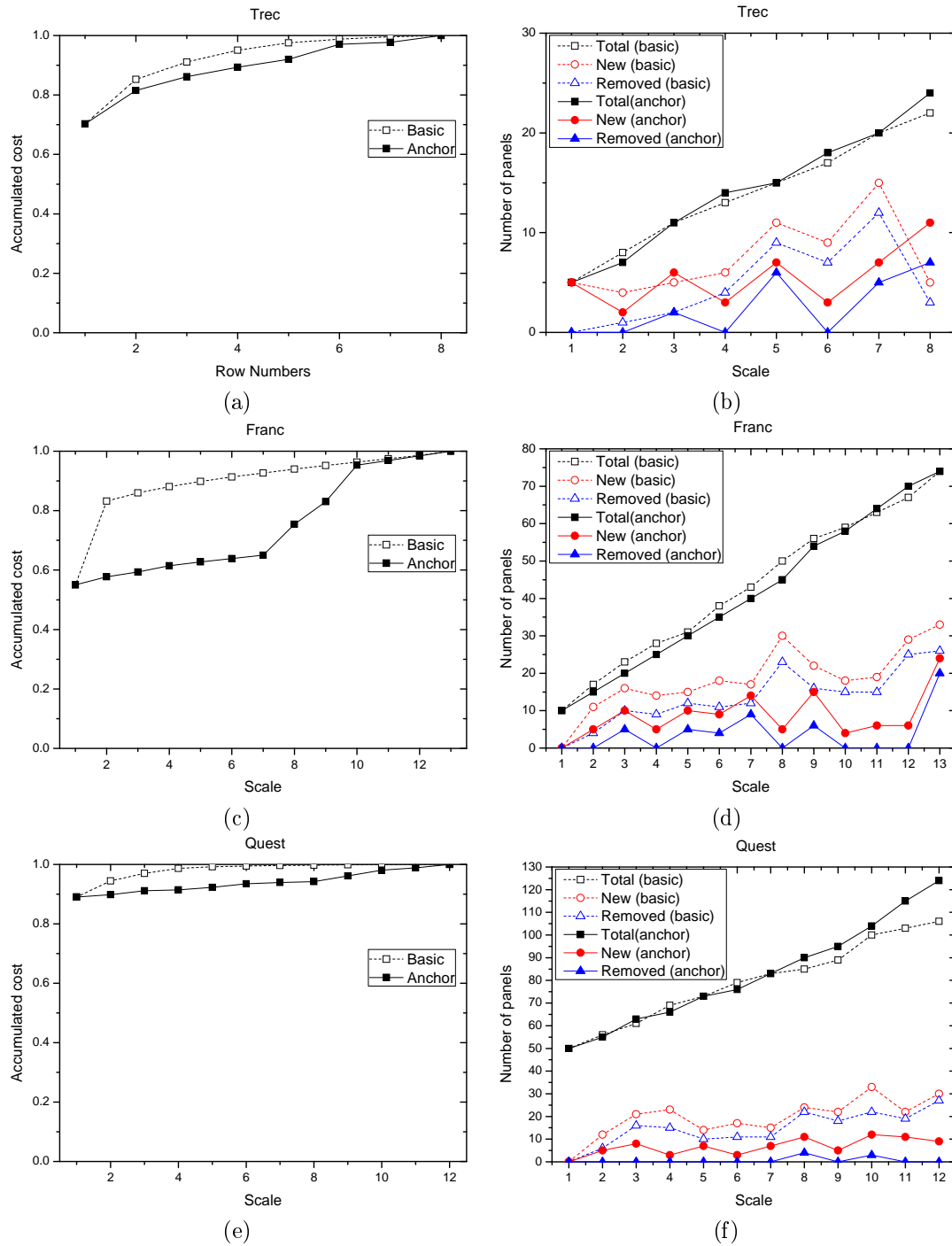


Figure 6.10: Comparison of basic and anchor algorithms: (a, c, e) accumulated cost, (b, d, f) inserted and removed panels.

Table 6.3: Subjective results for the interactive scenario.

		<i>Trec</i>	<i>Franc</i>	<i>Quest</i>
Satisfaction	Basic	4.4	4.0	3.9
	Anchor	4.4	3.9	4.0
Preference (5: Anchor - 1: Basic)		2.9	2.9	3.0
User interface		4.4	4.3	4.3

Table 6.4: Preference of the algorithms in the progressive scenario: *basic* (B), *anchor* (A) and *anchor* with highlighting (A+H).

%	<i>Trec</i>			<i>Franc</i>			<i>Quest</i>		
	1 <sup>st</sup>	2 <sup>nd</sup>	3 <sup>rd</sup>	1 <sup>st</sup>	2 <sup>nd</sup>	3 <sup>rd</sup>	1 <sup>st</sup>	2 <sup>nd</sup>	3 <sup>rd</sup>
B	28.6	14.3	<b>57.1</b>	14.3	28.6	<b>57.1</b>	21.4	14.3	<b>64.3</b>
A	14.3	<b>71.4</b>	14.3	14.3	<b>57.1</b>	28.6	21.4	<b>57.1</b>	21.4
A+H	<b>57.1</b>	14.3	28.6	<b>71.4</b>	14.3	14.3	<b>57.1</b>	28.6	14.3

### 6.8.3.2. Scenario 2: Progressive summaries

In this second scenario, the summary progressively includes more frames, which consequently changes the layout, and users are not allowed to interact with the summary. In the evaluations, the assessors had to visualize summaries presented in a progressive manner, from the coarsest to the finest scale, and at a fix rate of one scale per second. Three variations were evaluated: the *basic* method, the *anchor* method and the *anchor* method with new panels highlighted. The assessors were asked to sort them according to their preference (from higher to lower preference). Results (see Table 6.4) show a clear preference for the *anchor* method with highlighting, and, in second place, for the *anchor* method without highlighting. These results confirmed that the *anchor* algorithm can effectively reduce the disturbance, improving the utility of scalable comic-like summaries in this scenario, and also the importance of appropriate interface elements.

### 6.8.3.3. General evaluation

At the end of the evaluation, some general statements were posed to the assessors in order to evaluate the global opinion about the proposed summarization approach. The criteria and the statements were the following:

- Utility of comic-like summaries (“Comic-like summaries are useful and effective representations of video content.”)
- Utility of scalability (“Scalability, i.e. multiple levels of detail, is a useful feature in video summaries.”)
- Browsing interface (“The interface provides a useful way to browse summaries of video content.”)
- Utility of highlighting (“Highlighting feature is helpful in tracking changes across scales.”)
- Overall system (“The proposed system is useful for browsing video content.”)

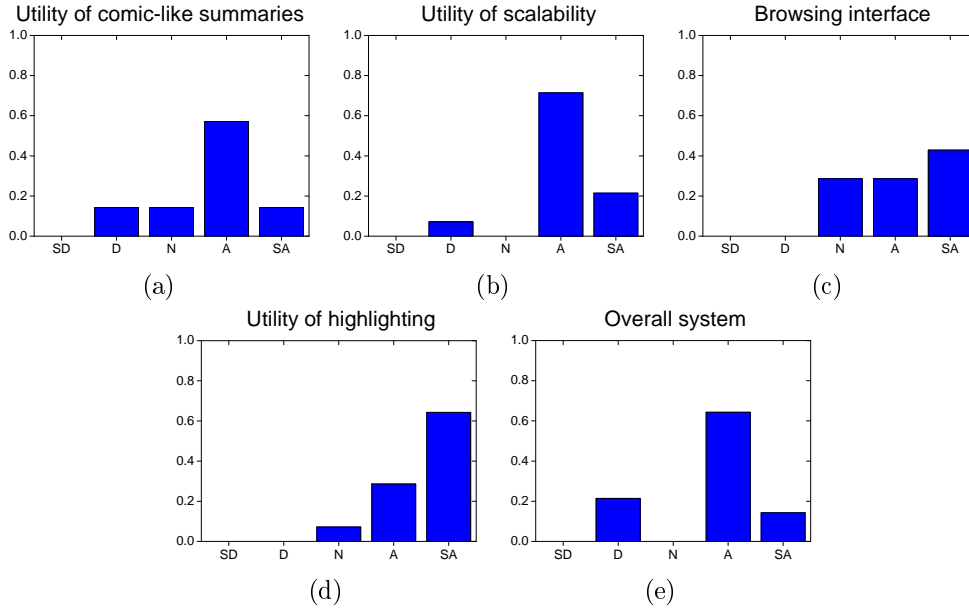


Figure 6.11: General assessments.

The results of this last part of the subjective evaluation are shown in Figure 6.11. In general, most of the assessors agreed with these statements, supporting the proposed scalable comic-like summaries as an effective and flexible approach to video summarization.

## 6.9. Summary and conclusions

In this chapter we have investigated the use of scalability in the context of comic-like summaries as a flexible and intuitive abstraction format based on the narrative structure of comics. In contrast to scalable storyboards, the need of a non-trivial visual structure makes the computation of comic-like summaries more difficult, and efficient methods are required. In our case, a suboptimal fast algorithm is used with satisfactory results.

We first explored the case of independent scales, suitable for applications that require the adaptation of the summary to a target length or size. The case of dependent scales, required for progressive and interactive visualization, was more complex, including an annoying effect due to the transition between scales when there are too many changes in the layout during a short amount of time. The term disturbance was used to refer to this effect. When this effect appears, the user feels uncomfortable and confused. Based on some previous observations, a heuristic algorithm is developed to localize these changes in limited areas. After some tests with users, we realized that the user interface is a key element in the whole system, and must be carefully designed to minimize disturbing effects and make the proposed abstraction approach appealing and pleasant. Elements driving the attention to the main changes are particularly useful (e.g. the highlighting feature in our system).

Experimental evaluation confirms the value of application of scalable comic-like summaries for

video retrieval and browsing. Models of disturbance, which can be included in the summarization process, as well as improved user interfaces, can be helpful to provide more appealing and user friendly summaries.



Part IV

Applications



## Chapter 7

# Applications of integrated summarization and adaptation

In this chapter we describe some potential applications of the extraction-based framework proposed in the second part of the thesis. As discussed previously, the main advantages of this framework are its simplicity, its efficiency and low resource requirements. These characteristics make this approach very suitable for a number of applications in which other generation-adaptation methods may be unsuitable. We also compare the approach with other alternatives, such as transcoding and variations.

### 7.1. Customized summaries

Current research in video summarization is shifting to a more user-centred approach[Tseng et al., 2004], in order to make easier the use of browsing and retrieval systems and also to enhance the experience of each individual user, according to his or her interests. There are several applications which enable a high level of personalization, such as sport portals[Babaguchi et al., 2007; Jaimes et al., 2002] and news portals[Maybury et al., 2004], where different users have different interests, and personalized summaries can be provided according to them. Recently, [Money and Agius, 2008a] studied some physiological responses to video summaries, which vary considerably among individual users, suggesting that summaries can be highly personalized based on the incorporation of external information, such as contextual data. However, personalization often means the generation of the summary on demand, once the specific preferences or user characteristics are known.

Personalization usually refers to a process of adaptation that is performed at the server according to some profile with user's preferences and usage environment. However, the user can be given even more freedom to build its own summary selecting which segments are included in the final digest. That kind of summaries is what we call here customized summaries.

In both personalized and customized summaries, the potential number of summaries is high,

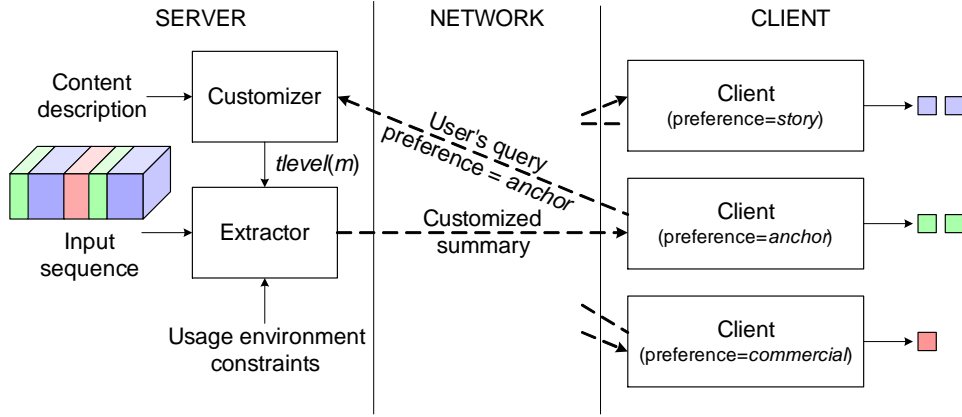


Figure 7.1: Customized summaries using extraction.

requiring the summary to be built on demand. The generation of these summaries often requires an adaptation process for each user, performed via transcoding, with significant computational cost or large processing delay. The proposed extraction framework is an efficient alternative which can also enable customization, in both server-side summarization-adaptation services and client-side browsing applications.

As an example of customized video summarization, let us consider the system shown in Figure 7.4, which depicts a simple application enabling personalized access to video summaries of broadcast news. Broadcast news data usually have a specific structure, which roughly consists of sequences of news items, and sometimes, with some commercials between them. Each news item generally consists of an introductory anchor segment and the news story segment. With this simple structure in mind it is possible to annotate a simple description of the segments and store it along with the sequence. Using the proposed summarization model, each SU is tagged with a category (*anchorperson*, *story* or *commercial*) and whether it must be included in the summary (e.g. keyframes for storyboards) or not. When a user request a summary, the client sends a query to the server with the preferences of the user and the type of summary. The server processes the preferences and generates a set of appropriate values for  $tlevel(m)$  (see Section 3.5.2) including the SUs tagged as belonging to the preferred categories and belonging to the summary, which is generated using the bitstream extractor. Thus, the summary delivered to the user includes only information matching his or her preferences. Although very simplistic, this example illustrates how video summaries can be generated on demand and delivered to the user with low delay using the proposed framework.

### 7.1.1. Audio extraction and multiplexing

Until this section, we have only considered the visual signal contained in an audiovisual stream. However, in practical applications, audio must be also considered. The audio stream must be edited according to the summarization model and multiplexed with the video stream.

The audio extraction process is similar to the video one. Audio streams are coded into

packets, which once decoded result in a number of audio samples. The length of the sample depends on the coding format, and other parameters such as the audio rate, the number of channels and the bitrate. We have implemented an audio extractor for MPEG-1 layer III (also known as MP3)[ITU-T and ISO/IEC, 1992; Musmann, 2006; Pan, 1995]. It can be extended also to other MPEG-1 layers and standards such as MPEG-2 Advanced Audio Coding (AAC)[ITU-T and ISO/IEC, 2007; Noll, 1997] and MPEG-4 AAC[ITU-T and ISO/IEC, 2005; Herre and Dietz, 2008], as the coding principles and structures are similar. The duration of each frame in MPEG-1 layer III is fixed, and it depends on the number of samples per frame (1152 samples), the sampling rate  $F_s$  and the number of channels  $N_{channels}$  as

$$t_{audio\_frame} = \frac{1152}{F_s \cdot N_{channels}} \quad (7.1)$$

Considering a frame rate of 22050 Hz and two channels, the duration of each audio frame is 26.12 milliseconds, which corresponds to an audio frame rate of 38.28 frames per second. The frame size is also fixed, depending also on the bitrate  $R_{audio}$  and the number of channels, given (in bytes) by

$$s_{audio\_frame} = \frac{144R_{audio}}{F_s + padding\_bit} \quad (7.2)$$

where *padding\_bit* is a parameter (a bit specified in the header of the frame) used, if necessary, to add extra data in order to adjust the bitrate. A bitstream description is used to specify the exact size of each packet.

The extraction process must be guided by the same summarization constraint  $t_{level}(m)$ , which is computed using the video stream as reference. Packets containing video frames (NAL units) have also a fixed duration. For instance, for a video frame rate of 29.97 frames per second, the duration of each video frame would be 33.37 milliseconds. The duration of audio and video frames differs, and the duration of a SU may not match an integer number of audio frames. In that case the extractor must decide either to include or to drop the last audio frame in order to select an integer number of audio frames.

Figure 7.2 shows an example of mismatch between audio and video frames, and how it results in a non synchronized video when they are multiplexed. The inclusion or dropping of audio frames at the boundaries of a new segment must be decided based on the instantaneous delay between both audio and video streams, in such a way that the delay is compensated. If it is done properly, the maximum delay should not be larger than  $t_{audio\_frame}/2$ . In the previous example it would be 13.06 milliseconds (26.12 milliseconds in the case of a single audio channel), which is almost imperceptible. However, if a better synchronization is required, a solution may be the dynamic adjustment of time stamps at system level. Thus, whenever a new segment of frames is included, the time stamp of both streams must be adjusted.

We evaluated experimentally<sup>1</sup> the efficiency of both transcoding and extraction with an audio file with a frame rate of 22050 Hz, a bitrate of 64 kbps and two channels. As shown in Figure 7.3

<sup>1</sup>Experiment performed in an Intel Core 2 at 2.83 Ghz (2 GB of RAM)

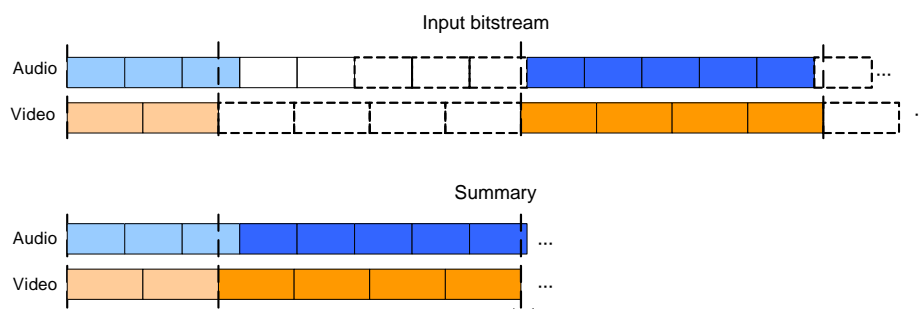


Figure 7.2: Audio extraction and multiplexing.

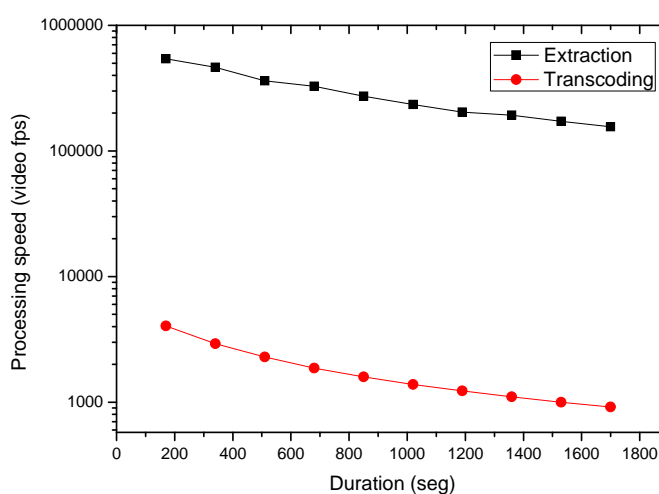


Figure 7.3: Processing speed of audio extraction and transcoding.

(the processing speed is measured in video frames per second, according to the corresponding video stream), with any of the two approaches, the generation can be performed much faster than real time, although extraction is notably faster. The performance degrades slightly as the length of the summary increases, but still being very efficient. However, in an audiovisual stream, the bottleneck for efficient processing is still the generation-adaptation of the video bitstream.

### 7.1.2. Web based demo

In order to demonstrate the utility of customized summaries in a practical application, we implemented a demo interface based on web technologies (see Figure 7.4) and H.264/AVC and MPEG-1 layer III. The semantic structure of the video, which is assumed to be available as metadata, is presented to the user as a list of video segments. The user can create a customized summary by selecting segments according to his or her preferences (e.g. news anchorperson, news story, commercials). Then, the user requests the customized summary and the server generates the bitstream using video and audio extraction. The audio and video bitstreams are multiplexed, into a suitable format (in this demo, Flash Video using H.264/AVC and MP3) for the embedded player, prior to its delivery to the client. As shown in the example of Figure 7.4, the summary

## Customized summaries demo

This is the structure of the original video. Which part do you want to be included in the customized summary?

- ☐ Header
- ☒ News 1. Anchor
- ☐ News 1. Story
- ☐ Bumper
- ☒ News 2. Anchor
- ☐ News 2. Story
- ☒ News 3. Anchor
- ☐ News 3. Story
- ☐ Commercials
- ☐ End news
- ☒ Comedy intro (cartoons)
- ☐ Comedy

The summary was generated successfully.

Report:

Video stream: Extraction time 937ms (844ms parsing+93ms generating)

Audio stream: Extraction time 94ms (62ms parsing+32ms generating)



Figure 7.4: Demo web interface for customized summaries.

is generated with a delay of approximately one second. The stream is delivered only after both streams are completely generated and multiplexed. The delay can be further reduced with a parallel implementation of the extractors and the multiplexer, streaming the video as soon as the first frames are available.

## 7.2. Local browsing

Image organizers are very useful to browse through personal and professional photo libraries[Shneiderman et al., 2006]. Many interfaces for image gallery navigation have been proposed with great success. Most of them are based on the idea of thumbnail (a reduced-size version) as an abstraction of the original image. Dozens of thumbnails are usually displayed together to reduce the time required to browse the contents of certain image gallery. The user can view a detailed version of an image by just selecting its thumbnail.

With the emergence of high capacity data storage devices, video has become an important source of content and personal video libraries are growing very fast. However, browsing large video libraries is much more time consuming than image browsing[Wildemuth et al., 2003]. For this reason, an effective abstraction is even more critical[Truong and Venkatesh, 2007]. Browsers and organizers specifically designed for video content can make easier the task. In video browsing, video summaries play a similar role as thumbnails in image browsing, as they effectively reduce the time the user needs to get an idea of what happens in the video.

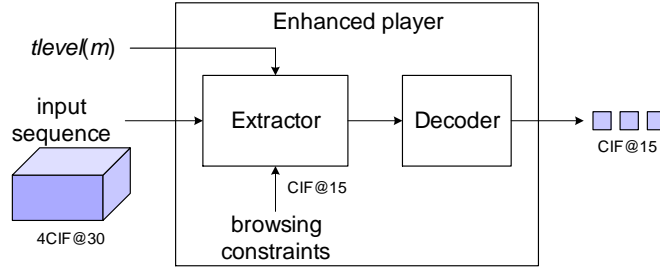


Figure 7.5: Architecture of a player capable of presenting a preview of the content based on summaries.

Video browsers and organizers can also benefit from the bitstream extraction approach for video summary generation, due to its low delay and flexibility. In a typical video library browser, thumbnails or short storyboards can be shown as a first coarse abstraction of the contents. If the user is interested on a specific piece of content, a video skim could provide a more detailed abstraction. Figure 7.5 shows a possible architecture of a module which can extract and display the summaries using the proposed summarization model. Extraction and decoding run in the same module. The summarization parameters (the different values of  $tlevel(m)$ ) are stored as metadata together with the video sequence. Using the corresponding values, the extractor builds the summarized bitstream which is then decoded. Thus only the required parts of the bitstream are processed by the decoder. Decoded frames are then presented in a suitable way (e.g. a set of still images for storyboards or a video sequence for video skims or fast forwards). Additional constraints can be used in extraction to obtain better performance or results. For instance, if different spatial scales are available, the lowest spatial one may be more suitable when thumbnails are required, saving some decoding effort.

Other applications, such as interactive video navigation or hierarchical video browsing[Bertino et al., 2003], can use the same approach with different combinations of summaries that can also be described with different values of  $tlevel(m)$ .

## 7.3. Comparison of architectures for summarization and adaptation

### 7.3.1. Application scenarios

In video retrieval and browsing, there are a number of scenarios in which summaries and adapted versions are very important for an efficient access and interaction with multimedia content. We briefly describe two representative scenarios.

#### 7.3.1.1. Multiple summaries and Universal Multimedia Access

The objective of the so called Universal Multimedia Access is to provide each user with a suitable version of the content according to the specific network, terminal and user's preferences.

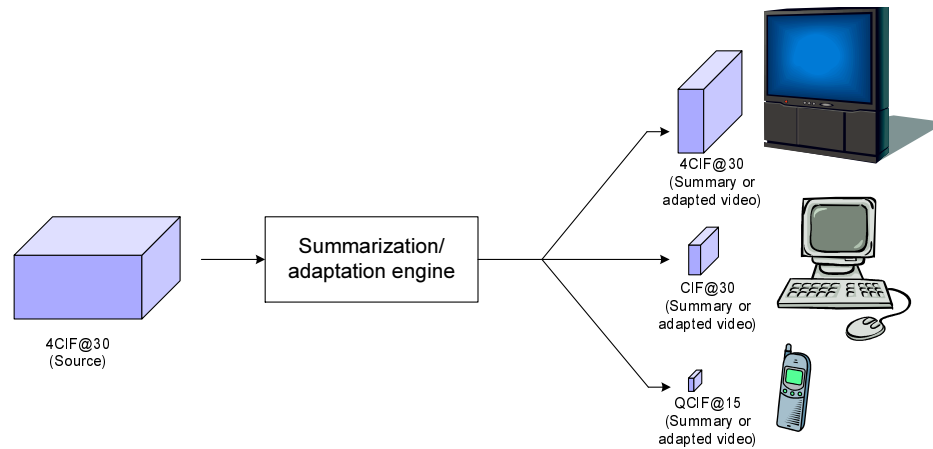


Figure 7.6: Access to video content from heterogeneous terminals and networks.

Usually, this access is carried out through search and browsing interfaces, in which video abstractions are required for efficient navigation. As ancillary content, summaries must be also adapted to the specific usage conditions.

An example is shown in Figure 7.6, in which users access video content from different terminals such as computers, mobile phones or TV set-top boxes. Users navigate through the content available in the video library. Eventually, a user may request a video summary of any of those clips. The server selects or generates the summary according to the request and it must be adapted to the specific usage environment (e.g. reduced resolution for mobile phones, lower quality for low-capacity networks). Note that the format of the summary itself may be also conditioned by the usage environment (e.g. if the terminal does not support video, a storyboard would be more suitable than a video skim). When the user selects a clip, the original video itself must be also adapted.

#### 7.3.1.2. Web-based video library

Adapted versions include low resolution images and video clips. Most web interfaces to access video libraries use these lightweight representations (e.g. thumbnails, embedded clips) to provide efficient navigation. Usually, these images and clips are stored as separated files (following the variation approach described in the next section). They are embedded into web pages at different points, according to a convenient web design, in order to provide an appealing and usable visual interface. Different types of summaries provide different levels of abstraction of the same piece of content. As they navigate, users can request more detailed summaries if required. Note that, in contrast to the preceding case, there is only one user, using an appropriate browser, who consumes different summaries and versions of video content as navigates through the web site.

Figure 7.7 shows a simple example of web-based video library. The entry point is a home page, in which the user can find news and information about the library. At that point, the interface may show some recommended videos (represented by a low resolution image or video skim). A

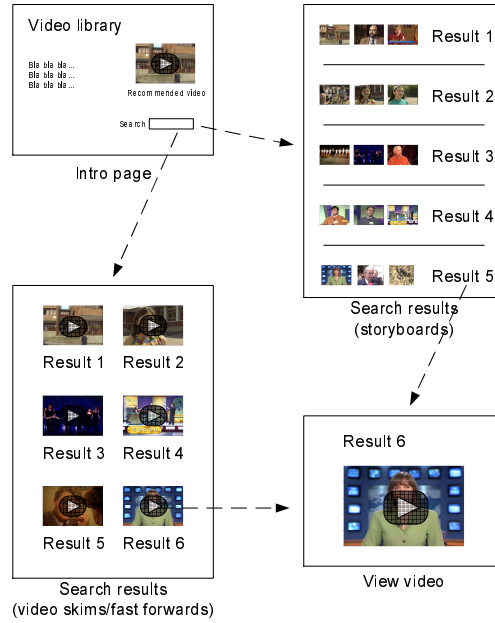


Figure 7.7: Example of web-based video library using different types summaries.

form enables text-based search. Search results are presented in different formats according to navigation profiles. The figure shows two cases in which one user prefers a storyboard with three images and the other prefers that each result is represented by a single image so the interface shows more results per page (when an image is clicked, it may also play an embedded video skim or fast forward). Finally, the user can play the original video in a dedicated page.

### 7.3.2. Summarization-adaptation architectures

Summaries and adapted versions can be provided using different approaches in which they are created at different points of the interaction process between client and server. We distinguish among three different basic architectures to provide adapted summaries and videos.

#### 7.3.2.1. Variations

A first approach is the use of a file for each of the versions (summaries and adapted versions). Following the nomenclature of MPEG-7 Multimedia Description Schemes (MDS)[ITU-T and ISO/IEC, 2001a], these versions are called variations. MPEG-7 MDS defines the *Variation* description scheme to describe each of these elements (e.g. different bitrate, spatial or temporal resolution)[van Beek et al., 2003; Böszörményi et al., 2003; Libsie and Kosch, 2004]. Each variation is created and encoded prior to the interaction with the users (see Figure 7.8a). A summary is a type of semantic adaptation of the content, and it can be considered as another variation, which in turn, may have different variations (e.g. bitrate, resolution). Typically, the content is analyzed and a number of summaries are created from the source video. Each of these summaries and the source video itself are adapted to a target profile (e.g. mobile phone, PDA, computer), with different temporal and spatial resolutions and bitrates.

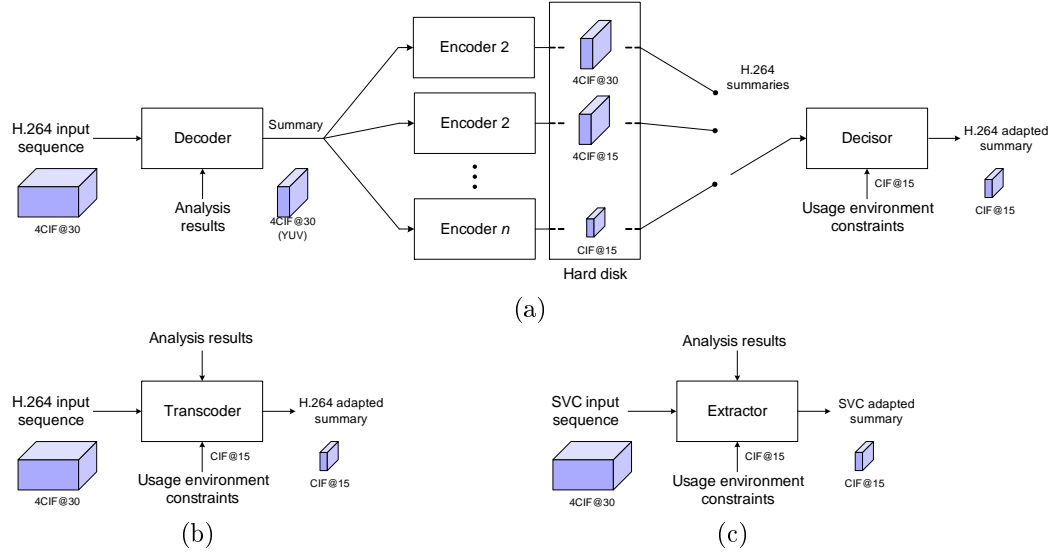


Figure 7.8: Architectures for summarization and adaptation: (a) variations based, (b) transcoding based, and (c) extraction based.

The advantage of variations is that, once the files are created and stored, the adaptation process is very simple, as the server only has to select the most appropriate variation, according to the request and the usage environment, and deliver it to the user. The most critical and time-consuming task, which is the generation and adaptation of summaries, is completely carried out prior to the interaction with users. However, it requires a significant amount of storage space and the potential adapted versions are restricted to those available as variations.

### 7.3.2.2. Transcoding

The transcoding approach, already described in Chapters 2 and 4, is based on the creation of summaries and transcoding content when the user selects a specific version, with specific target parameters (see Figure 7.8b). Thus, the only file required is the source video.

The main advantage of transcoding is its flexibility, as it can cope any possible adaptation (only limited by the actual capabilities of the transcoder). In contrast, transcoding video content is usually a very time consuming process.

### 7.3.2.3. Extraction

As shown in Chapters 2 and 4, under certain conditions, transcoding can be replaced by extraction, in order to achieve efficient generation of both summaries and adapted videos (see Figure 7.8c). In some sense, extraction has features of both variations and transcoding approaches, as each SVC file has embedded multiple summaries and versions of the same video content (as in the variations approach), which can be created on demand (as in transcoding) by selecting the appropriate packets of the bitstream.

Approach		Variations	Transcoding	Extraction
Efficiency		Very high	Low-very low (see Figures 3.21, 3.25, 4.6 and 4.7)	Very high (AVC, see Figures 3.21, 3.25) High (SVC, see Figures 4.6 and 4.7)
Quality		Best (if the source is uncompressed) Good (with high quality transcoding)	Medium-poor (depending on encoding parameters; see Figures 3.23, 3.24 and 4.8)	Best (AVC, see Figures 3.23 and 3.24) Good-medium (SVC, see Figure 4.8)
Storage requirements		High	Low	Low
Precision		Frame	Frame	SU/GOP
Delay		Very low	High	Low-very low
Coding	Support	Any (but must be available in the stored versions)	Any supported by the transcoder	AVC/SVC or other scalable codec
	New codecs	Yes (requires reencoding and extra storage space)	Yes (requires to include the codec in the transcoder)	No
	Potential adaptations	Any (but must be available in the stored versions)	Any (supported by the transcoder)	Those available in the coded bitstream Usually dyadic decompositions (e.g. 4CIF, CIF, QCIF, 15 Hz, 30 Hz)
Flexibility	Cost of a new version	Encoding and additional storage space	No additional cost	No additional cost if already embedded in the bitstream
	Cost of a new summary	Encoding and additional storage space	No additional cost	No additional cost
	Customized summaries	No	Yes	Yes

Table 7.1: Comparison of summarization-adaptation architectures.

#### 7.3.2.4. Hybrid architectures

In a practical framework, it is not necessary to use strictly only one of the previous architectures. Hybrid architectures, combining pre-stored variations with transcoding or extraction, may be more suitable, and will depend on the specific scenario and requirements.

One example of hybrid architecture would be a transcoding architecture with caching. In that case, summaries and adapted versions are generated by transcoding the source content on demand. However, the server stores all the previously generated files, as variations. Thus, if the user requests any variation that was requested previously, the server just delivers the cached file. Caching trades off efficiency and storage space.

Another example is the combination of different approaches for images and video files. Image based summaries, such as storyboards, require much less storage space than video based summaries. Thus, image based summaries can be stored as variations, while video based summaries are generated via transcoding or extraction.

A third example is the use of variations of the source file (e.g. store 4CIF, CIF and QCIF versions), and generate the rest of sub-variations by extraction or transcoding.

Transcoding and extraction can also be combined using a first extraction stage followed by a transcoder. That reduces significantly part of the cost of transcoding. This case was already studied in Chapter 4.

#### 7.3.3. Comparison of architectures

The differences between the three approaches are summarized in Table 7.1.

### 7.3.3.1. Efficiency

Efficiency of transcoding and extraction is extensively studied in Chapters 3 and 4. Here we just recall the results of the experiments shown in Figures 3.21 and 3.25 for H.264/AVC, and Figures 4.6 and 4.7 for SVC. This experiments showed that extraction is notably more efficient than transcoding, mainly because of the simplicity of the generation-adaptation process. In a server based on variations, the process is even simpler, i.e. select and forward a suitable pre-stored bitstream, which would be slightly faster than extraction.

Extraction and variations have also the advantage of consuming little computer resources such as memory and CPU usage, in contrast to transcoding, which is extremely demanding, especially for high resolution sequences. Thus, a single computer which could serve tens of clients using extraction and variations, could serve just a few using transcoding.

Closely related to efficiency, the generation-adaptation delay is another important factor which may be decisive in some applications. The delay would depend on the processing load which would also depend on the number of connections. In some applications, such as customized and scalable summaries, a low delay is critical for a satisfactory user experience.

### 7.3.3.2. Rate-distortion performance

Studied in Chapters 3 and 4, rate-distortion performance is also a relevant factor. Although not critical, in the sense that the content would reach the user even with lower quality, providing the user with the best video quality is important for a good user experience.

For single layer H.264/AVC, used when nor quality nor spatial adaptation are required, extraction preserves the original quality, while transcoding has some degradation (due to a second lossy stage, i.e. requantization). This degradation also depends on the configuration of the encoder (see Figures 3.23 and 3.24). Usually, efficiency and rate-distortion performance are traded off, although transcoding has always some quality loss (except for some special cases, such as idempotent coding[Zhu and Lin, 2010]). Using variations, if they are generated from the original uncompressed sequence, they have the same quality as those versions obtained using extraction. However, if the variations are generated from a previously coded bitstream (e.g. decoding and re-encoding), there would be a second quantization stage that would degrade the quality compared to the original uncompressed sequence. Using a transcoder with a high quality configuration (large motion estimation search window, advanced coding tools) to generate the variation would help to lessen the quality loss, although the encoding process would be very demanding and slow at the preprocessing stage.

For SVC, there is some quality loss compared to the single layer case (i.e. H.264/AVC). This quality loss is due to the coding penalty associated with layered coding. Thus, only the base layer does not degrade. However, the other enhanced versions may have worse quality than transcoding, depending on the operation point (see Figure 4.8). In this case, variations generated from the uncompressed original sequence provide the best quality, as they do not have requantization and each of them is a single layer bitstream.

### 7.3.3.3. Storage requirements

The main drawback of the use of variations is that each one must be stored in a separated file. Thus, systems using a large number of variations (due to a large number of summaries, adapted versions of summaries and/or adapted versions of the main video) may require large storage resources, especially for high resolution sequences. In contrast, transcoding and adaptation only require the storage of one file. Due to the coding penalty of layered coding, given the same quality (i.e. PSNR), extraction could require slightly more space.

As discussed in Chapter 5, a scalable summary is a special case in which the number of potential summaries may be very large, and each of them must be stored separately. While for storyboards may be feasible, for video skims the storage requirements could be unacceptable.

### 7.3.3.4. Coding

Variations and transcoding can provide codec adaptation. In principle, variations may be stored in any coding format supported by the encoder. Thus, the system can deliver different versions with the same characteristics but with different coding formats (e.g. MPEG-1, MPEG-2, H.263, H.264/AVC), useful in heterogenous scenarios in which the different terminals have different decoding capabilities (e.g. codecs, profiles). Transcoders may also adapt the content to any coding format, in principle. However, extraction relies on a specific scalable codec, either H.264/AVC for temporal scalability, or SVC for extended adaptation. All the terminals must support SVC decoding. There are two special cases in which H.264/AVC decoding would be enough. The first one is the case in which only the base layer is required. The second one is the use of SVC-to-AVC rewriting [Segall and Zhao, 2008; De Cock et al., 2008], in which the SVC bitstream is converted to an H.264/AVC single layer bitstream. However, the latter is closer to lightweight transcoding than to extraction. Adaptation to other codecs is not possible with extraction using SVC.

A similar capability is the adaptation to arbitrary resolutions and bitrates. Variations and transcoding may support any arbitrary spatial and temporal resolution and bitrate, provided that an appropriate encoder or transcoder is used. The transcoder would perform the adaptation on demand, while the system using variations must have created the variation previously. However extraction only supports those versions embedded in the original bitstream, which are typically encoded using dyadic decompositions in temporal (e.g. 15, 30 frames per second) and spatial dimensions (e.g, 4CIF, CIF), and possibly several bitrates.

### 7.3.3.5. Flexibility

Transcoding is the most flexible of the three approaches, as the inclusion of new summaries or versions not considered in an initial design do not require any additional processing (other than the description of the summary). The new version is generated on demand with the same cost of any other version. Extraction is still a flexible approach, although limited by the versions available in the source bitstream and a lower precision to describe summaries (i.e. the length of the summarization unit, in contrast to frame precision in variations and transcoding). However,

in most cases that flexibility is enough. In contrast, the use of variations does not provide any flexibility, as any version not considered initially cannot be generated.

## 7.4. Summary and conclusions

In this chapter we have explored the use of the extraction-based framework described in the second part of the thesis in different applications. Customized summaries and local browsing are two examples of applications that can benefit from the efficient and low delay generation of summaries.

Besides, we also explored the extension of the framework to video sequences with audio. Extraction can be also used for audio adaptation and can be integrated with the video extraction framework. However, both streams must be multiplexed carefully to avoid any noticeable delay between both streams.

Finally, we compared three architectures to provide access to video summaries and versions, adapted to different usage contexts. The same approaches can be used in web-based user interfaces to video libraries. Depending on the requirements of the application, some of them may be more appropriate. While variations are useful if few adapted versions and summaries are required, transcoding is a much more flexible approach with the drawback of being computationally expensive. Extraction is a flexible yet efficient approach that may be suitable for many applications, such as customized and scalable summaries. Often, these three basic approaches can be combined in hybrid architectures.



## Chapter 8

# Adaptation of scalable summaries

This chapter describes two applications of the methods proposed in the third part of the thesis, using mainly storyboards. Scalability is exploited by the proposed graphical user interface so the summary can automatically fit into an area that can be interactively resized by the user. Besides scalability, the method proposed in Chapter 5 has the advantage of being very efficient. This feature is used in a broadcast scenario in which the system must generate summaries of multiple channels.

### 8.1. Resizable pictorial summaries

#### 8.1.1. Introduction

Most of the graphical user interfaces of current operating systems are based on resizable windows. Each window covers a certain visual area (typically rectangular) displaying the visual components of processes. Giving the user the freedom to change the size of the windows is a very useful feature to organize visual information, so multiple documents and applications can be easily accessible in the same interface.

Similarly, pictorial summaries try to represent the content of video sequences as a combination of still images, laid out over a limited visual area. Conventional pictorial summaries represent this information in a fixed size summary [Pfeiffer et al., 1996; Yeung and Yeo, 1997; Chiu et al., 2004; Calic et al., 2007; Mei et al., 2009]. The amount of information depends on the available visual area. If this area is not large enough, the user has to resize it, scroll across the window or downscale the images in order to visualize the whole summary.

As described in Chapter 5, scalable summaries can adjust their length and level of detail on demand, without almost any further processing. The area covered by a pictorial summary is usually related with the amount of information conveyed by the summary (e.g. number of images in a storyboard). Thus, changing the scale of the scalable summary, the area covered can be adjusted to the available area in a window or canvas.

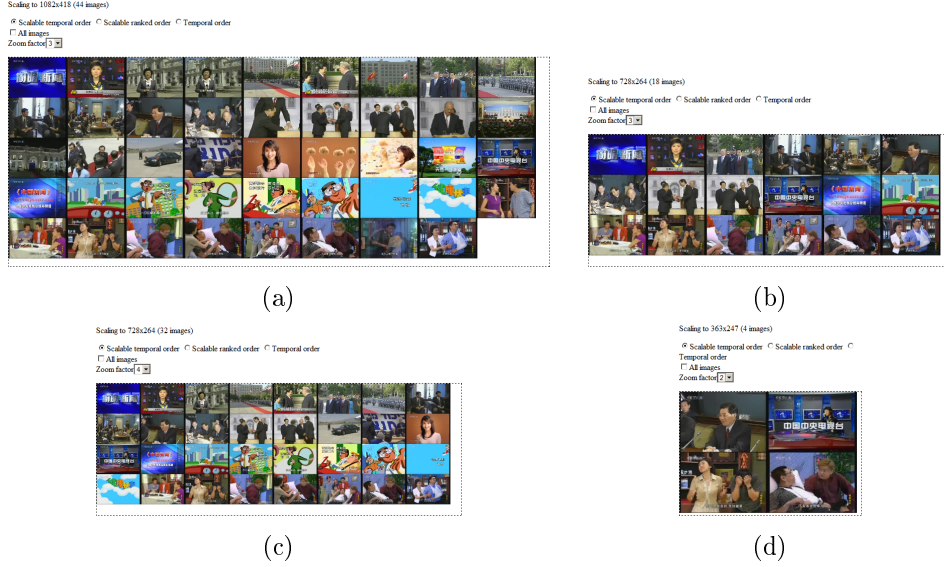


Figure 8.1: Example of window resizing using a scalable storyboard: (a) 44 images (full storyboard), (b) 18 images, (c) 32 images and (d) 4 images.

### 8.1.2. Resizable storyboards

In order to demonstrate the utility of scalable summaries for this application, we have designed a web interface in which the images are laid out in a resizable window. The user can resize the window, and consequently the canvas in which the storyboard is laid out. The number of images of the storyboard is computed from the available area in the canvas and the size of each individual image (which can be modified in the interface by adjusting a zoom factor).

If a conventional storyboard is displayed in a web page, and the area available is not large enough, the interface will show only images from the initial part of the video. If the user wants to have information about the end of the video, he or she would have to scroll across the storyboard or resize it. In contrast, using a scalable storyboard, the images cover the whole video, and if the area is constrained, only those considered more relevant are displayed. The user can interact with the interface to change the number of images in the storyboard [either changing the size of the window (see Figure 8.1a and b) or changing the size of each individual image (see Figure 8.1b and c, where the zoom factor is changed without changing the available area)].

## 8.2. Summarization of live TV streams

In this section, we describe a very challenging scenario and an application involving multiple channels and low delay. The objective is to provide the user with a quick summary of what was being broadcasted on several channels (e.g. favorite channels) when the user switches the TV on. For instance, the user would like to know what was on TV during the previous hour. If the terminal has proper recording resources to store the signal, the user could also view that content if desired.

### 8.2.1. Summarization in a multichannel broadcast scenario

The digital television broadcast scenario has several specific characteristics, compared with the typical offline scenario (i.e. content stored as files in a local repository) assumed in most summarization related literature[Truong and Venkatesh, 2007].

In first place, the content is broadcasted continuously. That implies that the system must be capable of processing the incoming amount of data at an adequate rate so that no information is lost. The data received from the input stream is stored in a temporary buffer which is being continuously fed with new data. An adequate size of the buffer and its management are critical.

The amount of data conveyed by a broadcast television channel is higher than in other sources of video, such as video available in multimedia libraries (e.g. YouTube). The bitrate required for each TV channel varies typically from 3 to 6 Mbps. Although high resolution content is becoming available in multimedia libraries, the resolution of broadcasting networks is also increasing due to the availability of High Definition channels. Most works in video summarization and available data sets deal with content in a lower resolution (e.g. TRECVID[Over et al., 2007, 2008] and MPEG-7 content set[ITU-T and ISO/IEC, 1998] use CIF or similar resolution sequences). Decoding and processing high resolution content requires significantly more efforts than a lower resolution version (see, for instance, Table 5.3). For this reason, efficient algorithms are required to process the content.

In addition, if the system must process several channels simultaneously, the requirement for efficient processing is even more critical, as the same processing must be performed in parallel for every channel. These particular characteristics of the scenario and application described previously make the summarization of multichannel TV broadcasts a very challenging problem.

### 8.2.2. Online architecture

A simple adaptation of the architecture shown in Figure 5.2 is the use of a storage drive that stores the data received for certain period of time (e.g. the last one or two hours of every channel) and that is updated continuously. Then, when the user request the summary, the system can process, offline, the channels required. That approach has two drawbacks: the enormous storage capacity required to store several channels and a significant processing delay, as the content is processed completely only after the user requests the summaries.

However, observing how the data is processed by the analysis algorithm described in Chapter 5, part of the processing does not require to have all the content available and it can be performed in advance as the content arrives (i.e. online processing). The other part requires all the content to have been processed (i.e. offline processing). Thus, we propose the architecture depicted in Figure 8.2, which combines both online and offline processing trying to process as much data as possible as it arrives, in order to reduce the analysis delay. A first stage includes partial decoding to extract DC images, feature extraction and shot detection. This stage can process data in a GOP basis, so a buffer storing the last GOP is enough at this level. Once the shot boundaries are detected, the last shot, which has been kept in a different buffer, is processed to select the keyframes. The GOPs containing the keyframes are then forwarded to

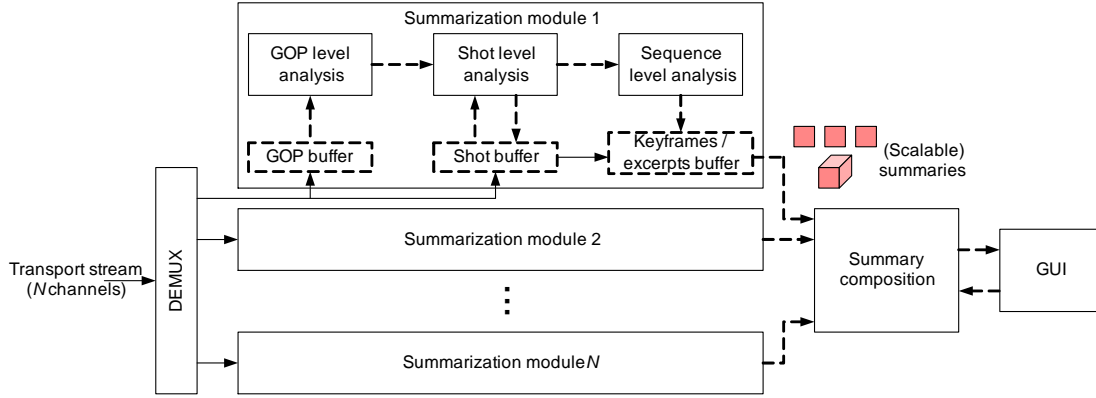


Figure 8.2: Architecture of the multichannel summarization system.

a last buffer. Additionally, the keyframes could be also transcoded to a more suitable format (e.g. JPEG) during the online stage to simplify the composition of the storyboards in the user interface. The last stage (clustering and ranking) is performed offline when the user request a summary. It only requires the feature vectors of the keyframes, which are already available from previous stages. Finally, the results from all the channels being processed are combined and presented to the user, who eventually may interact and change the amount of information shown (as in the application described previously).

Note that the system can be continuously processing the streams, extracting the feature vectors and storing the results (features and images in memory buffers or storage devices). Thus, the user can select the span to be summarized (e.g. 30 minutes, two hours, 24 hours). The only limitation would be the available storage space.

### 8.2.3. Delay analysis

With most of the computational effort shifted to an online stage, the actual delay perceived by the user is due to those processes which cannot be performed in advance. Assuming a bitstream with  $M$  GOPs and  $R$  shots, and including explicitly the delay due to demultiplexing and parsing, the delay due to analysis [adapted from Equation (5.17)] is

$$\begin{aligned}
 t_{analysis} &= t_{GOP\_level} + t_{shot\_level} + t_{clustering} + t_{ranking} \\
 &= \sum_{n=0}^{N-1} t_{parsing}(f_n) + \sum_{m=0}^{M-1} t_{GOP}(U_m) + \sum_{r=0}^{R-1} t_{shot}(s_r) + t_{clustering} + t_{ranking} \quad (8.1)
 \end{aligned}$$

where  $t_{parsing}(f_n)$ ,  $t_{GOP}(U_m)$  and  $t_{shot}(s_r)$  are the delays due to parsing the frame  $f_n$ , processing the GOP  $U_m$  and processing the shot  $s_r$ . In the online architecture, assuming that the frames are processed at a higher rate than the input rate, when the user requests the summary the delay is due to all the processing that could not be performed in advance. That is mainly the processing of the last units (frame, GOP and shot) and the offline analysis:

Sequence	Parsing and feature extraction		Clustering and ranking	Total delay	
	Offline (fps)	Online		Offline	Online
<i>dn2002-0228</i>	87.85 (593.65)	0.01	0.02	87.87	0.03
<i>Happy go lovely</i>	82.72 (630.46)	0.01	0.01	82.73	0.02
<i>Young man's fancy</i>	70.66 (738.07)	0.01	0.01	70.66	0.02
<i>BBS</i>	84.82 (614.84)	0.01	0.01	84.83	0.02
All	326.05 (639.80)	0.04	0.05	326.09	0.09

Table 8.1: Processing times and delays for the offline and online architectures (in seconds). Note: the precision to measure times was 10ms.

$$t_{analysis} \approx t_{parsing}(f_{N-1}) + t_{GOP}(\mathbf{U}_{M-1}) + t_{shot}(s_{R-1}) + t_{clustering} + t_{ranking} \quad (8.2)$$

We also assume that the keyframes are already available as JPEG images, so the delay due to the generation is negligible (there is no generation, only selection of the corresponding JPEG images).

#### 8.2.4. Experimental results

For the experiment carried out, we selected four public domain sequences from the Internet Archive and multiplexed them into a single MPEG-2 transport stream. The sequences cover news content (*dn2002-0228*), movies (*Happy go lovely*), sitcom (*Young man's fancy*) and documentary (*BBS*). All the sequences were encoded in MPEG-2 with a resolution of 720x480 pixels and a frame rate of 29.97 frames per second, using a GOP size of 13 frames. The sequences were shortened to 29 minutes (52152 frames), which is the duration of the shortest sequence (the sitcom).

Table 8.1 shows the processing time<sup>1</sup> for each of the test sequences in the transport stream, and the delay due to analysis since the user requests a summary (of the the last 29 minutes in this simulation). Clustering and ranking are common for both architectures, as they must be performed offline, once the feature vectors and shots have been extracted. Parsing (including demultiplexing and partial decoding) and feature extraction (including shot detection) can be performed either online or offline. In the online architecture, the same processing as in the offline architecture is carried out, although most was already performed before the interaction of the user. As we can observe, the processing speed is very high, around 600 frames per second per each channel. Even processing the four channels simultaneously, the system is still five times faster than real time, which guarantees the feasibility of the online architecture.

The delay with the online architecture is extremely low (around 90 ms in the experiment). However, deferring the processing burden to a completely offline stage leads to a very high delay (more than 5 minutes for 29 minutes of broadcast), which probably would be unacceptable for a practical application.

<sup>1</sup>Experiments performed in a Intel Core 2 Quad 2.83 Ghz processor (3.25 GB of RAM), but using only one processing core.

### 8.2.5. User interface

The purpose of the summarization user interface is to provide the user with as much information as possible, but in a appealing and easily understandable way. For this application we combined scalable storyboards and resizable canvases in a web interface that shows the four channels simultaneously (see Figure 8.3). The user can change the size of the images (changing the zoom factor) and automatically the scalable storyboard is adjusted to fill the canvas with the corresponding images (see Figure 8.3a and b). The user can also change the relative area given to each channel in order to have more detail of specific channels (see Figure 8.3c and d).

## 8.3. Summary and conclusions

In this chapter we have presented two applications of scalable summaries, focusing primarily on scalable storyboards and how they can be exploited in graphical user interfaces. With the proposed interfaces, the user has the freedom to adjust interactively the amount of detail and information.

Besides, the efficient analysis algorithm presented in Chapter 5 is used in a broadcast scenario for the problem of summarizing multiple TV channels. The architecture was adapted to perform most of the processing online so the delay perceived by the user is very low.

Although the analysis algorithm is generic and based on simple features, the results are satisfactory. However, more sophisticated analysis addressing the different types of content (e.g. news, sports, TV shows, movies), but still generating scalable summaries, would be very helpful to provide the user with a more structured and useful information.

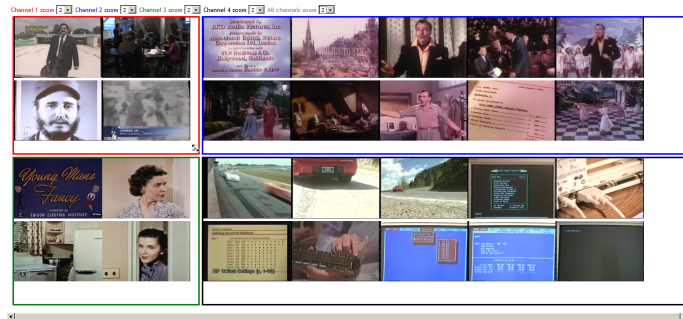
The application of scalable summaries is not limited to the scenarios and applications presented in this chapter. Scalable video skims could be also obtained with the same system and presented in a multichannel interface. The applications described in the preceding chapter can be easily enhanced including scalable summaries with adjustable length or scale. We believe that the underlying idea and adaptation mechanism of scalable summaries is powerful enough to be exploited in many applications involving browsing, adaptation and personalization.



(a)



(b)



(c)



(d)

Figure 8.3: User interface with resizable windows for multichannel summarization: (a) equally sized canvases (zoom factor 2), (b) equally sized canvases (zoom factor 4), (c) resized canvases (zoom factor 2) and (d) resized canvases (different zoom factors).



## Chapter 9

# Combined scalabilities and composite summaries

In previous chapters, we described applications using primarily either coding scalabilities (see Chapter 7) or scalable summaries (see Chapter 8). However, both techniques can be combined in order to obtain a highly scalable bitstream. In this chapter we briefly describe how to add a length scalability (from scalable summaries) to SVC bitstreams that are already scalable. In addition, we describe the application of these highly scalable bitstreams to composite summaries of news content.

### 9.1. Length scalable SVC bitstreams

#### 9.1.1. 4-D scalable bitstreams

Scalable video bitstreams, and particularly MPEG-4 SVC, usually deal with bitstreams that support three basic scalabilities: spatial, temporal and quality scalability. In the case of scalable summaries, their length is adjustable, according to the requirements at the time of adaptation. The original sequence is obtained as the summary with full length (highest scale). For convenience we will term this scalability as *length scalability*. Note that both temporal and length scalabilities change the number of frames, although using different methods. Temporal scalability changes the amount of frames per second without changing the duration of the sequence, while length scalability changes the duration of the sequence without changing its frame rate.

In both cases, adaptation is performed using a bitstream extractor. Thus, both adaptations can be integrated and performed in a single step, enabling bitstreams with four adaptation dimensions (i.e. spatial, temporal, quality and length). Figure 9.1 shows an example of a 4-D scalable bitstream and its adaptation. Conventional scalability is provided by using SVC with one base layer and three enhancement layers (in the example, two spatial scales, four temporal scales and two quality scales). Conceptually, length scalability is provided by arranging the

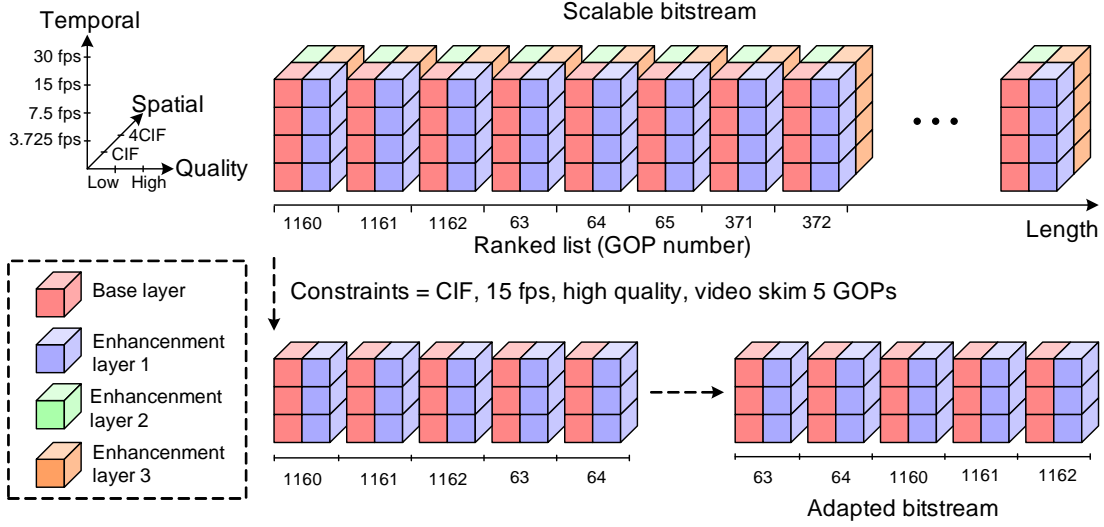


Figure 9.1: 4-D scalable bitstreams and the adaptation process.

GOPs (or SUs) in ranked order, according to a ranked list, so only a number of GOPs from the beginning are selected. In practice, it is not necessary to rearrange the GOPs, as the extractor can use the information in the ranked list to select or discard a GOP.

Spatial, temporal and quality scalabilities are enabled by the SVC syntax and information in SVC headers. However, additional information (i.e. ranked list) must be provided to enable length scalability.

### 9.1.2. Coding of ranked lists

The ranked list (or ranked lists if different sets of summaries are provided) is a simple list of indexes, which can be provided as metadata in text or binary formats. A binary format is more efficient in terms of compression, and it can be a simple list of unsigned integers (e.g. bytes or long integers, depending on the number of elements in the list). Although the amount of bits spent in coding the ranked lists is very low compared with the rest of the bitstream, we can easily exploit some properties of ranked lists to save some bits, especially for long lists.

For simplicity, we assume that the ranked list `list` is complete, i.e.  $M' = M$  in (5.2). In that case, the list is a permutation of  $I_V$ . Consequently, the entropy is  $\log_2 M$ , as the distribution of values in `list` is uniform. Figure 9.2a shows an example of ranked list (*news12* from the experiment described in Section 5.10.3), and the corresponding probability density function is shown in Figure 9.2b). The uniform distribution is the worst case in terms of source compression. However, ranked lists are usually highly correlated. For example, using the algorithm described in Chapter 5, a typical ranked list for video skims would be

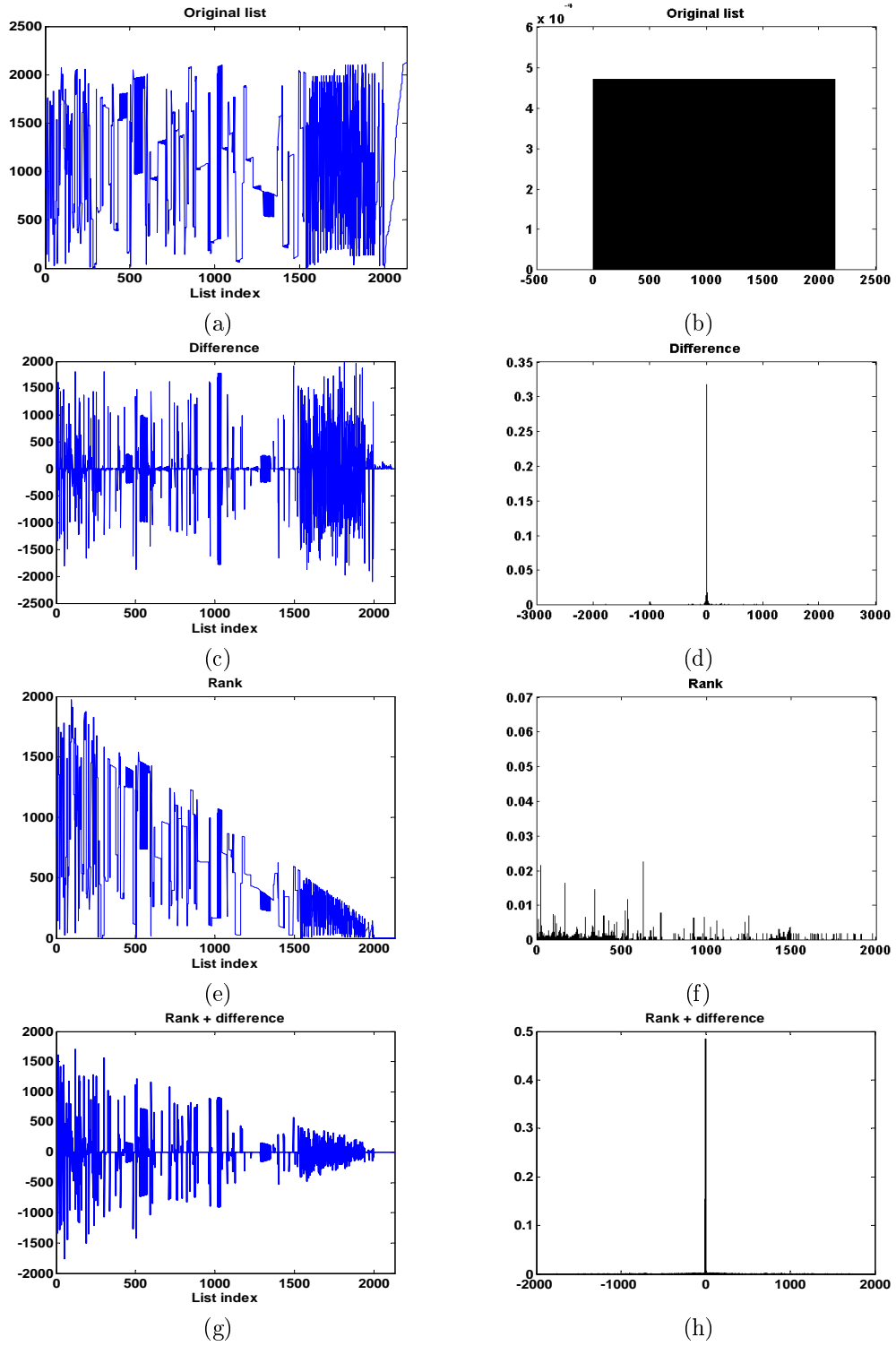


Figure 9.2: Lists and probability distributions after different transformations for *news12*: (a) and (b) original ranked list, (c) and (d) differential, (e) and (f) rank transformation, (g) and (h) rank transformation plus differential.

Test sequence			Source			Diff			Ranked			Ranked+diff		
Name	#GOP	S16	EN	LH	SH	EN	LH	SH	EN	LH	SH	EN	LH	SH
<i>MSNBCNEWS13</i>	3781	7.385	11.82	11.92	5.50	6.93	6.96	3.21	9.36	9.38	4.33	4.94	4.95	2.29
<i>contesting</i>	1735	3.389	10.76	10.82	2.29	6.85	6.88	1.46	8.53	8.56	1.81	4.74	4.79	1.01
<i>dn2002-0228</i>	4146	8.098	12.02	12.02	6.09	8.05	8.07	4.08	10.21	10.23	5.18	4.99	5.08	2.57
<i>MRS042538</i>	3683	7.193	11.85	11.89	5.34	8.14	8.17	3.67	9.33	9.35	4.21	4.83	4.89	2.20
<i>news12</i>	2130	4.16	11.06	11.08	2.88	6.57	6.59	1.71	8.64	8.66	2.25	4.64	4.65	1.21

Table 9.1: Coding results for ranked lists. S16: size of the list in KB using fixed-length integers (16 bits), EN: entropy, LH: mean length of the Huffman code, SH: size of the list in KB using Huffman.

$$\begin{aligned}
 &1160, 1161, 1162, 1163, \overbrace{63, 64, 65, 66}^{N_{exc}}, 371, 372, 373, 374, 789, 790, 791, 792, 532, 533, 534, 535, \dots \\
 &741, 720, 742, 719, 656, 651, 657, 650, 658, 649, 191, 648, 186, 192, 647, 185, 646, 1164, 1159, 1165, \dots
 \end{aligned}
 \tag{9.1}$$

where we can identify groups of several GOPs, that represent video excerpts from the cluster ranking stage. The indexes of GOPs included during the shot ranking stage also exhibit a strong correlation (see Figure 9.2a, especially when a shot is grown), as well as those included during the last stage.

With little extra cost, we can process the list to exploit these correlations and obtain a better coding efficiency. As many values are consecutive, coding the symbols differentially is very helpful to concentrate most of them in few low values (see Figure 9.2c and d). Another useful transformation is the rank encoding of a permutation. This transformation replaces each symbol in the permutation by its rank among the remaining symbols [Albert et al., 2003]. For example, the rank encoding of 341562 is 331221. The range of possible values is reduced progressively as the number of remaining symbols decreases (see Figure 9.2e). Combining both methods, almost half of the transformed symbols are zero (see Figure 9.2h).

We have tested these different coding options with the ranked lists obtained for the experiment of Section 5.10.3. Table 9.1 shows the experimental results, comparing the entropy, mean length and total size required to store each ranked list using Huffman codes for entropy coding. Both transformations (ranked and differential encoding) helps in effectively compressing the list information, by exploiting the properties of permutations. The best coding performance is achieved combining both.

## 9.2. Composite summaries

### 9.2.1. Introduction

As we have discussed in other chapters, the main objective of video summarization is to provide the user with a quick and informative representation of the content. A good trade-off between visualization time and amount of information is crucial for effective browsing, along with an intuitive and pleasant presentation format. Displaying several parts of the content (e.g.

frames, clips) at the same time reduces visualization time, and thus the summary is more compact. However, if the combination is not done properly, the user may become overwhelmed with an excessive amount of information, especially with moving images, resulting in useless summaries. We use the term composite summary to denote a summary which presents simultaneously information from different parts of the video.

Storyboard summaries could be considered as composite summaries of still images. Comic-like summaries and video trees are other examples. An example using moving images is described in [Dumont and Merialdo, 2007], which splits the frame into four windows presenting simultaneously four shots of the same video.

The structure of news content has been exploited to combine information from the anchorperson and the news story. In [Lie and Lai, 2005], the audio of the anchorperson is combined with a summary of the same length of the subsequent news story. This idea was extended in [Garcia et al., 2009; Valdés, 2010], including the video of the anchorperson overlaid in a small window. In both systems the summary is composed and created at the server.

### 9.2.2. Composite summaries of news video

Structured video content is usually composed of several segments in which some of them provide different types of information, either in the video or audio tracks. Sometimes, this structure can be exploited for a better and more compact abstraction. In this work, we focus primarily on news sequences, structured as an anchorperson introducing a subsequent news story. In most of the cases, this introduction is in fact a good, high level, summary of the news story. In that case the audio and video of both segments can be combined and presented simultaneously as they provide complementary information. Thus, the length of the abstract is reduced while the summary is more informative.

In our approach (see Fig. 9.3) we assume that the location of the anchorperson segment and the story segment are known, either by manual annotation or by automatic analysis [Avrithis et al., 2000; Lie and Lai, 2005; De Santo et al., 2006; Liu et al., 2009]. The anchorperson segment (both audio and video) is combined with a summary of the story with the same length and presented simultaneously. Anchorperson frames are reduced and overlaid in a small window over the summary of the story.

#### 9.2.2.1. Server side

Although summarization is performed at the server, the composition of the summary is performed at the client, in contrast to [Garcia et al., 2009; Valdés, 2010]. Thus, the cost of decoding, pixel domain composition and encoding at the server is avoided. Besides, the properties of scalable bitstreams and scalable summaries are used for efficient summarization. Although the approach is valid for non-scalable bitstreams, spatial scalability can be used to adapt the segment of the anchorperson, as only a smaller version is required at the client for composition, saving network bandwidth and decoding efforts at the client. Additionally, the summary of the story segment is obtained using lightweight summarization techniques, such as simple fast

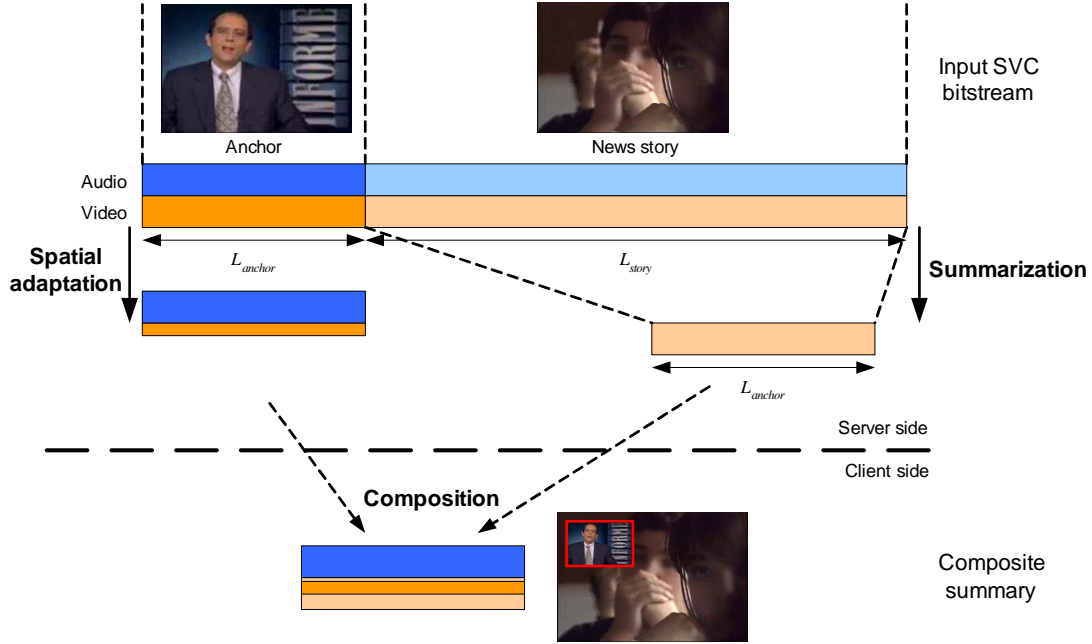


Figure 9.3: Overview of the composite summarization process.

forward or scalable summaries, which can create summaries of a specific length in adaptation time without additional summarization analysis. To avoid generation cost, bitstream extraction is used for the generation of both bitstreams, which are streamed to the client simultaneously.

#### 9.2.2.2. Client side

The client application was developed using web technologies. The composite summary is presented in a web page with two embedded players (one for each bitstream). Client side composition is much more dynamic and flexible than server side composition. Both videos can be laid out according to the preferences of the user and even changed dynamically (for example to uncover a part of the background video). The main difficulty is the synchronization of both streams, due to unequal buffering of both video streams (a spatially reduced anchorperson stream usually requires a much lower bitrate than the summary of the story).

To the best of our knowledge, SVC is not currently supported by any embeddable player, so for the subjective evaluation we emulated the client side functionality with two independent H.264/AVC streams (two spatial resolutions).

#### 9.2.3. Summarization approaches

In the proposed approach, the news story segment, of length  $L_{story}$ , must be condensed to a summary of length  $L_{anchor}$ , the length of the anchorperson segment. In most summarization algorithms, the length of the summary either cannot be adjusted to a target length or must be known prior to the analysis [Truong and Venkatesh, 2007]. We tested two different summarization techniques that can adjust the length of the summary without analyzing the content again.

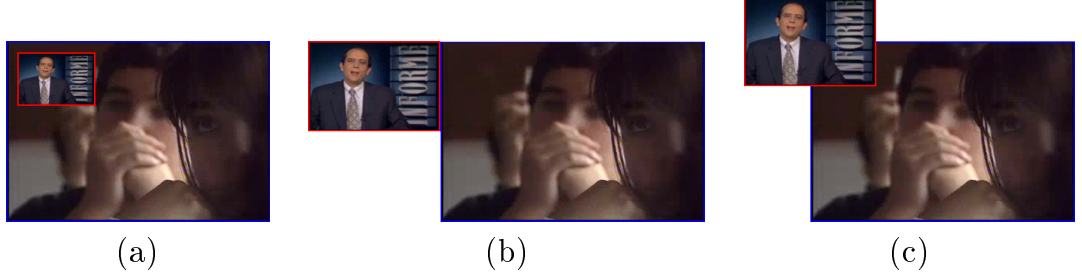


Figure 9.4: Different composition layouts in the web interface.

#### 9.2.3.1. Fast forwards

A fast forward summary is just the original source sequence played at a higher frame rate. This method does not require any analysis as it is not content-based. The compression (summarization) rate must be  $r = L_{\text{anchor}}/L_{\text{story}}$ , and thus, the speed-up must be  $R = L_{\text{story}}/L_{\text{anchor}} = 1/r$ . The summary can be easily obtained including one every  $R$  frames and played at the original frame rate. However it requires transcoding to generate the bitstream.

In contrast, we used a two step method based on temporal scalability. We first discard unnecessary temporal enhancement layers to avoid unnecessary consumption of network bandwidth and decoding at the client. Assuming a dyadic coding structure with hierarchical B-frames with  $T$  layers, the highest  $T_{\text{discard}} = \lfloor \log_2 R \rfloor$  temporal layers are discarded. If  $T_{\text{discard}} \geq T - 1$ , only frames from the lowest temporal level (i.e. intra frames) are selected, but skipping some of them in order to achieve a lower effective frame rate. Finally, as H.264/AVC and SVC allow arbitrary frame rates, we adjust more accurately the frame rate to  $\tilde{F} = rF2^{T_{\text{discard}}}$ , where  $F$  is the frame rate of the original sequence.

#### 9.2.3.2. Scalable video skims

Scalable summaries can easily adapt their length depending on the requirements. We used the algorithm proposed in Chapter 5, which generates scalable video skims with fine granularity. At any time a video skim of a given length  $L_{\text{skim}}$  is requested, the generation stage computes the number  $N$  of GOPs corresponding to that length. The first  $N$  values of the ranked list are selected and sorted in increasing order. The GOPs corresponding to those indexes are extracted from the original bitstream, obtaining the requested summary.

In the case of composite summaries, we assume that the ranked list for the news story segment is available. Then, the summary is generated with a target length of  $L_{\text{anchor}}$ . Due to the granularity of the scalable skim, the length of the skim  $L_{\text{skim}}$  might not be exactly  $L_{\text{anchor}}$ . It can be adjusted accurately modifying the frame rate to  $\tilde{F} = \frac{L_{\text{skim}}}{L_{\text{anchor}}} F$ . This adjustment is unnoticeable, as typically this mismatch is lower than 1%.

Test sequence		Summary		Fast forward				Video skim			
Name	Duration	Duration	Rate	IN	RT	OV	GT (sec)	IN	RT	OV	GT (sec)
<i>news1_TVE</i>	17m42.8s	40.2s	3.93%	4.2	2.4	3.2	8.37	4.5	4.6	4.4	8.42
<i>news8_CCTV4</i>	4m14.6s	11.6s	4.77%	3.5	2.1	2.6	0.91	3.6	4.2	3.5	0.89
<i>news5_NBC</i>	2m03.8s	9.6s	8.43%	3.8	2.3	3.1	1.23	3.8	4.3	3.9	1.22
<i>news6_NBC</i>	2m25.0s	16.6s	12.96%	4.3	3.1	3.6	0.81	4.3	4.5	4.4	0.81
<i>news3_TVE</i>	2m34.6s	24.2s	18.62%	4.4	2.8	3.6	1.00	4.4	4.5	4.4	1.01
<i>news2_TVE</i>	1m50.8s	19.5s	21.41%	4.6	2.9	3.7	1.17	4.6	4.6	4.5	1.16
<i>news9_CCTV4</i>	1m39.8s	19.7s	24.78%	4.1	3.2	3.5	1.17	4.2	4.5	4.2	1.16
<i>news4_NBC</i>	1m36.6s	19.1s	24.84%	4.5	3.2	3.6	2.06	4.4	4.5	4.3	2.08
<i>news7_NBC</i>	35.9s	13.1s	58.74%	4.4	3.8	3.9	0.81	4.4	4.4	4.1	0.83

Table 9.2: Results sorted by summarization rate. IN: Informative; RT: Rhythm; OV: Overall satisfaction; GT: Generation time

#### 9.2.4. Experimental results

We conducted a subjective evaluation of the proposed abstraction approach, in the context of news videos. A total of nine clips extracted from different data sets (MPEG-7 and TRECVID 2005) were encoded in SVC using the JSVM 9.18 encoder with two spatial layers (352x240/176x120 or 352x288/176x144) and a GOP length of 16 frames (5 temporal scales), and audio using MPEG-1 layer 3. These videos cover different TV sources, lengths and summarization rates (ratio between the lengths of the original video and its summary). Two composite summaries were generated for each video (i.e. using fast forward or video skim to summarize the news story), and were evaluated by 17 assessors. Three evaluation criteria (good information coverage, pleasant rhythm and overall satisfaction) were posed as affirmative statements and evaluated using a Likert scale (1:Strongly disagree; 3: Nor agree nor disagree; 5: Strongly agree)[Likert, 1932].

Table 9.2 shows the test videos sorted by summarization rate and the results. Figure 9.5 shows the scores obtained for information coverage, rhythm and overall satisfaction plotted in graphs for easier visualization. In general, results are satisfactory for both types of summaries. The results are better for video skims, which are also preferred to fast forward when the assessors were asked for their explicit preference (see Figure 9.6). As expected, very low summarization rates are very challenging, with degraded results, slightly for video skims and significantly for fast forwards, in which the dramatic speed-up makes the rhythm and consequently the summary more stressing and unpleasant.

Table 9.2 also shows the generation time (total extraction time of both video bitstreams and the audio bitstream) of the summaries<sup>1</sup>, supporting its suitability when efficiency or low delay are required. For video streams a modified JSVM 9.12.2 extractor was used. The extractor was not optimized so the generation time could be significantly reduced.

Finally, we asked the assessors (using the same Likert scale) about their satisfaction with the interface and dynamic layouts. The mean score was 4.53 out of 5 in overall satisfaction.

<sup>1</sup>Experiments performed in an Intel Core 2 Duo at 1.8Ghz (2GB of RAM)

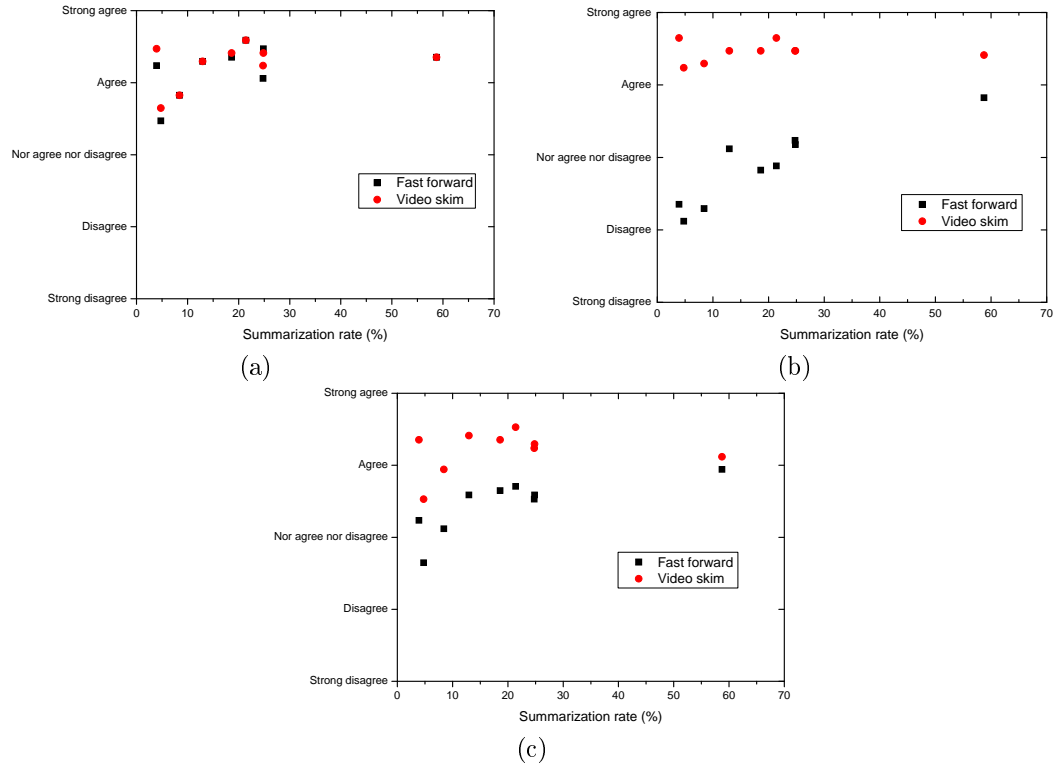


Figure 9.5: Subjective evaluation results: (a) information coverage, (b) rhythm, and (c) overall satisfaction.

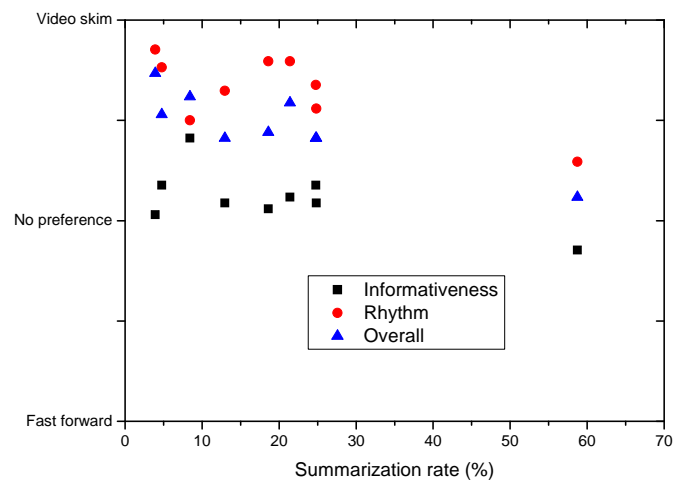


Figure 9.6: Preference between video skims and fast forwards in composite summaries.

### 9.3. Summary and conclusions

In this chapter we have described two applications which combine most of the ideas proposed in the thesis. A highly scalable video framework combining scalable summaries and conventional coding scalabilities has been described. In that framework, a single bitstream embeds multiple versions of the content, but also embeds multiple summaries. These ideas are combined also in the context of news video summarization. Exploiting the structure of news videos, the information is presented in a compact but appealing composite format, and it is generated using bitstream extraction and scalable summaries.

## Part V

# Conclusions



## Chapter 10

# Conclusions and future work

This thesis has proposed a novel integrated approach to video summarization and adaptation based on the idea of scalable bitstreams. Most of the techniques described in the thesis were inspired by the simplicity and flexibility of the adaptation of bitstreams in scalable video coding. The concept of scalable bitstream is applied in a novel way to obtain scalable summaries, which can be also adapted easily without any further content analysis. The use of a scalable representation format is a powerful tool in applications in which the content must be adapted. In this chapter we summarize the main results and conclusions yielded from the thesis.

### 10.1. Summary and conclusions

Chapter 1 introduced the main motivation and objectives of the thesis, which mainly are the exploration of scalable representations and their applications in video summarization and adaptation. Chapter 2 described the research and technology context in which the rest of the thesis was developed. A brief overview of the related video coding, summarization and adaptation techniques was provided, along with some review of scalable approaches in other different areas, as scalability was rarely used in the context of video summarization.

The next two chapters deal with how scalable video coding, an already available technology, can be used in a new context, namely video summarization. For convenience, in the summarization architecture proposed in Chapter 3, analysis and generation processes were decoupled. In contrast to most works in video summarization, the emphasis was laid on the generation of the bitstream, as a key part for applications such as low delay summarization and scalable summaries (as discussed later in Chapter 5). Analysis, generation and the description model are based on coding units (summarization units) rather than on single frames, which enables efficient processing using bitstream extraction. Experiments showed that, with this framework and model, extraction can generate the bitstream much faster than the alternative transcoding approach, and without any loss of quality.

Since extraction is also the main tool in scalable video adaptation, the summarization framework was extended in Chapter 4 to include adaptation. Using a single bitstream extractor, a

summary adapted to the usage environment is generated in a single and efficient step. Experiments showed that extraction is still much more efficient than transcoding, although there is some loss of quality in enhanced versions due to layered coding (as shown in Chapter 7).

The next two chapters of the thesis proposed a novel use of the concept of scalability as an intrinsic property of the summaries, in a different sense of that used in the previous part. With scalable summaries, the length (or duration), which is the most important characteristic of a summary, can be adjusted without any further processing. Chapter 5 discussed how the length of the summary can influence two important properties of a summary: semantic coverage and visual pleasantness. Based on these observations, a suitable representation and framework were proposed for scalable storyboards and video skims. The analysis algorithm is based on a novel iterative procedure which combines clustering and ranking to increase gradually the length of the summary including new visual information. This method enables a much finer granularity than hierarchical summaries and a very compact representation as a ranked list. The results of the evaluations were encouraging, and showed that even video skims can be generated with very low latency. Scalable summaries enable an easy and fast adaptation of summaries, which make them useful in many applications, such as customization, personalization or browsing.

The concept of scalable summaries was extended in Chapter 6 to comic-like summaries. These summaries were posed as enhanced storyboards, which enable the user to navigate across different scales with increasing level of details. In contrast to storyboards, the problem of laying the images out in a comic-like layout is not trivial, with implications in efficiency and visual disturbing effects in the transition between scales. Two layout algorithms were proposed: a basic method and an enhanced method to reduce the disturbing effects. Results showed that the enhanced method effectively reduces the changes in the transition, making the navigation across scales more comfortable.

The following three chapters described several applications of the previous methods. Chapter 7 extended the comparison of transcoding and extraction, including also variations, and describing how these architectures can be used in different application contexts. The problem of including and multiplexing audio was also discussed in this chapter, along with applications such as customized summaries and browsing.

Chapter 8 described two applications which can benefit from scalable storyboards. The first one is the use of storyboards that automatically adjust their size to the size of a window or canvas, which is very effective to dynamically adjust the amount of visual information. The second one combines the high efficiency of the analysis algorithm described in Chapter 5 with the previous application. The system is able to process a multiplex with several TV channels with an extremely low delay using an online architecture. The resulting storyboards are presented in an integrated manner, in which the user can distribute the amount of information of each channel using the dynamic user interface and the scalability of the storyboards. This last application shows the potential of scalable summaries and low delay summarization in a demanding environment (i.e. multiple high resolution channels).

Finally, Chapter 9 presented two applications combining all the scalabilities described in the thesis. Scalable summaries can be combined with scalable video coding, to obtain bitstreams

that can be scaled in four dimensions: spatial, temporal, quality and length. Such bitstreams can be very useful in browsing and adaptation systems. A second application summarizes news stories in composite summaries. The segment with the anchorperson introducing the story is combined with a summary (video skim or fast forward) of the segment with the story. They are obtained using different scalabilities, and composed at the client. Evaluations showed that this composite format effectively condenses the information in an appealing way.

Scalable representations have been the main objective of the thesis. The techniques and frameworks proposed in the different chapters have shown a number of advantages and potential applications of scalable bitstreams. Browsing, personalization, adaptation and user interfaces can benefit from easily adaptable videos or summaries.

Efficiency has also played an important role in the thesis. One of the requirements to fully benefit from scalable summaries is that the generation of the summary, once a certain scale is requested, must be very fast. Besides, although not strictly required by scalable summaries, fast analysis techniques were proposed. They can be useful in other demanding scenarios in which large amounts of data need to be processed and the summary must be presented without a considerable delay. An example is the proposed application to multichannel summarization of TV streams.

## 10.2. Main results and contributions

We can summarize the main results and contributions of the thesis as the following:

- A framework for efficient generation of summaries using bitstream extraction, and a suitable model to represent the summaries.
- The integration of the previous framework and scalable video into a joint framework that can generate the bitstream of summaries adapted to the usage environment.
- Study and comparison of the proposed frameworks to alternative architectures (i.e. transcoding and variations).
- Analysis of the applicability of the idea of scalability in video summarization and analysis of the requirements for practical utility. From that analysis, a suitable description (ranked list) and framework were proposed to generate scalable summaries.
- A suitable analysis algorithm to generate scalable storyboards and video skims. The algorithm was also designed to be very efficient.
- Method and architecture to generate scalable comic-like summaries. Analysis of the problem of disturbance and a heuristic layout method to lessen its effect.
- Novel applications of the proposed methods, such as low delay and customizable summaries, resizable interfaces using scalable storyboards, composite summaries of news content with client side composition, and efficient summarization of multichannel TV broadcasts.

Task/application	Video coding scalabilities			Summarization scalability	Summary description
	Temporal	Spatial	Quality	Length	
Adaptation (SVC)	A	A	A		
Summarization (Chapter 3)	S				Summarization curve
Adapted summaries (Chapter 4)	A,S	A	A		Summarization curve
Scalable summaries (Chapter 5)				X	Ranked list
Scalable comic-like summaries (Chapter 6)				X	Multiscale layout description or panelled description
Customized summaries (Chapter 7)	S				Summarization curve
Resizable summaries (Chapter 8)				X	Ranked list
Multichannel scalable summaries (Chapter 8)				X	Ranked lists
Adapted scalable summaries (Chapter 9)	A,S	A	A	X	Ranked list
Composite summaries (Chapter 9)	A,S	A,S	A	X	Summarization curves or ranked lists

Table 10.1: Use of the different scalabilities along the thesis. A: used for adaptation, S: used for summarization.

### 10.3. Scalability in the different proposed techniques

As discussed previously in Chapter 2, the term scalability has different interpretations depending on the field. Within the thesis, we have also used the word scalability with slightly different meanings. In this section we analyze how different scalabilities have been used throughout the thesis. We can distinguish between non-semantic (or content-blind) and semantic scalabilities. The former are the scalabilities that scalable video coding enables for video adaptation, and which some techniques in the thesis used for semantic purposes (i.e. summarization). In this thesis we also introduced a semantic scalability, namely the length scalability of scalable summaries.

Table 10.1 compares different techniques and indicates which scalabilities are used. Conventional adaptation of scalable video coding uses temporal, spatial and quality scalabilities. In Chapter 3 and the customized summaries of Chapter 7, temporal scalability is exploited also for summarization. In Chapter 4, temporal scalability is used for both summarization and adaptation purposes, while spatial and quality scalabilities are used only for adaptation. Length scalability is used for scalable storyboards and video skims, in Chapter 5, scalable comic-like summaries, in Chapter 6, and resizable summaries and multichannel summaries, in Chapter 8. Adapted scalable summaries and composite summaries, described in Chapter 9, use the four scalabilities.

### 10.4. Future work

Several research directions come up from the work developed in this thesis:

- Improving the analysis algorithms with a more advanced processing and higher level features. The thesis has focused mainly on generation and representation, providing a framework to generate scalable, fast and flexible summaries. However, the analysis algorithms have been based on low level characteristics and they can benefit from ideas from other works in video summarization[Truong and Venkatesh, 2007; Money and Agius, 2008b]. The following aspects may improve the semantic quality of the summaries:
  - Enhanced scoring approaches for scalable summaries. Additional features and criteria can be included in the scoring and ranking methods.
  - Specific analysis for different types of content (e.g. news, movies, sport), so the different scales can be created with a more intuitive distribution of information. Customized summaries and composite summaries can also benefit from an automatic structuring method.
- Extension of the analysis algorithms developed for MPEG-2 to H.264/AVC. Due to the complexity of H.264/AVC, the resulting analysis may be slower, so specific methods for efficient processing of H.264/AVC would be also convenient.
- Currently, the disturbance is the main drawback of the scalable comic-like summaries. A better study of these effects may be very helpful to better understand how to avoid it. Higher level semantic analysis can be also helpful to obtain a more meaningful distribution of images across panels (e.g. panels representing scenes).
- Studying of new ways of integrating scalable summaries into user interfaces. The easy and fast adaptation of scalable summaries can be a more dynamic alternative to conventional representations in browsing and adaptation interfaces (e.g. keyframes, thumbnails).
- Exploring new applications of low delay summarization.
- An efficient evaluation methodology for scalable summaries. One of the main handicaps of scalable summarization is that the number of summaries is much higher than in conventional summarization, which makes very desirable a better evaluation framework.

## 10.5. Published work

Part of the work in this thesis has also yielded some publications. In this section we classify these publications by chapter and research topic.

### Integrated summarization and adaptation

#### Chapter 3. Generation of video summaries by bitstream extraction

- L. Herranz, J.M. Martínez, "On the use of hierarchical prediction structures for efficient summary generation of H.264/AVC bitstreams", *Signal Processing: Image Communication*, vol. 24, no. 8, pp. 615-629, September 2009

- L. Herranz, J.M. Martínez, "On the advantages of the use of bitstream extraction for video summary generation", International Conference on Multimedia Modeling, Lecture Notes on Computer Science, vol 5916, pp. 755-760, Springer Verlag, Chongqing, China, January 2010

### Chapter 4. Integrated summarization and adaptation

The initial development of the integrated summarization and adaptation model was carried out in the context of a non-standard wavelet-based scalable video codec. However, the main framework is an early version of that described in this thesis. The following publications are from that early stage:

- L. Herranz, "A framework for online semantic adaptation of scalable video", Proc. IEEE International Workshop on Semantic Media Adaptation and Personalization, pp. 13-18, Athens, Greece, December 2006
- L. Herranz, "Integrating semantic analysis and scalable video coding for efficient content-based adaptation", Multimedia Systems, vol. 13, no. 2, pp. 103-118, August 2007
- L. Herranz, J.M. Martínez, "Use cases of scalable video based summarization within MPEG-21 DIA", International Conference on Semantic and Digital Media Technology, Lecture Notes on Computer Science, vol 4816, pp. 256-259, Springer Verlag, Genoa, Italy, December 2007

The following publications present the framework adapted to the standardized scalable extension of H.264/AVC:

- L. Herranz, J.M. Martínez, "Integrated summarization and adaptation using H.264/MPEG-4 SVC", Proc. International Conference on Visual Information Engineering, pp. 729-734, Xi'an, China, July 2008
- L. Herranz, J.M. Martínez, "An integrated approach to summarization and adaptation using H.264/MPEG-4 SVC", Signal Processing: Image Communication, vol. 24, no. 6, pp. 499-509, July 2009
- L. Herranz, J.M. Martínez, "Using MPEG Tools in Video Summarization", The Handbook of MPEG Applications: Standards in Practice, Ed. Marios C. Angelides and Harry Agius, John Wiley & Sons, 2011 (in press)

### Scalable summaries

#### Chapter 5. Scalable storyboards and video skims

- L. Herranz, J.M. Martínez, "Generation of scalable summaries based on iterative GOP ranking", Proc. International Conference on Image Processing, pp. 2544-2547, San Diego, California, October 2008

- L. Herranz, J.M. Martínez, "An efficient summarization algorithm based on clustering and bitstream extraction", Proc. International Conference on Multimedia and Expo, pp. 654-657, New York, New York, July 2009
- L. Herranz, J.M. Martínez, "A framework for scalable summarization", IEEE Transactions on Circuits and Systems for Video Technology, vol. 20, no. 9, pp. 1265-1270, September 2010



## Appendix A

# Conclusiones y trabajo futuro

En esta tesis se ha propuesto un enfoque integrado a la creación de resúmenes (*video summarization*) y a la adaptación de vídeo (*video adaptation*) basado en la idea de *bitstreams* (flujos de bits) escalables. La mayoría de las técnicas descritas en la tesis están inspiradas por la simplicidad y flexibilidad de adaptación de los bitstreams obtenidos mediante codificación de vídeo escalable. El concepto de bitstream escalable se aplica de una forma novedosa para obtener resúmenes escalables, los cuales pueden ser adaptados fácilmente sin necesidad de ningún análisis adicional del contenido (a diferencia de algoritmos convencionales no escalables, que deberían ejecutarse con diferentes parámetros cada vez que se necesita obtener un resumen adaptado). El uso de una formato de representación escalable es un potente herramienta en aplicaciones en las cuales el contenido debe ser adaptado. En este capítulo se recogen los resultados y conclusiones principales obtenidos de esta tesis.

### A.1. Resumen y conclusiones

El Capítulo 1 introduce la motivación y objetivos principales de la tesis, que fundamentalmente son la exploración de representaciones escalables y su aplicación a la creación y adaptación de resúmenes. El Capítulo 2 describe el contexto tecnológico y de investigación en el cual se enmarca la tesis. El capítulo también recoge una breve visión general de las técnicas relacionadas de codificación, generación de resúmenes y adaptación de vídeo, junto con la revisión de la aplicación de la idea de escalabilidad en diferentes áreas, ya que raramente se ha aplicado en el contexto de resúmenes de vídeo.

Los siguientes capítulos exploran como la codificación de vídeo escalable, una tecnología ya establecida y utilizada para adaptación de vídeo, puede utilizarse en un nuevo campo, el de los resúmenes de vídeo. Por conveniencia, la arquitectura propuesta en el Capítulo 3 distingue entre dos etapas en el procesado: análisis del contenido y generación del bitstream. A diferencia de la mayoría de trabajos en este campo, el énfasis se pone en la generación del bitstream como parte crucial para ciertas aplicaciones como pueden ser la generación de resúmenes con bajo retardo y los resúmenes escalables (como se verá posteriormente en el Capítulo 5). El análisis,

la generación y el modelo de descripción están basados en unidades de codificación (utilizando el término *summarization units*) en lugar de imágenes (*frames*) individuales. Esto permite procesar eficientemente el vídeo utilizando extracción de bits (es decir, el bitstream de salida se obtiene seleccionando ciertos paquetes del bitstream de entrada, sin necesidad de decodificar y volver a codificar). El sistema propuesto es flexible y permite representar *storyboards* (guiones gráficos hechos con imágenes estáticas), *video skims* (similares a los trailers de películas, creados mediante la concatenación de segmentos del vídeo original) y *fast forwards* (obtenidos mediante la aceleración de la secuencia original de forma que se reduce su duración). Los resultados experimentales muestran que, mediante la arquitectura basada en extracción, el bitstream se genera mucho más rápido que utilizando una arquitectura alternativa basada en transcodificación. Además, mediante extracción no se pierde calidad de imagen en el proceso, a diferencia de la transcodificación que sí la deteriora.

Dado que la extracción de bits es también la herramienta fundamental en la adaptación de vídeo escalable, la arquitectura se extiende en el Capítulo 4 para incluir adaptación. Mediante un único extractor de bits, se puede obtener un resumen adaptado a las condiciones de uso del usuario en un único paso y de forma muy eficiente. Los experimentos muestran que la extracción continúa siendo mucho más eficiente que la transcodificación, aunque en este caso sí se pierde algo de calidad en las versiones distintas a la básica, debido a la codificación de vídeo escalable (como se estudia en el Capítulo 7).

La siguiente parte de la tesis propone un uso novedoso del concepto de escalabilidad, en un sentido distinto al utilizado en la parte anterior. En este caso, la escalabilidad es una propiedad intrínseca de los resúmenes. Utilizando resúmenes escalables la longitud (o duración), que es la característica más importante de un resumen, se puede ajustar si necesidad de procesamiento adicional del contenido. El Capítulo 5 analiza como la longitud del resumen puede influir en dos propiedades importantes: cobertura semántica (es decir, que cubra suficiente información semántica) y agrado visual (es decir, sin que el resumen tenga artefactos visuales no deseables o sea incómodo de ver). Basado en este análisis, se propone una representación y una arquitectura adecuada para obtener storyboards y video skims escalables. El algoritmo de análisis está basado en un proceso iterativo que combina clustering y ranking para incrementar progresivamente la longitud del resumen incluyendo nueva información visual. Este método permite una mayor granularidad que los resúmenes jerárquicos, y una representación muy compacta (lista de índices). Los resultados de las evaluaciones muestran que los resúmenes se pueden generar con un retardo muy bajo. Los resúmenes escalables permiten una adaptación sencilla de la longitud de los resúmenes, lo cual les hace útiles en numerosas aplicaciones, tales como personalización o navegación.

El concepto de resumen escalable se extiende en el Capítulo 6 a resúmenes de tipo comic. En el capítulo, estos resúmenes se proponen como storyboards mejorados, que permiten al usuario navegar mediante diferentes escalas con un creciente nivel de detalle (más imágenes). A diferencia de los storyboards, el problema de distribuir las imágenes en una estructura de tipo cómic no es trivial, con implicaciones sobre la eficiencia y efectos visuales desagradables en las transiciones entre escalas. Se proponen dos algoritmos: un método básico y otro mejorado para reducir los

efectos visuales molestos. Los resultados muestran que el método mejorado permite reducir efectivamente la cantidad de cambios entre las transiciones, haciendo la navegación más cómoda.

Una última parte de la tesis propone diferentes aplicaciones de los métodos descritos en capítulos anteriores. El Capítulo 7 extiende la comparación entre transcodificación y extracción, incluyendo además una arquitectura alternativa basada en variaciones, y describiendo cómo estas arquitecturas se pueden utilizar en diferentes aplicaciones. El problema de incluir y multiplexar audio se estudia también en este capítulo, junto con aplicaciones tales como resúmenes personalizados y navegación.

Chapter 8 describe dos aplicaciones que pueden utilizar storyboards escalables. La primera es el uso de storyboards cuyo tamaño se ajusta automáticamente (cambiando el número de imágenes que se muestran) al tamaño de la ventana, permitiendo al usuario ajustar dinámicamente la cantidad de información visual. La segunda aplicación combina la eficiencia del algoritmo de análisis descrito en el Capítulo 5 con la aplicación anterior. El sistema es capaz de procesar un multiplex con varios canales de televisión digital con un retardo extremadamente bajo, utilizando una arquitectura online. Los storyboards resultantes se presentan de una forma integrada, en la cual el usuario puede distribuir la cantidad de información visual de cada canal dinámicamente utilizando el interfaz y la escalabilidad de los storyboards. Esta última aplicación muestra el potencial de los resúmenes escalables y la generación de resúmenes con bajo retardo en un entorno especialmente exigente (i.e. múltiples flujos de vídeo de alta resolución simultáneos).

Por último, el Capítulo 9 presenta dos aplicaciones combinando todos los tipos de escalabilidad descritos en la tesis. Los resúmenes escalables se pueden combinar con codificación de vídeo escalable para obtener bitstreams que se pueden escalar en cuatro dimensiones: espacial, temporal, calidad y longitud. Tales bitstreams pueden ser muy útiles en sistemas de visualización y adaptación. Una segunda aplicación genera un resumen de una noticia de un informativo en forma de resumen compuesto. El segmento con el presentador introduciendo la noticia se combina con un resumen (video skim o fast forward) del segmento con el reportaje. Ambos se obtienen utilizando diferentes escalabilidades, y se componen en el cliente. Las evaluaciones subjetivas muestran que el formato compuesto condensa eficazmente la información en una forma atractiva para el usuario.

Las representaciones escalables han sido el objetivo principal de la tesis. Las técnicas y arquitecturas propuestas en los diferentes capítulos han puesto de manifiesto una serie de ventajas y aplicaciones potenciales de los bitstreams escalables (en el contexto de resúmenes y adaptación). La personalización y adaptación de video, así como los interfaces de uso y herramientas de navegación y visualización pueden beneficiarse de vídeos y resúmenes fácilmente adaptables.

Por otro lado, la eficiencia también ha desempeñado un papel importante en la tesis. Uno de los requisitos para que los resúmenes escalables sean realmente útiles en la práctica es que la generación del resumen (bitstream), una vez que se solicita una escala, debe ser muy eficiente. Además, aunque no siendo estrictamente necesario para los resúmenes escalables, se han propuesto técnicas de análisis eficiente. Estas técnicas pueden ser útiles en otros escenarios en los que grandes cantidades de datos de vídeo deben ser procesados o los resúmenes presentados con un retardo pequeño. Un ejemplo es la obtención de resúmenes de múltiples canales del televisión.

## A.2. Resultados y contribuciones

Los principales resultados y contribuciones de la tesis son los siguientes:

- Una arquitectura para la generación eficiente de resúmenes utilizando extracción de bits, y un modelo de representación adecuado para representar los resúmenes.
- La integración de la arquitectura anterior y la codificación de vídeo escalable en una arquitectura conjunta que puede generar bitstreams de resúmenes adaptados al contexto de uso.
- Estudio y comparación de la arquitectura propuesta con otras arquitecturas alternativas (transcoding y variaciones).
- Estudio de la aplicabilidad de la idea de escalabilidad en el contexto de resúmenes de vídeo, y análisis de los requisitos necesarios para que tengan utilidad práctica. A raíz de tal estudio, una descripción adecuada (ranked list) y una arquitectura se proponen para generar resúmenes escalables.
- Un algoritmo de análisis adecuado para generar storyboard y video skims escalables. El algoritmo es además muy eficiente.
- Método y arquitectura para generar resúmenes de tipo cómic escalables. Análisis del problema de efectos no deseados y un método de layout heurístico para reducir su impacto.
- Nuevas aplicaciones de los métodos propuestos, tales como los resúmenes personalizables y de bajo retardo, resúmenes adaptables al tamaño de la ventana, resúmenes compuestos y la obtención rápida de resúmenes de múltiples canales de televisión.

## A.3. Trabajo futuro

Varias líneas de investigación surgen a raíz del trabajo desarrollado en la tesis:

- Mejora de los algoritmos de análisis con procesamiento más avanzado y características de más alto nivel. La tesis se ha centrado fundamentalmente en la parte de generación y representación, proponiendo un marco de trabajo para la generación rápida y flexible de resúmenes escalables. Sin embargo, los algoritmos de análisis se han basado en características de bajo nivel y podrían verse beneficiados por las ideas desarrolladas en otros trabajos en el campo de la abstracción de vídeo[Truong and Venkatesh, 2007; Money and Agius, 2008b]. Los siguientes aspectos podrían mejorar la calidad semántica de los resúmenes:
  - Métodos de puntuaciones (*scores*) mejorados en el algoritmo de análisis para resúmenes escalables. Características y criterios adicionales pueden incluirse en el cálculo de puntuaciones y ranking.

- Análisis específico para diferentes tipos de contenido (p.e. noticias, películas, deportes), de forma que las diferentes escalas se pueden crear con una distribución de información más intuitiva, de acuerdo con el tipo de contenido. Los resúmenes personalizados y compuestos también se pueden beneficiar de un método automático para estructurar el contenido.
- Extensión de los algoritmos de análisis desarrollados para MPEG-2 a H.264/AVC. Debido a la mayor complejidad de H.264/AVC, el análisis resultante puede resultar más lento, por lo que algoritmos específicos para H.264/AVC pueden ser convenientes.
- Actualmente, el problema de los efectos visuales molestos es el mayor problema de los resúmenes de tipo cómic escalables. Un estudio más en profundidad de estos efectos podría ser muy útil para entender mejor como evitarlo. Métodos de análisis semántico de más alto nivel pueden ser también útiles para obtener una distribución de imágenes en los paneles más intuitiva (p.e. paneles representando escenas).
- Estudio de nuevas formas de integrar resúmenes escalables en interfaces de usuario. La simple y rápida adaptación de los resúmenes escalables puede ser una alternativa más dinámica a representaciones convencionales utilizadas en interfaces de visualización, navegación y adaptación (p.e. imágenes clave, miniaturas).
- Explorar nuevas aplicaciones de la generación de resúmenes con bajo retardo.
- Una metodología eficiente de evaluación de resúmenes escalables. Uno de los mayores problemas de los resúmenes escalables es que el número de instancias a evaluar es mucho mayor que en los métodos convencionales, lo que hace muy conveniente un marco de evaluación más adecuado y eficiente.



# Glossary

ATSC	Advanced Television Systems Committee
AU	Access unit
AVC	Advance Video Coding
BSD	Bitstream Syntax Description
BSDL	Bitstream Syntax Description Language
CGS	Coarse grain scalability
CIF	Common intermediate format (352x288 pixels)
DCT	Discrete Cosine Transform
DI	Digital Item
DIA	Digital Item Adaptation
DPB	Decoded picture buffer
DVB	Digital Video Broadcasting
DVD	Digital Versatile Disc
EPZS	Enhanced Predictive Zonal Search
FGS	Fine grain scalability
GOP	Group of pictures
IDR	Instantaneous decoding refresh
IEC	International Electrotechnical Commission
ISDB	Integrated Services Digital Broadcasting
ITU	International Telecommunication Union
JPEG	Joint Pictures Experts Group

MDS	Multimedia Description Schemes
MGS	Medium grain scalability
MJPEG	Motion JPEG
MMCO	Memory management control operations
MPEG	Moving Pictures Experts Group
NAL	Network abstraction layer
PDA	Personal digital assistant
PPS	Picture Parameter Set
PSNR	Peak signal-to-noise ratio
SNR	Signal-to-noise ratio
SPS	Sequence Parameter Set
SU	Summarization unit
SVC	Scalable Video Coding
UED	Usage Environment Description
UMA	Universal Multimedia Access
VCL	Video coding layer
YUV	Color space including a luminance (Y) and two chrominance (U and V) channels

# Bibliography

- Acharya, S. and Smith, B. (1998). Compressed domain transcoding of MPEG. In *Proc. of International Conference on Multimedia Computing and Systems*, pages 295–304. 2.4.2, 5.8.2
- Adami, N., Signoroni, A., and Leonardi, R. (2007). State-of-the-art and trends in scalable video compression with wavelet-based approaches. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(9):1238–1255. 2.1.3
- Ahmad, I., Wei, X., Sun, Y., and Zhang, Y.-Q. (2005). Video transcoding: an overview of various techniques and research issues. *IEEE Transactions on Multimedia*, 7(5):793–804. 2.4.2
- Albanese, M., Fayzullin, M., Picariello, A., and Subrahmanian, V. (2006). The priority curve algorithm for video summarization. *Information Systems*, 31(7):679–695. 5.2, 5.4.5, 5.7.1
- Albert, M. H., Atkinson, M. D., and Ruskuc, N. (2003). Regular closed sets of permutations. *Theoretical Computer Science*, 306(1-3):85 – 100. 9.1.2
- Amon, P., Rathgen, T., and Singer, D. (2007). File format for scalable video coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(9):1174–1185. 2.2.3
- Amonou, I., Cammas, N., Kervadec, S., and Pateux, S. (2007). Optimized rate-distortion extraction with quality layers in the scalable extension of H.264/AVC. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(9):1186–1193. (document), 4.3, 4.2.2, 4.3.3
- Anastassiou, D. (1994). Digital television. *Proceedings of the IEEE*, 82(4):510–519. 1.1
- Ariki, Y., Kumano, M., and Tsukada, K. (2003). Highlight scene extraction in real time from baseball live video. In *Proc. of International Workshop on Multimedia information retrieval*, pages 209–214, New York, NY, USA. ACM. 2.3.3.1
- Avrithis, Y., Tsapatsoulis, N., and Kollias, S. (2000). Broadcast news parsing using visual cues: a robust face detection approach. In *Proc. of International Conference on Multimedia and Expo*, volume 3, pages 1469 –1472 vol.3. 9.2.2
- Babaguchi, N., Ohara, K., and Ogura, T. (2007). Learning personal preference from viewer’s operations for browsing and its application to baseball video retrieval and summarization. *IEEE Transactions on Multimedia*, 9(5):1016–1025. 2.3.3, 7.1

- Bae, T. M., Thang, T. C., Kim, D. Y., Ro, Y. M., Kang, J. W., and Kim, J. G. (2006). Multiple region-of-interest support in scalable video coding. *ETRI Journal*, 28(2):239–242. 2.4.3
- Bailey, R. A. (1996). Orthogonal partitions in designed experiments. *Designs, Codes and Cryptography*, 8(1):45–77. 5.10.2
- Battista, S., Casalino, F., and Lande, C. (1999). MPEG-4: a multimedia standard for the third millennium. 1. *IEEE Multimedia*, 6:74–83. 2.2.1
- Battista, S., Casalino, F., and Lande, C. (2000). MPEG-4: a multimedia standard for the third millennium. 2. *IEEE Multimedia*, 7(1):76–84. 2.2.1
- Benini, S., Bianchetti, A., Leonardi, R., and Migliorati, P. (2006). Extraction of significant video summaries by dendrogram analysis. In *Proc. of International Conference on Image Processing*, pages 133–136. 1.1, 5.2
- Bertino, E., Fan, J., Ferrari, E., Hacid, M.-S., Elmagarmid, A. K., and Zhu, X. (2003). A hierarchical access control model for video database systems. *ACM Transactions on Information Systems*, 21(2):155–191. 7.2
- Bescós, J. (2004). Real-time shot change detection over online MPEG-2 video. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(4):475–484. 5.5.2
- Bescós, J., Martínez, J. M., Herranz, L., and Tiburzi, F. (2007). Content-driven adaptation of on-line video. *Signal Processing: Image Communication*, 22:651–668. 2.3.3, 2.3.3.2, 3.7.1.3, 5.2
- Bondi, A. B. (2000). Characteristics of scalability and their impact on performance. In *Proc. of International Workshop on Software and Performance*, pages 195–203, New York, NY, USA. ACM. 2.1
- Bormans, J., Gelissen, J., and Perkis, A. (2003). MPEG-21: The 21st century multimedia framework. *IEEE Signal Processing Magazine*, 20(2):53–62. 2.4.1
- Brandenbrg, Karlheinz; Gill, B. (1994). First ideas on scalable audio coding. In *Audio Engineering Society Convention*. 2.1.3
- Brin, S. and Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. In *Proc. of International World-Wide Web Conference*. 2.1.2
- Böszörményi, L., Hellwagner, H., Kosch, H., Libsie, M., and Podlipnig, S. (2003). Meta-data driven adaptation in the ADMITS project. *Signal Processing: Image Communication*, 18(8):749 – 766. Special Issue on Multimedia Adaptation. 7.3.2.1
- Calic, J. and Campbell, N. W. (2007). Compact visualisation of video summaries. *EURASIP Journal on Advances in Signal Processing*, 2007(2):14 pages. 6.3, 6.4.1

- Calic, J., Gibson, D., and Campbell, N. (2007). Efficient layout of comic-like video summaries. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(7):931–936. (document), 2.3.2, 2.3.3.2, 6.1.1, 6.1.2, 6.2, 6.4.4, 6.5.1, 6.8.1, 8.1.1
- Cavallaro, A., Steiger, O., and Ebrahimi, T. (2005). Semantic video analysis for adaptive content delivery and automatic description. *IEEE Transactions on Circuits and Systems for Video Technology*, 15(10):1200–1209. 2.4.3
- CCITT (1992). Video codec for audiovisual services at p x 64 kb/s. 2.2.1
- Chang, H. S., Sull, S., and Lee, S. U. (1999). Efficient video indexing scheme for content-based retrieval. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(8):1269–1279. 2.3.3
- Chang, P., Han, M., and Gong, Y. (2002). Extract highlights from baseball game video with hidden markov models. In *Proc. of International Conference Image Processing*, volume 1. 2.3.3.1
- Chang, S.-F. and Vetro, A. (2005). Video adaptation: concepts, technologies, and open issues. *Proceedings of the IEEE*, 93(1):148–158. 1.1, 2.4.1, 2.4.3, 4.1
- Cheng, W.-H., Wang, C.-W., and Wu, J.-L. (2007). Video adaptation for small display based on content recomposition. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(1):43–58. 2.4.3
- Chiu, P., Girgensohn, A., and Liu, Q. (2004). Stained-glass visualization for highly condensed video summaries. In *Proc. of International Conference on Multimedia and Expo*, volume 3, pages 2059 – 2062. (document), 6.1.1, 6.1, 8.1.1
- Christopoulos, C., Skodras, A., and Ebrahimi, T. (2000). The JPEG2000 still image coding system: an overview. *IEEE Transactions on Consumer Electronics*, 46(4):1103–1127. 2.1.3
- Creusere, C. D. (2005). Understanding perceptual distortion in MPEG scalable audio coding. *IEEE Transactions on Speech and Audio Processing*, 13(3):422–431. 2.1.3
- Damjanovic, U., Piatrik, T., Djordjevic, D., and Izquierdo, E. (2007). Video summarisation for surveillance and news domain. In *Semantic Multimedia*, volume 4816 of *Lecture Notes in Computer Science*, pages 99–112. Springer-Verlag Berlin. 2.3.3
- De Bruyne, S., De Neve, W., De Wolf, K., De Schrijver, D., Verhoeve, P., and Van de Walle, R. (2006). Temporal video segmentation on H.264/AVC compressed bitstreams. In *Advances in Multimedia Modeling*, volume 4351 of *Lecture Notes in Computer Science*, pages 1–12. Springer-Verlag Berlin. 5.5.2
- De Bruyne, S., De Schrijver, D., De Neve, W., Van Deursen, D., and Van de Walle, R. (2007). Enhanced shot-based video adaptation using MPEG-21 generic bitstream syntax schema. In *Proc. of Symposium on Computational Intelligence in Image and Signal Processing*, pages 380–385. 3.1

- De Cock, J., Notebaert, S., Lambert, P., and Van de Walle, R. (2008). Advanced bitstream rewriting from h.264/avc to svc. In *Proc. of International Conference on Image Processing*, pages 2472–2475. 7.3.3.4
- De Cock, J., Notebaert, S., and Van de Walle, R. (2007). A novel hybrid requantization transcoding scheme for H.264/AVC. In *Proc. of International Symposium on Signal Processing and Its Applications*, pages 1–4. 3.6.1, 3.7.4.1, 3.7.4.1
- De Santo, M., Foggia, P., Sansone, C., Percannella, G., and Vento, M. (2006). An unsupervised algorithm for anchor shot detection. In *Proc. of International Conference on Pattern Recognition*, volume 2, pages 1238–1241. 9.2.2
- De Schrijver, D., De Neve, W., De Wolf, K., Notebaert, S., and Van de Walle, R. (2006). XML-based customization along the scalability axes of H.264/AVC scalable video coding. In *Proc. of International Symposium on Circuits and Systems*, page 4 pp. 3.1
- De Schrijver, D., De Neve, W., Van Deursen, D., Dhondt, Y., and Van de Walle, R. (2007). XML-based exploitation of region of interest scalability in scalable video coding. In *Proc. of International Workshop on Image Analysis for Multimedia Interactive Services*, pages 56–56. 2.4.3
- Delis, A. and Roussopoulos, N. (1992). Performance and scalability of client-server database architectures. In *Proc. of International Conference on Very Large Data Bases*, pages 610–623, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc. 2.1.2
- Devillers, S., Timmerer, C., Heuer, J., and Hellwagner, H. (2005). Bitstream syntax description-based adaptation in streaming and constrained environments. *IEEE Transactions on Multimedia*, 7(3):463–470. 3.1, 4.1
- Divakaran, A., Divakaran, A., Radhakrishnan, R., and Peker, K. (2002). Motion activity-based extraction of key-frames from video shots. In Radhakrishnan, R., editor, *Proc. of International Conference on Image Processing*, volume 1, pages 932–935 vol.1. 5.4.5
- Dumont, E. and Merialdo, B. (2007). Split-screen dynamically accelerated video summaries. In *Proc. of International Workshop on TRECVID video summarization*, pages 55–59, New York, NY, USA. ACM. 2.3.3, 9.2.1
- Dumont, E. and Merialdo, B. (2010). Rushes video summarization and evaluation. *Multimedia Tools and Applications*, 48:51–68. 10.1007/s11042-009-0374-9. 2.3.3.1
- Ekin, A., Tekalp, A., and Mehrotra, R. (2003). Automatic soccer video analysis and summarization. *IEEE Transactions on Image Processing*, 12(7):796–807. 2.3.3, 2.3.3.2
- Felzenszwalb, P. and Huttenlocher, D. (2004). Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181. 6.3, 6.3

- 
- Ferman, A. M. and Tekalp, A. M. (1998). Efficient filtering and clustering methods for temporal video segmentation and visual summarization. *Journal of Visual Communication and Image Representation*, 9(4):336–351. 5.9.1
- Filippone, M., Camastra, F., Masulli, F., and Rovetta, S. (2008). A survey of kernel and spectral methods for clustering. *Pattern Recognition*, 41(1):176–190. 5.6.1.1
- Fonseca, P. M. and Pereira, F. (2004). Automatic video summarization based on MPEG-7 descriptions. *Signal Processing: Image Communication*, 19(8):685–699. 2.3.3, 2.3.3.2
- Fu, Y., Guo, Y., Zhu, Y., Liu, F., Song, C.-M., and Zhou, Z.-H. (2010). Multi-view video summarization. *IEEE Transactions on Multimedia*, (99):1. Early Access. 2.3.3.2
- Gang, Z., Chia, L.-T., and Zongkai, Y. (2004). MPEG-21 digital item adaptation by applying perceived motion energy to H.264 video. In *Proc. of International Conference on Image Processing*, volume 4, pages 2777–2780. 3.1
- Garcia, A., Molina, J., Lopez, F., Valdes, V., Tiburzi, F., Martinez, J. M., and Bescos, J. (2009). Instant customized summaries streaming: a service for immediate awareness of new video content. In *Proc. of Workshop on Adaptive Multimedia Retrieval*. 9.2.1, 9.2.2.1
- Girgensohn, A. (2003). A fast layout algorithm for visual video summaries. In *Proc. of International Conference on Multimedia and Expo*, volume 2, pages II–77–80 vol.2. 6.1.1, 6.1.2
- Gong, Y. and Liu, X. (2000). Video summarization using singular value decomposition. In *Proc. of Conference on Computer Vision and Pattern Recognition*, volume 2, pages 174–180. 2.3.3
- Hanjalic, A. and Zhang, H. (1999). An integrated scheme for automated video abstraction based on unsupervised cluster-validity analysis. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(8):1280–1289. 2.3.3
- Hansen, H., Strahl, S., and Mertins, A. (2009). Fine-grain scalable audio coding based on envelope restoration and the spiht algorithm. In *Proc. of International Digital Signal Processing Conference*, pages 1–5. 2.1.3
- Hasebe, S., Sami, M. M., Muramatsu, S., and Kikuchi, H. (2005). Constructing storyboards based on hierarchical clustering analysis. In Li, S., Pereira, F., Shum, H.-Y., and Tescher, A. G., editors, *Visual Communications and Image Processing*, volume 5960, page 59601B. SPIE. 5.2
- Herranz, L. (2007). Integrating semantic analysis and scalable video coding for efficient content-based adaptation. *Multimedia Systems*, 13(2):103–118. 3.1, 3.7.1.3, 4
- Herranz, L. and Martínez, J. M. (2008a). Generation of scalable summaries based on iterative GOP ranking. In *Proc. of International Conference on Image Processing*, pages 2544–2547. 5

- Herranz, L. and Martínez, J. M. (2008b). Integrated summarization and adaptation using H.264/MPEG-4 SVC. In *Proc. of International Conference on Visual Information Engineering*, pages 729–734. 4
- Herranz, L. and Martínez, J. M. (2009a). An efficient summarization algorithm based on clustering and bitstream extraction. In *Proc. of International Conference on Multimedia and Expo*, pages 654–657. 5
- Herranz, L. and Martínez, J. M. (2009b). An integrated approach to summarization and adaptation using H.264/MPEG-4 SVC. *Signal Processing: Image Communication*, 24(6):499–509. Scalable Coded Media beyond Compression. 4
- Herranz, L. and Martínez, J. M. (2009c). On the use of hierarchical prediction structures for efficient summary generation of H.264/AVC bitstreams. *Signal Processing: Image Communication*, 24(8):615 – 629. 3, 3.7
- Herranz, L. and Martínez, J. M. (2010a). A framework for scalable summarization of video. *IEEE Transactions on Circuits and Systems for Video Technology*, 20(9):1265–1270. 5
- Herranz, L. and Martínez, J. M. (2010b). On the advantages of the use of bitstream extraction for video summary generation. In *Advances in Multimedia Modeling*, volume 5916 of *Lecture Notes in Computer Science*, pages 755–760. Springer-Verlag Berlin. 3, 3.7
- Herre, J. and Dietz, M. (2008). MPEG-4 high-efficiency AAC coding [standards in a nutshell]. *IEEE Signal Processing Magazine*, 25(3):137 –142. 7.1.1
- Hill, M. D. (1990). What is scalability? *ACM SIGARCH Computer Architecture News*, 18(4):18–21. 2.1
- Hoare, C. A. R. (1961). Algorithm 64: Quicksort. *Communications of the ACM*, 4(7):321. 2.1.1
- Homayounfar, K. (2003). Rate adaptive speech coding for universal multimedia access. *IEEE Multimedia*, 20(2):30–39. 2.1.3
- Hsiang, S. T. and Woods, J. W. (2001). Embedded video coding using invertible motion compensated 3-D subband/wavelet filter bank. *Signal Processing: Image Communication*, 16(8):705–724. 2.1.3
- Huang, M., Mahajan, A. B., and Dementhon, D. F. (2004). Automatic performance evaluation for video summarization. Technical report, University of Maryland. 2.3.3.1
- Huynh, D. F., Drucker, S. M., Baudisch, P., and Wong, C. (2005). Time quilt: scaling up zoomable photo browsers for large, unstructured photo collections. In *CHI '05 extended abstracts on Human factors in computing systems*, pages 1937–1940, New York, NY, USA. ACM. 6.1.1
- ISO/IEC (2004). Jpeg 2000 image coding system – part 1: Core coding system. 2.1.3

- Itti, L. (2004). Automatic foveation for video compression using a neurobiological model of visual attention. *IEEE Transactions on Image Processing*, 13(10):1304–1318. 2.4.3
- ITU-T (2000). Video coding for low bit rate communication, v3. 2.1.3
- ITU-T and ISO/IEC (1992). Coding of moving pictures and associated audio for digital storage media at up to about 1,5 mbit/s - part3: Audio. 2.2.1, 7.1.1
- ITU-T and ISO/IEC (1994). Generic coding of moving pictures and associated audio information - part 2: Video. 2.1.3, 2.2.1
- ITU-T and ISO/IEC (1998). Description of MPEG-7 content set. ISO/IEC JTC1/SC29/WG11 N2467. 5.10.2, 8.2.1
- ITU-T and ISO/IEC (1999). Coding of audio-visual objects - part 2: Visual. 2.1.3, 2.2.1
- ITU-T and ISO/IEC (2001a). ISO/IEC 15938-5, Information Technology - Multimedia Content Description Interface - Part 5: Multimedia Description Schemes. 7.3.2.1
- ITU-T and ISO/IEC (2001b). ISO/IEC 21000-1, Information Technology - Multimedia Framework (MPEG-21) - Part 1: Vision, Technologies and Strategy. 2.4.1
- ITU-T and ISO/IEC (2003a). Advanced video coding for generic audiovisual services. 2.2.2, 3.6.2.2
- ITU-T and ISO/IEC (2003b). ISO/IEC 21000-7, Information Technology - Multimedia Framework (MPEG-21) - Part 7: Digital Item Adaptation. 2.4.1
- ITU-T and ISO/IEC (2005). Coding of audio-visual objects - part 3: Audio. 7.1.1
- ITU-T and ISO/IEC (2007). Generic coding of moving pictures and associated audio information - part 7: Advanced audio coding. 7.1.1
- Jaimes, A., Echigo, T., Teraguchi, M., and Satoh, F. (2002). Learning personalized video highlights from detailed mpeg-7 metadata. In *Proc. of International Conference on Image Processing*, volume 1, pages I-133–I-136 vol.1. 7.1
- Jain, A. K., Murty, M. N., and Flynn, P. J. (1999). Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323. 2.1.1, 5.6, 5.6.1.2
- Jansen, M., Heeren, W., and van Dijk, B. (2008). Videotrees: Improving video surrogate presentation using hierarchy. In *Proc. of International Workshop on Content-Based Multimedia Indexing*, pages 560–567. (document), 2.3.3.1, 2.3.3.2, 6.1.1, 6.1
- Jeannin, S. and Divakaran, A. (2001). MPEG-7 visual motion descriptors. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(6):720–724. 3.7.1.3
- Jiang, X., Safaei, F., and Boustead, P. (2005). Latency and scalability: a survey of issues and techniques for supporting networked games. In *Proc. of International Conference on Communication and International Conference on Networks*, volume 1. 2.1.2

- Kandadai, S. and Creusere, C. D. (2008). Scalable audio compression at low bitrates. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(5):969–979. 2.1.3
- Kang, H.-B. (2002). Video abstraction techniques for a digital library. In *Distributed multimedia databases: techniques & applications*, pages 120–132. IGI Publishing. 2.3.3
- Kasutani, E. and Yamada, A. (2001). The MPEG-7 color layout descriptor: a compact image feature description for high-speed image/video segment retrieval. In Yamada, A., editor, *Proc. of International Conference on Image Processing*, volume 1, pages 674–677 vol.1. 5.5.3, 5.10
- Kelly, D. P., van Gestel, W., Hamada, T., Kato, M., and Nakamura, K. (2003). Blu-ray disc - a versatile format for recording high definition video. In *Proc. of Int. Conference on Consumer Electronics*, pages 72–73. 2.2.2
- Kozuka, M. (2004). An overview of Blu-ray disc application format for HD-movie distribution. In Kumar, B. V. K. V. and Kobori, H., editors, *Optical Data Storage*, volume 5380, pages 1–9. SPIE. 2.2.2
- Kraaij, W., Over, P., Ianeva, T., and Smeaton, A. (2006). TRECVID 2006 - an overview. In *Proceedings of TRECVID 2006*. NIST, USA. 6.8.1
- Lee, S. and Bovik, A. C. (1999). Motion estimation and compensation for foveated video. In *Proc. of International Conference on Image Processing*, volume 2, pages 615–619. 2.4.3
- Lee, S. and Bovik, A. C. (2000). Foveated video image analysis and compression gain measurements. In *Proc. of Southwest Symposium on Image Analysis and Interpretation*, pages 63–67. 2.4.3
- Lefol, D., Bull, D., and Canagarajah, N. (2006). Performance evaluation of transcoding algorithms for H.264. *IEEE Transactions on Consumer Electronics*, 52(1):215–222. 2.4.2, 3.7.4.1
- Lefol, D., Bull, D., Canagarajah, N., and Redmill, D. (2007). An efficient complexity-scalable video transcoder with mode refinement. *Signal Processing: Image Communication*, 22(4):421 – 433. 3.6.1, 3.7.4.1, 3.7.4.1
- Li, B. and Ibrahim Sezan, M. (2001). Event detection and summarization in sports video. In *Proc. of Workshop Content-Based Access of Image and Video Libraries*, pages 132–138. 2.3.3
- Li, W. (2001). Overview of fine granularity scalability in mpeg-4 video standard. *IEEE Transactions On Circuits And Systems For Video Technology*, 11(3):301 –317. 2.1.3, 2.2.1
- Li, Y., Lee, S.-H., Yeh, C.-H., and Kuo, C.-C. (2006). Techniques for movie content analysis and skimming: tutorial and overview on video abstraction techniques. *IEEE Signal Processing Magazine*, 23(2):79–89. 2.3.3
- Li, Z., Schuster, G., Katsaggelos, A., and Gandhi, B. (2005). Rate-distortion optimal video summary generation. *IEEE Transactions on Image Processing*, 14(10):1550–1560. 2.3.3

- 
- Libsle, M. and Kosch, H. (2004). Video adaptation using the variation factory. In *Proc. of Workshop on Multimedia Signal Processing*, pages 403–406. 7.3.2.1
- Lie, W.-N. and Lai, C.-M. (2005). News video summarization based on spatial and motion feature analysis. In *Advances in Multimedia Information Processing*, volume 3332 of *Lecture Notes in Computer Science*, pages 246–255. Springer-Verlag Berlin. 2.3.3, 9.2.1, 9.2.2
- Likert, R. (1932). A technique for the measurement of attitudes. *Archives of Psychology*, 22(140):1–55. 5.10.2, 6.8.3.1, 9.2.4
- Liu, T., Zhang, X., Feng, J., and Lo, K.-T. (2004). Shot reconstruction degree: a novel criterion for key frame selection. *Pattern Recognition Letters*, 25(12):1451 – 1457. 2.3.3.1
- Liu, W., Yang, G., and huang, X. (2009). Semantic features based news stories segmentation for news retrieval. In *Proc. of International Conference on Wavelet Analysis and Pattern Recognition*, pages 258 –265. 9.2.2
- Lotfallah, O. A., Reisslein, M., and Panchanathan, S. (2006). Adaptive video transmission schemes using MPEG-7 motion intensity descriptor. *IEEE Transactions on Circuits and Systems for Video Technology*, 16(8):929–946. 3.7.1.3
- Ma, Y.-F., Hua, X.-S., Lu, L., and Zhang, H.-J. (2005). A generic framework of user attention model and its application in video summarization. *IEEE Transactions on Multimedia*, 7(5):907–919. 2.3.3, 5.4.5
- Ma, Y.-F. and Zhang, H.-J. (2005). Video snapshot: A bird view of video sequence. In *Proc. of International on Multimedia Modelling Conference*, pages 94 – 101. (document), 6.1.1, 6.1
- MacQueen, J. B. (1967). Some methods of classification and analysis of multivariate observations. In *Proc. of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297. 2.1.1, 5.6.1.1
- Magalhaes, J. and Pereira, F. (2004). Using MPEG standards for multimedia customization. *Signal Processing: Image Communication*, 19(5):437–456. 2.4.3
- Marchionini, G., Wildemuth, B. M., and Geisler, G. (2006). The Open Video digital library: A Möbius strip of research and practice. *Journal of the American Society for Information Science and Technology*, 57(12):1629–1643. 2.3, 3.7.1
- Maybury, M., Greiff, W., Boykin, S., Ponte, J., McHenry, C., and Ferro, L. (2004). Personalcasting: Tailored broadcast news. *User Modeling and User-Adapted Interaction*, 14(1):119–144. 2.3.3, 2.4.3, 7.1
- McCloud, S. (1994). *Understanding Comics: The Invisible Art*. HarperCollins. 6.1.2, 6.4.1
- Meessen, J., Xu, L.-Q., and Macq, B. (2006). Content browsing and semantic context viewing through JPEG2000-based scalable video summary. *IEE Proceedings - Vision, Image and Signal Processing*, 153(3):274–283. 2.3.3.2

## BIBLIOGRAPHY

---

- Mei, T., Yang, B., Yang, S.-Q., and Hua, X.-S. (2009). Video collage: presenting a video sequence using a single image. *The Visual Computer*, 25(1):39–51. (document), 2.3.2, 2.3.3.2, 6.1.1, 6.1, 8.1.1
- Mietens, S., de With, P. H. N., and Hentschel, C. (2004). New complexity scalable mpeg encoding techniques for mobile applications. *EURASIP Journal on Advances in Signal Processing*, 2004:236–252. 2.1.3
- Milenova, B. L. and Campos, M. M. (2002). O-cluster: scalable clustering of large high dimensional data sets. In *Proc. of Int. Conference on Data Mining*, pages 290–297. 2.1.1
- Milliner, S., Bouguettaya, A., and Papazoglou, M. P. (1995). A scalable architecture for autonomous heterogeneous database interactions. In *Proc. of International Conference on Very Large Data Bases*, pages 515–526, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc. 2.1.2
- Money, A. G. and Agius, H. (2008a). Feasibility of personalized affective video summaries. In *Affect and Emotion in Human-Computer Interaction: From Theory to Applications*, volume 4868 of *Lecture Notes in Computer Science*, pages 194–208. Springer-Verlag, Berlin, Heidelberg. 7.1
- Money, A. G. and Agius, H. (2008b). Video summarisation: A conceptual framework and survey of the state of the art. *Journal of Visual Communication and Image Representation*, 19(2):121–143. 2.3.3, 5.2, 6.1, 10.4, A.3
- Mrak, M., Calic, J., and Kondo, A. (2009). Fast analysis of scalable video for adaptive browsing interfaces. *Computer Vision and Image Understanding*, 113(3):425–434. Special Issue on Video Analysis. 3.7.1.3
- Mundur, P., Rao, Y., and Yesha, Y. (2006). Keyframe-based video summarization using delaunay clustering. *International Journal of Digital Libraries*, 6(2):219–232. 2.3.3, 3.7.1.1
- Musmann, H. (2006). Genesis of the MP3 audio coding standard. *IEEE Transactions on Consumer Electronics*, 52(3):1043–1049. 2.2.1, 7.1.1
- Nakajima, Y., Ujihara, K., and Yoneyama, A. (1999). Shot change detection from partially decoded MPEG data. *Systems and Computers in Japan*, 30(8):11–22. 5.5.2, 5.5.2, 5.5.2
- Ng, A. Y., Jordan, M. I., and Weiss, Y. (2001). On spectral clustering: Analysis and an algorithm. In Dietterich, T. G., Becker, S., and Ghahramani, Z., editors, *Advances in Neural Information Processing Systems*, volume 14. 5.6.1.1
- Ngo, C.-W., Ma, Y.-F., and Zhang, H.-J. (2003). Automatic video summarization by graph modeling. In *Proc. of International Conference on Computer Vision*, volume 1, pages 104–109. 2.3.3.1, 5.10.2

- Noll, P. (1997). MPEG digital audio coding. *IEEE Signal Processing Magazine*, 14(5):59–81. 7.1.1
- Odobez, J.-M., Gatica-pérez, D., and Guillemot, M. (2003). Video shot clustering using spectral methods. In *Proc. of International Workshop on Content-Based Multimedia Indexing*, pages 94–102. 5.6.1.1
- Ohm, J.-R. (1994). Three-dimensional subband coding with motion compensation. *IEEE Transactions on Image Processing*, 3(5):559–571. 1.1, 2.1.3
- Ohm, J.-R. (2005). Advances in scalable video coding. *Proceedings of the IEEE*, 93(1):42–56. 1.1, 2.1.3
- Ohm, J. R., van der Schaar, M., and Woods, J. W. (2004). Interframe wavelet coding motion picture representation for universal scalability. *Signal Processing: Image Communication*, 19(9):877–908. 2.1.3
- Over, P., Smeaton, A. F., and Awad, G. (2008). The TRECVID 2008 BBC rushes summarization evaluation. In *Proc. of TRECVID Video Summarization Workshop*, pages 1–20, New York, NY, USA. ACM. 2.3.3, 5.3.4, 5.9.1, 5.10.2, 8.2.1
- Over, P., Smeaton, A. F., and Kelly, P. (2007). The TRECVID 2007 BBC rushes summarization evaluation pilot. In *Proc. of international Workshop on TRECVID video summarization*, pages 1–15, New York, NY, USA. ACM. 2.3.3, 5.9.1, 5.10.1, 5.10.2, 8.2.1
- Pan, D. (1995). A tutorial on MPEG/audio compression. *IEEE Multimedia*, 2(2):60–74. 7.1.1
- Panis, G., Hutter, A., Heuer, J., Hellwagner, H., Kosch, H., Timmerer, C., Devillers, S., and Amielh, M. (2003). Bitstream syntax description: a tool for multimedia resource adaptation within MPEG-21. *Signal Processing: Image Communication*, 18(8):721–747. 4.1
- Paridaens, T., De Schrijver, D., De Neve, W., and Van de Walle, R. (2007). XML-driven bitrate adaptation of svc bitstreams. In *Proc. of International Workshop on Image Analysis for Multimedia Interactive Services*, pages 49–49. 4.1
- Peña, J. M., Lozano, J. A., and Larrañaga, P. (1999). An empirical comparison of four initialization methods for the k-means algorithm. *Pattern Recognition Letters*, 20(10):1027–1040. 5.6.1.1
- Peker, K., Divakaran, A., and Sun, H. (2001). Constant pace skimming and temporal sub-sampling of video using motion activity. In *Proc. of International Conference on Image Processing*, volume 3, pages 414–417. 2.3.3, 3.7.1.3
- Peker, K., Otsuka, I., and Divakaran, A. (2006). Broadcast video program summarization using face tracks. In *Proc. of International Conference on Multimedia and Expo*, pages 1053–1056. 2.3.3, 3.7.1.2, 5.4.5

- Peng, W.-T., Huang, W.-J., Chu, W.-T., Chou, C.-N., Chang, W.-Y., Chang, C.-H., and Hung, Y.-P. (2008). A user experience model for home video summarization. In *Semantic Multimedia*, volume 5371 of *Lecture Notes in Computer Science*, pages 484–495. Springer-Verlag Berlin. 2.3.3
- Pfeiffer, S., Lienhart, R., Fischer, S., and Effelsberg, W. (1996). Abstracting digital movies automatically. *Journal of Visual Communication and Image Representation*, 7(4):345–353. 1.1, 2.3.3, 8.1.1
- Pritch, Y., Rav-Acha, A., and Peleg, S. (2008). Nonchronological video synopsis and indexing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(11):1971–1984. 6.1.1
- Ren, J. and Jiang, J. (2009). Hierarchical modeling and adaptive clustering for real-time summarization of rush videos. *IEEE Transactions on Multimedia*, 11(5):906–917. 2.3.3
- Ren, K. and Calic, J. (2009). Freeeye: interactive intuitive interface for large-scale image browsing. In *Proc. of International Conference on Multimedia*, pages 757–760, New York, NY, USA. ACM. 6.1.1
- Ren, K., Sarvas, R., and Calic, J. (2010). Interactive search and browsing interface for large-scale visual repositories. *Multimedia Tools and Applications*, Online:16 pages. (document), 6.1
- Ren, T., Liu, Y., and Wu, G. (2008). Full-reference quality assessment for video summary. In *Proc. of International Conference on Data Mining Workshops*, pages 874–883. 2.3.3.1
- Santini, S. (2007). Who needs video summarization anyway? In *Proc. of International Conference on Semantic Computing*, pages 177–184. 2.3.3.1
- Schafer, R. (1998). Mpeg-4: a multimedia compression standard for interactive applications and services. *Electronics & Communication Engineering Journal*, 10(6):253–262. 2.2.1
- Schierl, T., Stockhammer, T., and Wiegand, T. (2007). Mobile video transmission using scalable video coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(9):1204–1217. 2.2.3
- Schippers, J., Remke, A., Punt, H., Wegdam, M., and Haverkort, B. (2010). A massively scalable architecture for instant messaging & presence. *Electronic Notes in Theoretical Computer Science*, 261:109 – 130. Proceedings of the Fourth International Workshop on the Practical Application of Stochastic Modelling (PASM 2009). 2.1.2
- Schwarz, H., Marpe, D., and Wiegand, T. (2006). Analysis of hierarchical B pictures and MCTF. In *Proc. of International Conference on Multimedia and Expo*, pages 1929–1932. 3.2
- Schwarz, H., Marpe, D., and Wiegand, T. (2007). Overview of the scalable video coding extension of the H.264/AVC standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(9):1103–1120. 1.1, 2.1.3, 2.1.3, 2.2.3, 3.4.2

- Schwarz, H. and Wien, M. (2008). The scalable video coding extension of the H.264/AVC standard. *IEEE Signal Processing Magazine*, 25(2):135–141. 2.2.3
- Sedgewick, R. (1978). Implementing quicksort programs. *Commun. ACM*, 21(10):847–857. 2.1.1
- Segall, A. and Zhao, J. (2008). Bit stream rewriting for SVC-to-AVC conversion. In *Proc. of International Conference on Image Processing*, pages 2776–2779. 7.3.3.4
- Shneiderman, B., Bederson, B. B., and Drucker, S. M. (2006). Find that photo!: interface strategies to annotate, browse, and share. *Communications of the ACM*, 49(4):69–71. 7.2
- Sikora, T. (1997). MPEG digital audio-and video-coding standards. *IEEE Signal Processing Magazine*, 14(5):58. 2.2.1
- Smith, M. and Kanade, T. (1998). Video skimming and characterization through the combination of image and language understanding. In *Proc. of International Workshop on Content-Based Access of Image and Video Database*, pages 61–70. 2.3.3
- Smolic, A., Mueller, K., Stefanoski, N., Ostermann, J., Gotchev, A., Akar, G. B., Triantafyllidis, G., and Koz, A. (2007). Coding algorithms for 3dtv—a survey. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(11):1606–1621. 2.2.2
- Sprihan, N., Mrak, M., Abhayaratne, G., and Izquierdo, E. (2005). A scalable coding framework for efficient video adaptation. In *Proc. of International Workshop on Image Analysis for Multimedia Interactive Services*. 4.1
- Sullivan, G. J. and Wiegand, T. (2005). Video compression - from concepts to the H.264/AVC standard. *Proceedings of the IEEE*, 93(1):18–31. 2.2.2
- Tan, Y., Lee, W., Tham, J., Rahardja, S., and Lye, K. (2010). Complexity scalable h.264/avc encoding. *IEEE Transactions on Circuits and Systems for Video Technology*, (99):1. Early Access. 2.1.3
- Thang, T. C., Kim, Y. S., Ro, Y. M., Kang, J., and Kim, J.-G. (2006). SVC bitstream adaptation in MPEG-21 multimedia framework. *Journal of Zhejiang University - Science A*, 7(5):764–772. 4.1
- Theodoridis, S. and Koutroumbas, K. (2006). *Pattern Recognition*. Academic Press. 2.1.1, 5.6, 5.6.1.2, 5.6.2
- Tjondronegoro, D., Chen, Y.-P. P., and Pham, B. (2004). Highlights for more complete sports video summarization. *IEEE Multimedia*, 11(4):22–37. 2.3.3
- Tourapis, A. M., Leontaris, A., Sühring, K., and Sullivan, G. (2007). H.264/MPEG-4 AVC reference software manual. Jvt-x072, Int. Telecommun. Union-Telecommun. (ITU-T) and Int. Standards Org./Int. Electrotech. Comm. (ISO/IEC). 3.6.2.2, 3.7.1

- Truong, B. T. and Venkatesh, S. (2007). Video abstraction: A systematic review and classification. *ACM Transactions on Multimedia Computing, Communications and Applications*, 3(1):3. 2.3.3, 2.3.3.1, 3.3, 5.2, 5.4.5, 5.7.1, 6.1, 6.2.2, 7.2, 8.2.1, 9.2.3, 10.4, A.3
- Tseng, B., Lin, C.-Y., and Smith, J. (2004). Using MPEG-7 and MPEG-21 for personalizing video. *IEEE Multimedia*, 11(1):42–52. 2.3.3.2, 2.4.1, 2.4.3, 7.1
- Uchihashi, S., Foote, J., Girgensohn, A., and Boreczky, J. (1999). Video manga: generating semantically meaningful video summaries. In *Proc. of the international Conference on Multimedia*, pages 383–392, New York, NY, USA. ACM. 2.3.3.2, 6.1.1, 6.1.2
- Valdés, V. (2010). *On-Line Video Abstraction*. PhD thesis, Universidad Autónoma de Madrid. 2.3.3.1, 2.3.3.2, 9.2.1, 9.2.2.1
- Valdés, V. and Martínez, J. M. (2008). Binary tree based on-line video summarization. In *Proc. of TRECVID Video Summarization Workshop*, pages 134–138, New York, NY, USA. ACM. 2.3.3, 5.9.1
- Valdés, V. and Martínez, J. M. (2010). A framework for video abstraction systems analysis and modelling from an operational point of view. *Multimedia Tools and Applications*, Online first:–. 5.9.1
- van Beek, P., Smith, J., Ebrahimi, T., Suzuki, T., and Askelof, J. (2003). Metadata-driven multimedia access. *IEEE Signal Processing Magazine*, 20(2):40–52. 2.4.3, 7.3.2.1
- Vetro, A. (2004). MPEG-21 digital item adaptation: Enabling universal multimedia access. *IEEE Multimedia*, 11(1):84–87. 1.1, 2.4.1
- Vetro, A., Christopoulos, C., and Ebrahimi, T. (2003a). From the guest editors - universal multimedia access. *IEEE Signal Processing Magazine*, 20(2):16. 2.4.1
- Vetro, A., Christopoulos, C., and Sun, H. (2003b). Video transcoding architectures and techniques: an overview. *IEEE Signal Processing Magazine*, 20(2):18–29. 2.4.2
- Vetro, A. and Timmerer, C. (2005). Digital item adaptation: overview of standardization and research activities. *IEEE Transactions on Multimedia*, 7(3):418–426. 2.4.1
- Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 511–18. IEEE Comput. Soc IEEE Comput. Soc. Tech. Committee on Pattern Analysis & Machine Intelligence. 3.7.1.2
- Wang, F. and Merialdo, B. (2009). Multi-document video summarization. pages 1326–1329. 2.3.3.2
- Ward, J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58(301):236–244. 2.1.1, 5.6.1.2

- Wiegand, T., Sullivan, G., Reichel, J., Schwarz, H., and Wien, M. (2007). Joint draft ITU-T Rec. H.264 | ISO/IEC 14496-10 / Amd.3 Scalable video coding. Doc. JVT-X201. 2.1.3, 2.2.3
- Wiegand, T. and Sullivan, G. J. (2007). The H.264/AVC video coding standard. *IEEE Signal Processing Magazine*, 24(2):148–153. 2.2.2
- Wiegand, T., Sullivan, G. J., Bjontegaard, G., and Luthra, A. (2003). Overview of the H.264/AVC video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):560–576. 2.1.3, 2.2.2
- Wien, M., Cazoulat, R., Graffunder, A., Hutter, A., and Amon, P. (2007). Real-time system for adaptive video streaming based on SVC. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(9):1227–1237. 2.2.3
- Wildemuth, B., Marchionini, G., Yang, M., Geisler, G., Wilkens, T., Hughes, A., and Gruss, R. (2003). How fast is too fast? evaluating fast forward surrogates for digital video. In *Proc. of Joint Conference on Digital Libraries*, pages 221–230. 2.3.2, 7.2
- Xin, J., Lin, C.-W., and Sun, M.-T. (2005). Digital video transcoding. *Proceedings of the IEEE*, 93(1):84–97. 1.1, 2.4.2, 3.6.1, 3.7.4.1, 3.7.4.1
- Yang, Z., Cai, H., and Li, J. (2005). A framework for fine-granular computational-complexity scalable motion estimation [real-time video coding applications]. In *Proc. of International Symposium Circuits and Systems*, pages 5473–5476. 2.1.3
- Yeo, B.-L. and Liu, B. (1995). On the extraction of dc sequence from mpeg compressed video. In *Proc. of International Conference on Image Processing*, volume 2, pages 260–263 vol.2. 5.5.2
- Yeung, M. and Yeo, B.-L. (1997). Video visualization for compact presentation and fast browsing of pictorial content. *IEEE Transactions on Circuits and Systems for Video Technology*, 7(5):771–785. (document), 1.1, 2.3.3.2, 6.1.1, 6.1, 6.1.2, 8.1.1
- Zeng, W. and Gao, W. (2005). Shot change detection on H.264/AVC compressed video. In *Proc. of International Symposium on Circuits and Systems*, pages 3459–3462 Vol. 4. 5.5.2
- Zhang, H. J., Wu, J., Zhong, D., and Smoliar, S. W. (1997a). An integrated system for content-based video retrieval and browsing. *Pattern Recognition*, 30(4):643 – 658. Image Databases. 2.3.3
- Zhang, T., Ramakrishnan, R., and Livny, M. (1997b). Birch: A new data clustering algorithm and its applications. *Data Mining and Knowledge Discovery*, 1(2):141–182. 2.1.1
- Zhao, Z., Jiang, S., Huang, Q., and Zhu, G. (2006). Highlight summarization in sports video based on replay detection. In *Proc. of International Conference on Multimedia and Expo*, pages 1613–1616. 2.3.3

- Zhu, C.-Z., Hua, X.-S., Mei, T., and Wu, X.-Q. (2005a). Video booklet: a natural video searching and browsing interface. In *Proc. of International Workshop on Multimedia Information Retrieval*, pages 113–120, New York, NY, USA. ACM. 2.3.3.2
- Zhu, X., Elmagarmid, A., Xue, X., Wu, L., and Catlin, A. (2005b). InsightVideo: toward hierarchical video content organization for efficient browsing, summarization and retrieval. *IEEE Transactions on Multimedia*, 7(4):648–666. 1.1
- Zhu, X., Fan, J., Elmagarmid, A. K., and Wu, X. (2003). Hierarchical video content description and summarization using unified semantic and visual similarity. *Multimedia Systems*, 9(1):31–53. 2.3.3, 2.3.3.2, 5.2
- Zhu, X. Q., Wu, X. D., Fan, J. P., Elmagarmid, A. K., and Aref, W. F. (2004). Exploring video content structure for hierarchical summarization. *Multimedia Systems*, 10(2):98–115. 1.1, 2.3.3.1, 5.1, 5.2, 5.10.2
- Zhu, Z. and Lin, T. (2010). Idempotent H.264 intraframe compression. *Multimedia Tools and Applications*, 46(1):25–45. 7.3.3.2
- Zhuang, Y., Rui, Y., Huang, T., and Mehrotra, S. (1998). Adaptive key frame extraction using unsupervised clustering. In *Proc. of International Conference on Image Processing*, volume 1, pages 866–870. 2.3.3, 3.7.1.1