Personalizing quality aspects for video communication in constrained heterogeneous environments

Personalisatie van kwaliteitsaspecten voor videocommunicatie in beperkte, heterogene omgevingen

Sam Lerouge

Promotor: prof. dr. ir. R. Van de Walle

Proefschrift ingediend tot het behalen van de graad van
Doctor in de Ingenieurswetenschappen: Computerwetenschappen

Vakgroep Elektronica en Informatiesystemen
Voorzitter: prof. dr. ir. J. Van Campenhout
Faculteit Ingenieurswetenschappen
Academiejaar 2005-2006

UNIVERSITEIT
GENT

Dankwoord

Met het indienen van dit doctoraat sluit ik een periode van ruim vier jaar af, waarin ik de kans gekregen heb om in een boeiend domein aan wetenschappelijk onderzoek te doen. Het is dan ook een uitgelezen kans om de mensen te bedanken zonder wie dit werk niet mogelijk zou zijn geweest.

In de eerste plaats wil ik mijn promotor, prof. Rik Van de Walle bedanken. Het is dankzij zijn onuitputtelijke inzet en zijn pionierswerk in de wereld van multimedia binnen de Universiteit Gent dat het Multimedia Lab kon ontstaan. Deze jonge onderzoeksgroep vormde voor mij het ideale kader voor het behalen van mijn doctoraat. Verder wil ik benadrukken dat ik het bijzonder apprecieer dat mijn promotor altijd tijd heeft willen vrijmaken voor mij, ook wanneer de omstandigheden dat erg moeilijk maakten.

Ook mijn collega's van het Multimedia Lab wil ik uitgebreid bedanken. Niet alleen zorgde hun kennis en hun opbouwende kritiek ervoor dat mijn eigen onderzoek steeds is kunnen blijven vooruitgaan, ook zorgden ze voor een aangename werkomgeving waar ik mij altijd thuis heb kunnen voelen. Een aantal onder hen hebben grote delen van dit boek tot in het kleinste detail doorgenomen. Zonder hun kritische blik zou dit werk heel wat onnauwkeurigheden, inconsistenties en onduidelijkheden bevatten die ik nu dankzij hen heb kunnen wegwerken. Hen wil ik dan ook in het bijzonder danken voor de geleverde inspanningen.

Graag had ik ook Rita Breems langs deze weg bedankt, niet enkel omdat zij me door haar werk van heel wat administratieve rompslomp heeft verlost, maar ook omwille van de vele aangename gesprekken in de loop van mijn onderzoek.

Peter Dossche verdient ook een woordje van dank, omdat hij een aanzienlijk deel van dit boek heeft doorgenomen om me te wijzen op mijn grootste taalfouten. Doordat ik ervoor gekozen heb om dit werk niet in mijn moedertaal neer te pennen, was zijn bijdrage van groot belang voor de kwaliteit ervan. Verder wil ik iedereen danken die heeft meegewerkt aan het psychovisueel experiment dat in dit boek beschreven wordt. Hun medewerking was

noodzakelijk om tot de resultaten te komen die de kroon op het werk van mijn onderzoek vormen.

Dit dankwoord is ook een uitgelezen kans om mijn ouders te bedanken omdat ze me alle mogelijke kansen gegeven hebben om te groeien tot wat ik nu ben. Ze hebben me de kans gegeven om mijn studies aan te vatten en zijn me altijd blijven steunen, op alle mogelijke manieren. Ook mijn broer wil ik bedanken, net als mijn schoonfamilie, omdat ze altijd achter mij zijn blijven staan en mij alle nodige hulp hebben willen aanreiken.

Tot slot wil ik graag mijn vrouw Lieve uitvoerig bedanken. We hebben samen al een aantal schitterende jaren achter de rug, en hebben er hopelijk nog veel te gaan. Lieve heeft me al die tijd maximaal gesteund, ook op de moeilijkere momenten. Zonder haar zou ik wellicht nooit geraakt zijn waar ik nu sta.

Sam Lerouge
24 november 2005

Summary

The world of multimedia communication is drastically evolving since a few years. Advanced compression formats for audiovisual information arise, new types of wired and wireless networks are developed, and a broad range of different types of devices capable of multimedia communication appear on the market. The era where multimedia applications available on the Internet were the exclusive domain of PC users has passed. The next generation multimedia applications will be characterized by heterogeneity: differences in terms of the networks, devices and user expectations.

This heterogeneity causes some new challenges: transparent consumption of multimedia content is needed in order to be able to reach a broad audience. Recently, two important technologies have appeared that can assist in realizing such transparent *Universal Multimedia Access*. The first technology consists of new scalable or layered content representation schemes. Such schemes are needed in order to make it possible that a multimedia stream can be consumed by devices with different capabilities and transmitted over network connections with different characteristics. The second technology does not focus on the content representation itself, but rather on linking information about the content, so-called metadata, to the content itself. One of the possible uses of metadata is in the automatic selection and adaptation of multimedia presentations. This is one of the main goals of the MPEG-21 Multimedia Framework.

Within the MPEG-21 standard, two formats were developed that can be used for bitstream descriptions. Such descriptions can act as an intermediate layer between a scalable bitstream and the adaptation process. This way, format-independent bitstream adaptation engines can be built. Furthermore, it is straightforward to add metadata information to the bitstream description, and use this information later on during the adaptation process. Because of the efforts spent on bitstream descriptions during our research, a lot of attention is devoted to this topic in this thesis. We describe both frameworks for bitstream descriptions that were standardized by MPEG. Furthermore, we focus on our own contributions in this domain: we developed a number of bitstream

schemas and transformation examples for different types of multimedia content.

The most important objective of this thesis is to describe a content negotiation process that uses scalable bitstreams in a generic way. In order to be able to express such an application, we felt the need for a better understanding of the data structures, in particular scalable bitstreams, on which this content negotiation process operates. Therefore, this thesis introduces a formal model we developed capable of describing the fundamental concepts of scalable bitstreams and their relations. Apart from the definition of the theoretical model itself, we demonstrate its correctness by applying it to a number of existing formats for scalable bitstreams. Furthermore, we attempt to formulate a content negotiation process as a constrained optimization problem, by means of the notations defined in the abstract model.

In certain scenarios, the representation of a content negotiation process as a constrained optimization problem does not sufficiently reflect reality, especially when scalable bitstreams with multiple quality dimensions are involved. In such case, several versions of the same original bitstream can meet all constraints imposed by the system. Sometimes one version clearly offers a better quality towards the end user than another one, but in some cases, it is not possible to objectively compare two versions without additional information. In such a situation, a trade-off will have to be made between the different quality aspects. We use Pareto's theory of multi-criteria optimization for formally describing the characteristics of a content negotiation process for scalable bitstreams with multiple quality dimensions. This way, we can modify our definition of a content negotiation process into a multi-criteria optimization problem.

One of the most important problems with multi-criteria optimization problems is that multiple candidate optimal solutions may exist. Additional information, e.g. user preferences, is needed if a single optimal solution has to be selected. Such multi-criteria optimization problems are not new. Unfortunately, existing solutions for selecting one optimal version are not suitable in a content negotiation scenario, because they expect detailed understanding of the problem from the decision maker, in our case the end user.

In this thesis, we propose a scenario in which a so-called content negotiation agent would give some sample video sequences to the end user, asking him to select which sequence he liked the most. This information would be used for training the agent: a model would be built representing the preferences of the end user, and this model can be used later on for selecting one solution from a set of candidate optimal solutions.

Based on a literature study, we propose two candidate algorithms in this

thesis that can be used in such a content negotiation agent. It is possible to use these algorithms for constructing a model of the user's preferences by means of a number of examples, and to use this model when selecting an optimal version. The first algorithm considers the quality of a video sequence as a weighted sum of a number of independent quality aspects, and derives a system of linear inequalities from the example decisions. The second algorithm, called 1ARC, is actually a nearest-neighbor approach, where predictions are made based on the similarity with the example decisions entered by the user.

This thesis analyzes the strengths and weaknesses of both algorithms from multiple points of view. The computational complexity of both algorithms is discussed, possible parameters that can influence the reliability of the algorithm, and the reliability itself. For measuring this kind of performance, we set up a test in which human subjects are asked to make a number of pairwise decisions between two versions of the same original video sequence. The reliability of the two algorithms we proposed is tested by selecting a part of these decisions for training a model, and by observing if this model is able to predict other decisions entered by the same user. We not only compare both algorithms, but we also observe the result of modifying several parameters on both algorithms. Ultimately, we conclude that the 1ARC algorithm has an acceptable performance, certainly if the training set is sufficiently large. The reliability is better than what would be theoretically achievable by any other algorithm that selects one optimal version from a set of candidate versions, but does not try to capture the user's preferences.

Still, the results that we achieve are not as good as what we initially hoped. One possible cause may be the fact that the algorithms we proposed currently do not take sequence characteristics (e.g. the amount of motion) into account. Other improvements may be possible by means of a more accurate description of the quality aspects that we take into account, in particular the spatial resolution, the amount of distortion and the smoothness of a video sequence.

Despite the limitations of the algorithms we proposed, in their performance as well as in their application area, we think that this thesis contains an initial and original contribution to the emerging objective of realizing *Quality of Experience* in multimedia applications.

Samenvatting

In een periode van enkele jaren is de wereld van multimediacommunicatie drastisch veranderd, en deze veranderingen zetten zich door. Geavanceerde formaten voor de compressie van audiovisuele informatie worden ontwikkeld, nieuwe vormen van bedrade en draadloze netwerktoegang verschijnen, en een enorm bereik aan soorten toestellen die bruikbaar zijn voor multimediacommunicatie is beschikbaar. Het tijdperk waarin multimediatoepassingen op het Internet het exclusieve domein waren van PC-gebruikers is definitief voorbij. De volgende generatie multimediatoepassingen zal gekenmerkt zijn door heterogeniteit: er zullen verschillen zijn wat betreft de netwerken, de toestellen en de verwachtingen van de verschillende eindgebruikers.

Deze heterogeniteit zorgt voor nieuwe uitdagingen: wanneer men met multimediatoepassingen een breed publiek wil bereiken, zal transparante consumptie van die multimediale inhoud mogelijk moeten zijn. Recent zagen twee groepen van technologieën het levenslicht die beide kunnen helpen bij het realiseren van *Universele Multimediatoegang*. De eerste technologie is die van schaalbare codering van audiovisuele inhoud. Dit soort van codering is nodig om er op een vlotte manier voor te zorgen dat een bitstroom kan verwerkt worden door toestellen met verschillende mogelijkheden, en verstuurd kan worden over netwerken met verschillende eigenschappen. De tweede technologie richt zich niet zozeer op de voorstelling van multimediale informatie zelf, maar eerder op het koppelen van de eigenlijke inhoud met beschrijvende informatie over die inhoud, de zogenaamde metadata. Een van de mogelijke toepassingen van metadata is de automatische selectie en aanpassing van multimediale presentaties. Dit is precies een van de belangrijkste doelstellingen van het Multimediale Raamwerk dat ontwikkeld wordt onder de noemer MPEG-21.

Binnen de MPEG-21 standaard werden twee formaten ontwikkeld voor het beschrijven van de structuur van bitstromen. Zulke beschrijvingen kunnen gebruikt worden als tussenliggende laag tussen een schaalbare bitstroom en het proces dat instaat voor de adaptatie van zulke bitstromen. Hierdoor kan adaptatiesoftware ontwikkeld worden onafhankelijk van het eigenlijke for-

maat dat gebruikt wordt. Bovendien kan men eenvoudig metadata toevoegen aan die beschrijvingen, en die metadata later gebruiken voor het aansturen van de adaptatiesoftware. Omdat in ons onderzoek veel aandacht gegaan is naar deze bitstroombeschrijvingstalen, is ook een deel van deze thesis aan dit onderwerp gewijd. We beschrijven beide formaten zoals ze door MPEG gedefinieerd werden, en we lichten ook onze eigen bijdragen toe: we stelden een aantal zogenaamde bitstroomschema's op, samen met voorbeelden van mogelijke adaptatiestappen, en dat voor verschillende formaten voor multimediale inhoud.

De belangrijkste doelstelling van deze thesis was de beschrijving van het proces dat we inhoudsnegotiatie[1] noemen, wanneer er gebruik gemaakt wordt van schaalbare formaten. Om zo een toepassing te kunnen beschrijven, vonden we het nodig om de eigenschappen van de datastructuren die gebruikt werden, in dit geval de schaalbare bitstromen, beter te begrijpen. Daarom introduceert deze thesis een formeel model dat we ontwikkeld hebben dat in staat is om de fundamentele concepten, relaties en afhankelijkheden zoals ze voorkomen in schaalbare bitstromen, voor te stellen. Naast de definitie van het eigenlijke model, tonen we ook de correctheid van dit model aan door het toe te passen op een aantal bestaande schaalbare codeerformaten. Bovendien doen we een eerste poging om op basis van dit model het proces van inhoudsnegotiatie voor te stellen als een optimalisatieprobleem met beperkingen.

In sommige scenario's, bijvoorbeeld wanneer er schaalbare bitstromen met meerdere dimensies gebruikt worden, blijkt dat onze voorstelling van inhoudsnegotiatie als een optimalisatieprobleem met beperkingen onvoldoende de realiteit benadert. Bij zulke bitstromen kunnen er immers meerdere aanpassingen mogelijk zijn die aan alle beperkingen voldoen. Soms is het daarbij duidelijk dat de ene versie beter is dan de andere, maar soms is het onmogelijk om op een objectieve manier twee versies te vergelijken wanneer er geen bijkomende informatie beschikbaar is. In dat geval zal er een afweging moeten gebeuren tussen verschillende kwaliteitsaspecten die in rekening gebracht worden. We maken daarom gebruik van de theorie van Pareto rond multicriteria-optimalisatie bij het formuleren van het proces van inhoudsnegotiatie wanneer er schaalbare bitstromen met meerdere dimensies gebruikt worden. Op die manier herformuleren we het klassieke optimalisatieprobleem naar een multicriteria-optimalisatieprobleem.

Eén van de belangrijkste problemen bij dit soort optimalisatieproblemen is dat er meerdere oplossingen als optimaal beschouwd kunnen worden. In

[1] Met inhoudsnegotiatie bedoelen we het selecteren, aanpassen en aanbieden van een zo goed mogelijke multimediale presentatie, gegeven de beperkingen opgelegd door de gebruikte toestellen en netwerken.

dat geval is er bijkomende informatie nodig, bijvoorbeeld gebruikersvoorkeuren, om daaruit één oplossing te kiezen. Deze problematiek is uiteraard niet nieuw, maar jammer genoeg zijn de bestaande oplossingen niet geschikt om ons probleem op te lossen omdat ze telkens verwachten van de beslissingsmaker, in ons geval de eindgebruiker, dat die een zeer goed begrip heeft van het probleem.

Om die reden stellen we in deze thesis een scenario voor waarin een zogenaamde agent[2] voor inhoudsnegotiatie de eindgebruiker een aantal videosequenties laat beoordelen. De agent gebruikt deze informatie dan om een model op te bouwen dat de voorkeuren van de gebruiker beschrijft, en dat later kan gebruikt worden om één oplossing te kiezen uit een verzameling kandidaat-optimale oplossingen.

Op basis van een literatuurstudie selecteerden we twee mogelijke algoritmen die in zo een agent gebruikt kunnen worden. Beide algoritmen kunnen gebruikt worden voor het opbouwen van een model op basis van een aantal beoordelingen die de gebruiker als voorbeeld heeft ingegeven, en voor het gebruiken van dit model bij het selecteren van een optimale versie. Het eerste algoritme dat we bestuderen veronderstelt dat de kwaliteit van een videosequentie kan uitgedrukt worden als een gewogen som van kwaliteitsaspecten. De voorbeeldbeoordelingen worden in dit geval omgezet naar een stelsel van lineaire ongelijkheden bij het opbouwen van een model. In het tweede algoritme wordt er op zoek gegaan naar gelijkenissen tussen de te voorspellen gegevens en de beoordelingen die de gebruiker als voorbeeld heeft gegeven.

In deze thesis analyseren we de voor- en nadelen van beide algoritmen op verschillende vlakken. We bekijken de rekenkundige complexiteit van de algoritmen, mogelijke parameters die gewijzigd kunnen worden, en de betrouwbaarheid. Om die te bepalen, hebben we een zogenaamde subjectieve test opgesteld, waarbij de deelnemers een aantal keuzes tussen twee versies van dezelfde videosequentie moeten maken. De betrouwbaarheid van de algoritmen kan dan nagegaan worden door een deel van de beslissingen van een deelnemer te gebruiken als trainingsinformatie, wat overeenkomt met de voorbeeldbeoordelingen voor de agent, en na te gaan of we op basis hiervan ook andere keuzes van die deelnemer konden voorspellen. In deze thesis vergelijken we niet enkel de twee algoritmen met elkaar, maar gaan we ook voor elk van de algoritmen na wat de gevolgen zijn van het aanpassen van bepaalde parameters. Uiteindelijk zullen we kunnen besluiten dat het algoritme dat gebaseerd is op gelijkenissen een aanvaardbare betrouwbaarheid heeft, zeker

[2] In deze context is een agent een stuk software dat beslissingen neemt in de plaats van de eindgebruiker, onder andere op basis van voorbeeldbeslissingen die deze gebruiker genomen heeft.

wanneer er voldoende voorbeeldbeslissingen gebruikt worden. In dat geval is de betrouwbaarheid immers beter dan wat theoretisch mogelijk is voor gelijk welk algoritme dat ook één oplossing kiest uit een verzameling kandidaat-optimale oplossingen, maar daarbij geen gebruik maakt van informatie over de gebruikersvoorkeuren.

Toch blijken de resultaten die we bereiken minder goed te zijn dan wat we initieel gehoopt hadden. Een mogelijke oorzaak ligt in het feit dat geen enkele van de voorgestelde algoritmen rekening houdt met eigenschappen van de originele sequentie (zoals bijvoorbeeld de hoeveelheid beweging). Andere verbeteringen zijn wellicht mogelijk door een meer accurate beschrijving te bepalen van de verschillende kwaliteitsaspecten die in rekening gebracht worden: de resolutie, de distortie en de vloeiendheid van de videosequentie.

Ondanks de beperkingen van de algoritmen wat betreft hun betrouwbaarheid maar ook wat betreft hun toepassingsgebied, denken we dat we met deze thesis een initiële en originele bijdrage hebben geleverd voor het bereiken van wat men ervaringskwaliteit (Eng. *Quality of Experience*) noemt.

List of abbreviations

API	Application Programming Interface
AVC	Advanced Video Coding
BiM	Binary Format for Metadata
BSDL	Bitstream Syntax Description Language
CIF	Common Intermediate Format
CIFL	Coding-Independent Fair Layered multicast
DCT	Discrete Cosine Transform
DIA	Digital Item Adaptation
DID	Digital Item Declaration
DIDL	Digital Item Declaration Language
DOM	Document Object Model
FGS	Fine-Granularity Scalability
FGST	Fine-Granularity Scalability with Temporal scalability
FLAVOR	Formal Language for Audio-Visual Object Representation
gBSD	Generic Bitstream Syntax Description
GOP	Group Of Pictures
GPRS	General Packet Radio Service
HVS	Human Visual System
IP	Internet Protocol
ISO	International Standards Organisation
ITU	International Telecommunication Union
kbps	kilobits per second
LAN	Local Area Network
MC-EZBC	Motion-Compensated Embedded Zerotree Block Coding
MCTF	Motion-Compensated Temporal Filtering
MOS	Mean Opinion Score
MPEG	Moving Picture Experts Group
NTSC	National Television System Committee
PAL	Phase Alternating Line
PC	Personal Computer
PDA	Personal Digital Assistant
PSNR	Peak Signal-to-Noise Ratio
QCIF	Quarter CIF

QoE	Quality of Experience
QoS	Quality of Service
ROI	Region Of Interest
SAX	Simple API for XML
SNR	Signal-to-Noise Ratio
SSM	Structured Scalable Meta-formats
STB	Set-Top Box
UCD	Universal Constraints Description
UED	Usage Environment Description
UMA	Universal Multimedia Access
UMCTF	Unconstrained Motion-Compensated Temporal Filtering
UME	Universal Multimedia Experience
UML	Unified Modeling Language
UMTS	Universal Mobile Telecommunications System
VOP	Video Object Pane
VTC	Visual Texture Coding
VQEG	Video Quality Experts Group
WAP	Wireless Access Protocol
XML	Extensible Markup Language
XPath	XML Path Language
XSLT	Extensible Stylesheet Language Transformations

List of symbols

$\#(.)$	Number of elements in a vector
D	The set of all data blocks
$\hat{d}$	The label of data block d
$[d]$	The payload of data block d
$\|d\|$	The size of data block d
P	The set of all parcels
B	The set of all scalable bitstreams
$p(x_0, x_1, \ldots)$	A version of parcel p
p^+	The closure of parcel p, the set of all versions of p
$\|\|p\|\|$	The size of parcel p
$\mathbb{P}$	The set of all properties
C	The set of all constraints
$\langle p \rangle_{Cts}$	The set of all feasible versions of p according to the constraints Cts
$x \succ_F y$	Solution x dominates solution y according to criteria F
$PF_F(X)$	Pareto frontier of the set of (feasible) solutions X, according to criteria F
$\mathcal{O}(\cdot)$	Asymptotic upper bound of an algorithm
$\Omega(\cdot)$	Asymptotic lower bound of an algorithm
$\theta(\cdot)$	Asymptotic tight bound of an algorithm
$a \succ_u b$	According to user u, sequence a is preferred to sequence b
$a \approx_u b$	User u does not prefer sequence a to sequence b or vice versa
$a >_u b$	Sequence a is *implicitly* preferred to sequence b, according to user u
$\overline{F(v)}\|_{v:V}$	Average value of $F(v)$, for all possible values of $v \in V$
$\mathcal{C}_U^{\mathrm{alg}}(n)$	Rate of consistent training sets for algorithm alg for a training set size n, for all users in U
$\mathcal{T}_U^{\mathrm{alg}}(n)$	Reliability of algorithm alg, according to the test set method, for a training set size n, for all users in U
$\mathcal{B}_U^{\mathrm{alg}}(n)$	Reliability of algorithm alg, according to the Best in Group method, for a training set size n, for all users in U

Contents

Chapter 1

Introduction

1.1 Context

The boost of the use of the Internet during the second half of the nineties was a major step in the popularization of access to multimedia content. Before, people were only used to consume mass information in a passive way: listening to broadcast radio, watching television programs, reading newspapers, etc. The way information is consumed on the World Wide Web is different in multiple ways, the main difference being a more personalized way of interacting with the content: people could actively search for information, rather than having to wait for a particular television show.

Later on, technical evolutions enabled the distribution of more advanced media types. Efficient compression algorithms, faster microprocessors, and broadband internet access resulted in the possibility to consume audio and video fragments and interactive scenes.

The last few years, access of multimedia content through the Internet is no longer the exclusive domain of PC users. More limited devices, such as Personal Digital Assistants (PDAs), set-top boxes (STBs) and even mobile phones become capable of accessing the Internet and presenting audiovisual information.

This evolution towards a more heterogeneous network of devices willing to access the same information creates new challenges in the domain of multimedia presentation and distribution. The diversity of devices results in a diversity of ways of interacting with the multimedia content:

- Different kinds of networks are used for accessing information on the Internet. At the time of writing, about 77% of the Belgian internet users

has a broadband connection, while 23% still uses a dial-up line[1]. Mobile phones typically use GPRS or UMTS connections, that have a much lower bandwidth than a typical broadband connection. Some PDAs use a Wireless LAN access point to connect to the Internet. Such connections can have a high bandwidth, but significant fluctuations in terms of bandwidth and delay are possible.

- The screen resolution of these devices also varies a lot. A desktop PC has a high (and often configurable) resolution, whereas the resolution of a mobile phone or a PDA is much smaller. The aspect ratio of the resolutions on different devices is often different too.
- The differences in processing power result in different capabilities, which can be important, e.g., for the real-time decoding of a video clip.
- Battery-enabled devices also have to take into account the power consumption of particular operations.
- Different devices have different ways of interaction possibilities. A regular PC is the most flexible device regarding interaction possibilities; set-top boxes and mobile phones are usually much more limited.

To this day, this growing problem of diversity of device characteristics is solved by offering multiple versions of the same information. Web sites are produced in different versions for supporting WAP[2] or i-Mode. When video sequences are offered, multiple versions are made available for supporting different ranges of bandwidth and different resolutions. It is clear that such a solution can be quite costly in terms of production cost and storage requirements. When the diversity of devices accessing multimedia content keeps increasing, this cost will also increase.

1.1.1 Scalable Video Coding

As video is the kind of multimedia that is most demanding in terms of processing power and data capacity, the problem of diversity is in this case the most severe. Researchers recognized this problem and came up with a solution that is called scalable video coding.

In this kind of coding, an encoder produces an *embedded* bitstream. This means that within the produced bitstream, one can find reduced versions of the

[1] As reported by the Internet Service Providers Association of Belgium, February 2005.

[2] Wireless Access Protocol, a protocol for enabling mobile phones to access mostly text-based information that is available on the Internet.

original video sequence, by selecting the appropriate fragments of the stream and dropping the others. This way, the quality of a bitstream can be reduced by means of very basic editing operations, thus producing a new bitstream that has a lower quality, but also has less requirements with respect to the properties of the device of the end user.

When talking about the reduction of the quality of a scalable video sequence, we mostly think of three different kinds of quality reduction:

- Reduction of the frame rate or temporal resolution: the amount of frames per second can be reduced, which results in a video sequence that runs less smoothly or is more choppy than the original.
- Reduction of the spatial resolution: the number of pixels, horizontally or vertically, can be reduced, which results in a smaller image.
- Quality scalability: the amount of distortion that is visible in the images can be increased when decreasing the bit rate.

Only recently, coding schemes were developed that allowed all these types of quality reduction to be available in the same bitstream; in this case we talk about *fully* scalable video coding.

The first efforts in developing encoders that were able to produce scalable bitstreams all resulted in a certain quality loss with respect to non-scalable solutions: for a given bit rate the scalable codec would produce a bitstream of a lower quality than its non-scalable counterpart. Very recent evolutions in the scalable coding algorithms allowed the production of scalable bitstreams that achieve the same quality as non-scalable streams. Because of these evolutions, the Moving Picture Experts Group (MPEG) has started a standardization process for scalable video coding.

As the concept of scalable video coding is so important for the delivery of video presentations in constrained, heterogeneous environments, an entire section of the following chapter is devoted to the evolutions in this domain.

1.1.2 Quality of Experience

Nearly concurrently with the important evolutions in the domain of scalable video coding we just mentioned, a new idea about multimedia consumption started to emerge: the concept of *Quality of Experience* (QoE) [1]. This can be considered to be the natural evolution of the concept of Quality of Service (QoS), mostly used in the context of communication networks.

In order to be able to set up reliable multimedia applications over the Internet, the “best effort” nature of the Internet Protocol (IP) was a big problem.

Using QoS, this drawback can be avoided by means of some sort of traffic contract guaranteeing certain bounds on the throughput, latency, etc. of the connection.

For the next generation multimedia applications, obtaining guarantees about the service offered by the network is no longer sufficient. Extending QoS to QoE means that we will also have to take into account the subjective nature of the end users. In [1], Jain tries to summarize the challenges in realizing Quality of Experience in multimedia applications. The following quotation matches very well with what we will present in this thesis, as we will summarize at the end of this book.

> To do so, we will have to develop measures that will help us capture QoE in a given application and use it. We need to make these measures as applicable to our field as required by our practice, while capturing the subjective nature of experience.

Some time before, Pereira and Burnett published an article that looks forward towards the move from QoS to QoE as well [2]. Up to now, research in multimedia focusses mainly on adapting a multimedia presentation to the constraints imposed by the terminal and the network. This challenge is often called Universal Multimedia Access (UMA). Pereira and Burnett claim that ultimately, research will have to acknowledge that the end point of universal multimedia consumption is the user and not the terminal.

Such a shift in focus from data delivery to the terminal to experience delivery to the users is called Universal Multimedia Experience (UME). Just like Jain insists on the need for measures that can capture Quality of Experience, Pereira and Burnett say that one of the problems that we have nowadays for realizing UME is that there are not many mechanisms for measuring the quality of an experience.

The most important part of this thesis is a framework that tries to harmonize the new possibilities that are offered by scalable video coding with some of the challenges that have to be overcome when realizing Universal Multimedia Experience.

1.2 Outline

The structure of this thesis is as follows. In the next chapter, we discuss into detail two technologies that are essential in the development of applications for the delivery of multimedia presentations in constrained, heterogeneous environments. The third chapter introduces the concept of bitstream descriptions

as an intermediate layer for adapting (scalable) bitstreams to the constraints imposed by a particular environment. In chapter four, we define an abstract model describing the structure of a scalable bitstream. We link this with the bitstream description mechanisms explained in chapter three, and use this model for formally defining a content negotiation process in which a scalable bitstream is adapted, taking into account a number of constraints. Chapter five introduces the general concept of multicriteria optimization, and explains why and how this concept can be applied to a content negotiation process. We also describe two algorithms that can be used for capturing subjective preferences regarding video quality and for assisting the multicriteria optimization process. In chapter six, we describe a test we executed involving a number of human participants, for validating the two algorithms proposed for capturing the subjective preferences of users. We end this thesis with the major conclusions that can be drawn from the research described in this book.

The research that has lead to this thesis resulted in a number of publications. One paper [3] is published in a journal that appears in the Science Citation Index. Another paper is currently under review for publication in Elsevier's *Signal Processing: Image Communication*. Our work served as a contribution to a paper that is to be published in *Multimedia Systems* [4]. In addition, there were 4 contributions to international conferences as a first author [5–8], and contributions to 7 other papers that were presented at international conferences [9–15].

Chapter 2

Enabling technologies

2.1 Introduction

In this chapter, we introduce some important recent technological evolutions that can assist in the delivery of multimedia information in heterogeneous, constrained environments.

A first technology that we want to describe into detail is scalable video coding. A scalable video encoder produces a scalable bitstream; as we already have mentioned in the previous chapter, such a bitstream can easily be modified by means of some basic editing operations, such as the removal of certain fragments from the bitstream. This way, a version containing a reduced quality level is obtained, but at the same time this version imposes less requirements on the client and the network. This enables the possibility of adapting, for example, the bitrate to the available bandwidth in the network, or the resolution of the video to the screen resolution of the requesting device.

A second technology that is discussed in this chapter is MPEG-21. One of the main objectives of this *Multimedia Framework* is the transparent delivery of multimedia resources in heterogeneous networks, which corresponds very well with the problems we want to tackle in this thesis. Because of the way MPEG defines its standards, there is still a lot of freedom allowing competition between companies developing products that comply to these standards. Throughout this thesis, we use this freedom at several places for creating useful applications.

Together, scalable video coding and the MPEG-21 Multimedia Framework can help application builders to realize the so called *Universal Multimedia Access* (UMA): the ability to access multimedia content at any place, any time. Because of the heterogeneous nature of terminals and networks involved, content negotiation is essential for implementing UMA. The process of content

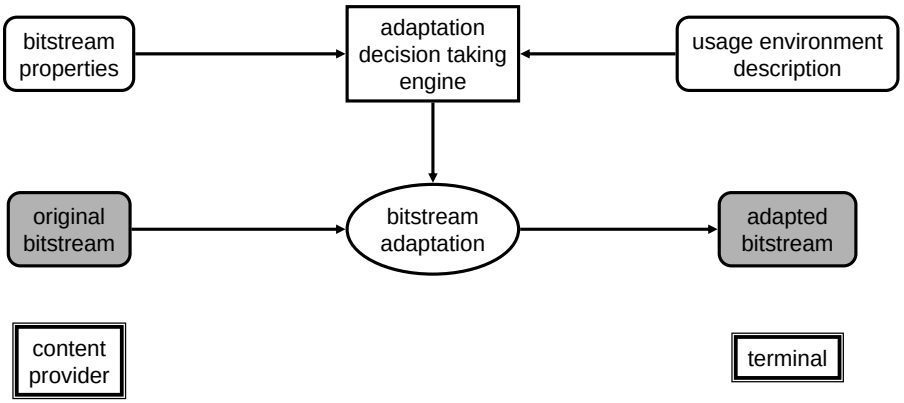


Figure 2.1: Generic representation of content negotiation and adaptation.

negotiation is schematically represented in Fig. 2.1. Here, an adaptation decision taking engine receives information about the capabilities of the terminal and the network, and information about the properties of the original bitstream. The engine tries to find a good adaptation decision, and transfers its decision to the adaptation engine, that creates an adapted bitstream from the original one.

Note that in this figure, the location of the adaptation decision taking engine and the bitstream adaptation process are not specified. This can happen on the server (content provider), client-side (at the terminal) or even in an intermediate node, such as a network gateway. The adaptation decision taking process and the bitstream adaptation process don't even have to take place on the same location, as long as the decision taking engine is able to pass its decision to the bitstream adaptation process.

2.2 Scalable video coding

Scalability is a principle that exists in the domain of video coding for a few years. The underlying idea is that a bitstream can be logically split up in different layers: a base layer offering a basic quality level, and one or more enhancement layers at different levels, where each layer improves the quality of the bitstream when it is combined with all layers at lower levels. This principle is schematically described in Fig. 2.2.

Quality can mean different things: it can point to the spatial resolution (number of pixels in each image), the number of frames (temporal resolution),

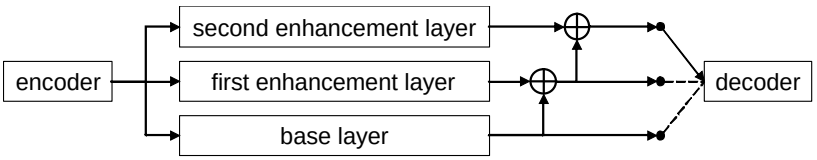


Figure 2.2: Basic principle of scalable coding.

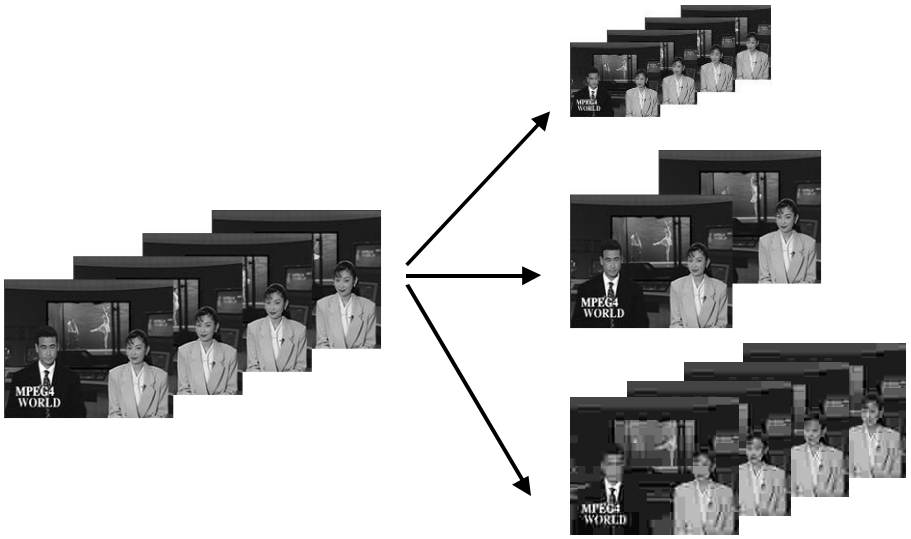


Figure 2.3: Three types of scalability in video coding.

or the severity of visual artifacts (distortion). These three different types of scalability are represented in Fig. 2.3. In some of the coding schemes presented in this section, only one of these types of scalability is possible. In other schemes, classified as fully scalable coding schemes, all of these types are available at the same time, and can be exploited concurrently.

A huge number of applications can benefit from scalable video coding. We focus on video distribution in heterogeneous environments, where scalability can be used for adapting a bitstream to the complexity of the decoder or the capacity of the network. Other useful scenarios are the transmission of digital video signals over error prone channels, where the lower layers can be transmitted in a more reliable way [16], region of interest coding [17], fast

browsing capabilities, encryption of higher layers for Digital Rights Management [18, 19], etc. In this section, we give an overview of the recent developments in scalable video coding.

2.2.1 First efforts in scalable video coding

Layered video coding was first introduced by MPEG in the MPEG-2 standard. An exhaustive overview of these techniques can be found in [20]. We only give a brief summary of the techniques used to obtain a two-layered video bitstream.

MPEG-2 provides multiple types of scalability. The objective was not to adapt a video sequence to a given bit rate, but rather to adapt the decoding complexity to the decoding device.

A first type of scalability provided by MPEG-2 is called *data partitioning*. Here, the coefficients obtained after the Discrete Cosine Transform (DCT) are split up: the most significant coefficients (low frequency information) are part of the base layer, the others belong to the enhancement layer.

MPEG-2 also provides *SNR scalability*. This is obtained by using two encoders: for each frame, a base layer encoder uses the normal coding scheme to construct the base layer using a coarse quantization. The enhancement layer coder performs a requantization of the DCT coefficients in order to construct an enhancement layer containing finer quantized information, thus offering a more detailed image and a higher SNR (Signal to Noise Ratio) value.

A third form of scalability provided by MPEG-2 is *spatial scalability*. The base layer is obtained by first downsampling the input signal, and encoding the resulting frames. Then the resulting base layer is decoded and upsampled to the original resolution; these frames are then fed to the motion prediction part of the enhancement layer encoder, thus enabling a spatio-temporal prediction from both previous enhancement layer frames and the current base layer frame.

Another possibility for scalable video coding using the MPEG-2 standard is *temporal scalability*. Technically speaking, this type of scalability is the most simple one. It can be achieved by encoding only a reduced number of frames in the base layer, and encoding the other frames in the enhancement layer. The frames that belong to the enhancement layer are allowed to use interlayer motion estimation: both base layer and enhancement layer frames can be used as reference frames to do motion estimation. The base layer encoder can only use base layer frames for the motion estimation.

The MPEG-2 standard also supports hybrid scalability, in which two different types of scalability are combined, resulting in a three-layered bitstream.

An important step forward in scalable video coding was taken during the development of the MPEG-4 standard. The schemes that exist in MPEG-2 are also present; all different types of scalability can be combined, resulting in a bitstream that consists of more than two layers. In addition, object-based scalability is also possible, in which the (temporal, spatial or SNR) quality of specific parts of a scene can be improved. The most spectacular innovation however, is surely the fine-granularity scalability (FGS).

2.2.2 Fine-granularity scalability

Fine-granularity scalability was developed to answer the need for a mechanism that easily allows real-time adaptation of the transmission bit rate in scenarios where video is offered using streaming technologies. At the same time, it had to be possible to encode the video sequence only once, before the transmission. Instead of optimizing the bitstream during encoding for one particular bit rate, it is optimized for a range of bit rates. This could be a very useful tool for supporting unicast and multicast streaming over the Internet [21, 22], where the available bandwidth changes frequently.

An FGS-encoded video sequence actually consists of one base layer and a single enhancement layer that improves the quality of the resulting video. A very important difference with the SNR scalability found in MPEG-2 is that this enhancement layer can be truncated by the streaming server at an arbitrary point within each Video Object Pane (VOP, the MPEG-4 terminology for a frame). This is shown in Fig. 2.4, where the streaming server will only transmit the gray parts of the enhancement layer data. The size of these parts is dynamically adapted to the actual available bandwidth between server and client at transmission time. In contrast, in the SNR scalability of MPEG-2, the bit rate of the enhancement layer had to be selected during the encoding process.

Without giving too much details, we want to introduce the basic principles of the FGS encoding scheme. The base layer is encoded following the non-scalable, DCT-based MPEG-4 encoding scheme. For the enhancement layer, *embedded* DCT coding is applied to the residual frames. In such a coding scheme, each frame consists of a number of bitplanes. The first bitplane contains the most significant bits of the DCT coefficients of the macroblocks of the residual frames, the second bitplane contains the second most significant bits, etc. An FGS bitstream is called embedded because reduced versions of the data can be found inside the bitstream itself; they can be obtained by truncating the bitstream at any point, thus removing the lower bitplanes.

As an addition to the embedded DCT coding technique, frequency weight-

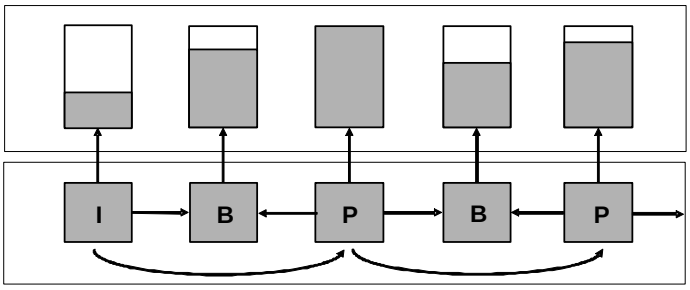


Figure 2.4: Real-time bit rate adaptation in MPEG-4 FGS. Only the gray parts of the enhancement layer data will be transmitted.

ing and selective enhancement can be applied. The first technique is used for shifting the low frequency DCT coefficients, thus hoping to improve the visual quality of reduced bitstreams. Selective enhancement is used for region of interest (ROI) coding: macroblocks that belong to the ROI are shifted during encoding, such that less information is dropped from these macroblocks when reducing the bit rate.

In addition, MPEG-4 FGS can be combined with temporal scalability. This is often called FGST, and this type of hybrid scalability allows to choose, at transmission time, between temporal scalability and SNR scalability.

Here, the base layer offers a video at a low SNR quality and a lower frame rate than the original sequence. The FGS scheme can be applied to improve the SNR values of these frames. Frames that were not encoded in the base layer are predicted from the base layer using motion estimation, and the residual frames are encoded using the FGS scheme. Therefore, an FGST frame consists of two parts: a set of motion vectors allowing the reconstruction of a predicted frame, and a set of bitplanes that represent the encoded residual frame. As shown in Fig. 2.5, FGST frames can be placed in the same enhancement layer, or in a second enhancement layer.

2.2.3 Fully scalable wavelet-based video coding

In still image coding, wavelet subband filter banks are used to do a transformation on the image to obtain information that is easier to compress. This can be used as a replacement for the block-based DCT transformation, and has already proven its efficiency in image compression, for example in the JPEG2000 standard [23].

An additional advantage of such coding schemes is the possibility to in-

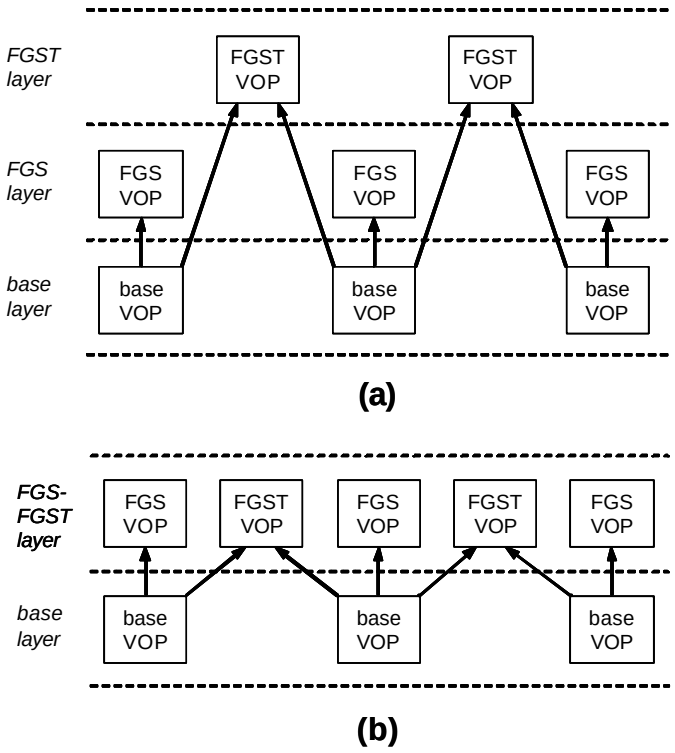


Figure 2.5: Two options to split up an FGST stream in layers: (a) the FGST VOPs belong to the temporal enhancement layer, (b) FGS and FGST VOPs are found in the same layer.

troduce scalable coding. Spatial scalability can be obtained very easily, as it follows inherently from the subband coding scheme, in which a spatial decomposition is applied. In addition, most schemes arrange the wavelet coefficients in an embedded way. This means that low quality versions of the image can be obtained from the bitstream by truncating it at a given point. This is obtained by applying a bitplane coding technique to the wavelet coefficients: the most significant bits are placed earlier in the bitstream (note the similarity with MPEG-4 FGS coding).

The successes of such coding schemes for still images inspired researchers to start exploring the possibilities of wavelet-based video coding. The objective is to develop a video coding scheme that creates *fully* scalable bitstreams. This means that three types of scalability can be provided at the same time:

spatial, SNR and temporal scalability. Moreover, we do not want to lose compression efficiency compared with the current state-of-the-art coding schemes.

Early work by Jens-Rainer Ohm resulted in algorithms for so-called Motion Compensated Temporal Filtering (MCTF), where 3-D wavelet-based subband coding is applied after a motion compensation step [24, 25].

In [26], Hsiang and Woods present a successfully implemented wavelet-based coding scheme that works in a similar way. It starts with a temporal filtering on the motion trajectory, based on well known motion compensation techniques [27]. A wavelet coder for still images is extended to three dimensions, and is applied to the 3-D signal. This coding scheme offers three types of scalability at the same time, and achieves a compression performance that is better than nonscalable MPEG-2 video.

Other approaches work the other way around: first, frames are decomposed using a wavelet transform as is used in still image wavelet coding, and after the decomposition, a temporal decomposition is executed in the wavelet domain [28, 29]. This class of algorithms is also referred to as *2D+t*, while the algorithms presented in the previous paragraphs belong to the *t+2D* class.

Recent evolutions regarding the temporal filtering, called Unconstrained Motion Compensated Temporal Filtering (UMCTF), allow an increased coding performance and a reduced delay [30].

2.2.4 Scalability in H.264/AVC

The best standardized video coding scheme in terms of compression efficiency, is currently H.264/AVC, also known as MPEG-4 Part 10 [31, 32]. In itself, this codec is not meant to offer scalable bitstreams, even though temporal scalability can be offered in certain situations [33, 34].

Because of the high bit rate savings of H.264/AVC compared to other video coding schemes, scalable extensions to the existing H.264/AVC are currently developed within MPEG. Temporal scalability can be realized by implementing a Motion Compensated Temporal Filtering (MCTF), as used in wavelet-based video coding, and for which the standardized syntax from H.264/AVC can be reused [35]. By iteratively applying the temporal decomposition, multiple temporal levels can be offered.

SNR scalability can be implemented by requantizing the quantization error of the previous SNR layer, while using a finer quantization step size. This can be added to the current H.264/AVC specification with only minor additions.

For enabling resolution scalability, more extensions are needed, because an encoder should have the option of using up-sampled lower-resolution macroblocks for prediction, for reusing motion vectors from lower resolution lay-

ers, and for reusing the up-sampled residual signal of the lower resolution layer. When motion vectors are reused, a motion vector refinement will have to be transmitted for accurately describing the predicted motion in the higher resolution layers [36].

2.3 MPEG-21: The multimedia framework

The problem of heterogeneous devices that are used to access the same multimedia content was one of the main reasons why the MPEG standardization committee started considering the definition of an entire *multimedia framework* for delivery and consumption, which resulted in the new MPEG-21 standard.

Rather than creating a world in which each community develops its own standard for exchanging multimedia content, MPEG-21 seeks to create *the big picture* of multimedia standards. It tries to establish interoperability between the different parts of a multimedia production chain by focusing on how the elements of a multimedia application infrastructure should interact. Where open standards for such interaction are missing, MPEG is creating new parts to fill the gaps [37, 38].

The global vision of MPEG-21 can be summarized as follows: to define a multimedia framework to enable transparent and augmented use of multimedia resources across a wide range of networks and devices used by different communities [39].

The fundamental unit for distribution and transaction in the MPEG-21 framework is the Digital Item. This is a combination of resources, metadata and structural information, expressing the relationships that exist between these resources and their metadata.

The MPEG-21 standard is split up into a number of distinct parts. Some parts are already finalized, while others are still in development. The following list is a complete overview of all parts of the MPEG-21 framework at the time of writing.

1. *Vision, Technologies and Strategy*: defines an architectural overview of the MPEG-21 framework, together with a number of requirements.

2. *Digital Item Declaration*: specifies a model (semantics) of the concept of a Digital Item, together with its representation (syntax). We discuss this part in Sect. 2.3.1.

3. *Digital Item Identification*: defines how to uniquely identify a Digital Item or a particular part of it.

4. *Intellectual Property Management and Protection*: provides an interoperable mechanism for the reliable protection of multimedia content across different devices.

5. *Rights Expression Language*: defines a language for declaring rights and permissions; the terms used in this language are defined in the Rights Data Dictionary.

6. *Rights Data Dictionary*: specifies a dictionary of key terms used to describe the rights of different users.

7. *Digital Item Adaptation*: defines description tools that allow transparent access to multimedia content by shielding users from the properties of the terminals and networks in use. We discuss this part in Sect. 2.3.2.

8. *Reference software*: bundles software packages that implement the different tools specified in other MPEG-21 parts.

9. *File Format*: defines a file format for storing and distributing Digital Items in an efficient way.

10. *Digital Item Processing*: defines how Digital Items can be processed in an interoperable way.

11. *Evaluation methods for persistent association technologies*: documents best practices in evaluating persistent association technologies using a common methodology.

12. *Test bed for MPEG-21 resource delivery*: provides a test bed for delivering scalable content and for evaluating the behavior of the delivery of this content in a streaming environment.

13. *Scalable Video Coding*: moved to the MPEG-4 specification for scalable extensions for Advanced Video Coding, and is no longer part of the MPEG-21 standard.

14. *Conformance*: defines testing procedures for different parts of the MPEG-21 framework, in order to validate compliance of particular implementations of the MPEG-21 standard.

15. *Event reporting*: standardizes mechanisms for monitoring and communicating events relating to Digital Items and/or the programs and devices that operate on them.

16. *Binary Format*: enables the efficient storage and exchange of several kinds of MPEG-21 descriptions by means of an efficient compression method for XML documents.

17. *Fragment identification of MPEG resources*: specifies a syntax for addressing specific parts of resources of different MPEG media types.

18. *Schemas*: contains all XML Schema files for the different MPEG-21 parts.

The main advantage of splitting up the standard into distinct parts is that they don't have to be used as a whole: each part can equally be used standalone in particular applications. This is the approach we use further on in the following chapter: we use a number of technologies defined in MPEG-21 Part 7, the Digital Item Adaptation specification, without using other MPEG-21 parts. As will become clear in the following section, the Digital Item Declaration can easily be used in a meaningful way without the other MPEG-21 parts.

In the remainder of this section, we go deeper into detail on two parts of the MPEG-21 specification that are particularly of interest when delivering multimedia content in heterogeneous environments: *Digital Item Declaration* [40] and *Digital Item Adaptation* [41]. The former defines a language for describing structured digital items, and provides hooks for offering different versions of the same content. The latter is entirely devoted to the seamless adaptation of multimedia presentations to a particular environment.

2.3.1 Digital Item Declaration

In MPEG-21 Part 1, *Vision, Technologies and Strategy* [37], Digital Items are defined as *structured digital objects, with a standard representation, identification and metadata within the MPEG-21 framework*. The actual representation of a Digital Item is defined in the *Digital Item Declaration* [40, 42, 43]. This document consists of three parts:

- An abstract model defining a set of abstract elements and concepts that are relevant for declaring Digital Items.

- The representation of the model gives a formal description of the syntax and semantics of the elements and concepts of the model, expressed by means of XML. This XML representation is called the Digital Item Declaration Language (DIDL).

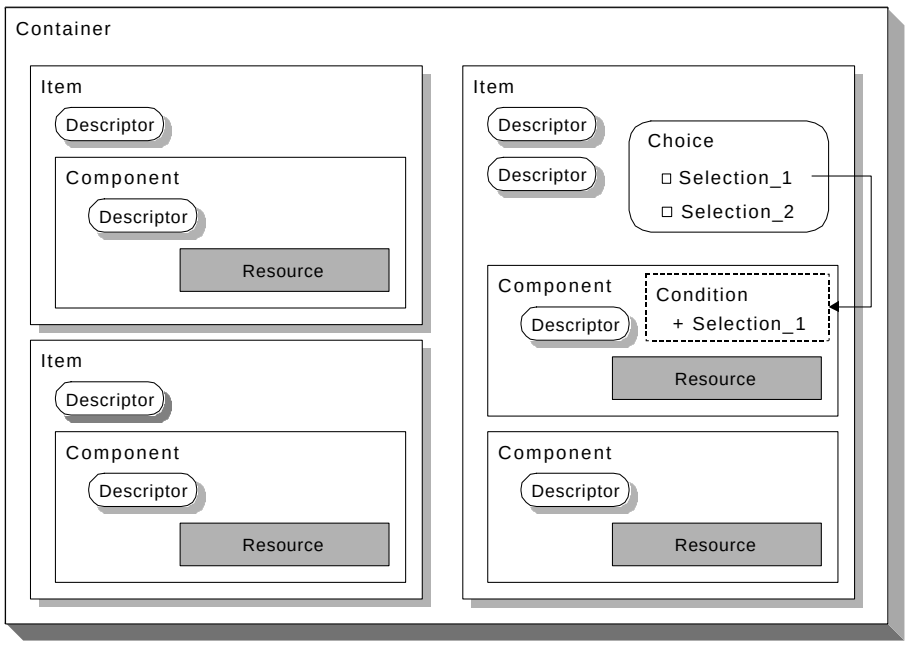


Figure 2.6: Building blocks of a Digital Item.

- The MPEG-21 DIDL Schema is an XML Schema defining the grammar of the Digital Item Declaration model and Digital Item Declaration Language in XML.

The abstract model has been designed to be as flexible and general as possible, providing hooks that enable higher level functionality and interoperability. This way, the model can be used in higher level models, for example within other parts of MPEG-21. The abstract model can also help to perform mappings between existing languages that have related mechanisms for defining digital items.

In Fig. 2.6, an example Digital Item is shown, in which several building blocks of the abstract model are used: **resources** that identify the actual content, **descriptors**, for expressing descriptive information, **components**, that are used to bind resources to descriptors, **items**, that are used to declare logically indivisible works or compilations, and **containers** for grouping related items or other containers.

The Digital Item Declaration provides a powerful mechanism for realizing Universal Multimedia Access: most elements can be made conditionally

available, by means of **conditions**. This is achieved by means of **choices** and **selections**, where the state of certain **predicates** can be determined. Choices and selections can contain human-readable or machine-readable information contained in descriptors for assisting in selecting the appropriate choices.

As an example of a complete Digital Item Declaration, the DIDL document in Listing 2.1 represents the Digital Item that is graphically represented in Fig. 2.6.

Listing 2.1: A basic Digital Item Declaration.

```
<DIDL xmlns="urn:mpeg:mpeg21:2002:02-DIDL-NS">
  <Container>
    <Item>
      <Descriptor>
        <Statement mimeType="text/plain">
          This is the first item in our collection
        </Statement>
      </Descriptor>
      <Component>
        <Descriptor>
          <Statement mimeType="text/plain">
            Here you can find the actual resource
          </Statement>
        </Descriptor>
        <Resource mimeType="video/mpeg"
          ref="http://myserver/videos/video_1.mpg"/>
      </Component>
    </Item>
    <Item>
      <Descriptor>
        <Statement mimeType="text/plain">
          This is the second item in our collection
        </Statement>
      </Descriptor>
      <Component>
        <Descriptor>
          <Statement mimeType="text/plain">
            Here you can find the actual resource
          </Statement>
        </Descriptor>
        <Resource mimeType="video/mpeg"
          ref="http://myserver/videos/video_2.mpg"/>
      </Component>
    </Item>
    <Item>
      <Descriptor>
        <Statement mimeType="text/plain">
          This is the third item in our collection
```

```
        </Statement>
      </Descriptor>
      <Descriptor>
        <Component>
          <Resource mimeType="image/jpeg"
            ref="http://myserver/images/thumbnail_3.jpg"/>
        </Component>
      </Descriptor>
      <Choice minSelections="1" maxSelections="1">
        <Selection select_id="high_bandwidth"/>
        <Selection select_id="low_bandwidth"/>
      </Choice>
      <Component>
        <Condition require="high_bandwidth"/>
        <Descriptor>
          <Statement mimeType="text/plain">
            Here you can find the high bandwidth version
          </Statement>
        </Descriptor>
        <Resource mimeType="video/mpeg"
          ref="http://myserver/videos/video_3_300kbps.mpg"/>
      </Component>
      <Component>
        <Descriptor>
          <Statement mimeType="text/plain">
            Here you can find the low bandwidth version
          </Statement>
        </Descriptor>
        <Resource mimeType="video/mpeg"
          ref="http://myserver/videos/video_3_100kbps.mpg"/>
      </Component>
    </Item>
  </Container>
</DIDL>
```

The DID we just presented, contains an example of how the MPEG-21 Digital Item Declaration specification can be used for offering multimedia content in heterogeneous environments. In the example, a high bit rate and a low bit rate version of the same video sequence exist, but the high bit rate version is only available if the *predicate* `high_bandwidth` is true. This *predicate* is assigned a value when evaluating a *choice*, in which one of three options (high, medium or low available bandwidth) has to be selected.

In a practical example, the `Choice` element and the `Selection` elements will contain descriptors. This can be human-readable information, that can be used for presenting the user a dialog box where he has to make an appropriate selection. It can also contain machine-readable information, such

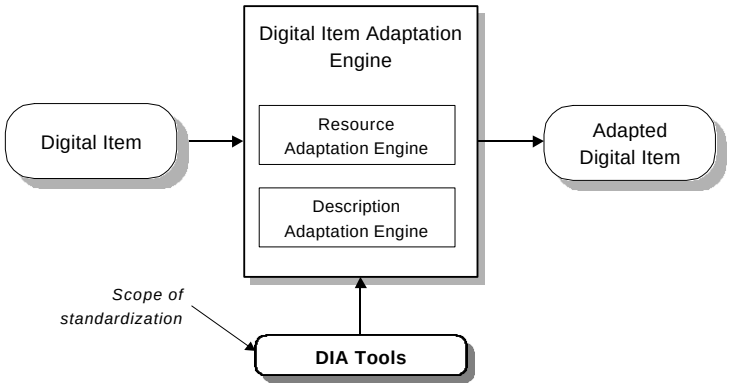


Figure 2.7: Digital Item Adaptation architecture [41, 45].

as MPEG-7 metadata [44] or a usage environment description as defined in MPEG-21 Digital Item Adaptation [41].

2.3.2 Digital Item Adaptation

Another part of MPEG-21 that focuses on enabling the distribution of multimedia content in heterogeneous environments is surely MPEG-21 Part 7, Digital Item Adaptation [41, 45, 46]. Its goal is to offer tools for the effective adaptation of multimedia presentations. In this section, we give an overview of the DIA specification.

The conceptual architecture of Digital Item Adaptation is shown in Fig. 2.7. Here, we can see that a Digital Item is subject to a resource adaptation engine as well as a description adaptation engine. Together, these engines produce an adapted Digital Item. An important remark in Fig. 2.7 is that the DIA specification does not standardize the adaptation engines themselves, but only the tools that are used for steering these engines [39, 45]. This way, MPEG opens competition between different companies offering DIA engines: they can implement the engines in several ways, e.g., focussing more on the speed of implementation or rather on the resulting quality.

The DIA specification consists in itself of different parts that can be used stand-alone or in combination with other MPEG-21 tools. These parts can be grouped into the following eight clusters; each of them is described in the following sections.

- The *Usage Environment Description Tools* provide a language for describing different aspects of the usage environment: user characteristics,

terminal capabilities, network capabilities and natural environment characteristics.

- The *BSDLink* tool provides facilities to link several parts of an adaptation chain into an entire adaptation architecture.
- The *Bitstream Syntax Description* tools define mechanisms for describing the high-level structure of a bitstream. This can facilitate the adaptation of these bitstreams.
- The next category is referred to as *Terminal and Network Quality of Service*, and allows to describe relationships between QoS constraints, adaptation operations and the resource qualities that result from these operations.
- The *Universal Constraint Description Tools* provide a mechanism for expressing limitation and optimization constraints on adaptations.
- *Metadata adaptability* specifies hint information for reducing the complexity of adapting the metadata contained in a Digital Item.
- The goal of the *Session Mobility* tool is to capture and transmit the configuration state of Digital Item when it is transferred from one device to another.
- The *DIA Configuration Tools* can assist an adaptation engine in its configuration.

Usage Environment Description Tools

A *Usage Environment Description* provides descriptive information about several aspects of the usage environment. This information can be used by a Digital Item Adaptation Engine for taking into account the constraints imposed by the environment, preferences expressed by the end user, etc.

The following aspects of the usage environment can be covered in a Usage Environment Description.

- Characteristics of the end user, such as general user information, content and presentation preferences, accessibility characteristics, mobility characteristics and destination.
- Characteristics of the terminal that is used for accessing Digital Items. This can be information about the decoding and encoding capabilities, input-output capabilities and device properties, such as power and storage characteristics.

- Static and dynamic characteristics of the network that is available, such as the maximum bandwidth, the guaranteed minimum bandwidth, delay characteristics, packet loss rate, bit error rate, etc.
- The characteristics of the natural environment of the usage of a Digital Item. This comprises the location and the time of usage, but also audio-visual environment characteristics, such as background noise and illumination.

In Listing 2.2, a Usage Environment Description is shown, giving information about the characteristics of the network between the terminal and the server. This Usage Environment Description contains only information about the network characteristics, in particular its capabilities (maximum and minimum available bandwidth) and its specific conditions at that moment: the maximum and average bandwidth, the delay that is measured, and the packet loss rate. A Resource Adaptation Engine can use this information when it wants to reduce the bit rate of a scalable video sequence when transmitting it to the terminal.

Listing 2.2: A usage environment description containing information about the network characteristics.

```
<DIA xmlns="urn:mpeg:mpeg21:2003:01-DIA-NS">
  <Description xsi:type="UsageEnvironmentType">
    <UsageEnvironment xsi:type="NetworkCharacteristicsType">
      <NetworkCharacteristics xsi:type="NetworkCapabilityType"
        maxCapacity="384000" minGuaranteed="32000"/>
      <NetworkCharacteristics xsi:type="NetworkConditionType">
        <AvailableBandwidth maximum="256000" average="80000"
          interval="330"/>
        <Delay packetTwoWay="330" delayVariation="66"/>
        <Error packetLossRate="0.05"/>
      </NetworkCharacteristics>
    </UsageEnvironment>
  </Description>
</DIA>
```

BSDLink

The main goal of the *BSDLink* tool is to link any kind of steering description tools (e.g., a Usage Environment Description) with the bitstream description tools in a flexible and extensible way. A BSDLink links steering description information, such as AdaptationQoS information, bitstream description information, and possible transformations for this bitstream description together.

The output of the steering description is used as an input parameter for the transformation on the bitstream description.

Bitstream Syntax Description

Within the MPEG-21 Digital Item Adaptation specification, two mechanisms for expressing *bitstream descriptions* by means of XML are defined. These bitstream descriptions assist a Resource Adaptation Engine in transforming a (possibly scalable) bitstream by means of rather straightforward editing operations. Because of the importance of bitstream descriptions for this thesis, and because of our original contributions in this domain, an entire chapter (see Chapter 3) of this thesis is devoted to the subject of bitstream descriptions.

Terminal and Network Quality of Service

Another cluster in Digital Item Adaptation is called *Terminal and Network QoS*. This is a set of tools that enable the expression of relationships between Quality of Service constraints imposed by the network (e.g., the available bandwidth) and the terminal (e.g., computational capabilities), the possible adaptation operations, and the influence of these adaptations on the resulting quality or utility. This way, an effective adaptation strategy can be formulated that can help in determining an optimal adaptation decision.

Universal Constraints Description tools

The *Universal Constraints Description* (UCD) tool allows the expression of constraints by means of resource and environment characteristics, as well as optimization guidelines.

The UCD tool becomes very powerful when used in conjunction with AdaptationQoS information from the Terminal and Network QoS tool. Using both sources of information, a Resource Adaptation Engine can construct a generic constrained optimization problem, as described in [47].

The variables in such a problem are expressed by means of *IOPins*. Dependencies between these variables are expressed by means of *modules*. Different types of modules exist: look-up tables, numeric functions, and lists of possible utility values [48]. Information in the Usage Environment Description can be referenced from the Terminal and Network QoS tools. The constraints and the optimization functions of the optimization problem can be expressed by means of the UCD tool.

As a concrete example, consider a fully scalable video sequence. Each Group of Pictures is treated as a single adaptation unit. Relevant variables that

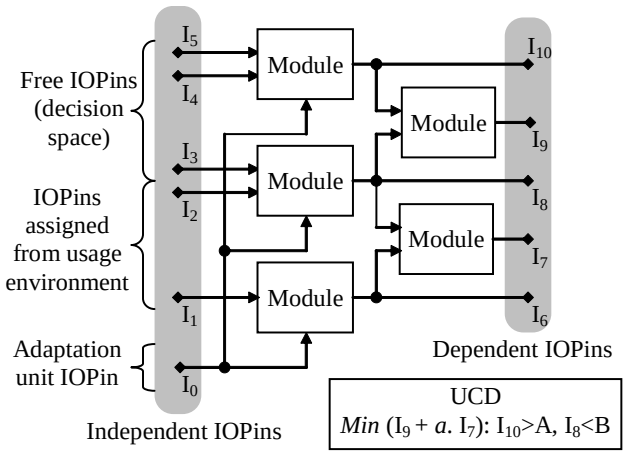


Figure 2.8: Illustrations of IOPins and Modules in Terminal and Network QoS [47].

are assigned from the UED, expressed by means of IOPins, are the screen resolution and the available bandwidth. They are used for expressing constraints on the acceptable solutions, by means of Modules that compare the resolution of the adapted sequence and the screen resolution of the terminal, and the bit rate and the available bandwidth. The free variables, also expressed as IOPins, are the number of spatial and temporal levels, and the number of bit planes that are allowed. Probably, a weighted sum of the resulting PSNR value, resolution and frame rate will be used as an optimization function.

Fig. 2.8 shows an example of a Terminal and Network QoS description, consisting of IOPins and Modules from the AdaptationQoS tool, as well as UCD information. As can be seen from the figure, some IOPins are independent, while others depend on other IOPins by means of Modules. Within the independent IOPins, some are assigned from the usage environment, one is used for referring to a particular *adaptation unit*, and the remaining IOPins are free variables that can be modified during the adaptation process. The UCD in the example consists of two constraints on the possible values of the dependent IOPins, and a minimization function derived from two dependent IOPins.

Metadata Adaptability

The next cluster of DIA tools deals with *Metadata Adaptability*. This description tool specifies hint information that can be used to reduce the complexity of adapting the metadata contained in a Digital Item, by filtering the Digital Item from less interesting or too detailed information.

Session Mobility

The *Session Mobility tool* offers a mechanism for preserving the current state of interaction between a particular User and a Digital Item when the User wants to change to another device. This configuration state consists of the states of the different *predicates* of the Digital Item, but also application-state information. It is transmitted by means of a so-called *context Digital Item*.

DIA Configuration tools

The *DIA Configuration tool* assists a terminal in resolving the choices and selections of a Digital Item. It allows the author of a Digital Item to specify which DIA descriptors are needed from the sender and receiver side, and which choices of the Digital Item should be automatically configured by the terminal and which choices should be presented to the end user.

Chapter 3

Bitstream descriptions

3.1 Introduction

In the introduction of this thesis, we already briefly discussed the properties of scalable video coding. The most important property of compressed bitstreams that are encoded in a scalable way, is that they can be adapted to a lower quality version imposing less requirements by means of rather straightforward editing operations. This way, a reduced bitstream can be produced, that is suitable for the constraints imposed by the environment.

As an alternative to scalable video coding, real-time transcoding exists for quite some time [49]. Here, fast dedicated implementations operate on the bitstreams. The same approach can be used for developing software for adapting scalable bitstreams. Even though these editing operations are supposed to be straightforward, it is still an annoying and error-prone task to produce software that is capable of executing the adaptation process. It is often difficult to reuse such application-specific and format-specific software in similar applications operating on the same coding format.

Moreover, in order to steer this adaptation in an intelligent way, it can be useful to attach descriptive information, metadata, to the bitstream itself. This information can be used, for example, for filtering a multimedia presentation based on its content [50], or for inserting so called *generic complexity metrics* that can help estimating the complexity needed for decoding a bitstream [51].

To overcome these problems, it is useful to have a high-level description of the structure of a scalable bitstream, in parallel with the bitstream itself. Such bitstream descriptions can then serve as a level of abstraction operating on top of the bitstream level: manipulations executed on the bitstream descriptions reflect manipulations on the bitstream itself. The preferred language for representing these structural descriptions is XML, the eXtensible Markup Lan-

guage [52], for a number of reasons.

- XML offers a very flexible way of describing highly structured data.
- In a few years time, XML has become the de facto language for describing metadata [53]. This way, bitstream descriptions can be easily integrated with existing metadata standards, such as MPEG-7 [44].
- A huge amount of tools exist for processing XML information in a fast, reliable and flexible way.
- XML is extensible: it is possible to add information to an existing XML document in such a way that applications that are not familiar with this new information will ignore it.
- As it is represented by means of plain text, XML is platform independent.

Within Part 7 of the MPEG-21 standard, that offers the tools needed for Digital Item Adaptation (DIA) [41], two related frameworks are defined that can be used for bitstream descriptions.

In this chapter, we first describe these frameworks. Next, we discuss our original contributions in this domain, in particular the construction of a number of useful examples of how the frameworks proposed by MPEG-21 can be used for producing descriptions of bitstreams that comply to existing scalable formats. Before concluding the chapter, we describe some other languages for bitstream descriptions that are developed outside MPEG.

In general, a bitstream description generation and adaptation can be described as is shown in Fig 3.1. In the remainder of this chapter, this figure is reused for describing the typical architectural properties of the different bitstream description frameworks that are discussed in this chapter.

3.2 Bitstream Syntax Description Language

3.2.1 Introduction

The main objective of BSDL, the Bitstream Syntax Description Language [54–57], is that it should be possible to automatically produce bitstream descriptions in XML, based on a formal definition of the syntax of these bitstreams. This syntax definition should also be sufficient for regenerating a bitstream from an adapted bitstream description.

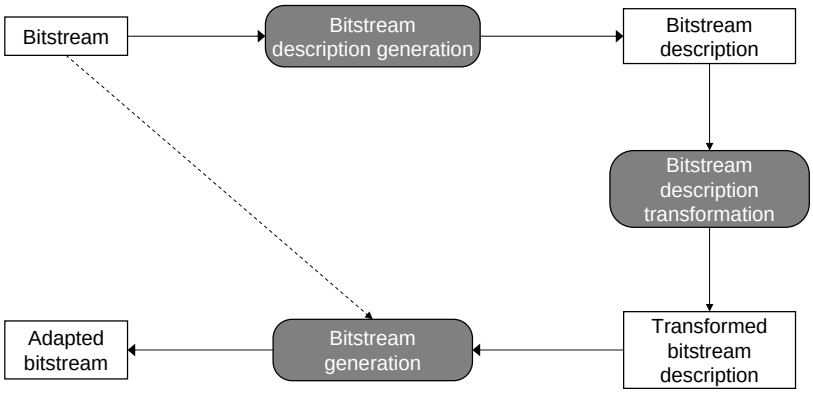


Figure 3.1: General architecture for a bitstream description generation and transformation system [45].

The entire work flow of the BSDL framework is shown in Fig. 3.2. In this framework, a bitstream that complies to a certain syntax, described by means of a BSDL Schema, is fed to the BinToBSD process. This process uses the BSDL Schema to produce a bitstream description. This description can be adapted using any tool for manipulating XML data, such as XSLT (eXtensible Stylesheet Language - Transformations) [58], an application that uses an XML parser that implements an API such as DOM (Document Object Model)[1] or SAX (Simple API for XML)[2], etc. In principle, one can even use a basic text editor for adapting the bitstream description. It is important to note that the appropriate selection of a good transformation mechanism is very important, as it has a significant impact on the execution speed and memory consumption of the transformation step [4, 59].

After manipulating the bitstream description in an appropriate way, we end up with an adapted bitstream description. The BSDToBin tool is capable of using this description to generate the adapted bitstream itself, using the information that can be found in the BSDL Schema. If the bitstream description contains references to the original bitstream, this bitstream is also needed as an input to the BSDToBin process, as can be seen from Fig. 3.2.

The most innovative idea behind the Bitstream Syntax Description Language is that the language for expressing a BSDL Schema is an extension of the XML Schema [60] language. Whereas an XML Schema defines a syntax

[1] DOM, the Document Object Model, is a language-independent API for manipulating XML documents.

[2] http://www.saxproject.org

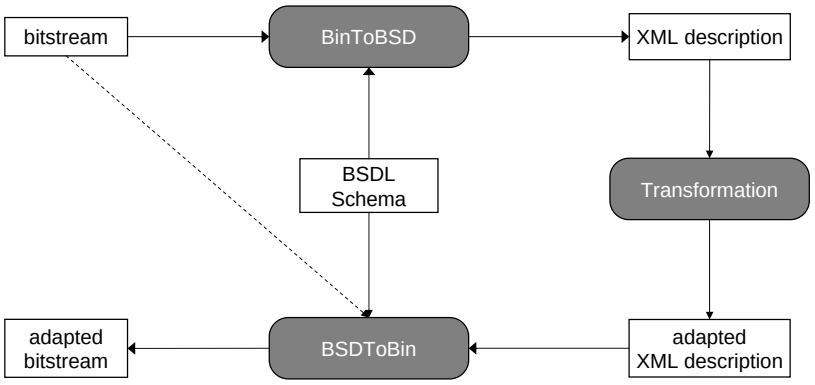


Figure 3.2: Data flow in the BSDL framework.

for a certain class of XML documents, a BSDL Schema defines a syntax for a certain class of XML bitstream descriptions, and is at the same time a syntax for the bitstreams themselves[3]. This way, existing XML Schema tools can be used for validating the correctness of a BSDL bitstream description with respect to its BSDL Schema.

In the following section, we give a complete description of all constructs that can appear in a BSDL Schema that complies to the MPEG-21 DIA specification. A number of example BSDL Schemas for several types of bitstreams can be found in Sect. 3.4.

3.2.2 Specification

As we just mentioned, the BSDL Schema language is an extension of XML Schema. It also defines a limited number of restrictions on what can occur in a Schema. The extensions are expressed by means of two XML Schemas. The *Schema for BSDL-1 Extensions* defines a number of attributes and data types that can be used in a BSDL Schema, but that do not exist in XML Schema. The BSDToBin process only needs these extensions for producing correct bitstreams from a bitstream description. The *Schema for BSDL-2 Extensions* defines a number of additional extensions to XML Schema that can be needed to resolve ambiguities in the BinToBSD process.

[3]This explains the name Bitstream *Syntax* Description Language.

Restrictions on XML Schema

Some constructs and data types that can occur in an XML document and that can be expressed in an XML Schema are not useful for bitstream descriptions as they can cause certain ambiguities. Therefore, BSDL defines a number of restrictions on the XML Schema language.

A first restriction is that data that occurs in the bitstream can only appear inside an XML element, not in an attribute. The reason for this restriction is that the order of appearance of an attribute in an XML element has no meaning, and can therefore be modified by any tool operating on the XML document.

Mixed content is not allowed either, as all elements must be assigned a type. Therefore, `xsd:mixed` cannot occur in a BSDL Schema. Elements with no type, expressed by means of `xsd:any`, `xsd:anyType` or `xsd:anySimpleType` are not allowed either.

Only a limited number of data types that exist in XML Schema are allowed in BSDL. Only those for which a binary representation is possible, are permitted. For instance, `xsd:integer` is not allowed, as the number of bytes needed for representing such an element is not specified. An element of type `xsd:int` is allowed, because in this case, the number of bytes is fixed. In Table 3.1, all data types that belong to XML Schema that are allowed to occur in a BSDL Schema, are shown. Note that some of these data types do not have an a priori length. If such a data type occurs in a BSDL Schema, BinToBSD will need information belonging to the BSDL-2 Extensions for determining the actual length. This problem does not occur for BSDToBin as the length of the data becomes clear when evaluating the content of the element.

In addition, other data types can be defined in a BSDL Schema, by restricting the range of one of the existing unsigned data types, using the `xsd:maxExclusive` restricting mechanism of XML Schema. Listing 3.1 shows how we can define a new data type that consists of exactly 5 bits.

Listing 3.1: Example of the use of xsd:maxExclusive in BSDL.

```
<!-- the following data type exists of 5 bits -->
<xsd:simpleType name="a5bitsToken">
  <xsd:restriction base="xsd:unsignedByte">
    <xsd:maxExclusive value="32"/>
  </xsd:restriction>
</xsd:simpleType>
```

Table 3.1: Overview of all XML Schema data types that are allowed in BSDL, together with their length, expressed in bytes.

data type	length
`xsd:string`	unlimited
`xsd:float`	4
`xsd:double`	8
`xsd:hexBinary`	unlimited
`xsd:base64Binary`	unlimited
`xsd:long`	8
`xsd:int`	4
`xsd:short`	2
`xsd:byte`	1
`xsd:unsignedLong`	8
`xsd:unsignedInt`	4
`xsd:unsignedShort`	2
`xsd:unsignedByte`	1

BSDL-1 Extensions

The Schema for BSDL-1 Extensions defines two new data types: `byteRange` and `bitstreamSegment`, both used for referring to fragments of the original bitstream. Elements of the `byteRange` data type consist of two non-negative integer values. The first value refers to a location in the current bitstream, represented as a byte offset, where the fragment begins, and the second value indicates the length, in bytes, of this fragment. A `bitstreamSegment` element has the same function, but elements of this data type have a `start` and a `length` attribute. This data type only exists for compatibility with the gBSD standard, also part of MPEG-21 Digital Item Adaptation, that is discussed in Sect. 3.3.

The bitstream where a `byteRange` or a `bitstreamSegment` refers to, is declared by means of the `bitstreamURI` attribute. It can occur in the root element of a bitstream description, thus providing a default value for that description, but can also occur in a `byteRange` or a `bitstreamSegment`, thus overwriting the default location of the referred bitstream.

A final extension defined in the Schema for BSDL-1 Extensions is the `ignore` attribute. Elements in a bitstream description that carry this attribute will be ignored by the BSDToBin process if its value is set to *true*, either in the bitstream description itself, or in the BSDL Schema by setting its default value to *true*. This way, it is easy to add metadata information to a bitstream descrip-

tion in such a way that it does not influence the generation of the (adapted) bitstream itself.

BSDL-2 Extensions

The most important extension in BSDL-2 is the use of XPath[4] expressions. Sometimes, the presence, the number of occurrences or the length of a particular element in a bitstream depends on the value of another element of the bitstream. Such situations can be expressed in a BSDL Schema. The BinToBSD process needs these expressions for generating the correct bitstream description.

When the number of occurrences of a particular element depends on the value of another element, this can be expressed by using the `nOccurs` attribute, that can contain an XPath expression. If it is absent, its value is supposed to be 1. In order to make sure that the BSDL Schema is a correct XML Schema for the produced bitstreams, the values for `minOccurs` and `maxOccurs` must correspond with all possible values for `nOccurs`.

Often, elements of a bitstream are conditional. In a BSDL Schema, this is expressed by means of the `if` or the `ifNext` attribute. An element containing an `if` attribute only occurs in the bitstream description if the XPath expression in the attribute evaluates to *true*. An `ifNext` attribute contains a hexadecimal value or a range of hexadecimal values (2 values, separated by a dash), as is shown in Listing 3.2. In this case the element will only occur if the following bytes in the stream correspond with the value or range of values mentioned in the `ifNext` attribute. As it was the case for `nOccurs`, the `minOccurs` and `maxOccurs` attributes should have appropriate values, if we want to be able to use the BSDL Schema as a valid XML Schema.

Listing 3.2: Example of the use of ifNext in a BSDL Schema.

```
<xsd:element name="elem" type="bt:b3" bs2:ifNext="20-FF"
    minOccurs="0" maxOccurs="unbounded"/>
<!-- elem element occurs repeatedly
     until the following 3 bits are all zero -->
<xsd:element name="stop" type="bt:b3" fixed="0"/>
```

We deliberately omitted the namespace declarations of BSDL, for reasons of compactness. The same goes for the other BSDL and gBSD fragments of this chapter. For completeness, all relevant namespaces and the prefixes we used for them in this chapter, can be found in Table 3.2.

[4] XPath is a language for selecting fragments of an XML document.

Table 3.2: Namespaces for BSDL and gBSD.

specification	namespace	prefix
BSDL-1	urn:mpeg:mpeg21:2003:01-DIA-BSDL1-NS	`bs1`
BSDL-2	urn:mpeg:mpeg21:2003:01-DIA-BSDL2-NS	`bs2`
gBSD	urn:mpeg:mpeg21:2003:01-DIA-gBSD-NS	`gbsd`

Another attribute defined in the Schema for BSDL-2 Extensions is the `rootElement` attribute in the `xsd:schema` element. This attribute tells the BinToBSD process which element of the schema should be used as the root element of the bitstream description.

In BSDL-1, some XML Schema data types can have an unlimited representation size, as can be seen in Table 3.1. The same goes for the `byteRange` data type. In these cases, the BinToBSD process needs additional information to know how long the actual elements are, otherwise all remaining bytes in the bitstream are considered to be part of that element. Because XML Schema does not support such constructs directly, the use of `xsd:annotation/xsd:appinfo` inside an `xsd:restriction` element is needed.

One of the mechanisms we can use for limiting the range of unlimited data types is the `length` attribute, that can contain an XPath expression determining the number of bytes the element consists of. In Listing 3.3, an example of the use of the `length` attribute to determine the length of a `byteRange` data type is shown: the number of bytes in the second field corresponds with the value of the first field minus 2. The `length` attribute cannot be used in combination with `xsd:length`, `startCode` or `endCode` constructions.

Listing 3.3: BSDL fragment in which the length (in bytes) of the second element is determined by the value of the first element.

```
<xsd:element name="header">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="firstfield" type="xsd:unsignedShort"/>
      <xsd:element name="secondfield">
        <xsd:simpleType>
          <xsd:restriction base="bs1:byteRange">
            <xsd:annotation>
              <xsd:appinfo>
                <bs2:length value="../this:firstfield - 2"/>
              </xsd:appinfo>
            </xsd:annotation>
          </xsd:restriction>
```

```
        </xsd:simpleType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Another way of determining the length of an unlimited data type is the use of `startCode` or `endCode` types. When the length of an element is restricted using these constructions, the BinToBSD process looks for the next occurrence of the hexadecimal pattern or range that is mentioned in the `value` attribute of the `startCode` or `endCode` element. When `startCode` is used, the pattern does not belong to the current element, in the case of an `endCode`, it does. An example of the use of start codes in BSDL is given in Listing 3.4. The end of both payloads is determined by the start code of the following element, which can either fall within the given range (from `0000` up to `0010`) or correspond with a fixed value (in this case `0020`).

Listing 3.4: Example of the use of startCode in a BSDL Schema.

```
<xsd:simpleType name="PayloadType">
  <xsd:restriction base="bs1:byteRange">
    <xsd:annotation><xsd:appinfo>
      <bs2:startCode value="0000-0010"/>
      <bs2:startCode value="0020"/>
    </xsd:appinfo></xsd:annotation>
  </xsd:restriction>
</xsd:simpleType>
<xsd:element name="init" type="bt:b8"/>
<xsd:element name="payload_1" type="PayLoadType"/>
<xsd:element name="marker" type="bt:b16"/>
<xsd:element name="payload_2" type="PayLoadType"/>
```

A final construction that belongs to BSDL-2 is the `ifUnion` construction. This is comparable with the well known switch/case constructions in most programming languages, and is used when an element of a BSDL Schema is defined using `xsd:union`. In that case, one type from a list of types, enumerated in the `memberTypes` attribute, is allowed to occur.

In order to be able to select the appropriate type, BinToBSD needs the information that occurs in the `ifUnion` elements. The index of the first `ifUnion` element for which the XPath expression evaluates to *true* will determine which element will be selected. If all n expressions evaluate to *false*, the member type with position $n + 1$ is selected. This way, this element is actually treated as a default type.

The following example (Listing 3.5) should make the use of this construction easier to understand. When the first expression evaluates to true, type `bt:b1` is selected. If all expressions are false, type `bt:b4` is selected.

Listing 3.5: Example of the use of xsd:union combined with ifUnion elements in a BSDL Schema.

```
<xsd:element name="elt1">
  <xsd:simpleType>
    <xsd:union memberTypes="bt:b1 bt:b2 bt:b3 bt:b4">
      <xsd:annotation>
        <xsd:appinfo>
          <bs2:ifUnion value="../pref:someOtherElt &lt; 2"/>
          <bs2:ifUnion value="../pref:someOtherElt &lt; 4"/>
          <bs2:ifUnion value="../pref:someOtherElt &lt; 8"/>
        </xsd:appinfo>
      </xsd:annotation>
    </xsd:union>
  </xsd:simpleType>
</xsd:element>
```

3.3 Generic Bitstream Syntax Description

One drawback of BSDL is that, even though it makes abstraction of the bitstreams that are used, and therefore is generic to some extent, the BSDToBin process needs to know the BSDL Schema that is in use, as can be seen from Fig. 3.2. As a consequence, BSDToBin needs to know all Schemas for all bitstream descriptions it has to be able to process, and therefore is not truly format independent. This can cause problems in case BSDToBin has to be executed on a constrained device. For this reason, the generic Bitstream Syntax Description, gBSD [50, 57], was developed. This format is also standardized as part of MPEG-21 Digital Item Adaptation. In Fig. 3.3, a data flow diagram is presented for the gBSD framework, in a similar way as in Fig. 3.2. The most important difference is that there is no longer a format specific BSDL Schema in use.

In gBSD, a bitstream description also describes a high-level structure of the logical units of a bitstream. In contrast to BSDL, the notation of these logical units, standardized in the gBSD Schema, is format independent. This is achieved by using only two types of elements that can occur in any generic bitstream description: `gBSDUnit` and `Parameter`.

A logical unit in the original bitstream is referred to by means of the `gBSDUnit` element. These elements refer to the original bitstream in a sim-

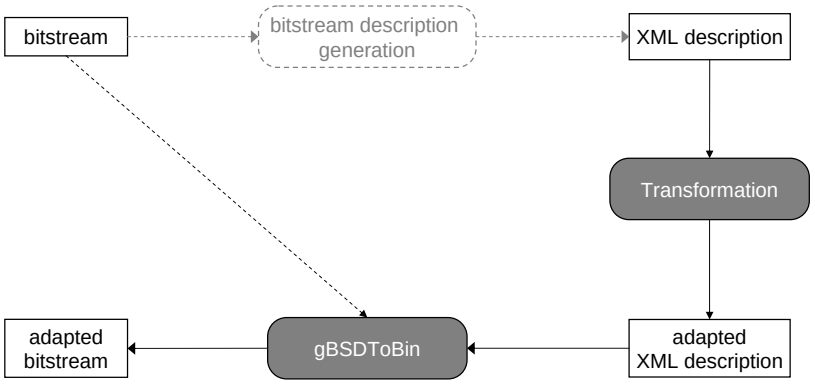


Figure 3.3: Data flow in the gBSD framework.

ilar way as the `byteRange` data type of BSDL. A `gBSDUnit` can contain a number of other `gBSDUnit` or `Parameter` elements, thus enabling the possibility of creating a hierarchical description of the bitstream structure.

A `gBSDUnit` can also contain a number of arguments. The `start` and `length` attributes point to the beginning and the length of the bitstream fragment referred to by the current representation. They only get processed when the unit has no child elements. A `syntacticalLabel` attribute can be used for inserting codec-dependent information, thus assisting a transformation engine that is aware of the coding format in use. The `marker` attribute is used for inserting semantic, application-specific information that can be used for adapting the bitstream.

The `Parameter` element is used for situations in which the value of a syntactical element of the bitstream (e.g., the resolution of an image or a video) might need to be changed during the adaptation process. A `Parameter` element contains the numerical value of this syntactical element, but also the data type that is used for representing this value in the bitstream. The data types that can be used are the same as for BSDL-1.

Three attributes are defined for determining the exact way in which a bitstream fragment is located. The `bitstreamURI` has the same meaning as in BSDL, and uses the BSDL-1 namespace. The `addressUnit` can be either `bit` or `byte`, and denotes the unit for the values of the `start` and `length` attributes. The value of the `addressMode` attribute is `Absolute` when the `start` attribute denotes the distance from the beginning of the bitstream, `Consecutive` can be used when it is assumed that each unit immediately follows the preceding unit (in this case, the `start` attribute is not needed), or

`Offset`, when the start `attribute` denotes the distance from the beginning of the previous unit.

The addressing attributes can occur in the top-level element of type `gBSDType` (usually a `Description` element of the Digital Item Adaptation namespace), thus denoting the default values. These values can be overwritten when any of the addressing attributes occurs in a `gBSDUnit` or a `Parameter`.

When looking at Fig. 3.3, we see that in the gBSD data flow there is no replacement for the BinToBSD process. As gBSD is truly format independent, there is no longer a bitstream syntax description for a class of bitstreams, as it is the case for BSDL by means of a BSDL Schema. As a consequence, there is no standardized way of generating bitstream descriptions, and different options are possible. For example, a video encoder might be extended in such a way that it no longer only produces a compressed bitstream, but in addition also produces a bitstream description that complies to the gBSD format. A second possibility is demonstrated in Sect. 3.4, where we translate a BSDL description into a gBSD description by means of a small piece of software written specifically for the conversion from BSDL to gBSD. Such software is format-dependent, as it needs to understand the syntax and semantics of the BSDL bitstream description.

3.4 Producing bitstream descriptions

In this section, we describe a number of example bitstream descriptions that comply to the BSDL or the gBSD specifications. We developed them ourselves during the research that lead to this thesis, in order to demonstrate the possibilities of high-level bitstream descriptions.

3.4.1 Uncompressed video in the YUV domain

A first example of a BSDL Schema is a very simple one, that was developed as an introduction exercise for people willing to discover the possibilities of BSDL. Even though such a BSDL Schema will probably never be used in practice, as uncompressed video is normally not used for the storage or transmission of a video signal[5], the example can be used for demonstrating the most important properties of BSDL.

[5] In a production environment, where any quality loss is to be avoided, YUV information is frequently used for exchanging digital video data. However, in our opinion, bitstream descriptions do not offer much additional value in such an environment.

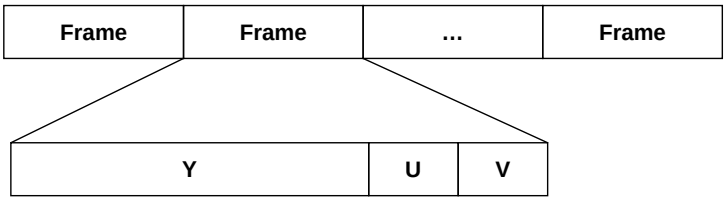


Figure 3.4: Structure of a YUV 4:2:0 bitstream.

The YUV file format

Before a video signal is compressed, the video data exists somehow in an uncompressed way: for every pixel in the image, a number of values are used for representing the actual color of that pixel. The most well known color space is probably the RGB color space, in which a color is decomposed into a red, a green and a blue channel. For each of these channels, a value is used for representing the intensity of the color.

In digital video compression, the YUV color space is more common. Again, a color is decomposed into three channels: the Y component represents the luminance of the color, and the chrominance (color information) is represented by the U and the V channel (sometimes, the names Cr and Cb are used). The term YUV 4:4:4 is used when one Y, U and V value is used for every pixel.

An interesting property of the Human Visual System is that it is much more sensitive to luminance information than chrominance information. Because of this property, the U and V data are often downsampled, thus drastically reducing the amount of data with nearly any quality loss. When for every horizontal pair of pixels only 1 U and 1 V value is stored, the term YUV 4:2:2 is used. The BSDL Schema that we present in this section is targeted at the YUV 4:2:0 format, in which case only 1 U and 1 V value for every block of 4 pixels, while for each pixel, one Y value remains.

The files that are commonly used for storing uncompressed YUV data do not have any header information. In order to be able to work with such files, information about the resolution and the frame rate has to be provided manually. The actual file is just a concatenation of bytes representing Y, U and V values. When a video sequence has a spatial resolution of N by M pixels, each frame consists of $N \times M$ luminance values, followed by $\frac{N \times M}{4}$ U values and $\frac{N \times M}{4}$ V values. Each Y, U or V value consists of exactly one byte. This structure is schematically represented in Fig. 3.4.

BSDL Schema

Listing 3.6 shows the entire BSDL Schema for YUV data. For convenience, we skipped the namespace declarations, as we also did for the remaining XML listings of this chapter. The `Sequence` element is defined as an unlimited number of `Frames`, and each Frame consists of one `luma` component, called `Y`, of 176×144 bytes, followed by the two `chroma` components, called `U` and `V` respectively, each having $\frac{176 \times 144}{4}$ bytes.

Listing 3.6: A BSDL Schema for uncompressed YUV 4:2:0 data having a resolution of 176x144 pixels.

```
<xsd:schema bs2:rootElement="yuv:Sequence">
    <!-- **** Root element declaration **** -->
    <xsd:element name="Sequence">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="yuv:Frame"
                    minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:attribute ref="xml:base"/>
        </xsd:complexType>
    </xsd:element>

    <!-- **** Frame declaration **** -->
    <xsd:element name="Frame">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="Y" type="yuv:luma"/>
                <xsd:element name="U" type="yuv:chroma"
                    minOccurs="0"/>
                <xsd:element name="V" type="yuv:chroma"
                    minOccurs="0"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <!-- **** Declaration of basic types **** -->
    <xsd:simpleType name="luma">
        <xsd:restriction base="bs1:byteRange">
            <xsd:annotation>
                <xsd:appinfo>
                    <bs2:length value="176 * 144"/>
                </xsd:appinfo>
            </xsd:annotation>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="chroma">
        <xsd:restriction base="bs1:byteRange">
```

```
            <xsd:annotation>
                <xsd:appinfo>
                    <bs2:length value="176 * 144 div 4"/>
                </xsd:appinfo>
            </xsd:annotation>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:schema>
```

In this Schema, the length of the luma and chroma parts is defined by means of the `bs2:length`. To improve readability, we used XPath expressions for calculating the actual number of bytes present in each part.

When we apply this BSDL Schema to the BinToBSD process, the bitstream description of a YUV file will look as is shown in Listing 3.7.

Listing 3.7: Bitstream description of a YUV file.

```
<Sequence bs1:bitstreamURI="stefan_qcif.yuv" xmlns="YUVqcif">
    <Frame>
        <Y>0 25344</Y>
        <U>25344 6336</U>
        <V>31680 6336</V>
    </Frame>
    <Frame>
        <Y>38016 25344</Y>
        <U>63360 6336</U>
        <V>69696 6336</V>
    </Frame>
    <!-- and so on -->
</Sequence>
```

Possible transformations

Even though uncompressed YUV data cannot be considered to be a scalable format, we can still apply some interesting transformations. A first transformation is the reduction of the frame rate to half of the original frame rate. It is fairly straightforward to implement such a transformation in XSLT, as is shown in Listing 3.8.

Listing 3.8: Transformation for reducing the frame rate using the bitstream description of a YUV sequence.

```
<xsl:stylesheet xmlns:yuv="YUVqcif">
  <xsl:output method="xml" indent="yes"/>
```

```
  <!-- Match all -->
  <xsl:template match="@*|node()">
    <xsl:copy>
      <xsl:apply-templates select="@*|node()"/>
    </xsl:copy>
  </xsl:template>
  <!-- Match even frame numbers  -->
  <xsl:template match="yuv:Frame[position() mod 2 = 0]">
    <!-- Nothing ! -->
  </xsl:template>
</xsl:stylesheet>
```

In this XSLT Stylesheet, all elements of the original XML document are copied, except those `frames` that have an even position in the sequence. When the resulting bitstream description is used for generating an adapted version of the sequence, it will contain only half of the frames. Note that when watching the sequence using a YUV player, the reduced frame rate will have to be entered.

Another interesting transformation that can be executed is the removal of the chroma information, thus ending up with a grayscale version of the video sequence. This transformation can be implemented using XSLT as shown in Listing 3.9. Here, all XML elements are copied, except the `U` and `V` elements, which contain chrominance information.

Listing 3.9: Transformation for removing the color information in the bitstream description of a YUV sequence.

```
<xsl:stylesheet xmlns:yuv="YUVqcif">
  <xsl:output method="xml" indent="yes"/>
  <!-- Match all -->
  <xsl:template match="@*|node()">
    <xsl:copy>
      <xsl:apply-templates select="@*|node()"/>
    </xsl:copy>
  </xsl:template>
  <!-- Match chroma components  -->
  <xsl:template match="yuv:U | yuv:V">
    <!-- Nothing ! -->
  </xsl:template>
</xsl:stylesheet>
```

3.4.2 MPEG-4 Visual

MPEG-4 Visual is a popular standard for video compression. It is used in well known multimedia products such as QuickTime, DivX and XViD. The BSDL Reference Software comes with a BSDL Schema that can be used for MPEG-4 Visual Elementary Streams. However, this schema was tailored towards the MPEG-4 reference software, and failed to produce bitstream descriptions for compressed bitstreams produced by other implementations.

Therefore, we corrected and expanded this schema so that it would be useful for other bitstreams compliant with the MPEG-4 Visual specification. Where possible, we also tried to simplify it. The resulting modified Schema is presented in Listing 3.10.

Listing 3.10: BSDL Schema for bitstreams that comply to the MPEG-4 Visual specification.

```
<xsd:schema targetNamespace="MPEG4" xmlns:mp4="MPEG4"
    bs2:rootElement="mp4:VOS">
  <!-- **** Visual Object Sequence declaration **** -->
  <xsd:element name="VOS">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="mp4:VisObj" minOccurs="0"
          maxOccurs="unbounded"/>
        <xsd:element name="VOS_endcode"
          type="mp4:StartCodeType" fixed="000001B1"
          minOccurs="0"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <!-- **** Visual Object declaration **** -->
  <xsd:element name="VisObj" bs2:ifNext="000001B0">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="VOS_code" type="mp4:StartCodeType"
          fixed="000001B0"/>
        <xsd:element name="profile_level" type="bt:b8"/>
        <xsd:element name="VisObj_code"
          type="mp4:StartCodeType" fixed="000001B5"/>
        <xsd:element name="VisObj_data">
          <xsd:simpleType>
            <xsd:restriction base="bs1:byteRange">
              <xsd:annotation><xsd:appinfo>
                <bs2:startCode value="00000100"/>
            </xsd:appinfo></xsd:annotation>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
```

```
        <xsd:element name="VidObj_code"
          type="mp4:StartCodeType" fixed="00000100"/>
        <xsd:element ref="mp4:VOL"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <!-- **** Video Object Layer declaration **** -->
  <xsd:element name="VOL">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="VOL_code" type="mp4:StartCodeType"
          fixed="00000120"/>
        <xsd:element name="VOL_data" type="mp4:PayloadType"/>
        <xsd:element ref="mp4:VOP" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <!-- **** VOP declaration **** -->
  <xsd:element name="VOP" bs2:ifNext="000001B6">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="VOP_code" type="mp4:StartCodeType"
          fixed="000001B6"/>
        <xsd:element name="VOP_coding_type" type="bt:b2"/>
        <xsd:element name="stuffing" type="bt:b6"/>
        <xsd:element name="payload" type="mp4:PayloadType"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <!-- **** Declaration of basic types **** -->
  <xsd:simpleType name="StartCodeType">
    <xsd:restriction base="xsd:hexBinary">
      <xsd:length value="4"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="PayloadType">
    <xsd:restriction base="bs1:byteRange">
      <xsd:annotation><xsd:appinfo>
        <bs2:startCode value="000001B0"/>
        <bs2:startCode value="000001B6"/>
      </xsd:appinfo></xsd:annotation>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:schema>
```

In Fig. 3.5, the syntactical structure of an MPEG-4 Visual Elementary Stream is shown. We have included this figure because of the similarities with the bitstream description we just showed: both in Listing 3.10 and in Fig. 3.5,

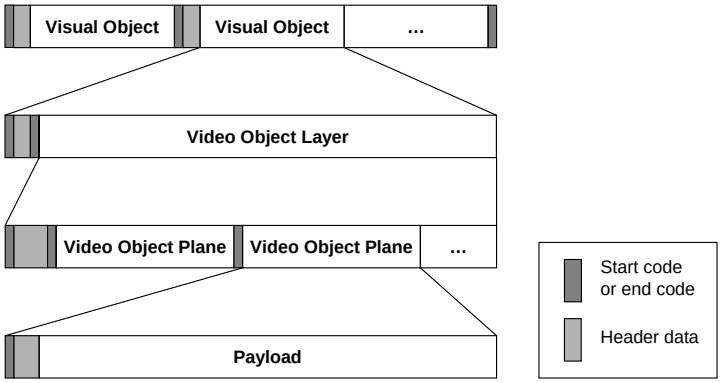


Figure 3.5: Structure of an MPEG-4 Visual bitstream.

it can be seen that an MPEG-4 bitstream can consist of a sequence of Visual Objects, and that such an object can correspond with a Visual Object Layer, that in turn is a sequence of Video Object Planes. Most of these elements begin with a start code and a certain amount of header information.

Listing 3.11 shows how a bitstream description of a particular MPEG-4 sequence may look like.

Listing 3.11: BSDL description of an MPEG-4 Visual bitstream.

```
<VOS bs1:bitstreamURI="news.cmp" xmlns="MPEG4">
  <VisObj>
    <VOS_code>000001B0</VOS_code>
    <profile_level>244</profile_level>
    <VisObj_code>000001B5</VisObj_code>
    <VisObj_data>9 1</VisObj_data>
    <VidObj_code>00000100</VidObj_code>
    <VOL>
      <VOL_code>00000120</VOL_code>
      <VOL_data>18 23</VOL_data>
      <VOP>
        <VOP_code>000001B6</VOP_code>
        <VOP_coding_type>0</VOP_coding_type>
        <stuffing>16</stuffing>
        <payload>46 5747</payload>
      </VOP>
      <VOP>
        <VOP_code>000001B6</VOP_code>
        <VOP_coding_type>1</VOP_coding_type>
        <stuffing>17</stuffing>
        <payload>5798 1119</payload>
```

```
      </VOP>
      <VOP>
        <VOP_code>000001B6</VOP_code>
        <VOP_coding_type>2</VOP_coding_type>
        <stuffing>16</stuffing>
        <payload>6922 150</payload>
      </VOP>
      <!-- etc. -->
    </VOL>
  </VisObj>
  <!-- multiple VisObj elements possible -->
</VOS>
```

An MPEG-4 Visual Elementary Stream can contain three different types of frames: an I-frame is an intra-coded frame that uses no temporal prediction, a P-frame is predicted from the previous I- or P-frame, and a B-frame is predicted from the previous and the following I- or P-frame. Because B-frames are never used for prediction, they can safely be removed from the bitstream, thus realizing a basic kind of temporal scalability. According to the MPEG-4 specification, the value of the *VOP_coding_type* field in the header of a frame is 0 in case of an I-frame, 1 for a P-frame and 2 for a B-frame. As a consequence, we can express the removal of B-frames in a bitstream description using XSLT as follows: all elements are copied, except `VOP` elements for which the `VOP_coding_type` is 2. This is shown in Listing 3.12.

Listing 3.12: Removal of B-frames in MPEG-4 Visual sequences.

```
<xsl:stylesheet xmlns:m="MPEG4">
  <xsl:output method="xml" indent="yes"/>
  <!-- Match all -->
  <xsl:template name="tplAll" match="@*|node()">
    <xsl:copy>
      <xsl:apply-templates select="@*|node()"/>
    </xsl:copy>
  </xsl:template>
  <!-- Match B_VOP - Overrides tplAll -->
  <xsl:template name="tplB_VOP"
      match="m:VOP[m:VOP_coding_type=2]">
    <!-- Nothing ! -->
  </xsl:template>
</xsl:stylesheet>
```

3.4.3 MPEG-4 FGS

In the previous section, we utilized the most commonly used subset of the MPEG-4 Visual specification, called Advanced Simple Profile. More complex coding schemes also exist, such as the Fine-Granular Scalability (FGS) [21], that we already discussed in Sect. 2.2. In MPEG-4 FGS, a bitstream consists of a base layer that is encoded using the techniques that are equally available in the Advanced Simple Profile. In addition, there is an enhancement layer that offers SNR scalability in a fine-grained way.

This fine-grained nature means that within each frame, the data of the enhancement layer can be truncated at any arbitrary point, as is described in Fig. 2.4. This way, a streaming server can dynamically adapt the bit rate of the transmitted video sequence to the actual bandwidth available between server and client at transmission time.

When MPEG-4 FGS is combined with temporal scalability, we talk about FGST. Because of the very flexible scalability properties, MPEG-4 FGST is a very interesting test case for both bitstream description mechanisms (BSDL and gBSD) that we introduced in this chapter. In the remainder of this section, we show how we can use BSDL for generating high-level bitstream descriptions of FGS and FGST bitstreams. We also show how we can generate gBSD descriptions from these BSDL descriptions.

BSDL Schema

Because the entire BSDL Schema for the case of MPEG-4 FGS is rather large, we only show a small fragment of it in Listing 3.13. The entire schema can be found in Appendix A. The most important difference with the Schema for MPEG-4 bitstreams as shown in Listing 3.10, is that the regular Video Object Planes are replaced with the definition of an `FGSVOP`. The most interesting property of such Video Object Planes, is that they consist of a number of `BitPlanes` that can be addressed easily by means of their start codes.

Listing 3.13: Fragment of the BSDL Schema for FGS enhancement layer streams.

```
<xsd:schema targetNamespace="MPEG4"
  bs2:rootElement="mp4:Bitstream">
    <!-- **** Root element declaration **** -->
    <xsd:element name="Bitstream">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="mp4:VOS" minOccurs="0"/>
                <xsd:element ref="mp4:VO"  minOccurs="0"/>
                <xsd:element ref="mp4:VOL" minOccurs="0"/>
```

```
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<!-- Skipped some definitions -->
<!-- **** VOL declaration **** -->
<xsd:element name="VOL" bs2:ifNext="00000120-0000012F">
    <xsd:complexType>
        <xsd:sequence>
          <xsd:element
            name="video_object_layer_start_code"
            type="mp4:StartCodeType"/>
          <xsd:element name="random_accessible_vol"
            type="bt:b1"/>
          <xsd:element
            name="video_object_type_indication"
            type="bt:b8"/>
          <!-- More header information -->
          <xsd:element ref="FGSVOP" minOccurs="0"
            maxOccurs="unbounded"
            bs2:if="mp4:video_object_type_indication
                    = 18"/>
      </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<!-- Skipped some definitions -->
<xsd:element name="FGSVOP" bs2:ifNext="000001B9">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="fgs_vop_start_code"
              type="mp4:StartCodeType" fixed="000001B9"/>
            <xsd:element name="vop_coding_type"
              type="bt:b2"/>
            <xsd:element name="modulo_time_base"
              type="bt:b1" fixed="1"
              minOccurs="0" maxOccurs="unbounded"
              bs2:ifNext="80-FF"/>
            <xsd:element name="modulo_time_base"
              type="bt:b1" fixed="0"/>
            <xsd:element name="marker_bit"
              type="bt:b1" fixed="1"/>
            <!-- More header information -->
            <xsd:element ref="mp4:BitPlane"
              maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<xsd:element name="BitPlane"
  bs2:ifNext="00000140-0000015F">
    <xsd:complexType>
```

```
            <xsd:sequence>
                <xsd:element name="fgs_bp_start_code"
                  type="mp4:StartCodeType"/>
                <xsd:element name="BP_data"
                  type="mp4:PayloadType" />
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <!-- Skipped some definitions -->
</xsd:schema>
```

In Table 3.3, we show a fragment of the specification of the syntax of FGS bitstreams. The relation between the MPEG-4 specification and the BSDL Schema for these bitstreams should be visible from this table.

Table 3.3: Fragment of the MPEG-4 specification for FGS bitstreams.

FGSVideoObjectPlane() {	**No. of bits**
fgs_vop_start_code	32
fgs_vop_coding_type	2
do {	
modulo_time_base	1
} while (modulo_time_base != '0')	
marker_bit	1
/* skipped some header information */	
if (nextbits_bytealigned () == fgs_bp_start_code) {	
while(nextbits_bytealigned() != '000 0000 0000 0000 0000 0000' || nextbits_bytealigned () == fgs_bp_start_code) {	
if (start_of_bit_plane())	
fgs_bp_start_code	32
else { /* not relevant here */	
}	
fgs_macroblock()	
}	
next_start_code()	
}	
}	

In Listing 3.14, we show the most interesting part of an FGS bitstream description. We removed most of the header information, so that the bitplane

information is more visible. Again, a more complete example of a bitstream description can be found in Appendix A.

Listing 3.14: Bitstream description generated using the BSDL Schema for MPEG-4 FGS.

```
<Bitstream bs1:bitstreamURI="stockholm_fgs.cmp" xmlns="MPEG4">
    <VOS>
      <!-- Skipped -->
    </VOS>
    <VO>
      <!-- Skipped -->
    </VO>
    <VOL>
        <video_object_layer_start_code>
          00000121
        </video_object_layer_start_code>
        <random_accessible_vol>1</random_accessible_vol>
        <video_object_type_indication>
          18
        </video_object_type_indication>
        <fgs_layer_type>1</fgs_layer_type>
        <!-- Skipped some header information -->
        <FGSVOP>
            <fgs_vop_start_code>000001B9</fgs_vop_start_code>
            <vop_coding_type>0</vop_coding_type>
            <modulo_time_base>0</modulo_time_base>
            <marker_bit>1</marker_bit>
            <!-- Skipped some header information -->
            <BitPlane>
                <fgs_bp_start_code>00000140</fgs_bp_start_code>
                <BP_data>31 176</BP_data>
            </BitPlane>
            <BitPlane>
                <fgs_bp_start_code>00000141</fgs_bp_start_code>
                <BP_data>211 3945</BP_data>
            </BitPlane>
            <BitPlane>
                <fgs_bp_start_code>00000142</fgs_bp_start_code>
                <BP_data>4160 9742</BP_data>
            </BitPlane>
            <BitPlane>
                <fgs_bp_start_code>00000143</fgs_bp_start_code>
                <BP_data>13906 14137</BP_data>
            </BitPlane>
        </FGSVOP>
        <!-- and so on -->
    </VOL>
</Bitstream>
```

Conversion to gBSD description

For the case of FGS bitstreams, we tested the possibility of converting a BSDL description of a base layer, an FGS enhancement layer and an additional FGST enhancement layer to a gBSD description using dedicated software.

A first problem that occurs in this conversion is that there are three original bitstreams, as shown in Fig. 2.5 (a), each producing its own bitstream description. In order to facilitate the adaptation process, it is more interesting that a single generic bitstream description is used for all these bitstreams.

Fortunately, this can easily be implemented in gBSD by using the `globalAddressInfo` attribute to denote the location of the original bitstream appropriately. When generating an adapted bitstream using such a bitstream description, the description will still have to be preprocessed, such that separate descriptions are produced, which can be used for producing each of the adapted bitstreams.

The entire data flow is shown in Fig. 3.6. The encoder produces three bitstreams, of which each produces a BSDL bitstream description. These descriptions are merged and transformed into a single gBSD description, that can be transformed by an adaptation engine. In order to reproduce the three adapted bitstreams, a filtering process is needed for generating the bitstream descriptions belonging to the correct bitstream. These descriptions can then be used by the gBSDToBin process for producing the appropriate bitstreams.

The Java code for merging three BSDL bitstream descriptions for FGS bitstreams into a single gBSD bitstream description is too exhaustive to show here; interested readers can find it in Appendix A. Listing 3.15 shows how the resulting gBSD bitstream description looks like. The meaning of the information that is present in the `marker` attributes of the `gBSDUnits` is not important right now, but will be explained in the next chapter.

Listing 3.15: Bitstream description in gBSD format of MPEG-4 FGS streams.

```
<DIA xmlns="urn:mpeg:mpeg21:2003:01-DIA-NS">
  <Description xsi:type="gBSDType"
      gbsd:addressUnit="byte" gbsd:addressMode="absolute"
      gbsd:globalAddressInfo="stockholm.cmp">
    <gbsd:gBSDUnit marker=":parcel:0" start="0" length="56155">
      <gbsd:gBSDUnit marker=":label:(0,0) :fps:15.0 :kbps:56"
        start="0" length="29"
        globalAddressInfo="stockholm.cmp"/>
      <gbsd:gBSDUnit marker=":label:(0,0)"
        start="0" length="18"
        globalAddressInfo="stockholm_fgs.cmp"/>
      <gbsd:gBSDUnit marker=":label:(0,0)"
        start="0" length="28"
```

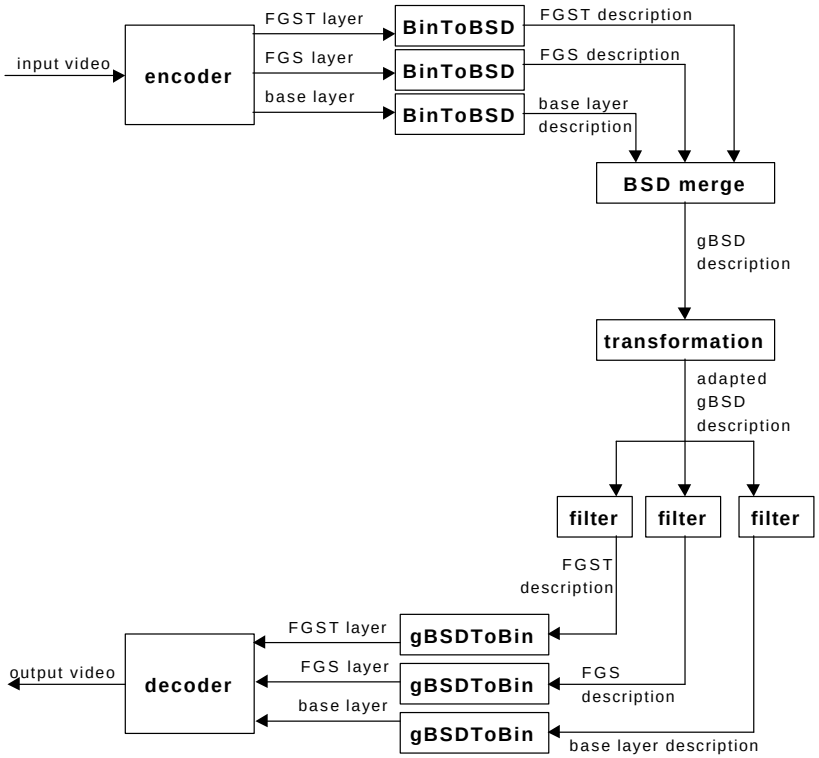


Figure 3.6: Adapting FGS bitstreams using gBSD.

```
  globalAddressInfo="stockholm_e.cmp"/>
<gbsd:gBSDUnit marker=":label:(0,0)"
  start="29" length="10321"/>
<gbsd:gBSDUnit marker=":label:(0,1) :fps:15.0 :kbps:73"
  start="18" length="9"
  globalAddressInfo="stockholm_fgs.cmp"/>
<gbsd:gBSDUnit marker=":label:(0,1)"
  start="27" length="180"
  globalAddressInfo="stockholm_fgs.cmp"/>
<gbsd:gBSDUnit marker=":label:(0,2) :fps:15.0 :kbps:176"
  start="207" length="3949"
  globalAddressInfo="stockholm_fgs.cmp"/>
<gbsd:gBSDUnit marker=":label:(0,3) :fps:15.0 :kbps:378"
  start="4156" length="9746"
  globalAddressInfo="stockholm_fgs.cmp"/>
<gbsd:gBSDUnit marker=":label:(0,4) :fps:15.0 :kbps:437"
  start="13902" length="14141"
  globalAddressInfo="stockholm_fgs.cmp"/>
<gbsd:gBSDUnit marker=":label:(1,0) :fps:30.0 :kbps:63"
```

```
        start="28" length="454"
        globalAddressInfo="stockholm_e.cmp"/>
      <gbsd:gBSDUnit marker=":label:(1,1) :fps:30.0 :kbps:81"
        start="482" length="101"
        globalAddressInfo="stockholm_e.cmp"/>
      <gbsd:gBSDUnit marker=":label:(1,2) :fps:30.0 :kbps:192"
        start="583" length="610"
        globalAddressInfo="stockholm_e.cmp"/>
      <gbsd:gBSDUnit marker=":label:(1,3) :fps:30.0 :kbps:432"
        start="1193" length="3860"
        globalAddressInfo="stockholm_e.cmp"/>
      <gbsd:gBSDUnit marker=":label:(1,4) :fps:30.0 :kbps:604"
        start="5053" length="9586"
        globalAddressInfo="stockholm_e.cmp"/>
      <gbsd:gBSDUnit marker=":label:(0,0)"
        start="10350" length="3093"/>
      <gbsd:gBSDUnit marker=":label:(0,1)"
        start="28043" length="9"
        globalAddressInfo="stockholm_fgs.cmp" />
      <!-- and so on -->
    </gbsd:gBSDUnit>
    <gbsd:gBSDUnit marker=":parcel:1"
      start="56155" length="56363">
      <!-- and so on -->
    </gbsd:gBSDUnit>
    <!-- and so on -->
  </Description>
</DIA>
```

A straightforward way of adapting this kind of bitstream descriptions is by using XSLT for removing those bitplanes and temporal levels that are not needed, by means of the information available in the `marker` attributes of the `gBSDUnit` elements. This is shown in Listing 3.16. Note that, when we want to fully exploit the features of FGS, we can also modify the `length` attribute of the highest bitplane that appears in the adapted bitstream description.

Listing 3.16: Removing all but the first two bitplanes and all temporal levels of an FGS bitstream description.

```
<xsl:stylesheet>
  <xsl:output method="xml" indent="yes"/>
  <!-- Match all -->
  <xsl:template name="tplAll" match="@*|node()">
    <xsl:copy>
      <xsl:apply-templates select="@*|node()"/>
    </xsl:copy>
  </xsl:template>
```

```
  <!-- Drop certain blocks - Overrides tplAll -->
  <xsl:template
      match="gbs:gBSDUnit[contains(@marker,':label:')]">
    <xsl:choose>
      <xsl:when test="contains(@marker,':label:(0,0)')">
        <xsl:copy>
          <xsl:apply-templates select="@*|node()"/>
        </xsl:copy>
      </xsl:when>
      <xsl:when test="contains(@marker,':label:(0,1)')">
        <xsl:copy>
          <xsl:apply-templates select="@*|node()"/>
        </xsl:copy>
      </xsl:when>
      <xsl:otherwise>
        <!-- Nothing -->
      </xsl:otherwise>
    </xsl:choose>
  </xsl:template>
</xsl:stylesheet>
```

The resulting bitstream description cannot be used directly for generating an adapted bitstream, because the gBSD description contains the descriptions of multiple bitstreams. Therefore, some preprocessing is needed. This can be achieved for the first enhancement layer as is shown in Listing 3.17.

Listing 3.17: Filtering the entire gBSD description such that only the description of one bitstream remains.

```
<xsl:stylesheet>
  <xsl:output method="xml" indent="yes"/>
  <!-- Match all -->
  <xsl:template name="tplAll" match="@*|node()">
    <xsl:copy>
      <xsl:apply-templates select="@*|node()"/>
    </xsl:copy>
  </xsl:template>
  <!-- Only copy elements belonging to enhancement layer -->
  <xsl:template
      match="gbs:gBSDUnit[contains(@marker,':label:')]">
    <xsl:if test="@globalAddressInfo = 'stockholm_fgs.cmp'">
      <xsl:copy>
        <xsl:apply-templates select="@*|node()"/>
      </xsl:copy>
    </xsl:if>
  </xsl:template>
</xsl:stylesheet>
```

3.4.4 Bitstream descriptions for other formats

The list of examples discussed in this chapter for coding formats for which bitstream descriptions can be generated, is of course not complete. We only discussed those cases for which we delivered an original contribution.

Within the Multimedia Lab, other BSDL Schemas were developed as well. Davy De Schrijver and Chris Poppe did this for a fully scalable wavelet-based video codec [59]. Wesley De Neve created a BSDL Schema for the recent and successful H.264/AVC standard for video coding, also known as MPEG-4 Part 10 [14, 33, 34].

In other publications, we can find bitstream descriptions in both BSDL and gBSD formats for the JPEG2000 still image coding standard, that allows multiple types of scalability concurrently [57]. Lafruit et al. use BSDL for implementing view-dependent transmission of texture images for 3-D scenes using the scalable MPEG-4 Visual Texture Coding (VTC) [61]. Depending on the angle and the distance of the viewer, the level of detail and quality is dynamically selected and triggers the transformation of the BSDL description of the texture information. Di Giacomo et al. apply BSDL for the adaptation of human facial and body animations [62].

3.5 Related work

BSDL and gBSD are standardized frameworks for using bitstream descriptions, but are not the only languages for working with such descriptions. In this section, we briefly discuss some other frameworks that have similar properties and goals.

3.5.1 FLAVOR and XFLAVOR

FLAVOR, the Formal Language for Audio-Visual Object Representation [63], is originally targeted at the automatic generation of parsers for multimedia bitstreams. This is achieved by describing the syntactical structure of a bitstream in FLAVOR code, which can be translated by the FLAVOR compiler into Java or C++ code to be integrated in a parser, decoder, ... for that type of bitstreams.

The FLAVOR code has a similar functionality as a BSDL Schema, but is in itself not targeted at the generation of bitstream descriptions. In the introduction of this chapter, we mentioned that the preferred language for bitstream descriptions is XML. This was recognized by the FLAVOR developers. As an extension to the FLAVOR framework, XFLAVOR adds support for XML [64, 65].

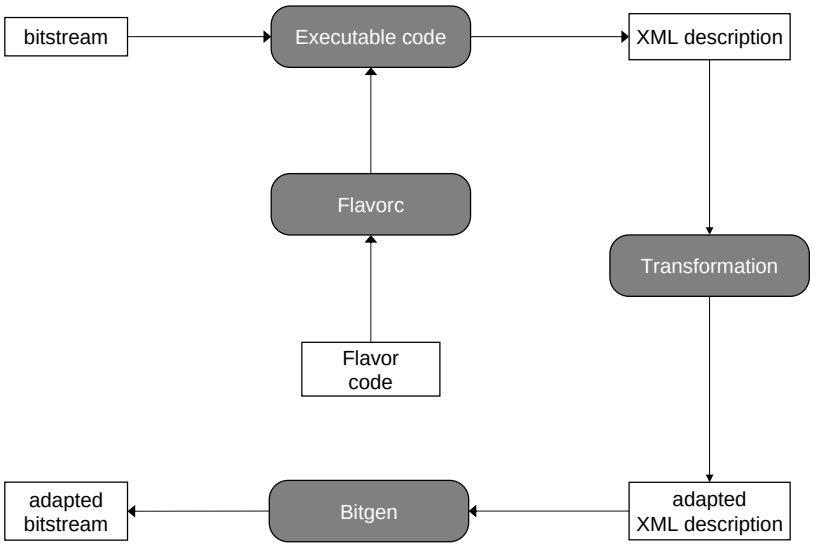


Figure 3.7: Data flow in the XFLAVOR framework.

XFLAVOR enables the creation of a bitstream description in XML and the generation of a bitstream from a (possibly adapted) bitstream description. The entire data flow is represented in Fig. 3.7. The FLAVOR code is compiled by the FLAVOR compiler (marked *Flavorc* in Fig. 3.7) into executable code that can produce a bitstream description in XML. This description can be transformed, prior to the regeneration of the adapted bitstream by means of the *Bitgen* software.

Note that, in contrast with the BSDL data flow (see Fig. 3.2), there is no connection between the original bitstream and the bitstream generation process. This is because the bitstream description contains all information needed for reproducing the bitstream. As a consequence, this description is not as compact as a bitstream description in BSDL: a FLAVOR bitstream description is even larger than the original bitstream.

An additional property of XFLAVOR is the automatic generation of an XML Schema for the bitstream descriptions generated with XFLAVOR. Note the difference between this schema and a BSDL Schema: a BSDL Schema is generated manually and is used for producing BSDL bitstream descriptions, whereas the XML Schema in XFLAVOR is automatically generated from the FLAVOR code.

3.5.2 SSM

In [66, 67], Mukherjee and Said present the Structured Scalable Meta-formats framework (SSM), a methodology for the representation and adaptation of scalable content. This methodology is based on a model of the structure of the most common types of scalable bitstreams.

The SSM framework also defines an XML description of the structure of a scalable bitstream, that reflects the model defined by the SSM framework. Because of the tight coupling between the SSM model and its language for expressing bitstream descriptions, this language is less flexible than BSDL or gBSD, but at the same time also less ambiguous: a transformation implementation that is familiar with the SSM model can transform any description in a meaningful way. This way, a truly format agnostic transformation and adaptation process can be implemented. This is one step further than gBSD, where the bitstream generation process is format agnostic, but the transformation is not: the transformation implementation must understand the meaning of the contents of the `marker` and `syntacticalLabel` attributes when executing a meaningful transformation.

3.6 Conclusions and original contributions

In this chapter, we introduced the added value that can be offered in multimedia distribution environments when using bitstream descriptions, preferably in XML. Such bitstream descriptions can help in developing more generic adaptation engines, and allow the coupling of the actual content of a multimedia stream with additional metadata, that can be used for steering the adaptation process.

We described into detail two languages for bitstream descriptions that are standardized in MPEG-21 Digital Item Adaptation, in particular the Bitstream Syntax Description Language (BSDL) and the generic Bitstream Syntax Description language (gBSD).

Our own original contributions introduced in this chapter consist of the definition of BSDL Schemas for uncompressed raw video data in the YUV format, MPEG-4 Visual Elementary Streams and bitstreams following the MPEG-4 Fine-Granular Scalability. For this last case, we also showed how we can use such BSDL bitstream descriptions for generating format-independent gBSD bitstream descriptions.

For each of these classes of bitstreams, we gave at least one example of a possible transformation that can be executed in a meaningful way on a bitstream description. These transformations were implemented using XSLT and

exploited the scalable aspects of the bitstreams. After transformation, the bitstream descriptions correspond with bitstreams that are more compact, but offer a lower visual quality.

In this chapter, we also looked at bitstream descriptions for other bitstream formats, and we briefly discussed two related methodologies for generating and using bitstream descriptions.

Three important disadvantages of using bitstream descriptions were not mentioned. First, the generation of a bitstream description, its transformation and the generation of the adapted bitstream require an additional computational cost that can become significant in some scenarios [59]. Another disadvantage is that the generation and processing of bitstream descriptions does not fit nicely in streaming scenarios, where the acceptable delay is limited. Solutions to this problem are explored in [68]. A final disadvantage is that the size of the bitstream descriptions can become rather large with respect to the bitstream itself. This problem can be reduced by means of one of the existing frameworks for XML compression [69], such as MPEG-7 BiM (Binary format for MPEG-7) [44].

Our contributions in the domain of bitstream descriptions can be found in the following publications.

1. Sam Lerouge, Boris Rogge, Dimitri Van De Ville, Rik Van de Walle, and Jan Van Campenhout. An XML-based framework for content adaptation. In *Proceedings of Euromedia*, pages 175–179, Modena, Italy, April 2002.

2. Wesley De Neve, Sam Lerouge, Peter Lambert, and Rik Van de Walle. A performance evaluation of MPEG-21 BSDL in the context of H.264/AVC. In *Proceedings of SPIE 2004: Signal and Image Processing and Sensors*, volume 5558, pages 555–566, Denver, Colorado, USA, August 2004.

3. Davy De Schrijver, Chris Poppe, Sam Lerouge, Wesley De Neve, and Rik Van de Walle. MPEG-21 bitstream syntax descriptions for scalable video codecs. *Multimedia Systems*, to appear.

Chapter 4

An abstract model for scalable bitstreams

4.1 Introduction

As described in the previous chapters, scalable video coding is a key technology for enabling video communication in constrained, heterogeneous environments. The process of automatically selecting the version offering most quality within the limits forced by the capabilities of the requesting device, is usually called content negotiation, which was the most important research topic leading to this thesis.

During research on content negotiation procedures for transmitting scalable video bitstreams, we discovered some unexpected complexities [3, 6]. We were convinced it would be useful to define an abstract model describing the properties of scalable bitstreams, and to use this model as a tool for reasoning about applications using such bitstreams. This would help us in formally defining a content negotiation process in a mathematical way: a process that finds the optimal adaptation operation, that maximizes the utility, the overall user's satisfaction, while meeting all constraints imposed by the system.

Important advantages of formalizing the description of data structures (in our case scalable bitstreams) and applications processing these structures, are the possibility to reason about these applications before actually implementing them, but also the possibility of validating of the correctness of implementations. Some formalisms, such as the Unified Modeling Language (UML) [70] can even be used to some extent for automatic code generation.

We were not the only ones that felt the need for formalizing the process of multimedia content negotiation and adaptation. In [71], Chang and Vetro propose a formalization of video adaptation by defining the fundamental entities

and concepts in such a scenario, in particular resources, adaptation operations and resulting utilities. Using these concepts and their relations, they describe a video adaptation process as a constrained optimization problem.

The Structured Scalable Meta-formats framework (SSM) [66, 67] that we already mentioned in the chapter on bitstream descriptions, developed by a team from HP Labs, is a methodology for the representation and adaptation of scalable content. It is based on a model of the structure of scalable bitstreams. The model that we present in this chapter, is strongly influenced by the SSM model. As opposed to the work of Mukherjee and Said, we aimed at defining the model in a more formal way. In this approach, it is possible to describe any kind of application that is based on the properties of scalable bitstreams, as we show in Sect. 4.4.

4.2 The abstract model

4.2.1 Informal semantics

Before moving on to the actual definitions of the model, we explain the underlying principles of the model in a more informal way.

As we have mentioned in Chapter 2, a scalable bitstream consists of a base layer and one or more enhancement layers (see Fig. 2.2). More advanced algorithms for encoding scalable multimedia data allow different types of scalability at the same time. In our model, each type of scalability corresponds to one **scalability axis**. An entire bitstream can be split up into a sequence of **parcels** (sometimes also called adaptation units [72]). When talking about video coding, a parcel will typically consist of a number of frames; it is the unit of information for which adaptation can be applied.

Figure 4.1 schematically shows the structure of a parcel that consists of three scalability axes: a temporal, a spatial and an SNR axis. The temporal axis (the frame rate axis on the figure) consists of three levels, because we suppose three different frame rates are possible: 30, 15 and 7.5 frames per second. These levels correspond with a base layer and two enhancement layers in the temporal dimension. The bitstream consists of four bitplanes, therefore allowing four levels for the SNR axis. In addition, three different resolutions are possible (4CIF, CIF, and QCIF[1]). This is shown in the figure by the three levels along the resolution axis.

[1]CIF is the Common Intermediate Format, a resolution of 352 by 288 pixels that is frequently used in digital video coding. QCIF, or Quarter CIF, is obtained by taking the half of the horizontal and spatial resolution. 4CIF is obtained by doubling the horizontal and spatial resolution and corresponds approximately to the resolution of a television signal.

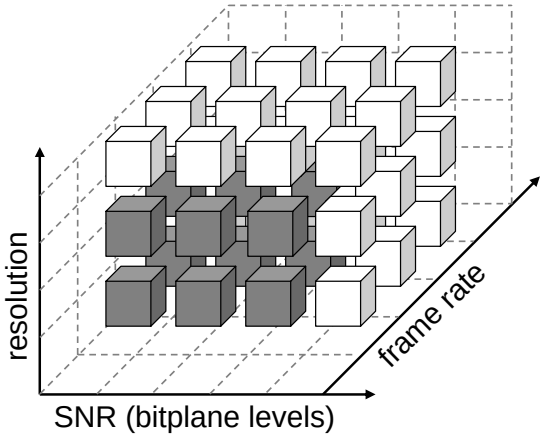


Figure 4.1: Graphical representation of the data blocks of a parcel that consists of three scalability axes.

In Fig. 4.1, for every possible combination of levels (that can be identified with a **label**) there is exactly one **data block**. This is not a requirement of our model: in the example that is shown in Listing 4.1, multiple data blocks carrying the same label exist within the same parcel. The gray data blocks in the figure are the data blocks that form one specific **version** of the parcel, in particular the one that offers a 15 frames per second sequence at CIF resolution, and that consists of 3 bitplanes.

4.2.2 Definitions

In this section, we introduce the definitions that we developed, and that produce the core of the abstract model for scalable bitstreams we developed.

A **bitstream** is a sequence of **parcels**. Such a parcel consists of a number of **data blocks**. Each data block has a **label**, which is an N-dimensional vector with non-negative integer components. N is the number of **scalability axes**; each component of a label can be considered to be a coordinate on the scalability axes of the corresponding parcel, and thus identifies to which layer the data block belongs to, regarding that scalability axis.

Definition 4.1. *The set of all **data blocks** is denoted D. The **label** of a data block $d \in D$ is represented as a vector $\hat{d}$, where $\hat{d} \in \{(x_0, x_1, \ldots, x_{N-1}) | x_i \in \mathbb{N}_0\}$.*

The actual data that is contained in the data block and that will be the interesting part for the decoder is called the **payload** and is just a sequence of

bits. The **size** of a data block is the number of bits in the payload of a block.

Definition 4.2. *The **payload** of a data block d is denoted $[d]$, where $[d] = (b_0, b_1, \ldots, b_{n-1})$, with $b_i \in \{0, 1\}$. The **size** of a data block $d \in D$ is denoted $|d|$, where $|d| = \#[d]$, and $\#$ is the operator that maps a vector to its number of components: if $[d] = (b_0, b_1, \ldots, b_{n-1})$, then $|d| = n$.*

In content negotiation, the decision about the adaptation of a scalable bitstream is not taken at the bitstream level, but at the parcel level. This way, the process is capable of reacting to changes in the usage environment. A **parcel** is a set of data blocks that are logically related: as we have said before, it is the unit of information for which adaptation can be applied. In a video stream, this is typically one frame or a group of sequential frames (such as a GOP, a Group of Pictures, in MPEG terminology).

Definition 4.3. *A **parcel** p is defined as a set of data blocks. P is the set of all possible parcels: $(\forall p \in P)(p \subset D)$.*

Within each parcel, there can be any number of **scalability axes**. The scalability axes determine in which way scalability can be applied to the bitstream. In scalable video coding, typical scalability axes are the temporal, bitplane (or SNR), and resolution axis. When a bitstream has a temporal scalability axis, it means that it can be adapted to versions having a lower frame rate.

The number of scalability axes in a parcel corresponds to the number of components of the label of each data block (denoted $\hat{d}$) in the parcel. It should be clear that this number is supposed to be the same for all data blocks that belong to the same parcel. This is stated in the next rule.

Rule 4.1. $(\forall p \in P)(\forall d \in p)(\#(\hat{d}) = S)$ *where S is the number of **scalability axes** in p.*

The number of **levels** of one scalability axis determines the number of options one has when using the scalability possibilities of the parcel along this axis. It corresponds to the number of available layers as in Fig. 2.2, when considering only one scalability axis.

Rule 4.2. *The number of **levels** of the i-th scalability axis of parcel p is denoted p^i, where $p^i = \max_{d \in P} (\hat{d}_i + 1)$, when $\hat{d}_i$ is the i-th component of vector $\hat{d}$.*

The next definition states that in our model, a scalable **bitstream** is a sequence of parcels.

Definition 4.4. *The set of all scalable **bitstreams** is denoted B, where $B = \{(p_0, p_1, \ldots) | p_i \in P\}$.*

According to this definition, a scalable bitstream $b \in B$ is a vector of parcels $p_i \in P$. This vector should be interpreted as a sequence of parcels, as we do not use the properties of n-dimensional vectors when looking at bitstreams: all relevant operations are performed at the parcel level.

Usually, the number of scalability axes and their levels would be the same for each parcel of a scalable bitstream. However, we chose not to force this requirement in our model, as it is not necessary for the applications we are thinking of. This way the model is still valid for scalable bitstreams that would not have a constant number of scalability axes and levels.

Now we are ready to define how we can generate a reduced bitstream from a scalable bitstream by selecting the appropriate data blocks, while dropping the other blocks.

First, we introduce the concept of a **version** of a parcel. A version of a parcel is in itself a parcel, a subset of the original parcel. It is identified by the original parcel and an N-dimensional vector, where N corresponds to the number of scalability axes of the original parcel.

Definition 4.5. *A **version** of a **parcel** p can be denoted $p(x_0, x_1, \ldots, x_{S-1})$, when S is the number of scalability axes of p. A particular version is composed as follows: $p(x_0, x_1, \ldots, x_{S-1}) = \{d \in p | (\forall i \in \{0, 1, \ldots, S-1\})(x_i \leq \hat{d}_i)\}$. Note that $p(x_0, x_1, \ldots, x_{S-1}) \subseteq p$.*

According to this definition, a version of a parcel is a subset of the original parcel, composed in such a way that it contains all the data blocks for which all of the levels are not greater than the corresponding level of the version itself.

For a better understanding, Fig. 4.2 shows a parcel containing two scalability axes, one having four levels, a second one having three levels. For each combination of levels, one data block is shown with its label. All blocks marked in gray belong to the same version. In a similar way, the gray blocks in Fig. 4.1 belong to the same version of a parcel that consists of three scalability axes.

The base layer of a parcel can be defined as the set of data blocks d of that parcel that have $\hat{d} = \overrightarrow{\mathbf{0}}$ as a label. According to Def. 4.5, the base layer of a parcel will be part of any possible version of that parcel. Because of this property, global header information should be part of the base layer, as this guarantees that this information will be part of all possible versions.

The **closure** of a parcel can now be defined as all versions that can be derived from a certain parcel. Note that the closure of a parcel is in itself a set of parcels, and all these parcels are subsets of the original parcel.

Definition 4.6. *The **closure** of a parcel $p \in P$ is denoted p^+, where $p^+ = \{q \in P | (\exists x_0, x_1, \ldots, x_{S-1})(q = p(x_0, x_1, \ldots, x_{S-1}))\}$.*

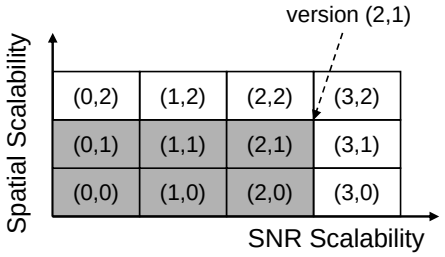


Figure 4.2: Representation of the data blocks of a parcel that consists of two scalability axes. The blocks marked in gray belong to version (2,1).

An important concept in the content negotiation process for scalable bitstreams is the notion of **properties**. A property is actually a function that maps a version of a parcel to some value.

Definition 4.7. *The set of all **properties** is denoted* $\mathbb{P}$*, where* $\mathbb{P} = \{f | f : P \mapsto \mathbb{R}\}$.

A useful example of a property is the $||.||$ operator applied to a parcel, which is defined as the size of all data blocks belonging to that parcel. It should be obvious that $||.|| \in \mathbb{P}$.

Definition 4.8. *The **size** of a parcel p is denoted* $||p||$*, and is defined as follows:* $p \in P \Rightarrow ||p|| = \sum_{d \in p} |d|$.

In the context of scalable video coding, other useful properties are the frame rate, the spatial resolution, and the visual quality of the individual frames, typically expressed using PSNR (Peak Signal-to-Noise Ratio) values. Note that some of these properties cannot be derived immediately from the bitstream structure as defined by the model, but can be supplied as additional metadata.

In Def. 4.7, we have defined properties as functions that can be applied to all possible parcels. In our examples, however, we only look at the properties of the parcels that are in the closure of one particular parcel.

4.3 Mapping existing coding formats onto the abstract model

A model does not offer any added value if there is no clear connection with the reality. In this section, we give some examples of how the model that we

just defined can be applied to some of the existing video coding schemes that were introduced in Chapter 2. For the case of fine-granularity scalability with temporal scalability, we also show how we can use a bitstream description framework such as gBSD for explicitly making the link between a bitstream and our model. We developed these examples in order to prove that our model corresponds well with existing structures for scalable video coding.

4.3.1 Fine-granularity scalability

According to the MPEG-4 standard, the base layer and enhancement layer data of an FGS video sequence are placed in separate streams. As a consequence, one should bear in mind that the payload of the data blocks of the base layer are located in a different stream than the payload of the other data blocks. Apart from that, this separation between streams is not a problem for our model.

According to the model, a scalable bitstream is considered to be a sequence of parcels. In the FGS case, we can choose to declare one parcel as corresponding with the data belonging to one VOP (one frame) in the video sequence. If this division is considered too fine, one could decide to place all frames of one GOP (Group of Pictures) in the same parcel.

The only possibility for exploiting the scalability of an FGS stream is in the SNR axis. In theory, an FGS stream can be cut off at any arbitrary position within a certain bitstream. Therefore, we could consider every single byte as a data block. However, it seems to be more convenient to group all data belonging to the same bitplane into one data block.

When an enhancement layer bitstream consists of four bitplanes in the current parcel, we can split up the two bitstreams into five data blocks for each VOP. The payload of the first data block, identified with label (0), is the data in the base layer that corresponds with the current VOP. The data block having label (1) is found in the first bitplane of the current VOP of the enhancement layer, the one with label (2) corresponds with the second bitplane, and so on.

Interesting properties of the different versions of an FGS parcel that can be used in content negotiation scenarios may be the size, the computational complexity and the estimated or calculated PSNR value (or some other distortion measure) of each possible version.

4.3.2 FGS with temporal scalability

When we consider an FGST sequence, things become a little more complicated because now there is a second scalability axis because of the temporal scalability that is available.

Suppose we start from a video sequence that has a frame rate of 30 fps when no data is dropped. When removing the FGST enhancement layer, a 15 fps sequence is left. The bitplane structure is the same as in the FGS example. As in the previous case, we can consider an entire GOP as one parcel. Another possibility would be to consider two successive frames as one parcel, as the only dependencies across the parcel borders are targeting the base layer, and the base layer is present in all versions.

As there are only two options in the temporal dimension, this axis has two levels. Just like in the previous example, the SNR axis consists of 5 levels.

We can split up the payload of the different streams as follows. The data blocks of the current parcel that consist of the base layer frames have label $(0, 0)$. Bitplane i in the FGS VOP has label $(0, i)$, as in the FGS case. The bitplanes of the FGST VOP have label $(1, i)$. When we split up the motion vector information of the FGST VOP from the rest of the VOP, we can consider this information as the payload of data block $(1, 0)$.

In Fig. 4.3, a graphical representation of the structure of such an FGST sequence is shown, along with the different available data blocks and their labels. The first two frames represent the same information as in Fig. 2.5 (a). In the following two frames, we marked the different bitplanes and motion vectors, and in the last two frames, the labels are shown of the data blocks corresponding with all parts of the frames.

As described in Chapter 3 when discussing bitstream descriptions, we developed a mechanism for generating a gBSD description of the structure of an FGST video sequence. In a first step, a BSDL description is generated, based on a BSDL schema for MPEG-4 FGS/FGST bitstreams. As the entire sequence consists of three bitstreams, we end up with three BSDL descriptions.

For the second step, we developed a DOM-based Java application that parses the existing descriptions, transforms them and merges them into one single gBSD description. During this step, information about properties that can be calculated immediately (in particular, the frame rate and the bit rate) is added to the bitstream description. The code for generating these descriptions can be found in Appendix A.

In a third phase, all versions are generated, and their PSNR values are calculated. This information is manually added to the existing gBSD description. Listing 4.1 shows how the bitstream description looks like after all these steps.

Listing 4.1: Example gBSD description of an FGST video sequence.

```
<dia:DIA>
  <dia:Description xsi:type="gBSDType"
      gbsd:addressUnit="byte" gbsd:addressMode="absolute"
```

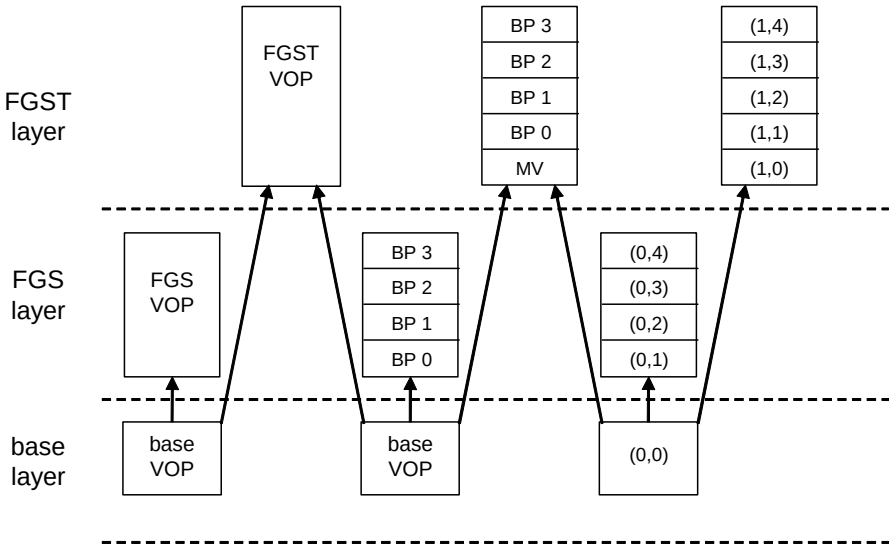


Figure 4.3: Representation of an FGST sequence in the model. The sequence consists of base layer VOPs, FGS VOPs and FGST VOPs. The FGS and FGST VOPs each consist of 4 bitplanes; FGST VOPs also contain motion vector information. In the last VOPs, the labels for the corresponding data blocks are shown.

```
    gbsd:globalAddressInfo="base.cmp">
  <gbsd:gBSDUnit marker=":parcel:0" start="0" length="56155">
    <gbsd:gBSDUnit marker=":label:(0,0) :fps:15.0 :kbps:449
      :psnr:33.6" start="0" length="29"/>
    <gbsd:gBSDUnit marker=":label:(0,0)"
      start="0" length="18"
      globalAddressInfo="fgs.cmp"/>
    <gbsd:gBSDUnit marker=":label:(0,0)"
      start="0" length="28"
      globalAddressInfo="fgst.cmp"/>
    <gbsd:gBSDUnit marker=":label:(0,0)"
      start="29" length="10321"/>
    <gbsd:gBSDUnit marker=":label:(0,1) :fps:15.0 :kbps:584
      :psnr:34.2" start="18" length="9"
      globalAddressInfo="fgs.cmp"/>
    <gbsd:gBSDUnit marker=":label:(0,1)"
      start="27" length="180"
      globalAddressInfo="fgs.cmp"/>
    <gbsd:gBSDUnit marker=":label:(0,2) :fps:15.0 :kbps:1410
      :psnr:37.5" start="207" length="3949"
      globalAddressInfo="fgs.cmp"/>
    <gbsd:gBSDUnit marker=":label:(0,3) :fps:15.0 :kbps:3029
```

```
        :psnr:43.0" start="4156" length="9746"
        globalAddressInfo="fgs.cmp"/>
      <gbsd:gBSDUnit marker=":label:(0,4) :fps:15.0 :kbps:3498
        :psnr:44.5" start="13902" length="14141"
        globalAddressInfo="fgs.cmp"/>
      <gbsd:gBSDUnit marker=":label:(1,0) :fps:30.0 :kbps:506
        :psnr:32.6" start="28" length="454"
        globalAddressInfo="fgst.cmp"/>
      <gbsd:gBSDUnit marker=":label:(1,1) :fps:30.0 :kbps:652
        :psnr:33.0" start="482" length="101"
        globalAddressInfo="fgst.cmp"/>
      <gbsd:gBSDUnit marker=":label:(1,2) :fps:30.0 :kbps:1541
        :psnr:35.0" start="583" length="610"
        globalAddressInfo="fgst.cmp"/>
      <gbsd:gBSDUnit marker=":label:(1,3) :fps:30.0 :kbps:3460
        :psnr:38.6" start="1193" length="3860"
        globalAddressInfo="fgst.cmp"/>
      <gbsd:gBSDUnit marker=":label:(1,4) :fps:30.0 :kbps:4863
        :psnr:41.3" start="5053" length="9586"
        globalAddressInfo="fgst.cmp"/>
      <gbsd:gBSDUnit marker=":label:(0,0)"
        start="10350" length="3093"/>
      <gbsd:gBSDUnit marker=":label:(0,1)"
        start="28043" length="9"
        globalAddressInfo="stockholm_cif_fgs.cmp"/>
      <!-- and so on -->
    </gbsd:gBSDUnit>
    <gbsd:gBSDUnit marker=":parcel:1"
        start="56155" length="56363">
      <!-- and so on -->
    </gbsd:gBSDUnit>
    <!-- and so on -->
  </dia:Description>
</dia:DIA>
```

In the generated description, we used the marker mechanism of gBSD for defining the structural elements as defined in the abstract model. The top-level `gBSDUnit` elements represent the parcels of the bitstream. They are marked by `:parcel:i`, in which `i` is the sequence number of the parcel.

Within the parcels, any number of data blocks can be found, each corresponding with a `gBSDUnit` that carries a label, marked by the `:label:(x,y,...)` syntax.

For assigning properties to the different versions of each parcel, we employed the following convention: a property consists of a name, enclosed by colons, followed by a value, for example `:fps:15`, and can be added to the marker of any data block. This value corresponds with the value of the prop-

Table 4.1: Conventions for mapping a gBSD description on the abstract model

Marker	Location	Link with the model
`:parcel:i`	top level, always occurs	Represents the i-th parcel $p \in P$ of the bitstream.
`:label:(x,y,...,z)`	second level, always occurs	Data block $d \in D$, with $\hat{d} = (x, y, \ldots, z)$.
`:prop:value`	second level	Property prop $\in \mathbb{P}$; when this occurs in data block d of parcel p, with $\hat{d} = (x, y, \ldots, z)$, then $\text{prop}(p(x, y, \ldots, z)) = \text{value}$. Note that any name for the property can occur, except *parcel* and *label*.

erty that is mentioned, for the version of the current parcel that has the same identifier as the label of the data block that is marked by the property.

All the conventions we just mentioned for mapping a gBSD description on our abstract model are summarized in Table 4.1. The following remarks have to be taken into account.

- The entire bitstream $b \in B$, a sequence of parcels, is found in a `dia:Description` of type `gBSDType`.
- According to Rule 4.1, the number of elements in the label of all data blocks of the same parcel should be the same. This is supposed to be reflected in gBSD descriptions that follow the conventions of Table 4.1.
- The payload of a data block can be found by means of the addressing information and the `start` and `length` attributes of the `gBSDUnit` that corresponds with that data block.
- The size of a data block ($|d|$) can be determined by means of the `length` attribute of the `gBSDUnit` that corresponds with d. If the `addressMode` is `byte`, the value of the `length` attribute has to be multiplied by 8.

The approach that we followed here for MPEG-4 FGST sequences can be applied for other scalable bitstreams for which a mapping to our model can be defined, and for which a bitstream description can be produced.

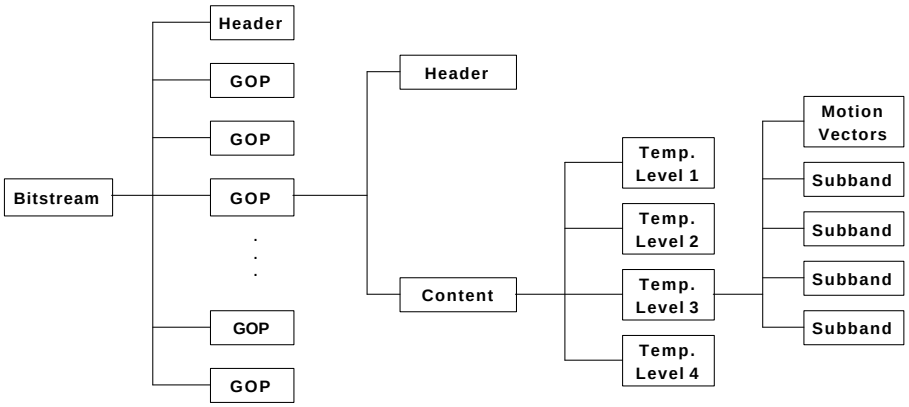


Figure 4.4: Structure of an MC-EZBC bitstream.

4.3.3 Wavelet-based video coding

The most interesting example of coding formats for which our model can be applied, is probably the one that is based on the fully scalable video coding scheme called MC-EZBC (Motion-Compensated Embedded Zerotree Block Coding, an improvement of what is presented in [26]). Parcels in these bitstreams have three scalability axes: an SNR axis, a resolution axis and a temporal axis.

The entire bitstream in MC-EZBC consists of header information followed by a sequence of Groups of Pictures (GOPs). Each GOP starts with a GOP header containing general motion information, followed by the information of a number of distinct temporal subbands, with motion vectors in between. Each temporal subband is split up into a number of spatial subbands, that are coded by means of a bitplane coding technique. This structure is shown in Fig. 4.4.

In our model, an MC-EZBC parcel corresponds with one GOP in the video sequence, and is typically 16 or 32 frames long (we can think of a video fragment of approximately one second). The number of levels for each axis is determined during the encoding process: for the temporal axis, it corresponds with the number of temporal levels, for the resolution axis the number of spatial levels, and for the SNR axis, it depends on the maximum number of bitplanes available within each spatial subband.

The label of each data block is determined by the temporal level, the spatial subband, and the bitplane where it belongs to. For each possible label, there exists exactly one data block in the current parcel. Therefore, Fig. 4.1 can

represent a MC-EZBC parcel. The first temporal level of Fig. 4.4 is composed of the blocks that appear closest to the viewer in Fig. 4.1. The first spatial subband of this temporal level corresponds with the first row in Fig. 4.1. In turn, each block within this row corresponds with one bitplane of that subband.

4.4 Content negotiation for scalable bitstreams

Now we are ready to describe how we can formally define a content negotiation process in a scenario for streaming video over the Internet, on top of the abstract model. The aim of a content negotiation process is, given the limitations of the usage environment, to select a value on each scalability axis (in the case of a fully scalable bitstream an SNR level, a resolution and the number of frames per second) that corresponds with a reduced version of the parcel that meets all limitations, but at the same time maximizes the quality as experienced by the user. We need to introduce some additional definitions that are not part of the core of our model, but rather belong to this specific application.

4.4.1 Introduction

One application that can benefit from scalability in video coding is multimedia content negotiation, an essential part of what is often called Universal Multimedia Access [73]. Content negotiation for scalable bitstreams can be defined as a process that has a scalable bitstream and a usage environment description as input, and the output is a reduced version of the bitstream, suitable for the given usage environment. In this context, the usage environment consists of the following aspects.

- The characteristics of the terminal (this captures both hardware and software).
- The network characteristics (such as average bandwidth and error rate).
- User preferences (such as preferred language).
- The natural environment of the user (such as location).

A schematic representation of a content negotiation process is already given in Fig. 2.1 of Chapter 2. Such a process can be decomposed into several phases.

1. The first step, which could be considered to be a preprocessing phase, is the translation of the usage environment description into constraints.

Such a translation can be based on a format for usage environments, such as MPEG-21 DIA. A more future-proof solution, that should be able to cope with new vocabularies, can be based on ontologies [74]. In our experiments, the constraints were expressed manually.

2. In the second phase, all possible versions are compared with the constraints, in order to construct a set of feasible versions. It is obvious that the version that ultimately will be selected, must be feasible.

3. The actual selection step is the third phase. Its aim is to choose one particular version from the set of feasible versions, in such a way that a maximum quality is offered towards the end user.

4. The fourth phase consists of offering the actual payload of the selected version to the decoding device. This can be done by means of one of the bitstream description mechanisms discussed in the previous chapter.

4.4.2 Constraints

We define constraints in the same way we constructed our model: by means of definitions. A **constraint** is actually a function that maps a version to a boolean value. Semantically, a constraint tells if a parcel is **feasible** (evaluates to true) or not.

Definition 4.9. *The set of all **constraints** is denoted C, where $C \ni c : P \mapsto \{true, false\}$. We say that a parcel $p \in P$ is **feasible** according to a set of constraints $Cts \subset C$ iff $(\forall c \in Cts)(c(p) = true)$.*

Typically, such constraints depend on the usage environment. As an example, the resolution of the screen of the receiving device might limit the spatial resolution of the images and video data that can be accepted. However, it is important to note that a constraint may also be imposed by the content creator. A musician may for example state that a song can be adapted, but during the adaptation, the quality is not allowed to drop below a certain threshold. As a third example, when a streaming video sequence is transmitted over the internet, the network will impose a constraint, in particular an upper bound for the bit rate, corresponding with the available bandwidth.

Within MPEG-21 Digital Item Adaptation, the Universal Constraint Descriptor (UCD) is introduced as a mechanism for describing different types of constraints. These constraints can use constant values, but can also refer to bitstream properties or properties coming from the Usage Environment Description (UED). As such, any of the types of constraints that we just described can be expressed using the UCD mechanism.

Listing 4.2 shows how a bandwidth constraint can be expressed in UCD. There can be several limit constraints, each expressed by means of a stack function. The values that can occur in these stack function expressions will often refer to the UED of the terminal or information about the properties of the adapted media.

Listing 4.2: Example of a bandwidth constraint expressed using UCD.

```
<DIA>
  <DescriptionMetadata>
    <!-- omitted definition of classification schemes -->
  </DescriptionMetadata>
  <Description xsi:type="UCDType">
    <AdaptationUnitConstraints>
      <!-- a LimitConstraint is expressed
           by means of a stack function -->
      <LimitConstraint>
        <!-- actual bit rate of the media,
             should occur somewhere else, e.g., in
             AdaptationQoS or in the bitstream description -->
        <Argument xsi:type="SemanticalRefType"
          semantics=":MEI:7"/>
        <!-- available bandwidth, found in the UED -->
        <Argument xsi:type="ExternalIntegerDataRefType"
          uri="my_UED.xml#
               xmlns(dia=urn:mpeg:mpeg21:2003:01-DIA-NS)
               xpointer(//dia:NetworkCharacteristics
                         /dia:AvailableBandwidth/@average)"/>
        <!-- boolean IsLessThanOrEqualTo operator -->
        <Operation operator=":SFO:38"/>
      </LimitConstraint>
    </AdaptationUnitConstraints>
  </Description>
</DIA>
```

In Sect. 4.3.2, we proposed a mechanism for inserting information about the properties of the different possible versions in a bitstream description. This information can also be expressed in a standardized format compliant with MPEG-21 Digital Item Adaptation, by means of the AdaptationQoS tool.

Consider Listing 4.3, where different properties of the available versions of a fully scalable video sequence are represented. The labels in our model are composed from the `TEMPORAL_LAYERS`, `SPATIAL_LAYERS`, and `QUALITY_LAYERS`. This can be marked in the relevant IOPins in the AdaptationQoS description by means of the `semantics` attribute.

The `FRAMERATE`, `FRAMEWIDTH`, `FRAMEHEIGHT`, and `CODESIZE` are

properties of the different versions that can be calculated from the identifier (the different layers) of each version. The CODESIZE is different for each parcel, which is why a ContentSwitch is needed. Another property, REQD_GOP_BW, the required bandwidth, is expressed as a function of the code size: as each GOP consists of 16 frames, the original frame rate is 30 frames per second, and the CODESIZE is expressed in bytes, the required number of bits per second available is calculated as CODESIZE $\times \frac{8}{16/30}$, or CODESIZE.15.

Listing 4.3: Example of expressing properties of a scalable video sequence using AdaptationQoS.

```
<DIA>
  <DescriptionMetadata>
    <!-- omitted definition of classification schemes -->
  </DescriptionMetadata>
  <Description xsi:type="AdaptationQoSType">
    <Module xsi:type="LookUpTableType">
      <AxisRef iOPinRef="TEMPORAL_LAYERS" />
      <Content iOPinRef="FRAMERATE">
        <ContentValues xsi:type="FloatMatrixType"
            mpeg7:dim="5">
          <Matrix>1.875 3.75 7.5 15 30</Matrix>
        </ContentValues>
      </Content>
    </Module>
    <Module xsi:type="LookUpTableType">
      <AxisRef iOPinRef="SPATIAL_LAYERS" />
      <Content iOPinRef="FRAMEWIDTH">
        <ContentValues xsi:type="FloatMatrixType"
            mpeg7:dim="6">
          <Matrix>11 22 44 88 176 352</Matrix>
        </ContentValues>
      </Content>
      <Content iOPinRef="FRAMEHEIGHT">
        <ContentValues xsi:type="FloatMatrixType"
            mpeg7:dim="6">
          <Matrix>9 18 36 72 144 288</Matrix>
        </ContentValues>
      </Content>
    </Module>
    <Module xsi:type="LookUpTableSwitchType"
        switchIOPinRefs="GOP">
      <AxisRef iOPinRef="TEMPORAL_LAYERS" />
      <AxisRef iOPinRef="SPATIAL_LAYERS" />
      <AxisRef iOPinRef="QUALITY_LAYERS" />
      <ContentSwitch iOPinRef="CODESIZE">
        <ContentDataSwitch switchValues="0">
```

```
        <ContentValues xsi:type="FloatMatrixType"
            mpeg7:dim="5 6 5">
          <Matrix>
            123 136 148 161 173
            533 612 686 763 838
            1543 1866 2188 2500
            <!-- and so on -->
          </Matrix>
        </ContentValues>
      </ContentDataSwitch>
      <ContentDataSwitch switchValues="1">
        <!-- more CODESIZE information -->
      </ContentDataSwitch>
    </ContentSwitch>
  </Module>
  <Module xsi:type="StackFunctionType"
      iOPinRef="REQD_GOP_BW">
    <StackFunction>
      <Argument xsi:type="InternalIOPinRefType"
          iOPinRef="CODESIZE" />
      <Argument xsi:type="ConstantDataType">
        <Constant xsi:type="FloatType">
          <Value>15</Value>
        </Constant>
      </Argument>
      <!-- multiply -->
      <Operation operator=":SFO:18" />
    </StackFunction>
  </Module>
  <IOPin id="CODESIZE" />
  <IOPin id="FRAMERATE" semantics=":MEI:20" />
  <IOPin id="FRAMEWIDTH" semantics=":MEI:17" />
  <IOPin id="FRAMEHEIGHT" semantics=":MEI:18" />
  <IOPin id="TEMPORAL_LAYERS">
    <Axis>
      <AxisValues xsi:type="IntegerVectorType">
        <Vector>0 1 2 3 4</Vector>
      </AxisValues>
    </Axis>
  </IOPin>
  <IOPin id="SPATIAL_LAYERS">
    <Axis>
      <AxisValues xsi:type="IntegerVectorType">
        <Vector>0 1 2 3 4 5</Vector>
      </AxisValues>
    </Axis>
  </IOPin>
  <IOPin id="QUALITY_LAYERS">
    <Axis>
```

```
        <AxisValues xsi:type="IntegerVectorType">
          <Vector>0 1 2 3 4</Vector>
        </AxisValues>
      </Axis>
    </IOPin>
    <IOPin id="GOP" semantics=":SEG:1">
      <Axis>
       <AxisValues xsi:type="IntegerVectorType">
         <Vector>
           0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
         </Vector>
       </AxisValues>
      </Axis>
    </IOPin>
    <IOPin id="REQD_GOP_BW" semantics=":MEI:7" />
  </Description>
</DIA>
```

We are now ready to describe the second phase of a content negotiation process, the construction of the set of feasible versions, in a more formal way: we define the set of feasible versions of a parcel according to a set of constraints.

Definition 4.10. *The set of all **feasible versions** of a parcel $p \in P$, according to a set of constraints $Cts \subset C$, is denoted $\langle p \rangle_{Cts}$, where $\langle p \rangle_{Cts} = \{v \in p^{+} | (\forall c \in Cts)(c(v) = true)\}$.*

According to the definition, the set of feasible versions of a parcel is the set of all versions v that are part of the closure of the parcel (all versions that can be generated from the original parcel), for which all constraints evaluate to true. Note that we suppose that this set of constraints is composed in the first phase.

As an example, suppose that a first constraint limits the bit rate of the parcel to a certain amount, corresponding with the available bandwidth, and a second constraint limits the spatial resolution, corresponding with the resolution of the display of the receiving device. Only those versions that fulfill both constraints, can be considered feasible versions.

4.4.3 Selecting the best version

In the third phase, we start from a set of feasible versions, and the objective is to select one version that offers the best quality towards the end user. We suppose we have a specific property, a quality function $Q : P \mapsto \mathbb{R}$ for which

$Q(p)$ represents the quality offered by parcel p towards the end user. Now we are ready to define a content negotiation application.

Definition 4.11. *A **content negotiation problem** can be defined as follows: Given:*

- *a bitstream $b \in B$,*
- *a quality function $Q \in \mathbb{P}$,*
- *a set of constraints $Cts \subset C$.*

Find an adapted bitstream $b' \in B$, subject to:

- $\#b = \#b'$, *and*
- $(\forall i \in \{0, 1, \ldots, \#b - 1\})(b'_i = \arg \max_{p \in \langle b_i \rangle_{Cts}} Q(p))$

More informally, we define a content negotiation problem as the process of deriving an adapted bitstream b' from an original bitstream b, in such a way that in each parcel b_i, the adapted parcel b'_i satisfies all constraints Cts imposed by the usage environment ($b'_i \in \langle b_i \rangle_{Cts}$), and at the same time maximizes a certain quality function Q.

In Def. 4.11, we have used a fixed set of constraints Cts. In practice, however, it is expected that the constraints imposed by the environment will change during the transmission of the bitstream [12]. To model such a situation, we need to change the definition, and use a sequence of sets of constraints instead of one such set. Each set within this sequence then corresponds to the constraints that exist at the transmission time of the corresponding parcel.

An additional problem might occur in case of heavily fluctuating constraints. This can cause significant fluctuations in the observed visual quality. As a consequence, it might be necessary to use modified constraints, rather than the actual constraints that are immediately implied by the usage environment, in order to obtain more graceful changes in terms of visual quality.

When only one scalability axis is available, and the usage environment is only constrained by the bandwidth, we could use the size-function (as defined in Def. 4.8) as a quality function ($Q \equiv ||.||$), because we can expect that the more layers are received, the more quality is offered. In this case, the version that contains most layers and that is still feasible, will offer most quality and will be selected for transmission.

In more complicated cases, we can use other properties as quality functions, such as the PSNR value of a particular parcel. In [72], some useful

examples of combinations of constraints and quality functions can be found, in the context of MPEG-21 Digital Item Adaptation.

Within MPEG-21, the UCD can be used for defining what is called an optimization constraint, which is a quality or utility function that has to be maximized or a cost function that has to be minimized. Listing 4.4 shows how to express such a quality function in UCD.

Listing 4.4: Expressing by means of UCD that a particular quality function needs to be maximized.

```
<DIA>
  <Description xsi:type="UCDType">
    <AdaptationUnitConstraints>
      <LimitConstraint>
        <!-- details about one particular constraint -->
      </LimitConstraint>
      <!-- several LimitConstraints can be expected -->
      <OptimizationConstraint optimize="maximize">
        <Argument xsi:type="ExternalIOPinRefType"
          iOPinRef="#VISUAL_QUALITY" />
      </OptimizationConstraint>
    </AdaptationUnitConstraints>
  </Description>
</DIA>
```

4.5 Other applications of the abstract model

In the previous section, we showed how the abstract model for scalable bitstreams can be used for describing a content negotiation process. For validating the usefulness of the model, we also used it for describing a multicast protocol that was designed for scalable bitstreams [7].

Because the topic of new network protocols for multimedia sessions does not entirely fall within the scope of this thesis, we only briefly discuss the way we applied our model to such a protocol.

This protocol, called CIFL (Coding-Independent Fair Layered Multicast), is designed to be a generic, stable, TCP-friendly protocol for setting up layered multicast sessions. Details about the protocol can be found in the original paper by El Khayat and Leduc [75].

Because of a number of requirements imposed by the protocol, only a subset of the bitstreams that are valid according to the definitions of the model, can be used. Fortunately, we were able to demonstrate that it is possible to translate a bitstream that does not belong to this subset, in such a way that

the transformed bitstream does, by creating empty data blocks and by merging multiple data blocks together.

An important addition to the basic abstract model defined in Sect. 4.2, is the concept of timing information. We achieved this by introducing *time stamps*, that are defined as the moment when the processing of a particular parcel begins.

A concept that is in use in the CIFL protocol, and that can easily be defined using our model, is a channel. This is a sequence of data blocks belonging to subsequent parcels, all having the same label. A *subscription set* is a set of channels that a user receives at some time. In order to receive exactly those data blocks necessary for reconstructing a particular version, a subscription set has an identifier, which has a similar meaning as the identifier of a version of a parcel. CIFL also uses *synchronization points*, which are time stamps used for notifying when a user is allowed to modify its subscription set.

When such a synchronization point occurs, two actions are possible for a receiver: a *join* means that a number of channels is added to the subscription set, a *leave* means that a number of channels is removed. Based on the description of the CIFL protocol, we formally defined these actions by means of a number of preconditions that all have to be satisfied before the action can take place, and a number of postconditions, describing the state changes that occur when the action is executed.

This way, we demonstrated that the abstract model that we defined in this chapter can be used for describing other applications that use scalable bitstreams, apart from the content negotiation problem for which the model was defined.

4.6 Conclusions and original contributions

In this chapter, we formally defined an abstract model that can be used for describing the structure of any kind of scalable bitstream. We achieved this by means of a number of definitions and rules, but we also described the more informal semantics of this model.

We validated the correctness of this model by checking different algorithms for scalable video coding, to see if they can be mapped onto the model definitions. For one particular case, MPEG-4 FGST, we used the generic gBSD bitstream descriptions for an explicit mapping onto the model. By means of the facilities of gBSD for adding metadata information, we created explicit links between particular bitstream fragments, represented by means of gBSD units, and the structural elements defined in the model.

The main goal of setting up such an abstract model for scalable bitstreams

is the possibility of formally describing applications that use these bitstreams. In this chapter, we showed how this could be done for two different applications. The first one described the problem of multimedia content negotiation, in which the best possible available version has to be transmitted to the client, taking into account a number of constraints that can be imposed by the terminal, the network or other parts of the environment.

The second application of which we showed how we could formally describe it, was the definition of a protocol that was designed for multicast transmission of scalable bitstreams. All relevant concepts were formally defined using our model, and the possible actions were described by means of preconditions and postconditions.

As we will discuss in the next chapter, our definition of a content negotiation process does not take into account certain aspects of the reality. As a consequence, we will have to further extend our definition of a content negotiation process to be able to describe these aspects.

The research that lead to this chapter of this thesis is also discussed in the following of our publications.

1. Sam Lerouge, Boris Rogge, Robbie De Sutter, Jeroen Bekaert, Dimitri Van De Ville, and Rik Van de Walle. A generic mapping mechanism between content description metadata and user environments. In *Internet Multimedia Management Systems III*, volume 4862 of *Proceedings of SPIE*, July 2002.

2. Sam Lerouge, Peter Lambert, and Rik Van de Walle. Multi-criteria optimization for scalable bitstreams. In *Visual Content Processing and Representation, 8th International Workshop VLBV 2003*, volume 2849 of *Lecture Notes in Computer Science*, September 2003.

3. Sam Lerouge, Robbie De Sutter, Peter Lambert, and Rik Van de Walle. Fully scalable video coding in multicast applications. In *Electronic Imaging 2004*, volume 5308 of *Proceedings of SPIE*, pages 555–564, San Jose, January 2004.

Chapter 5

Multi-criteria optimization in video communication

5.1 Introduction

In video coding, compression is achieved by lowering the quality of the incoming signal somehow. The most common way of reducing the quality is by means of what is usually called adaptive quantization: rounding errors are allowed in order to reduce the bit rate, but at the same time distortion is introduced. The formal definition of a content negotiation process, as introduced in Sect. 4.4, where one quality function is used, reflects such a scenario. It also sufficiently describes adaptation scenarios based on transcoding or scalable coding where only one quality aspect can be modified, e.g. MPEG-4 FGS.

Sometimes, better results are possible when multiple quality aspects can be modified at the same time. This is possible during encoding, transcoding, and also with some types of scalable coding, such as MPEG-4 FGST and fully scalable video coding, using wavelet-based coding or based on H.264/AVC.

In such scenarios, it no longer makes sense to describe the quality of the video as a single quality function. Rather, a multi-dimensional adaptation is possible, where multiple quality functions have to be maximized at the same time. Existing theoretical frameworks for optimizing the quality in video communication, such as the well known rate-distortion optimization frameworks based on Lagrangian cost [31], are not capable of describing such a situation.

As we have said in the previous chapter, the objective of a content negotiation process is to maximize the overall quality or utility towards the end user, given all the constraints imposed by the system. Because of these constraints (e.g. the available bandwidth), the solution where all quality functions are optimal probably does not fulfill all constraints. In such a situation, a *trade-*

off will have to be made between the different quality functions, which is a problem that is often not straightforward.

The problem of making trade-offs between multiple optimization functions is not new. The first to describe such situations in a mathematical way was Vilfredo Pareto [76], a teacher in economics. In the following section, we introduce all important definitions belonging to the Pareto theory. Later on, we show how we can apply these principles to the domain of content negotiation.

5.2 Background of multi-criteria optimization

The first definition that we introduce in this section is the one of a **multi-criteria optimization problem** [77]. The main difference between such an optimization problem, and what we could call classic optimization problems, is that in this case, more than one optimization function exists.

Definition 5.1. *A **multi-criteria optimization problem** is defined as follows:*

$$\begin{array}{ll} \textit{Find} & \boldsymbol{x} = (x_0, x_1, \ldots, x_{n-1}) \\ \textit{Maximizing:} & \boldsymbol{F(x)} = (F_0(\boldsymbol{x}), F_1(\boldsymbol{x}), \ldots, F_{k-1}(\boldsymbol{x})) \\ \textit{Subject to:} & g_i(\boldsymbol{x}) \leq 0; i = 0, 1, \ldots, l-1 \\ & h_j(\boldsymbol{x}) = 0; j = 0, 1, \ldots, m-1 \end{array}$$

In this definition, each F_i is called an objective function or optimization function, and g_i and h_j are constraints, limiting the set of candidate solutions. A solution **x** that meets all of these constraints is said to be **feasible**.

The concept of **dominance** is crucial in multi-criteria optimization. We say that solution x dominates solution y if and only if x performs at least as well as y for all objective functions, and better than y for at least one objective function.

Definition 5.2. *A solution $\boldsymbol{x} = (x_0, x_1, \ldots, x_{n-1})$ **dominates** another solution $\boldsymbol{y} = (y_0, y_1, \ldots, y_{n-1})$ according to $\boldsymbol{F} = (F_0, F_1, \ldots, F_{k-1})$*

$$\Leftrightarrow$$

$$(\forall i \in \{0, \ldots, k-1\})(F_i(\boldsymbol{x}) \geq F_i(\boldsymbol{y})) \wedge (\exists j \in \{0, \ldots, k-1\})(F_j(\boldsymbol{x}) > F_j(\boldsymbol{y})).$$

To improve readability, we note $\boldsymbol{x} \succ_{\boldsymbol{F}} \boldsymbol{y}$.

In Fig. 5.1, an example is given of a set of candidate solutions for which two criteria X and Y need to be optimized. It should be clear that solution a dominates solution b, as it is better than b on one criterion, and not worse than

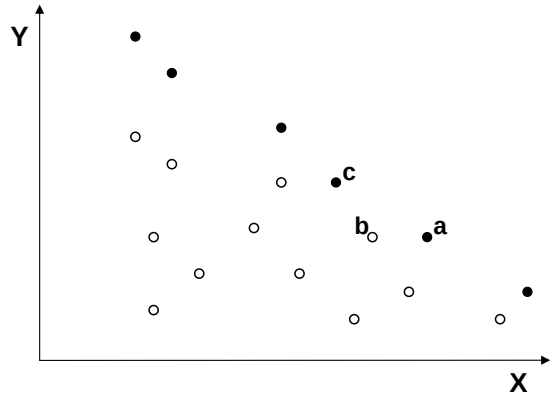


Figure 5.1: Example of a Pareto frontier for two optimization criteria.

b on the other criterion. At the same time, a and c cannot be compared using the concept of dominance: going from one of these solutions to the other one can only happen by improving the value of one criterion, while lowering the other one.

The **Pareto frontier** of a multi-criteria optimization problem is defined as the set of feasible solutions that are not dominated by any other feasible solution. It should be obvious that all candidate solutions that belong to the Pareto frontier can be considered to be optimal solutions.

Definition 5.3. *The **Pareto frontier** of a multi-criteria optimization problem having candidate solutions X and a vector of objective functions $\boldsymbol{F}$ is defined as follows:*

$$PF_{\boldsymbol{F}}(X) = \{\boldsymbol{x} \in X | (\nexists \boldsymbol{y} \in X)(\boldsymbol{y} \succ_{\boldsymbol{F}} \boldsymbol{x})\}.$$

*A solution that belongs to the Pareto frontier is said to be **Pareto optimal**.*

In this definition, the set of candidate solutions X only consists of solutions that are feasible according to all constraints g_i and h_i, as in Def. 5.1.

More informally, a Pareto optimal solution is a feasible solution for which any other feasible solution is worse on at least one optimization criteria. Moving from one Pareto optimal solution to another one is therefore only possible by improving on at least one criterion but at the same time worsening another criterion. As all solutions in the Pareto frontier are considered optimal solutions, it often occurs that a multi-criteria optimization problem does not produce one optimal solution but rather a set of optimal solutions that cannot further be compared if no additional information is available.

In Fig. 5.1, all white solutions are dominated by at least one other solution. The black solutions are not dominated, and therefore belong to the Pareto frontier.

If only one solution has to be selected from the Pareto frontier, a trade-off will have to be made between the different optimization functions. It will be necessary to somehow make clear which criteria are more important than others, and how much more important they are. In the domain of Multi-Criteria Decision Analysis [78], several frameworks for assisting a human, the so-called decision maker, in selecting an appropriate solution are developed. In Sect. 5.6, we briefly present an overview of these frameworks.

5.3 Complexity of calculating the Pareto frontier

In general, one of the main issues in solving multi-criteria optimization problems using the Pareto frontier, is the complexity of calculating the frontier.

Because the candidate solutions in the problems we are thinking of are the available versions of a parcel, only a finite number of candidate solutions has to be taken into account. In such situations, it is possible to use an exhaustive search algorithm, such as the one shown in Algorithm 5.1.

5.3.1 Complexity analysis

Because of the exhaustive nature of the presented search algorithm, we should be concerned with its computational complexity. When determining the theoretical complexity of an algorithm, we use the well-known $\mathcal{O}(\cdot)$ (or Big-O), $\Omega(\cdot)$ and $\theta(\cdot)$ notations for describing the asymptotic behavior of algorithms. Respectively, they describe the asymptotic upper bound, lower bound, and tight bound. The latter means that the lower and upper bounds are the same.

When doing a basic analysis of the algorithm as described here, we start with noting that the *isFeasible* function is linear with the number of constraints. For simplicity, we only consider the number of feasible versions n.

Let k be the number of criteria involved in the optimization process. The *dominates* function has to do a comparison for each criterion, until it finds one where the second solution has a higher value than the first one. This function has a complexity of $\mathcal{O}(k)$.

First, we discuss the lower bound for the complexity. In this case, the first solution will dominate all other solutions. The set O will therefore always contain one element, except for the first iteration. Therefore, the complexity of the body of the loop between line 6 and line 14 is $\Omega(k)$. This loop will be executed n times, so the total complexity is $\Omega(n.k)$.

Algorithm 5.1 An exhaustive search algorithm for determining the Pareto frontier.

```
    function GETPARETOFRONTIER(S)          //S is the set of all versions
 2:     O := ∅                             //current set of non-dominated solutions
        for all s in S do
 4:         if isFeasible(s) then
                dom := false               //initially, suppose s is not dominated
 6:             for all o in O do
                    if dominates(o, s) then
 8:                     dom := true        //s cannot belong to O
                        break
10:                 end if
                    if dominates(s, o) then
12:                     O := O - o         //remove solution o
                    end if
14:             end for
                if dom = false then
16:                 O := O + s
                end if
18:         end if
        end for
20:     return O
    end function
```

For the upper bound, we consider the worst case situation where *all* solutions are part of the Pareto frontier. In this case, the result of the *dominates* function is always false. The complexity of the body of the loop between line 6 and line 14 is $\mathcal{O}(k) + \mathcal{O}(k) = \mathcal{O}(k)$. This loop is averagely executed $n/2$ times (the average size of the set O), so the complexity for the loop is $\mathcal{O}(n.k)$. As this part is executed n times, the total complexity will be $\mathcal{O}(n^2.k)$.

5.3.2 Measurements

In [3], we performed some measurements using an implementation of the presented algorithm, to see if the calculation of the Pareto frontier can become problematic in a content negotiation application.

These measurements were done by profiling a Java implementation of Algorithm 5.1 for calculating the Pareto frontier using three 'real-life' descriptions of scalable bitstreams in the SSM framework [66]. We compared the time spent on the parsing of the XML descriptions with the time spent on calculat-

Table 5.1: Parsing time compared to the time needed for calculating the Pareto frontier, for three different SSM descriptions.

sequence	versions	criteria	size (KiB)	time (ms)	parsing	frontier	other
VTC	54	2	5 098	413	97.4%	1.5%	1.1%
MPEG-4	20	3	22	2	97.0%	1.3%	1.7%
MC-EZBC	210	4	85	21	70.5%	27.6%	1.9%

ing the Pareto frontier. The results of these measurements, executed in 2003, can be found in the original paper [3].

The most important conclusion at that time was that the real bottleneck was not the calculation of the Pareto frontier, but rather the time needed for parsing XML documents, even though an efficient SAX parser was used.

Since then, the performance of XML parsers has seriously evolved. Therefore, we decided to run the same code with more recent XML libraries. These measurements were performed on a computer running a Pentium 4 2.8 GHz, using 1 GiB of RAM. The software ran on a Windows XP Professional, using Java 1.5.0_02 with Xerces as XML parser. In order to avoid any influence of the speed of the disk, the SSM description was loaded into the memory before the actual measurements started. The results can be found in Table 5.1.

The first column refers to the test sequence used (more information can be found in [3]). The second column is the number of available versions in the SSM description. In the third column, the number of criteria is shown. In the fourth column, the size of the SSM description can be found, expressed in KiB. The fifth column shows the total amount of time needed for processing the description, expressed in ms. The following two columns show the relative amount of time spent on parsing the description and calculating the Pareto frontier. The remaining time, shown in the last column, covers initializations, such as class initializations.

It is clear that for the data we used for testing, the time spent for the calculation of the Pareto frontier remains acceptable, as it is far below the time needed for parsing the bitstream descriptions. As expected, the time needed for calculating the Pareto frontier increases when the number of versions and the number of criteria increases. The large time consumption for the VTC case can be explained by the large size of the description. For the MC-EZBC case, the relative amount of time spent for calculating the Pareto frontier is so large because of the large number of versions and criteria taken into account.

5.4 Multi-criteria optimization in video coding

As we already mentioned in the introduction, in video coding it can be possible to consider several optimization functions at the same time, when not only the distortion, but also the resolution and the frame rate are allowed to change. Because we want to end up with one optimal solution, a trade-off will have to be made between these quality aspects, taking into account the constraints, in particular the target bit rate.

In what we could call *traditional* video coding, this decision has to be made during the encoding step. Because in most scenarios hardly any information about the end user is available, it makes most sense to take only into account information on the input video sequence itself. This is the approach followed by Reed and Lim [79]. They propose a mechanism for automatically making these trade-offs, based on the properties of the input signal. They define an integer programming formulation and present an algorithm for computing an optimal solution when controlling the bit rate by jointly adapting the frame rate, the resolution and the distortion.

These are not the only situations in which trade-offs between different quality aspects have to be made. In analog television broadcasting, a trade-off between the resolution and the frame rate of the images had to be made. In different television standards, this trade-off is taken in a different way. An NTSC signal carries images containing typically 485 visual lines at a refresh rate of 60 Hz (interlaced), whereas a PAL signal has 575 visual lines, but at only 50 interlaced frames per second.

The use of interlaced frames instead of progressive frames can also be considered a trade-off: in order to achieve a high frame rate, the actual resolution is only half of the resolution of a television screen. Some other relevant situations in which trade-offs can occur in multimedia distribution can be found in the following list.

- In very low bit rate transmission of audiovisual information, the allocation of bandwidth to the audio and the video signal can become very important. In such a situation, an appropriate choice for making this trade-off will greatly influence the quality perceived by the end user [80].
- A utility-based network adaptation system is presented in [81]. It contains a utility-fair algorithm for scaling multiple media objects, while taking into account user preferences.
- In [82], a generalized framework is proposed for determining an optimal adaptation strategy when multiple objects of a multimedia presentation can be adapted.

Table 5.2: Properties of the different versions of a video sequence.

version	frame rate (Hz)	PSNR value (dB)	bit rate (kbps)
1	15	28	46
2	15	34	128
3	15	40	400
4	30	26	64
5	30	32	256
6	30	36	1200

- Transmission over a network where the client (or the server) has to pay per byte, rather than proportional with the connection time or a fixed fee. Such scenarios currently occur in GPRS networks, amongst others. In such a situation, a user might not be willing to receive a version that is twice as big as another one but that only offers 5 percent more quality. A formalism for making trade-offs between network cost and user satisfaction is presented in [83].

- In mobile devices, one of the critical aspects is battery life. A user might not be willing to sacrifice a lot of his battery capacity to watch the news at full quality, if he can watch it at a lower quality while saving a lot of battery power.

Let us consider a practical example to make the problem of multi-criteria optimization in video communication more concrete. Suppose we want to transmit a video sequence over the Internet. Different versions of the sequence are available (either because it was encoded using a scalable coding algorithm, or because simulstore[1] is used, or because a real-time transcoder is available), all having different properties, as summarized in Table 5.2. The available bit rate is limited to 200 kbps. Under these circumstances, we want to maximize both the frame rate and the PSNR value. More formally, we can state that $Q = \{\mathrm{PSNRvalue}, \mathrm{framerate}\}$, and that $c(v) \equiv \mathrm{bitrate}(v) \leq 200$ is the only constraint involved. The entire sequence is treated as one parcel p.

Because of the bit rate constraint, 3 versions are not available: versions 3, 5 and 6, so $\langle p \rangle_{\{c\}} = \{1, 2, 4\}$. Within the set of feasible versions, we see that version 1 is worse than version 2, as it is not better than that version on any of the two criteria that we want to maximize. We say that version 2 *dominates* version 1 ($2 \succ_Q 1$). Therefore, version 1 cannot be part of the Pareto frontier.

[1]The term *simulstore* is used in situations where a server has different versions of the same multimedia presentation stored on its disks, all available for download.

Version 2 and version 4 however cannot be compared in the same way: both are better regarding one criterion but worse regarding the other one. As there is no other feasible version dominating version 2 and version 4, they both belong to the Pareto frontier: $PF_Q(\langle p \rangle_{\{c\}}) = \{2, 4\}$.

It should be clear that it is impossible to determine which of both versions is actually the best, if we don't have any additional information. We need to know if the user is willing to reduce the frame rate from 30 to 15 frames per second (fps), if he would gain 8 dB in terms of PSNR quality.

5.5 Content negotiation redefined

In Def. 4.11, we considered a content negotiation problem as the process of selecting one feasible version that has the highest score according to a maximization function. When we want to optimize multiple criteria, we need to redefine this process as follows.

Definition 5.4. *A **multi-criteria content negotiation problem** can be defined as follows:*
Given:

- *a bitstream $b \in B$,*
- *a set of quality functions $\boldsymbol{Q} = (Q_0, Q_1, \ldots, Q_{k-1})$, where $Q_j \in \mathbb{P}$,*
- *a set of constraints $Cts \subset C$.*

Find an adapted bitstream $b' \in B$, subject to:

- $\#b = \#b'$, *and*
- $(\forall i \in \{0, 1, \ldots, \#b - 1\})(b'_i \in PF_{\boldsymbol{Q}}(\langle b_i \rangle_{Cts}))$

Possible quality functions that can be used, according to the examples given in the previous section, are the frame rate, the spatial resolution, a distortion measure such as PSNR, the audio quality, the network cost (that needs to be minimized), battery power consumption (that also needs to be minimized), etc.

We were the first to describe a content negotiation process as a multi-criteria optimization problem [3]. Later, this approach was adopted by Mukherjee et al. [72], whose formulation appears in the specification of the Universal Constraints Description (UCD) of MPEG-21 Digital Item Adaptation. According to Def. 5.4, any solution that is part of the Pareto frontier, is

considered to be an optimal solution of a content negotiation problem. This is the way it is defined in the MPEG-21 Digital Item Adaptation as well: when multiple optimization functions are taken into account, any Pareto optimal solution is acceptable.

This way, there is no trade-off to be made. In the MPEG philosophy, it is up to the companies implementing intelligent Adaptation Decision Taking Engines to develop smart algorithms for making these trade-offs.

We believe that this can be done in an intelligent way when more information is available, in addition to the information of the bitstream, the quality functions and the constraints. We propose to consider the preferences of the end user for making a trade-off when selecting one optimal solution. This approach matches well with the concept of Quality of Experience, which is introduced in Chapter 1.

5.6 Selecting one solution from the Pareto frontier

In the domain of Multi-Criteria Decision Analysis [78], several methods exist for finding one optimal solution from a set of candidate solutions (also called *alternatives*), most of them taking into account the preferences of the user (also called the *decision maker*).

In **weighing methods**, the key concept of describing the preferences of the decision maker is the *weight*. In these methods, a weight is to be considered a measure of the relative importance of the criteria according to the decision maker.

The most widely spread method in this group of methods is the method of the weighted sum. Here, the values of the alternatives have to be normalized to a $[0, 1]$ interval, such that the ratio w_i/w_j of two weights can be considered a "substitution rate": when the utility of attribute i decreases by an amount δ, the utility of attribute j has to increase by a factor $\delta w_i/w_j$ to keep the global utility constant.

Several possibilities exist for obtaining the weights from the decision maker. In the simple classification method, the simple cardinal evaluation method and the ratio method, the decision maker has to declare the weights directly, without much considerations. In the method of successive comparisons, the decision maker has to compare additions of the importance of the criteria, which leads to more accurate estimations, but is cognitively more difficult for the decision maker.

Another well known weighing method is the Analytic Hierarchy Process [84]. This is an eigenvalue method, in which the decision maker is asked to compare the different criteria two by two and to assign a value rep-

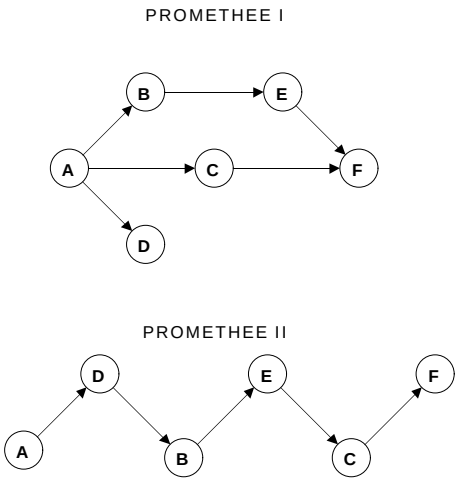


Figure 5.2: Output of the PROMETHEE methods.

resenting their relative importance. These values are placed in a matrix, for which the eigenvalues are calculated in order to retrieve the weights themselves.

Even though **outranking methods** also take into consideration the weights defined by the decision maker, the approach is quite different. The basic concept is the notion of outranking: an alternative a outranks an alternative b when a is at least as good as b on a majority of the criteria, and not performing considerably worse on any of the other criteria.

One class of outranking methods is the series of ELECTRE methods. These methods define a concordance set for two alternatives a and b as the set of criteria for which a scores better than b. The discordance set is the set of criteria for which b scores better than a. These sets are used for defining a concordance and discordance index, in which the weights of the different criteria are taken into account. These indices are then used for determining if one alternative outranks another one. More advanced versions of the ELECTRE methods also take into account a preference threshold and an indifference threshold.

The PROMETHEE method uses preference functions for each criterion. Such a function defines how much one alternative is preferred over another one, according to that criterion, in function of the deviation of the values of both alternatives. These preference functions are used for calculating a positive and negative outranking flow. In turn, these values are used for defining a partial preorder (PROMETHEE I) or a less reliable complete preorder

(PROMETHEE II). This is represented graphically in Fig. 5.2, where version A is obviously considered the best alternative according to PROMETHEE I, but versions B, C and D cannot be compared. When using the less restrictive PROMETHEE II method, it appears that D can be considered better than B or C.

5.7 Quality agents in content negotiation

The methodologies presented in the previous section demand a lot of input from the decision maker, and are difficult to use in a content negotiation scenario. Still, in order to make an intelligent choice among the Pareto optimal solutions, we need additional input from the end user. The problem is that in practice, a user will not want to select one optimal version manually. We identified two reasons for that.

- Most users have insufficient knowledge about the meaning of the different quality aspects (optimization criteria) used in video coding. As an example, most users do not know what it means when a sequence has a PSNR value of 28 dB. This also explains why it does not make sense to obtain weights from the user explicitly.

- When consuming a multimedia presentation, the user does not want to be interrupted with questions for selecting the optimal version from the Pareto frontier. As these decisions have to be taken for every parcel (see Def. 5.4), it is very well possible that this will occur every second.

It is clear that if we want to offer a personalized adaptation, we need to obtain the additional user preference information in a different, less explicit way. One option that we found was to consider the possibilities of agents that can act on behalf of the user as much as possible in taking decisions, in order to minimize the amount of explicit user interaction.

Some 10 years ago, research in Artificial Intelligence partly evolved into the exploration of so-called *interface agents*. Such agents provide active assistance to a user and can act as a personal assistant that imitates tasks that are executed regularly by the user. The agent can achieve this by learning from the user's behavior as well as from other more experienced agents. During this learning process, the agent becomes more and more effective.

According to Maes [85], a learning agent acquires its competence from four different sources.

1. The agent learns by continuously "looking over the shoulder" of the user as the user is performing actions. The agent can then look for recurrent patterns, and try to automate these.

2. The agent can learn from indirect feedback, when the user takes a different action than the one offered by the agent. The user can also give explicit feedback for actions automated by the agent.

3. The agent can learn from examples given explicitly by the user. This way, the user can train the agent by giving it hypothetical examples of events and situations and telling the agent what to do in those cases.

4. The agent can ask for advice from agents that assist other users with the same task and that may have built up more experience. Additionally, the agent can learn to trust agents that have proven to recommend actions that the user appreciated.

The application of these machine learning techniques for building autonomous agents has proven to be successful for automated e-mail handling, meeting scheduling, news filtering, entertainment selection, and web page annotation [85, 86] amongst others.

The concept of agents acting on behalf of the user has already appeared for a while in the domain of multimedia. Within MPEG-7, the *Multimedia Content Description Interface*, hooks are provided for allowing software agents to automatically figure out the personal tastes of users, and use this information to discover, select and recommend new multimedia content using this information [87].

To our knowledge, we were the first to apply the basic principles of agents to a content negotiation process. Here, the objective is to construct a model of the preferences of the end user regarding the different quality aspects. When looking at the principles of interface agents, we think that such a model can be constructed from the following sources of information.

1. Every time the agent fails taking a decision because his model seems not reliable enough, the choice is left to the user who will have to take the decision explicitly. The agent will learn from this decision by refining the user model.

2. The user should always be able to recall a decision that has been taken by the agent. Again, the agent will have to update its user model.

3. If the user wants to avoid interaction as much as possible, it might be useful to train the agent in advance. The agent can offer some predefined

examples to the user; the decisions taken by the user can be used to initiate or refine the user model.

4. It might be useful that the agent learns from other, more experienced agents that assist other users.

As an additional requirement, the agent has to operate in real time: it has to be able to resolve a decision before the next request arrives. Because of this requirement, the last option for learning the preferences might not be appropriate. In addition, we expect that models for different users will differ significantly. The first source for learning a model is also not recommended because of the reasons mentioned in the beginning of this section.

As a consequence, the major requirements that we identified for a model capable of capturing the preferences of the end user regarding different quality aspects, are the following.

- As users are not willing to spend a long time taking example decisions, the algorithm has to be able to construct a model from a very small set of example decisions.
- Constructing a model from a set of example decisions should be possible in a reasonable amount of time, even though there are no hard time constraints imposed.
- Selecting one version from the set of candidate versions has to be possible in real time: such decisions must be taken immediately, and at a high rate, as we want to be able to respond quickly to changes in the constraints imposed by the system (e.g., changes in the available bit rate, a battery running low, etc.).
- A user must have the possibility to recall the decision of the agent. The algorithm for building a model must be able to use this information for updating the model.

Figure 5.3 shows how the training process could look like. When a user starts using his multimedia player for the first time, the agent asks him to express his preferences by means of some example decisions that are used for training the agent. These example decisions can be obtained by means of pairwise comparisons, as in Fig. 5.3. When not only visual quality aspects are taken into account, for example when considering audio quality as an additional optimization criterion, it is necessary to use a different approach. The most obvious solution would be to use absolute scores, where the user is asked

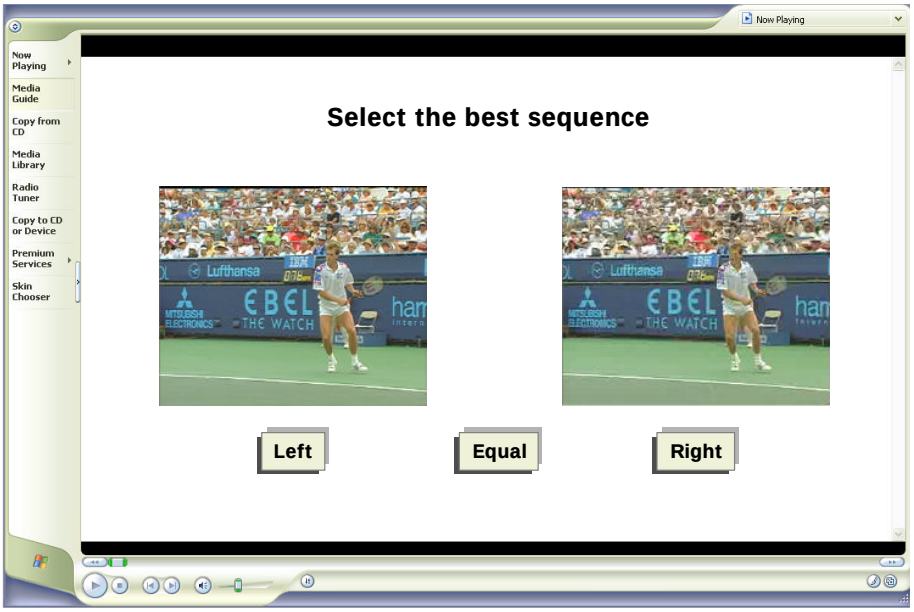


Figure 5.3: Example of how an agent that is part of a multimedia player can be trained.

to assign a number between 0 and 10 to a video sequence, expressing the quality of the sequence as observed by the user.

In the domain of psychophysics, tests are often conducted where the responsiveness of observers with respect to a certain signal, also called a stimulus, is measured. Already in 1876, Fechner conceived the idea that a psychophysical experiment could be conducted where the stimuli would have no obvious single numerical quantity [88, 89]. This could be used in studies on esthetics, where observers have to choose the one of two objects that is most pleasant. In 1927, Thurstone published his *law of comparative judgement* [90], which is a mathematical method for constructing a psychological scale from comparative judgements. Pairwise comparisons are also commonly used in well-known approaches in the domain of multi-criteria optimization, such as the Analytic Hierarchy Process [84].

The popularity of using pairwise comparisons when information has to be obtained from users comes from the fact that people often find it easier to compare two objects than to assign some absolute value to related objects. This way, the reliability of the information obtained from the user is increased. Because of this, we prefer to use pairwise relative example decisions.

5.8 Algorithms for capturing user preferences

During an exhaustive literature study, we selected two classes of algorithms that can be used for capturing user preferences based on some example decisions expressed by means of pairwise comparisons. Both algorithms meet the requirements we defined in the previous section. We describe the main advantages and disadvantages of both algorithms.

The actual performance, in particular the reliability when predicting decisions from the end user, is measured by means of a subjective test we performed. This test and its results are described in the next chapter.

5.8.1 Systems of inequalities

General description

This algorithm is related to the weighing methods for solving multi-criteria problems. In such methods, the overall utility (or cost) of a particular solution is determined as a weighted sum of its criteria. The weights are supposed to be provided by the end user.

The basic assumption of our proposed algorithm is that we can describe the personalized quality of a video sequence as a weighted sum of the individual quality aspects of the sequence. These quality aspects, such as frame rate, resolution and Peak Signal-to-Noise Ratio (PSNR), only depend on the video sequence, and are not user-dependent. The weights depend on the preferences of the end user, and it is the task of the agent to learn them from the example decisions. To summarize, we can describe the quality of a version S of a sequence according to a user u as follows:

$$Q_u(S) = \sum_{f \in F} w_f^u S_f; \tag{5.1}$$

with F a set of measurable quality aspects (properties as in Def. 4.7: $F \subset \mathbb{P}$) derived from the sequence features, w_f^u the (strictly positive) value of the weight for feature f according to user u, and S_f the value of feature f for sequence S. Note that we consider the sequence features F to be optimization criteria (quality features), rather than cost functions. In case a sequence feature f should be minimized (e.g., power consumption), a derived feature f' that has to be maximized will have to be used, e.g., by inverting the original feature.

With this approach, we have translated the multi-criteria content negotiation problem, as defined in Def. 5.4, back to a content negotiation problem with a single optimization criterion, as defined in Def. 4.11. More precisely, we can rewrite Def. 5.4 as follows.

Definition 5.5. *A **content negotiation problem** that takes into account the preferences of a user u by means of a weighted sum, can be defined as follows: Given:*

- *a bitstream $b \in B$,*
- *a set of features $F \subset \mathbb{P}$,*
- *a set of user-dependent weights w_f^u for each of the features $f \in F$,*
- *a set of constraints Cts.*

Find an adapted bitstream $b' \in B$, subject to:

- $\#b = \#b'$, *and*
- $(\forall i \in \{0, 1, \ldots, \#b - 1\})(b'_i = \arg \max_{v \in \langle b_i \rangle_{Cts}} \sum_{f \in F} w_f^u v_f)$.

In this definition, we presume that the weights are known at the time of solving the content negotiation problem. It is the task of the agent to construct a model of the preferences of the end user; this model is actually the set of weights w_f^u.

The algorithm that we propose learns these weights from a set of example decisions by means of a system of inequalities. In the remainder of this and in the following chapter, the algorithm for modelling user preferences by means of systems of inequalities is abbreviated as the *SoI* algorithm.

Constructing a model

The information that the agent uses for learning user preferences is a set of pairwise example decisions obtained from the user. Suppose that user u states that sequence A is better than sequence B when training his agent[2]. We can translate this statement into the following inequality: $Q_u(A) > Q_u(B)$. According to Eq. 5.1, this is equivalent to:

$$\sum_{f \in F} w_f^u (A_f - B_f) > 0. \tag{5.2}$$

Thus, each example decision that is obtained from the user when training the agent results in such a linear inequality. The entire training set therefore

[2]From now on, for reasons of simplicity, we treat the sequences that are used for training the agent as consisting of only one parcel.

generates a system of linear inequalities. Note that this type of inequality is equivalent to the following inequality:

$$\sum_{f \in F} w'^{u}_{f}(A_f - B_f) > 0; \quad w'^{u}_{f} = \frac{w^{u}_{f}}{c}, \quad c > 0. \tag{5.3}$$

In other words, we are allowed to rescale the weights with some constant value, as long as it is positive. In particular, if we select one of the weights as this value ($c = w^{u}_{i}$), we can remove one unknown value from the system of inequalities. This corresponds with stating that $w'^{u}_{i} = 1$.

In general, a system of inequalities does not produce a single solution but rather a region of feasible solutions. To solve this problem, we decided to choose the coordinates of the centroid of this region as the ideal solution of the region. Note that, as the inequalities are all linear, when the region is bounded, it will be convex, and therefore it is guaranteed that the centroid will be part of the solution region.

In case of three features, there are two unknown feature weights to be determined, and therefore we can represent this process graphically, as is shown in Fig. 5.4. In this figure, the dashed lines represent the inequalities derived from the example decisions of the user: sequence A_1 is better than B_1, and A_2 is better than B_2. Implicitly, there are lower bounds for all weights, as they are supposed to be strictly positive. In this situation, it is relatively easy to calculate the centroid of the solution region.

When more features are taken into consideration, it might be more difficult to calculate the centroid. In such a situation, it may be more appropriate to search for any solution that is part of the solution region, e.g. by means of the Fourier-Motzkin algorithm [91].

A problem occurs when the system of inequalities that is produced from the example decisions is inconsistent. In such a case, no solution can be found that fulfills all the inequalities in the system, and we have to remove one or more inequalities in order to make the system consistent. An example can be found in Fig. 5.5, where one example decision indicated a low importance for weight w_{f_1}, and the other decision indicated a high importance for the same weight.

When the system is consistent, the solution region can also be unbounded, and in that case it is not possible to determine a centroid. We can overcome this problem by adding upper bounds to the possible values of the different weights as shown in Fig. 5.6. If there were no upper bounds in this situation, it would not be possible to determine a centroid.

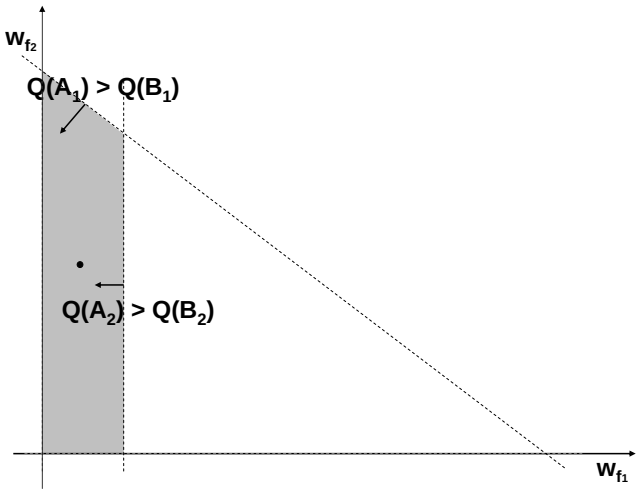


Figure 5.4: Graphical representation of the process of determining the weights using a system of inequalities.

Updating a model

In case the user rejects a certain decision taken by the agent, the agent has to update its model of the user preferences accordingly. This can be done by adding an inequality as in Eq. 5.3 to the entire system of inequalities, where version A is the sequence selected by the user and B the version selected by the agent.

In case the resulting system of inequalities becomes inconsistent, one or more inequalities have to be removed. For selecting which inequalities should be removed, the following rules can be applied.

- The newly added inequality should not be removed.
- The number of inequalities that are removed should be as low as possible.
- In case several options are possible for removing inequalities, it is preferred to remove the oldest inequalities.

When removing older inequalities, a model will be easily capable of adapting itself to changing user expectations. On the other hand, older statements have proven to be reliable, and therefore it might make more sense not to remove them so easily. Ideally, we would be able to define a metric expressing the amount of (in)consistency of one statement with respect to a number of

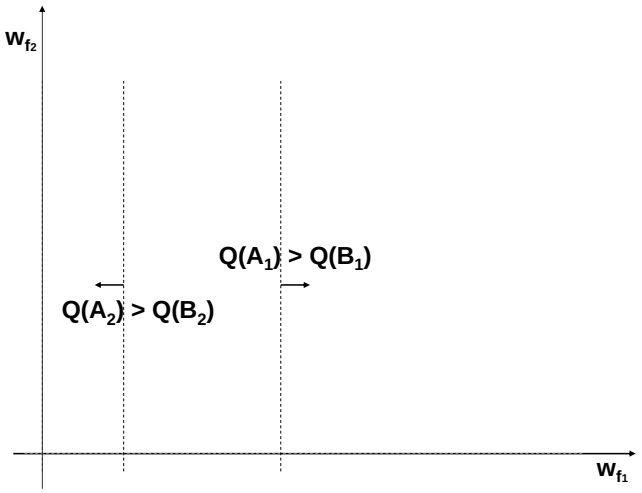


Figure 5.5: Inconsistent decisions in the SoI algorithm.

other statements, and use this metric for removing the most inconsistent statement, rather than using age as a decision criterium for removal.

Advantages

The major advantage of the SoI algorithm is that once the weights for the preferences of a particular user are determined, it is easy to determine the expected quality of the different available versions and select that version that has the highest score.

A first advantage that follows from this is that this algorithm is very interesting from a computational point of view: only a small number of multiplications, additions and comparisons are needed to find an optimal version from the set of candidate versions.

More specifically, calculating the score of one sequence corresponds to the calculation of Eq. (5.1), and has a complexity of $\theta(k)$, when k is the number of features considered in the model. When we want to select one version from a set of n candidate best versions, the overall complexity for calculating the quality values is $\theta(nk)$. In addition, for determining the best version, we need to perform $n - 1$ comparisons, which results in an overall complexity of $\theta(nk) + \theta(n) = \theta(nk)$. As we can consider k as a (small) constant value (in our test we describe in the next chapter, three features were used), we can safely say that this selection process is linear in terms of the amount of candidate solutions.

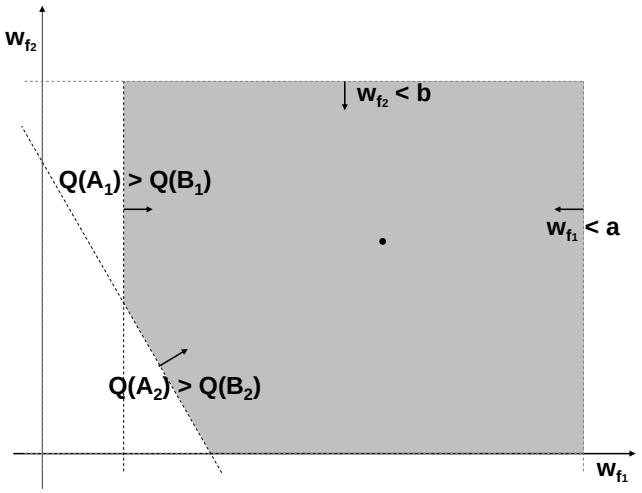


Figure 5.6: An unbounded solution region in the SoI algorithm.

Another advantage that is a consequence of the simplicity of the algorithm is that we can express the content negotiation problem as defined in Def. 5.5 by means of the Universal Constraint Descriptor (UCD) that is standardized within MPEG-21 Digital Item Adaptation. Because we can translate multiple optimization criteria into a single optimization function, any MPEG-21 compliant Adaptation Decision Taking Engine will produce the same result.

A final interesting property of using this algorithm is that it is not necessary that the set of candidate solutions only contains Pareto optimal solutions. It is straightforward to prove that the version that is considered to be the best version according to this algorithm, will always belong to the Pareto frontier.

Theorem 5.1. *Suppose we have a content negotiation problem that takes into account the preferences of a user u by means of a weighted sum, with constraints $Cts \subset C$ and features $F \subset \mathbb{P}$. Then, the optimal version p' of a parcel $p \in P$ is always a Pareto optimal version:*

$$p' \in PF_F(\langle p \rangle_{Cts}). \tag{5.4}$$

Proof. From Def. 5.5, we know that

$$p' = \arg \max_{v \in \langle p \rangle_{Cts}} \sum_{f \in F} w_f^u v_f. \tag{5.5}$$

Suppose that there exists a feasible version q that dominates p' in the Pareto sense:

$$(\exists q \in \langle p \rangle_{Cts})(q \succ_F p'). \tag{5.6}$$

Then, according to Def. 5.2:

$$(\forall f \in F)(q_f \geq p'_f) \wedge (\exists f \in F)(q_f > p'_f). \tag{5.7}$$

Therefore, as all weights are defined as strictly positive:

$$\sum_{f \in F} w^u_f q_f > \sum_{f \in F} w^u_f p'_f, \tag{5.8}$$

which is in conflict with our initial assumption: $\sum_{f \in F} w^u_f p'_f$ is supposed to be the maximum possible value. As a consequence, we can conclude that

$$(\nexists q \in \langle p \rangle_{Cts})(q \succ_F p'), \tag{5.9}$$

and therefore, according to Def. 5.3,

$$p' \in PF_F(\langle p \rangle_{Cts}). \tag{5.10}$$

□

Disadvantages

As we have already mentioned, one of the disadvantages of the way of calculating the weights in the SoI algorithm is that it is not guaranteed that we will find a solution based on a set of example decisions. Therefore, it may be possible that one or more example decisions have to be removed when training the agent, in case the resulting system is inconsistent.

When no upper bounds on the values are used, it is also possible that no solution can be found because the system yields an unbounded region. In that case, additional training information may be needed, which can be annoying for the user. If we do use upper bounds for avoiding such problems, the choice of the values for the upper bounds will have a serious impact on the solution that will be generated. This can be seen in Fig. 5.4: when the upper bounds for w_{f_1} or w_{f_2} are modified, the centroid of the solution region will change as well. As a consequence, the upper bounds have to be selected very carefully, e.g. based on the results of an experiment in which several test persons are involved.

Another important disadvantage of this algorithm is that it relies on a linear behavior of the quality measures F in terms of the actual value or utility as experienced by the user. More specifically, a change δ in terms of feature $f \in F$ should cause a change in the subjective quality of approximately $k \cdot \delta$, with k some constant value. Unfortunately, this is not always the case. As an example, suppose that we use the frame rate as a quality measure. The

difference in frame rate between 30 fps and 15 fps is 15 fps, but in terms of subjective quality it is much less significant than the difference between 15 fps and 7.5 fps. Similarly, but less extreme, this is also the case for PSNR values: the difference between 46 and 48 dB will probably be less noticeable than the difference between 32 and 34 dB.

5.8.2 1ARC

General description

The second algorithm that we present for capturing user preferences related to visual quality, is called 1ARC [92], which is a nearest-neighbor approach. In this approach, each object (in our case a video sequence) is represented by its coordinates, based on the features of the object (this can be the same set F that we used in the previous algorithm). When a user states that version A is better than version B, an arrow will be drawn from B to A (denoted $B \rightarrow A$), as can be seen in Fig. 5.7. This happens for every pair of sequences that is used for training the model.

When we want to use such a model for predicting whether version X is better than version Y, we draw an arrow $Y \rightarrow X$, and detect which arrow belonging to the model best resembles the new arrow, based on the distances between the beginning and the end of the arrows. The same is done for an arrow $X \rightarrow Y$. Ultimately, the arrow that seems to be the most reliable is selected as the predicted decision.

In Fig. 5.7, arrow $B_2 \rightarrow A_2$ best resembles $Y \rightarrow X$, and $B_1 \rightarrow A_1$ best resembles $X \rightarrow Y$. The former seems to be the most reliable among both options, so we conclude that $Y \rightarrow X$ is more likely, and therefore X is considered to be better than Y.

Both during the search for the best resembling version, as well as when determining the most reliable arrow, a distance measure is needed. Of course, the accuracy of this distance measure has a significant impact on the reliability of the algorithm, as we will demonstrate in the next chapter.

For a better understanding, the algorithm we just explained is written down in pseudo code in Algorithm 5.2. In this algorithmic description, s_X denotes the similarity of $B \rightarrow A$ with the most resembling $Y \rightarrow X$ (lower values indicate a higher similarity), and s_Y the most resembling $X \rightarrow Y$, in which smaller values denote a higher resemblance. If s_X is smaller than s_Y, the most likely situation would be $Y \rightarrow X$, and X would be selected as the best version.

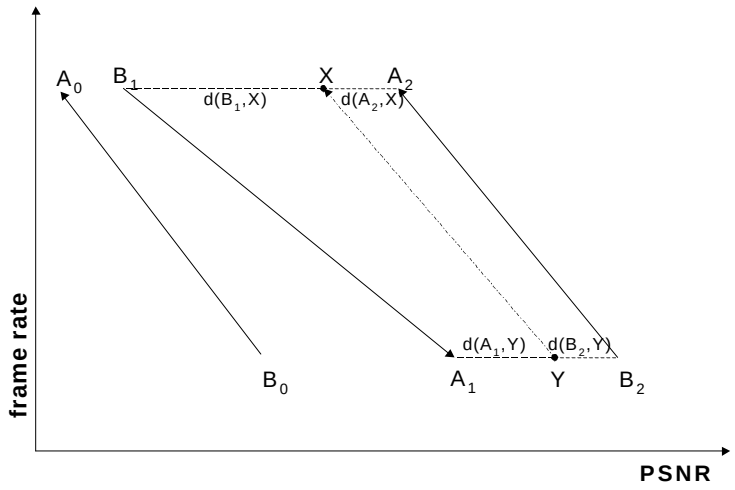


Figure 5.7: Explanation of the 1ARC algorithm.

Constructing a model

The construction of a model is straightforward in the 1ARC algorithm when a number of example pairwise preferences is available. The model itself is nothing more than a set of arrows, as in Fig. 5.7, that correspond with the example preferences given by the user.

Updating a model

Updating a model when the user recalls a decision taken by the agent can also be rather straightforward. In the first place, the arrow that lead to the wrong decision taken by the agent, is removed from the model. Second, a new arrow is added, having its tail at the version that was wrongly selected by the agent and its head at the version selected by the end user.

Advantages

The most important advantage of this algorithm is the simplicity for generating and updating a model, as well as for representing it.

A second interesting property of the 1ARC algorithm is that it is less sensitive to the nonlinear behavior of the properties of the different versions that are taken into account. It is the distance measure that is used, rather than the features themselves, that has a serious influence on the reliability of the algorithm.

Algorithm 5.2 An algorithmic description of comparing solutions X and Y using the 1ARC algorithm. In this description, $d(.,.)$ is a metric describing the *distance* or resemblance between the characteristics of two video sequences.

```
function FINDBEST(X,Y,model)  //X and Y are versions to be compared,
the model is a set of comparisons of the form B_i → A_i
    s_X := ∞
    s_Y := ∞
    for all B → A in model do
        if d(A,X) + d(B,Y) < s_X then
            s_X := d(A,X) + d(B,Y)
        end if
        if d(B,X) + d(A,Y) < s_Y then
            s_Y := d(B,X) + d(A,Y)
        end if
    end for
    if s_X ≤ s_Y then
        return X
    else
        return Y
    end if
end function
```

Disadvantages

The main disadvantage of this approach lies in the complexity of the evaluation of the model. Whereas the construction of a model is easy and cheap in terms of computational cost, predicting a decision by comparing two sequences is more costly. Unfortunately, the first activity will not occur frequently, whereas the second activity will.

It would be interesting to determine the computational complexity of selecting the best version from a set of candidate versions, and compare this with the complexity we calculated for the SoI algorithm. If we suppose that we use a Euclidian or Manhattan distance as the distance metric $d(A, B)$, this calculation has a complexity of $\theta(k)$, with k the number of features taken into account. The for-loop of the algorithm described in Algorithm 5.2, will be executed t times, with t the number of comparisons that are part of the training set. Therefore, the process of comparing two candidate versions has a complexity of $\theta(tk)$. When we want to find the best version from a set of n versions, the total complexity is $\theta(ntk)$.

When comparing the complexity of selecting one best version from a set

Algorithm 5.3 Description of the algorithm for determining an optimal version using the 1ARC algorithm. The algorithm is constructed in such a way that it is certain that a Pareto optimal solution will be found.

```
    function 1ARC(V,model)              //V is the set of candidate versions
 2:     o := V[0]                       //Currently optimal version
        V := V - V[0]
 4:     for all v in V do
            if not dominates(o,v) then
 6:             if dominates(v,o) then
                    o := v
 8:             else
                    o := FindBest(o,v,model)
10:            end if
            end if
12:    end for
       return o
14: end function
```

of versions, we see that the complexity of the 1ARC algorithm is a factor t larger: it increases linearly with the size of the training set. Note that this factor does not occur in the case of using a weighted sum, because the size of the training set only influences the complexity of determining the weights, and this is a process that will not occur frequently. As a consequence, from a computational point of view, using systems of inequalities is more interesting than using the 1ARC algorithm.

In the calculation of the computational complexity, we did not take into account another disadvantage of the 1ARC algorithm: in theory, it may be possible that we end up with a solution that is not Pareto optimal. This can be avoided in two ways: either we do a pre-filtering on the set of candidate versions using Algorithm 5.1, or we check for domination when comparing two versions before we apply the nearest neighbor algorithm. This approach is shown in Algorithm 5.3.

When we apply one of these techniques for ensuring that a Pareto optimal solution will be found, we need to recalculate the computational complexity. According to Sect. 5.3.1, the lower bound for the pre-filtering process is $\Omega(nk)$ and the upper bound is $\mathcal{O}(n^2k)$. In the best case, only one Pareto optimal solution is present; in that case no additional calculations are needed, so the total complexity stays at $\Omega(nk)$. In the worst case, all n solutions are in the Pareto frontier. In that case, the total complexity is $\mathcal{O}(n^2k) + \mathcal{O}(ntk) =$

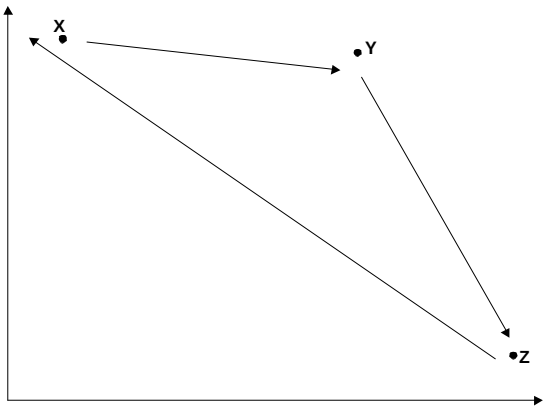


Figure 5.8: Influence of the order of evaluation in the 1ARC algorithm.

$\mathcal{O}(nk(t+n))$.

In the method described in Algorithm 5.3, the lower bound occurs when the first solution dominates all others. In that case, the *dominates* function, with complexity $\theta(k)$ is executed exactly n times, and no other calculations are needed. As a consequence, the lower bound is $\Omega(nk)$. In the worst case, no solution dominates another one. In this case, in each loop, the *dominates* function is executed twice and the *FindBest* function once, so the total complexity for the body of the loop is $2\mathcal{O}(k) + \mathcal{O}(tk) = \mathcal{O}(tk)$. As this loop is executed n times, the total upper bound is $\mathcal{O}(ntk)$.

As a consequence, Algorithm 5.3 is the most interesting way of using the 1ARC algorithm. It is more efficient than pre-filtering the Pareto frontier, and it is not even worse than an algorithm that does not take into account the domination of the candidate solutions.

A final, but important disadvantage of the 1ARC algorithm is that it is possible that in certain situations the outcome of the algorithm depends on the order in which the candidate versions are evaluated. This is shown in Fig. 5.8. The information that belongs to the training set is represented by means of arrows, as in Fig. 5.7. Suppose that we want to determine which of the candidate solutions X, Y and Z is the best. According to the 1ARC algorithm, any of the three candidate versions can be considered the best version, depending on the order of evaluation: if we follow the alphabetic order, Z is the best version, but if we first compare X and Z, and then Y, Y is considered to be the best version.

We developed an approach that is aimed at suppressing this problem. It starts from a set of Pareto optimal candidate solutions. Within this set, all candidate solutions are compared, using a modified version of Algorithm 5.2:

instead of returning a boolean value, the difference between s_Y and s_X is returned. We call this value the likeliness that version X is better than version Y; when it is positive X is considered to be better than Y:

$$l(X, Y) = s_Y - s_X. \tag{5.11}$$

The total likeliness that a version v_i is the best version of the set of Pareto optimal solutions PF, is defined as follows:

$$L(v_i) = \sum_{v_j \in PF} l(v_i, v_j). \tag{5.12}$$

Note that $l(X, Y) = -l(Y, X)$ and that $l(X, X) = 0$. The version v_i that maximizes $L(v_i)$ is selected as the best version.

A disadvantage of the proposed approach is that the complexity of calculating an optimal solution is increased even more: it is no longer sufficient to do $n - 1$ comparisons in the set of candidate solutions. As $l(Y, X)$ can be calculated from $l(X, Y)$, and $l(X, X)$ doesn't need to be calculated, $\frac{n.(n-1)}{2}$ comparisons have to be made to determine all $L(v_i)$ values. As a consequence, the complexity of selecting an optimal version is $\theta(n^2tk)$ instead of $\theta(ntk)$.

Moreover, the proposed method expects a set of Pareto optimal solutions. This means that, in the worst case, the total complexity is $\mathcal{O}(n^2k)+\mathcal{O}(n^2tk) = \mathcal{O}(n^2tk)$. Therefore, in situations where a large number of Pareto optimal solutions can occur, this method might not be useful.

5.9 Related work

In the previous sections, we proposed to use software agents that model the preferences of the end user for solving the problem of making trade-offs between multiple quality aspects of a video sequence. We were the first to propose such a user-centric solution, but other approaches can be found in the literature as well.

In video transcoding [49], it was recognized before that an intelligent combination of different types of quality reduction could achieve a better overall quality or utility. In this context, the broader term *utility*, originating from the domain of economics, is often used instead of *quality*; it expresses the overall user's satisfaction.

Several implementations of making a trade-off between reducing the frame rate and allowing distortions exist. In [93], a dynamic frame rate control algorithm is proposed, that uses motion information in order to reduce the frame rate without introducing too much jerkiness. A similar approach, where end

users are allowed to express their preferences by means of a jerkiness threshold, is presented in [94].

In [79], Reed and Lim propose a mechanism for automatically making trade-offs between the temporal and spatial resolution and the distortion. They define an integer programming formulation and present an algorithm for computing an optimal solution when the bit rate is controlled by jointly adapting the frame rate, the resolution and the distortion.

Since a few years, fully scalable video coding is becoming more and more mature. This stimulated research on formalizing and solving the decision taking problem in making trade-offs between multiple quality aspects.

In [72], Mukherjee et al. present an architecture based on MPEG-21 Digital Item Adaptation for realizing the adaptation and the delivery of a scalable video sequence in a format-agnostic way. They also incorporate a decision-taking mechanism in this architecture, but they do not try to solve the problem of making trade-offs between quality aspects. In [47], they further formalize this decision-taking process.

The formalization proposed by Chang and Vetro [71] has some similarities with the formulation of Mukherjee et al. Both describe a content negotiation problem as a constrained optimization problem.

A possible solution based on this formulation can be found in [95]. The major difference between the approach of Wang et al. and our user-centric approach, is that they try to learn the optimal adaptation by means of a subjective quality evaluation and by assuming that sequences with similar characteristics are likely to have a similar optimal adaptation. In other words, whereas we look at the preferences of individual users, they look at the characteristics of the sequence that is to be adapted.

In [96], Önür and Alatan describe the optimal adaptation selection as a multi-criteria adaptation process. They consider the overall utility of an adapted version of a video sequence as a weighted sum of several independent utility functions. As it is easier to obtain an accurate modelling of these independent utility functions, by means of exhaustive subjective tests, the overall utility value will be more reliable as well. Similar to what is described in this paper, they also produce a set of Pareto optimal candidate solutions. They consider the solution that utilizes the available resources to the fullest extent the one that is expected to be the best from the Pareto set.

5.10 Conclusions and original contributions

In this chapter, we demonstrated that it is better to define a content negotiation process as a multi-criteria optimization problem instead of a constrained opti-

mization problem with one criterion, as it better reflects the reality. We were the first to define such a problem in this way [3], but this approach is already adopted by others [47, 72], and is even reflected in the MPEG-21 Digital Item Adaptation specification.

For a better understanding of the consequences of using a multi-criteria approach, we introduced the most important concepts of the Pareto theory, which is a mathematical approach for handling multi-criteria optimization problems. We also modified the formal definition of a content negotiation problem with one criterion, as defined in the previous chapter, to a content negotiation problem with multiple optimization criteria.

Because solving a multi-criteria optimization problem is more complex than an optimization problem involving one criterion, we looked at the consequences of following this approach regarding the computational complexity. We did this from a theoretical point of view, by means of complexity analysis, as well as by means of time measurements on some practical use cases.

As a multi-criteria optimization problem can produce multiple possible solutions that cannot be compared without further information, we gave a brief overview of existing frameworks for assisting a user in making a good decision. Unfortunately, due to the nature of the problem we want to handle, these approaches could not be used.

Instead, we looked at the fundamental concepts behind interface agents that act on behalf of an end user. From these concepts, we derived a number of requirements for possible algorithms that can be used by an agent that selects appropriate solutions in a content negotiation scenario.

We presented two different algorithms that are capable of constructing a model of the preferences of an end user from a set of example decisions. Once constructed, such a model can be used for predicting other decisions that should be taken by the user.

The first algorithm, referred to as the System of Inequalities (SoI) algorithm, is based on the assumption that the overall quality of a video sequence can be determined by means of a weighted sum of the features of the sequence. These weights are user-dependent and represent the preferences of the user. From a set of example decisions, a system of linear inequalities can be derived, in which the weights are the unknown values.

The second algorithm that we proposed is the 1ARC algorithm, which is a kind of nearest-neighbor approach. We were the first to apply this algorithm to the domain of video quality. In the 1ARC algorithm, each example decision is represented by means of an arrow, using the features of each instance as a vector of coordinates. When a decision has to be predicted, the arrow that best resembles the arrow that is to be predicted, is selected as the most reliable

prediction.

In the next chapter, we will describe how we set up a test for measuring the prediction accuracy of both algorithms. We will describe the results, and the influence of particular configuration parameters and minor modifications on the algorithms on their performance. This way, we can draw some additional conclusions on the algorithms that we presented in this chapter.

Our research in the domain of multi-criteria optimization, applied to video communication, resulted in the following publications.

1. Sam Lerouge, Peter Lambert, and Rik Van de Walle. Multi-criteria optimization for scalable bitstreams. In *Visual Content Processing and Representation, 8th International Workshop VLBV 2003*, volume 2849 of *Lecture Notes in Computer Science*, September 2003.

2. Sam Lerouge, Robbie De Sutter, and Rik Van de Walle. Personalizing quality aspects in scalable video coding. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME) 2005*, July 2005.

Chapter 6

Performance of capturing user preferences

6.1 Introduction

In the previous chapter, we presented two different algorithms that are capable of constructing a model of the preferences of a user regarding the aspects of visual quality, from a set of example decisions taken by the user by means of pairwise comparisons. We discussed some of the general properties, advantages and disadvantages of both methods.

The actual prediction performance, the reliability of both algorithms, was not discussed in the previous chapter. This performance can only be measured in a meaningful way by means of a subjective test. In such a test, a number of test persons is asked to observe a number of video sequences and assign a value to these sequences, either directly or indirectly. The results from this test are compared with the results obtained from objective data, i.e. data based on algorithms instead of user input.

In this chapter, we describe how we have set up this test. We use the results of the test for measuring the prediction performance of both proposed algorithms. We study the influence of particular parameters and minor modifications of the algorithms on the overall result, and we compare the best versions of both algorithms to see if one of them outperforms the other one.

Before describing the test setup and the results, we give an overview of the terminology that is used in this chapter, in order to avoid any confusion on terms such as algorithm, model, method, etc.

6.2 Terminology

In this and in the previous chapter, an **algorithm** is composed of two procedures. The first procedure is capable of constructing a model of the preferences of a user based on some example decisions. The second procedure is aimed at selecting one optimal version from a set of candidate versions, taking into account the information that can be found in the model.

In this thesis, two algorithms are considered. The weight-based algorithm uses a system of inequalities for constructing the model, and is called the *SoI algorithm*. The second algorithm is an existing nearest-neighbor approach for modelling user preferences, and is called the *1ARC algorithm*.

A **model** is the output of the first step of an algorithm. It is a structural representation of the preferences of one particular end user, that can be used for predicting decisions on behalf of that user.

The SoI algorithm produces a model that consists of a set of user-dependent weights, one for each quality feature that is considered in the algorithm. In the 1ARC algorithm, a model is a set of arrows representing the example decisions given by the user. The coordinates of the head and tail of each arrow are determined by the features of the sequences involved.

A quality **feature** is a property, in the sense of the properties defined in Def. 4.7 of Chapter 4. It is a function that assigns a particular value to a parcel, or in our case a version of an entire video sequence[1]. The term *quality* feature points to the nature of the property: it says something about a particular quality aspect of a video sequence, in the sense that it is proportional to the utility it offers towards an end user. A good example of a quality feature is the frame rate: the higher the frame rate, the more the sequence will be appreciated by an end user, if all other quality aspects remain constant.

A **method** is a way of analyzing the prediction performance (the reliability) of an algorithm. In this chapter, we use two such methods. In the *test set method*, the correct prediction of pairwise decisions is measured. In the best in group method, or the *BiG method*, the correct prediction of the best version from a set of candidate versions is evaluated. Both methods are described more in detail in Sect. 6.4.

A **sequence** is one video sequence that is observed by a participant of the subjective test. In the domain of psychophysics, the broader term stimulus is used. In the test, multiple sequences produced from the same original are used. All sequences that were created from the same original (uncompressed) video sequence, are part of the same **sequence group**.

[1] In our test, we treated each video sequence as consisting of one parcel.

Table 6.1: Overview of the 6 sequences used in the subjective test, together with the target bit rate used for the reduced versions.

sequence	description	bit rate (kbps)
coastguard	Two boats crossing each other across the shore; continuous camera panning	300
stefan	Tennis player; camera is following the player, high motion from camera and subject, complex textures	400
akiyo	News anchor; no camera movement, slow movement from subject	128
foreman	Man presenting a construction yard; moderate movement from both the subject and the camera	300
mother	Mother and daughter sitting in front of a camera; no camera movement	200
silent	Woman talking in sign language; no camera movement but significant movement from the subject	200

6.3 Test setup

As we already described in Chapter 5, we were convinced that when asking the user some example decisions for training an agent, pairwise comparisons would offer the most reliable information. We used this assumption in the algorithms described in Sect. 5.8. As a consequence, it makes most sense to apply this principle of pairwise comparisons in the setup of the subjective test.

6.3.1 Sequences

We used six different original sequences in this test. A brief description of the content of the sequences can be found in Table 6.1. In order to have a better understanding of the content of the sequences, for each sequence, a frame is shown in Appendix B. All sequences were encoded using a fully scalable video codec based on the MC-EZBC algorithm that we already mentioned in Sect. 4.3.3. We preferred to use a fully scalable video codec because it is this kind of video compression we have in mind for modelling user preferences. For each sequence, a target bit rate was determined in such a way that the distortion would certainly be visible for the version having the highest resolution

Table 6.2: Distortion of all versions that are used, expressed by means of PSNR values (expressed in dB).

	CIF			QCIF		
sequence	30 Hz	15 Hz	7.5 Hz	30 Hz	15 Hz	7.5 Hz
coastguard	29.34	30.45	31.98	33.62	35.21	38.36
stefan	25.28	27.51	30.33	29.31	32.68	37.63
akiyo	36.05	36.82	37.52	40.07	40.99	41.89
foreman	32.34	33.55	35.09	36.93	39.01	41.86
mother	38.20	39.09	40.22	41.61	42.83	44.53
silent	31.48	32.42	33.25	35.07	36.51	38.21

and the highest frame rate. Otherwise, if the distortion would not be visible, we expected that it would not be possible to capture the trade-offs a user is making between the distortion and other quality aspects. The bit rate used for each sequence group is also represented in Table 6.1.

The original sequences have a CIF resolution (352 by 288 pixels), a frame rate of 30 frames per second, and are 300 frames (10 seconds) long. From each original sequence, we generated six different versions, at two different resolutions (CIF and QCIF) and three frame rates (30, 15 and 7.5 frames per second). As said, each version was encoded using the same bit rate. The resulting distortion can be found in Table 6.2, expressed by means of PSNR values. For each original frame that is shown in Appendix B, we show the corresponding distorted frame of the version at 30 frames per second and the original resolution in the same Appendix.

For calculating the PSNR values shown in Table 6.2, a reference sequence is needed, having the same frame rate and resolution, and containing no distortion. These reference sequences are obtained by generating a version at the same frame rate and resolution but with an unlimited bit rate; this procedure generates a version in which no distortion is visible. As a consequence, the values in Table 6.2 are good representatives of the distortion of all versions.

6.3.2 Presentation

If we would have wanted to present each participant all possible pairs within one sequence group, for each group $\binom{6}{2} = \frac{6!}{4!2!} = 15$ comparisons would have to be shown. Not only would this limit the amount of sequence groups that we could show, we also learned from a preliminary test that so many repetitions of the same content had a negative impact on the concentration of the participants.

As a consequence, a different approach was needed, in which not all pos-

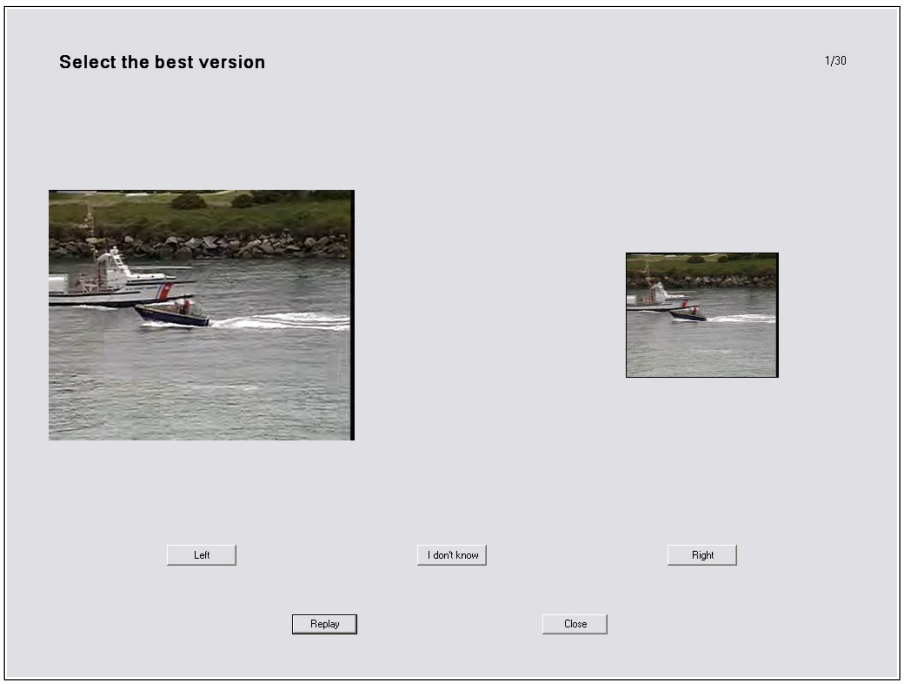


Figure 6.1: Screenshot of the subjective test.

sible pairs would be presented. We decided that during the test, we would only work with those versions that are considered to be the best among those that were already shown, according to the participant. This is possible because only one participant is interacting with the computer that is used for the test.

In addition, we wanted to avoid that a particular pair of sequences is always presented in the beginning or at the end of a session, because the concentration of the participants may be lower than in the middle of the session. Such guidelines can be found in the recommendations defined by the ITU for setting up subjective tests concerning video quality [97]. As a consequence, the selection of a particular sequence group is determined randomly during the session. Similarly, we wanted the position (left or right of the screen) to be determined randomly as well.

A screenshot of the test is shown in Fig. 6.1. After filling in a form with some personal information (such as the age, sex, education, etc.), the user is presented two sequences belonging to the same sequence group, one on the left side of the screen, another one on the right side of the screen. When the sequences are finished, the user has to select one of both versions as the best

one, or he can enter a neutral decision by declaring that he finds it impossible to choose one version. He can also ask for replaying both sequences. This kind of decision has to be taken 30 times.

The test was always performed on the same device, a laptop, in order to avoid any influence caused by the quality or the resolution of the screen that was used. Because the test was not always taken on the same location, we could not guarantee constant lighting conditions. The screen of the laptop that was used has a resolution of 1024x768 pixels, with a refresh rate of 60 Hz, and a 15 inch display size.

Algorithm 6.1 Selection process for determining the sequences that will be shown to the test subject.

```
S0 := [s0, s1, ..., s5]
   ⋮
S5 := [s30, s31, ..., s35]
S := [S0, ..., S5]
T:= [0, 0, ..., 0]
for i := 0 to 5 do
    t := random(S[i])          //Randomly select one sequence from the group
    S[i] := S[i] - t                    //Remove the sequence from its group
    T[i] := t                  //Mark the sequence as the best from its group
end for
for i := 1 to 30 do
    j := random(0..size(S)-1)            //Randomly select a sequence group
    top := T[j]
    new := random(S[j])        //Randomly select a sequence from the group
    S[i] := S[i] - new                  //Remove the sequence from its group
    if S[i] = ∅ then
        S := S - S[i]                          //Remove the sequence group
        T := T - T[i]
    end if
    if random(0..1) = 1 then
        present(top, new)            //Randomly select presentation position
    else
        present(new, top)
    end if
    if best(new) then
        T[j] := new            //Replace the best version of the current group
    end if
end for
```

The procedure for selecting which sequence pair is presented to the participant is as follows. Initially, for each sequence group, one randomly selected sequence is defined to be the best version. When a new decision has to be obtained from the user, the first thing that occurs is the random selection of one of the remaining sequence groups. From this group, a version that was not shown yet is selected. If only one version is available, the sequence group can be removed for future decisions. The selected version is then presented to the user, together with the best version of the sequence group at that moment. The location (left or right) is also determined randomly. If the user states that the new version is better, it becomes the best version in that group. The whole procedure of selecting and presenting pairs of video sequences, as we have just described, is written down in pseudo code in Algorithm 6.1.

6.3.3 A note on the presentation of different resolutions

Some people might argue that the spatial resolution of a video sequence should not be treated as a configurable quality aspect. Instead, the largest resolution that still fits the display resolution of the target device should be selected. We do not agree that this solution is correct in all scenarios. For example, consider a scenario in which a display with a high resolution is used (such as a laptop or a tablet PC), but with a limited network connection (such as a GPRS connection). If in such a situation a version would be selected that corresponds with the screen resolution, the distortion would probably be unacceptably high. As a consequence, we think that the spatial resolution is indeed an aspect that should be considered when selecting an optimal adaptation configuration.

As can be seen from Fig. 6.1, we chose to present the low resolution versions in their actual resolution, rather than to present an upsampled version of the sequence at the same resolution as the high resolution versions.

Both approaches have their advantages and disadvantages. When presenting upsampled versions, several algorithms can be used for the upsampling process [98], and there is a risk that the selected upsampling algorithm can have an influence on the result of the test. As a consequence, we might be testing the performance of the upsampling algorithm, rather than the subjective preference of a user towards high resolution or low resolution versions.

On the other hand, the approach that we used has some disadvantages as well. In particular, the low resolution versions always appear sharper and less distorted, not only because more bits are available per pixel, but also because distortions are less visible as their sizes are smaller.

When the approach presented in this thesis would be implemented in a commercial product (say, a multimedia player), both approaches are possible,

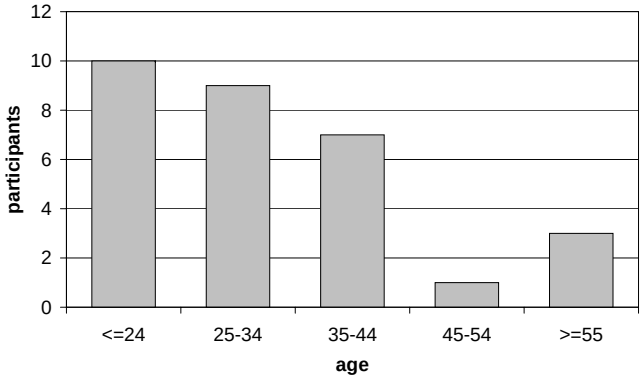


Figure 6.2: Age histogram of the 30 participants.

but it is recommended that the way the training sequences are presented to the user is the same as the way video sequences will be shown during actual usage. That is, when the player always presents its sequences in their actual resolution, the training sequences should also be presented in their actual resolution. In contrast, when the player always rescales video sequences, the training sequences should also be upsampled in the same way before they are presented to the user.

6.3.4 Participants

The results that we describe in this chapter are obtained from 30 participants, of which 19 men and 11 women. They had different educational backgrounds, and there was a large distribution in terms of the frequency of watching television or visiting a movie theater. From the 30 participants, 2 can be considered *visual experts*, i.e. they are familiar with the aspects of digital image and video coding from their professional background. Because their results did not differ much from the other participants, we decided not to exclude them from the test. The distribution of the age of the participants can be found in Fig. 6.2, where an age histogram is shown.

6.4 Evaluation methods

6.4.1 Basic definitions

Before explaining the two methods we use for evaluating the reliability of the algorithms presented in the previous chapter, we introduce some definitions

that will help formalizing the description of these methods.

Definition 6.1. *The set of all **sequences** is denoted $\mathbb{S}$. This set of sequences can be split up into sequence groups, where each sequence of the same group is generated from the same original sequence. We denote a **sequence group** as follows: $\mathbb{S}_i$.*

It is clear that $\mathbb{S}_i \subset \mathbb{S}$. Furthermore, there are no sequences that are not part of a sequence group: $\mathbb{S} = \bigcup \mathbb{S}_i$. Moreover, all sequence groups are disjunct: $(\forall i \neq j)(\mathbb{S}_i \cap \mathbb{S}_j = \emptyset)$.

In this test, we used six sequence groups, $\mathbb{S}_0, \ldots, \mathbb{S}_5$, each consisting of six different sequences: $\#\mathbb{S}_i = 6$.

Definition 6.2. *When user u states that sequence a is better than sequence b, we write $a \succ_u b$. When he states that he could not decide if he prefers a or b, we write $a \approx_u b$, which is equivalent to $b \approx_u a$.*

Note that, as can be seen in Algorithm 6.1, $a, b \in \mathbb{S}_i$: when two versions are compared, they are always part of the same sequence group.

In addition, it is useful to define a more general version of the $\succ_u$ operator, that makes indirect comparisons stated by the same user. This operator is defined in the following definition.

Definition 6.3. *A sequence a is preferred to a sequence b by user u, if one out of four conditions is satisfied:*

$$a >_u b \Leftrightarrow \begin{cases} a \succ_u b & \vee \\ (\exists c)(a \succ_u c \wedge c >_u b) & \vee \\ (\exists c)(a >_u c \wedge c \approx_u b) & \vee \\ (\exists c)(a \approx_u c \wedge c >_u b) & \end{cases}$$

The information that we obtain from a user that participated in our test, is a set of pairwise decisions of both types defined in Def. 6.2. Mostly, we are only interested in those decisions where the user stated a preference, not a neutral decision[2]. Therefore, we separated the following sets of decisions.

Definition 6.4. *The set of all statements for which user u stated that he preferred one sequence over another is called the **preference set**, and is denoted $\mathcal{P}_u$, with $\mathcal{P}_u = \{a_i \succ_u b_i\}$.*

[2]During a small test that was set up as a preparation of the test described in this chapter, we noticed that it was not correct to consider these neutral decisions as statements expressing that the quality of both sequences was approximately equal, because this would create too much inconsistent behavior.

Definition 6.5. *The set of all statements for which user u stated that he could not choose between two sequences is called the **indifference set**, and is denoted $\mathcal{I}_u$, with $\mathcal{I}_u = \{a_i \approx_u b_i\}$.*

Definition 6.6. *The set of all statements entered by user u is called the **decision set**, and is denoted $\mathcal{D}_u$, with $\mathcal{D}_u = \mathcal{P}_u \cup \mathcal{I}_u$.*

A model of the preferences of a participant is always constructed from a subset of the preference set, and will be used for predicting decisions taken on behalf of that user.

Definition 6.7. *A **model** of user u, based on the training set t, where $t \subset \mathcal{P}_u$, using algorithm alg, is denoted $\mathcal{M}_t^{\mathrm{alg}}$.*

A model can be used as a relational operator, as it is capable of comparing two sequences. We write $a\,\mathcal{M}_t^{\mathrm{alg}}\,b$ when the model considers a to be better than b.

When using the 1ARC algorithm, all training sets produce valid models. In the SoI algorithm, training sets can be inconsistent or unbounded; in that case no model can be produced. This situation is denoted as follows.

Definition 6.8. *When a training set t fails to produce a model, the model is said to be **inconsistent**. This is denoted as follows: $\mathcal{M}_t^{\mathrm{alg}} = \perp$. This is equivalent with saying that nothing can be predicted from the model: $\mathcal{M}_t^{\mathrm{alg}} = \perp \Leftrightarrow (\nexists x, y)(x\,\mathcal{M}_t^{\mathrm{alg}}\,y)$.*

Before concluding this section containing the basic definitions that are used in the following sections, we need to introduce the notation that we use for denoting the average of a series of values.

Definition 6.9. *The average of a series of function values $F(v)$, where v can be any element of a set V, is denoted by means of $\overline{F(v)}|_{v:V}$. This value can be determined as follows:*

$$\overline{F(v)}|_{v:V} = \frac{\sum_{v \in V} F(v)}{\#V}.$$

6.4.2 Amount of inconsistent training sets

For the SoI algorithm, where models can be inconsistent, it is useful to know the probability of ending up with an inconsistent model. Therefore, we define the amount of inconsistent models as follows.

Definition 6.10. *The rate of inconsistent models of size n for user u, using algorithm* alg*, is denoted $\mathcal{I}^{\text{alg}}(u,n)$, and is determined as follows: $\mathcal{I}^{\text{alg}}(u,n) = \overline{I(\mathcal{M}_t^{\text{alg}})}|_{t:T}$, where T contains all training sets of size n for user u: $T = \{t \subset \mathcal{P}_u | \#t = n\}$, and*

$$I(\mathcal{M}_t^{\text{alg}}) = \begin{cases} 1 & \textit{when} \quad \mathcal{M}_t^{\text{alg}} = \bot, \\ 0 & \textit{otherwise} \end{cases}.$$

The rate of inconsistent models of an algorithm alg *for a specific size of a training set n, for a set of users U, is denoted $\mathcal{I}_U^{\text{alg}}(n)$, and is defined as follows: $\mathcal{I}_U^{\text{alg}}(n) = \overline{\mathcal{I}^{\text{alg}}(u,n)}|_{u:U}$.*

In a similar way, we can define the amount of training sets that do produce a model. In that case, we talk of a consistent model.

Definition 6.11. *The rate of consistent models of size n for user u, using algorithm* alg*, is denoted $\mathcal{C}^{\text{alg}}(u,n)$, and is determined as follows: $\mathcal{C}^{\text{alg}}(u,n) = 1 - \mathcal{I}^{\text{alg}}(u,n)$.*

The rate of consistent models of an algorithm alg *for a specific size of a training set n, for a set of users U, is denoted $\mathcal{C}_U^{\text{alg}}(n)$, and is defined as follows: $\mathcal{C}_U^{\text{alg}}(n) = \overline{\mathcal{C}^{\text{alg}}(u,n)}|_{u:U}$.*

6.4.3 Test set method

In the test set method, the basic idea is that the preference set of a particular user is split up into a **training set** and a **test set**. The training set is used for constructing a model, and this model is used for predicting all decisions in the preference set that are not part of the training set. This is written down formally in the following definition.

Definition 6.12. *The reliability of a training set $t \subset \mathcal{P}_u$ for user u, using algorithm* alg*, is denoted $\mathcal{T}^{\text{alg}}(t)$, and is determined as follows:*

$$\mathcal{T}^{\text{alg}}(t) = \begin{cases} \bot & \textit{when} \quad \mathcal{M}_t^{\text{alg}} = \bot, \\ \frac{c}{m} & \textit{otherwise} \end{cases},$$

with $c = \#\{x \succ_u y \in \{\mathcal{P}_u \setminus t\} | x \; \mathcal{M}_t^{\text{alg}} \; y\}$, the number of correctly predicted statements in the test set $\mathcal{P}_u \setminus t$, and $m = \#\{\mathcal{P}_u \setminus t\}$, the size of the test set.

Mostly, we are interested in the average reliability of a particular algorithm for a fixed training set size, for one user or for all participants of our test.

Definition 6.13. *The reliability of all training sets of size n for user u, using algorithm* alg*, is denoted $\mathcal{T}^{\text{alg}}(u,n)$, and is determined as follows: $\mathcal{T}^{\text{alg}}(u,n) = \overline{\mathcal{T}^{\text{alg}}(t)}|_{t:T}$, with T the set of all valid training sets of size n for user u: $T = \{t \subset \mathcal{P}_u | \#t = n \wedge \neg(\mathcal{T}^{\text{alg}}(t) = \bot)\}$.*

The reliability of an algorithm alg *for a specific size of a training set* n*, for a set of users* U*, is denoted* $\mathcal{T}_U^{\text{alg}}(n)$*, and is defined as follows:* $\mathcal{T}_U^{\text{alg}}(n) = \overline{\mathcal{T}^{\text{alg}}(u,n)|_{u:U}}$.

The major advantages of the test set method are that the results are easier to understand (for example, an accuracy of about 50% is what we would obtain when taking random decisions), and that the information that is used for evaluation is independent of the training set.

6.4.4 Best in Group method

In the Best in Group method, sometimes abbreviated to BiG method, we observe in how many cases a prediction algorithm is capable of predicting a correct best version within a sequence group. Before explaining the method itself, we need to explain what we mean when we talk about the best sequence in a sequence group.

Definition 6.14. *The set of best versions of a sequence group* $\mathbb{S}_i$ *according to user* u *is denoted* $\widehat{\mathbb{S}}_{iu}$*, and is determined as follows:* $\widehat{\mathbb{S}}_{iu} = \{s \in \mathbb{S}_i | (\nexists s' \in \mathbb{S}_i)(s' >_u s)\}$.

When looking at Algorithm 6.1, we can conclude that the top sequence T[i] at the end of the test will always be part of the set of best versions from that group, together with all versions for which the user considered it impossible to declare if they where better or worse than the ultimate best version: T[i] $\approx_u s$, s being another version belonging to the same sequence group.

In a similar way, a prediction algorithm has to select one best version from a sequence group $\mathbb{S}_i$ using a model of user u. If that version is part of the set of best versions $\widehat{\mathbb{S}}_{iu}$, the algorithm made a correct prediction. Note that, as explained in Sect. 5.8.2, in the case of the 1ARC algorithm, the order of evaluation can have an influence on the ultimate decision.

Definition 6.15. *The best version of a sequence group* $\mathbb{S}_i$ *according to model* $\mathcal{M}_t^{\text{alg}}$ *is denoted* $\mathcal{B}(\mathbb{S}_i, \mathcal{M}_t^{\text{alg}})$.

In the BiG method, the reliability of a training set $t \subset \mathcal{P}_u$ *for user* u*, using algorithm* alg*, is denoted* $\mathcal{B}^{\text{alg}}(t)$*, and is determined as follows:*

$$\mathcal{B}^{\text{alg}}(t) = \begin{cases} \perp & \textit{when} \quad \mathcal{M}_t^{\text{alg}} = \perp, \\ \frac{c}{g} & \textit{otherwise} \end{cases},$$

with $c = \#\{0 \leq i \leq g-1 | \mathcal{B}(\mathbb{S}_i, \mathcal{M}_t^{\text{alg}}) \in \widehat{\mathbb{S}}_{iu}\}$*, the number of correctly predicted best versions, and* g *the total number of sequence groups.*

Again, we are mostly interested in the average reliability of a particular algorithm for a fixed training set size, for one user or for all participants of our test.

Definition 6.16. *Using the BiG method, the reliability of all training sets of size n for user u, using algorithm* alg*, is denoted $\mathcal{B}^{\mathrm{alg}}(u,n)$, and is determined as follows: $\mathcal{B}^{\mathrm{alg}}(u,n) = \overline{\mathcal{B}^{\mathrm{alg}}(t)}|_{t:T}$, with T the set of all valid training sets of size n for user u: $T = \{t \subset \mathcal{P}_u | \#t = n \wedge \neg(\mathcal{T}^{\mathrm{alg}}(t) = \bot)\}$.*

The reliability of an algorithm alg *for a specific size of a training set n, for a set of users U, is denoted $\mathcal{B}_U^{\mathrm{alg}}(n)$, and is defined as follows: $\mathcal{B}_U^{\mathrm{alg}}(n) = \overline{\mathcal{B}^{\mathrm{alg}}(u,n)}|_{u:U}$.*

The most important advantage of the BiG method is that in practice, only the correct selection of the best version is important. In the test set method, an incorrect ordering of two bad versions will be taken into account, even though such an error is not so problematic, because none of both versions will be selected as the best version in a content negotiation scenario. The BiG method does not take these mistakes into account, as it better matches with the context of the content negotiation process.

A disadvantage is that the evaluation information (the $\widehat{\mathbb{S}}_{iu}$ sets) is not independent of the training set: all statements belonging to the preference set, including those that are part of the training set, are needed for determining the best versions. Furthermore, the interpretation of the resulting numbers is not as obvious as in the case of the test set method. In the next section, we introduce some reference points to overcome this problem.

6.5 General analysis

In this section, we want to observe the behavior of the participants of the test as a whole. We treat them as a homogeneous group, to see what sorts of conclusions we can draw.

For each sequence s, we counted the amount of users that selected that version as the best version for its sequence group: $\#\{u \in U | s \in \widehat{\mathbb{S}}_{iu}\}$. For each sequence group, we can plot this information in a chart. This is what is shown in Fig. 6.3 for the *mother* sequence group. Note that because participants may have selected multiple versions of a sequence group as the best versions ($\#(\widehat{\mathbb{S}}_{iu}) \geq 1$), the sum of the percentages may exceed 100%. We can clearly see that there is no agreement regarding the preference of the different participants. Even the most popular version can only satisfy 50% of the participants; the other participants considered at least one of the other versions to be better.

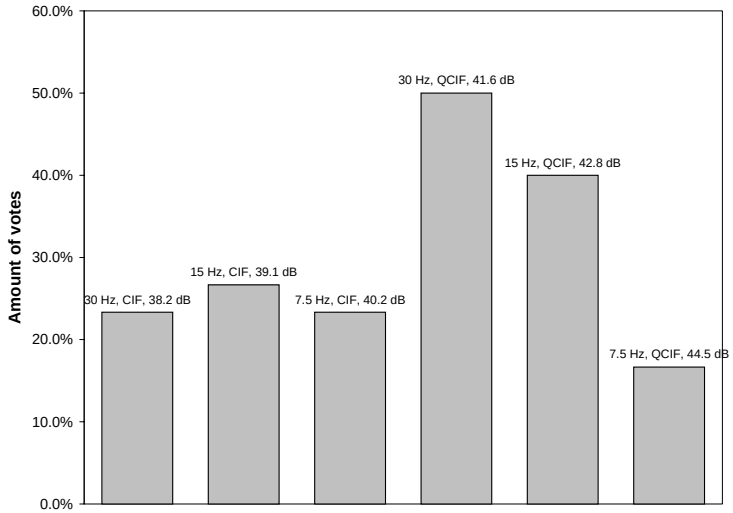


Figure 6.3: Percentage of users that selected a particular version of the *mother* sequence as the best.

In Table 6.3, we show for all sequences which version was mostly preferred by the participants, and how frequently they were ranked as (one of) the best version(s) of that sequence group.

From this information, we can deduce a number of *reference points* that can be used in the best in group method. A first reference point is the theoretically optimal reliability that can be obtained when using no personalization. This corresponds with a scenario where a content provider uses a test panel that has to select the best version for each sequence and for each bit rate the content provider wants to offer. It is the optimal result that any algorithm can achieve

Table 6.3: Maximum result possible when selecting the best version of a sequence without doing personalization.

sequence	maximum	properties
coastguard	53.3 %	15 Hz, CIF
stefan	56.7 %	30 Hz, QCIF
akiyo	66.7 %	15 Hz, QCIF
foreman	60.0 %	30 Hz, QCIF
mother	50.0 %	30 Hz, QCIF
silent	50.0 %	30 Hz, QCIF

that is targeted at selecting an optimal version from a set of candidate versions, but that does not take any user-specific information into account. Examples of such algorithms can be found in [95] and [96], amongst others.

This first reference point is calculated by taking the average of the maximum reliability for each sequence, as is shown in Table 6.3, which is in this case 56.1%. Formally, this value is determined as follows. First, this is how the maximum reliability for a particular sequence group is calculated:

$$RP_1(\mathbb{S}_i) = \max_{s \in \mathbb{S}_i} \frac{\#\{u \in U | s \in \widehat{\mathbb{S}}_{iu}\}}{\#U}. \tag{6.1}$$

From this, the average value is determined:

$$RP_1 = \overline{RP_1(\mathbb{S}_i)}|_{i:\{0,\ldots,g-1\}}, \tag{6.2}$$

g being the amount of sequence groups that are considered.

If one of the prediction algorithms is capable of crossing this border when using the BiG method, we can say that we achieved true personalization when using that algorithm, because its performance is better than the optimal performance any algorithm can achieve when using no user-specific information.

The scenario of a content provider using a test panel, as associated with the first reference point, is probably not very realistic: it would be very costly to use a test panel for every sequence and for every bit rate that would be offered. In a more practical situation, a test panel would only be contacted once, and for every range of bit rates, the optimal overall frame rate and resolution would be determined. The performance of using this approach will be used further on as a second reference point (RP_2).

Table 6.4 summarizes for each possible configuration (i.e., frame rate and resolution) the average amount of users that were satisfied. From this table, we can conclude that the highest amount of users that can entirely be satisfied when selecting a fixed frame rate and resolution (RP_2) is 48.3%, when we would always offer a version at QCIF resolution and a frame rate of 30 frames per second, in the bandwidth range of 100 to 400 kbps.

A final interesting reference point (RP_3) is what we would obtain when selecting a random version of the sequence group as the best possible version. If we would do that, we would obtain an average reliability of 28.7%. This number should be interpreted in the same way as the 50% border in the test set method: if we go below this number, we would have done better by randomly selecting a version as the best.

Table 6.4: Maximum reliability possible when choosing a constant frame rate and resolution for selecting the best version.

frame rate	resolution	reliability
30 Hz	CIF	20.6 %
15 Hz	CIF	22.2 %
7.5 Hz	CIF	15.0 %
30 Hz	QCIF	48.3 %
15 Hz	QCIF	42.8 %
7.5 Hz	QCIF	23.3 %

6.6 Different versions for the SoI algorithm

6.6.1 Initial settings

In the initial version of evaluating the performance of the algorithm described in Sect. 5.8.1, referred to as **SoIinit**, we used the frame rate level, the resolution level, and the PSNR value of the sequence as the different quality measures:

$$F_{\mathrm{SoIinit}} = \{\mathrm{psnr}, \mathrm{frlevel}, \mathrm{reslevel}\}. \tag{6.3}$$

More precisely, S_{frlevel} is 2 if sequence S contains 30 frames per second, 1 if it has 15 fps, and 0 for 7.5 fps. Similarly, S_{reslevel} is 1 for CIF sequences and 0 for QCIF sequences. Furthermore, $w_{\mathrm{psnr}} = 1$, and w_{frlevel} and w_{reslevel} are user-dependent, and need to be determined while solving the system of linear inequalities. To summarize:

$$S_{\mathrm{frlevel}} = \begin{cases} 2 & \text{, when } S_{\mathrm{fr}} = 30 \\ 1 & \text{, when } S_{\mathrm{fr}} = 15 \\ 0 & \text{, when } S_{\mathrm{fr}} = 7.5 \end{cases}, \tag{6.4}$$

$$S_{\mathrm{reslevel}} = \begin{cases} 1 & \text{, when } S_{\mathrm{res}} = \mathrm{CIF} \\ 0 & \text{, when } S_{\mathrm{res}} = \mathrm{QCIF} \end{cases}. \tag{6.5}$$

When using the SoI model, a problem that can occur is that the system is inconsistent. In Fig. 6.4, the dashed line shows $\mathcal{C}_U^{\mathrm{SoIinit}}(n)$, with $2 \leq n \leq 6$, the average amount of training sets that produce a solution, for the settings just presented. As expected, this number first increases because the probability of ending up with an unbounded region decreases when more inequalities are present. When we keep increasing the size of the training set, the probability of obtaining an inconsistent system is also increased. This phenomenon already causes a global decrease of the amount of systems that produce a solution with

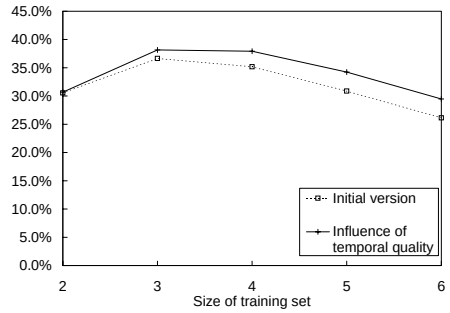


Figure 6.4: Amount of training sets that produce a consistent system with a bounded solution region.

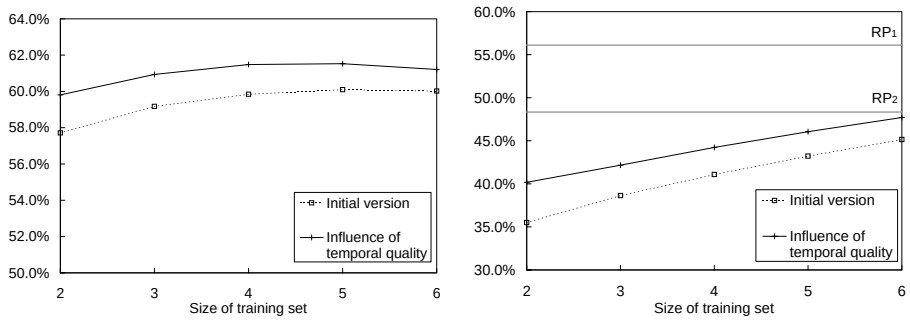


Figure 6.5: Performance of two versions of the SoI algorithm, using (a) the test set method and (b) the best in group method.

a training set of size 4. Note that even in the optimal case, only 36.7% of the training sets produce a preference model.

The reliability of the training sets that did produce a solution, $\mathcal{T}_U^{\mathrm{SoIinit}}(n)$ and $\mathcal{B}_U^{\mathrm{SoIinit}}(n)$ can be found in Fig. 6.5. The performance of the initial version is again represented by means of a dashed line. In the test set method, we see that the reliability slowly increases when the training set is increased. The optimal result is with a training set of size 5, with a 60.1% reliability, which is a disappointingly poor result. A similar conclusion can be drawn from the results of the BiG method. Here, we see that the initial version of the SoI algorithm cannot satisfy more users than what would be achieved when we would always offer a version at QCIF resolution at 30 frames per second (denoted by means of the gray line marked RP_2).

6.6.2 Influence of temporal quality

In the initial version, the difference between 30 and 15 frames per second was considered equally important as the difference between 15 and 7.5 frames per second. This however does not correspond well with the subjective quality observed by most people: the difference between 15 and 7.5 fps is very significant, whereas the difference between 30 and 15 fps is sometimes hardly observable.

In this section, we discuss the influence of the accuracy of the individual quality aspects used by both methods. More precisely, we try to predict the temporal quality in a more accurate way. In particular, in the **SoItemp** algorithm, we considered the difference between 15 and 7.5 fps three times more significant than the difference between 30 and 15 fps, based on our intuition. Therefore, we modified the definition of S_{fr} in as follows: if sequence S contains 30 frames per second it has value 4, for a sequence of 15 fps, S_{fr} is given the value 3, and 0 in case of 7.5 fps.

More formally, the quality features that are now considered are the following:

$$F_{\text{SoItemp}} = \{\text{psnr}, \text{frqual}, \text{reslevel}\},$$

in which frqual is defined as follows:

$$S_{\text{frqual}} = \begin{cases} 4 & \text{, when} \quad S_{\text{fr}} = 30 \\ 3 & \text{, when} \quad S_{\text{fr}} = 15 \\ 0 & \text{, when} \quad S_{\text{fr}} = 7.5 \end{cases} .$$

In Fig. 6.5, we can compare the improvement obtained when using a better estimation of the temporal quality for the SoI model. In the test set method, a gain of 1.2% up to 2.1% is observed between $\mathcal{T}_U^{\text{SoIinit}}(n)$ and $\mathcal{T}_U^{\text{SoItemp}}(n)$. In the best in group method, the improvement when going from $\mathcal{B}_U^{\text{SoIinit}}(n)$ to $\mathcal{B}_U^{\text{SoItemp}}(n)$ is more significant: from 2.6% up to 4.7%. This way, the SoI model approaches the reference point of using fixed adaptation parameters. Unfortunately, we failed to cross this border.

The probability of finding a consistent training set increases as well when using a more reliable way of estimating the temporal quality. For a training set of size 5, the amount of consistent training sets increased from 30.9% to 34.2%. The differences for other sizes are smaller, as can be seen in Fig. 6.4.

6.6.3 Influence of handling upper bounds

When analyzing the performance of the two versions of the model described in Sect. 6.6.1 and 6.6.2, we noticed that some users had unexpectedly bad results:

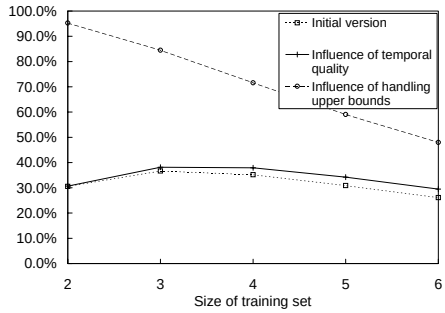


Figure 6.6: Amount of training sets that produce a consistent system with a bounded solution region.

not even half of the decisions in the test set were correctly predicted, which means that a random guess would be more reliable than using a model of the user's preferences. When investigating the numbers more closely, we noticed that these users had a high number of training sets that yielded an unbounded solution region, and therefore no weights could be deduced.

Ending up with a system producing an unbounded region of solutions means something totally different than having an inconsistent system. In the case of an unbounded region, it means that at least one of the unknown weights (in our case, w_{reslevel} and w_{frqual}) should be much larger than the weight that was assigned a value of 1 (in our case, w_{psnr}), as the corresponding quality aspect is considered more important for that user. Instead of ignoring this case, we should be capable of incorporating this information in our model.

The easiest way to do this, is to add upper bounds to the values of the unknown weights. We determined these upper bounds empirically: for a subset of all participants, we tried several possibilities for these upper bounds. From these experiments, we found that $w_{\mathrm{frqual}} \in [0, 8]$ and $w_{\mathrm{reslevel}} \in [0, 8]$ would be a good choice. These settings are incorporated in the **SoIbounds** algorithm.

The most remarkable improvement when using upper bounds can be expected from the amount of training sets that produce a solution. This is represented in Fig. 6.6, where all three versions of the SoI algorithm are presented. As expected, when the size of the training set is small, most training sets are capable of producing a model of the user's preferences.

In Fig. 6.7, we see that this addition of upper bounds has a positive effect on the performance of the algorithm. The gain observed in the test set method ($\mathcal{T}_U^{\mathrm{SoItemp}}(n)$ and $\mathcal{T}_U^{\mathrm{SoIbounds}}(n)$) ranges between 2.4% and 2.7%. In the BiG method, the gain $\mathcal{B}_U^{\mathrm{SoIbounds}}(n) - \mathcal{B}_U^{\mathrm{SoItemp}}(n)$ is even more remarkable: it

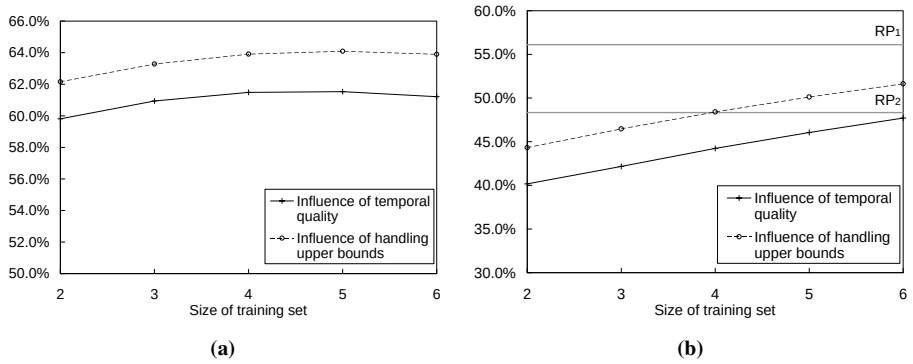


Figure 6.7: Performance of the two best versions of the SoI algorithm, using (a) the test set method and (b) the best in group method.

fluctuates around 4%. Because of this, the RP_2 performance is now easily achieved. Still, the reliability of the SoI algorithm remains below the RP_1 value, which is the theoretically optimal result that can be achieved when no user preferences are taken into account.

6.7 Different versions for the 1ARC algorithm

6.7.1 Initial settings

In the 1ARC algorithm, as described in Sect. 5.8.2, the way the distance between two alternatives is measured, can significantly influence the performance of the algorithm. In the initial version, that we call $\mathbf{1ARCinit}$, we used the Euclidian distance between the coordinates of the points, and those coordinates are given by $(\mathrm{psnr}, \mathrm{frlevel}, \mathrm{reslevel})$. The meaning of these parameters is exactly the same as in Sect. 6.6.1. More formally, the distance $d_{1\mathrm{ARCinit}}(X, Y)$ between two sequences X and Y, as it occurs in Algorithm 5.2, is defined as follows:

$$d_{1\mathrm{ARCinit}}(X, Y) = \sqrt{\Delta^2_{\mathrm{psnr}} + \Delta^2_{\mathrm{frlevel}} + \Delta^2_{\mathrm{reslevel}}}, \tag{6.6}$$

in which

$$\Delta_{\mathrm{f}} = X_f - Y_f. \tag{6.7}$$

In the 1ARC algorithm, there is no problem of training sets that do not produce any model, so we do not have to discuss this when looking at its performance. In Fig. 6.8, we show the reliability of this version, $\mathcal{T}_U^{1\mathrm{ARCinit}}(n)$ and $\mathcal{B}_U^{1\mathrm{ARCinit}}(n)$, by means of the dashed line. We see that for all training sets,

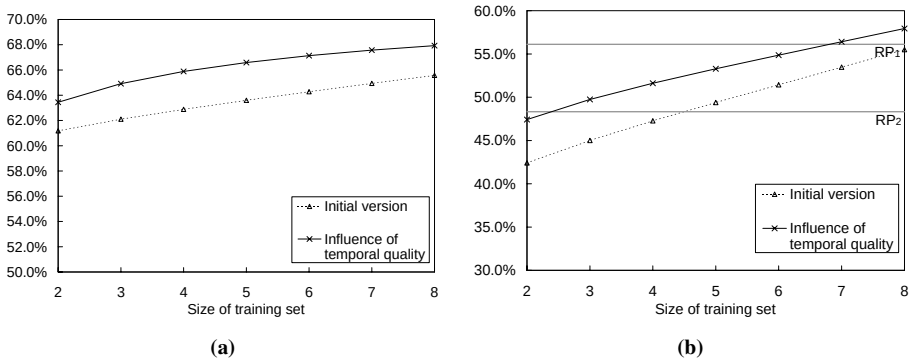


Figure 6.8: Performance of two versions of the 1ARC algorithm, using (a) the test set method and (b) the best in group method.

the performance is at least 60%, and increases when the size of the training set increases. We also see that this initial version approaches the optimal reference point for a training set of size 8, but that it is not capable of crossing this border.

6.7.2 Influence of temporal quality

In Sect. 6.6.2, we observed a better performance when a more accurate description of the temporal quality is used. It is useful to observe the influence of using the frqual measure, as defined in Sect. 6.6.2, in the $1\mathrm{ARCinit}$ algorithm. We use the name $\mathbf{1ARCtemp}$ for this improved version.

This is not the only modification in the improved version of the 1ARC algorithm. In the definition of a useful distance measure, it is important that the different quality aspects use a common scale. In this case, we chose to rescale the values that where used onto a 1 to 5 scale, which is a commonly used range for expressing Mean Opinion Scores (MOS) in subjective quality measurement, 1 expressing an unacceptably bad quality, and 5 expressing an excellent quality.

This way, the coordinates of this version were given by $(\mathrm{MOSpsnr}, \mathrm{MOSfr}, \mathrm{MOSres})$. As a consequence, the distance between two sequences is calculated as follows:

$$d_{\mathrm{1ARCtemp}}(X,Y) = \sqrt{\Delta^2_{\mathrm{MOSpsnr}} + \Delta^2_{\mathrm{MOSfr}} + \Delta^2_{\mathrm{MOSres}}}, \tag{6.8}$$

In this definition, $\mathrm{MOSpsnr}$ was defined as follows:

$$S_{\mathrm{MOSpsnr}} = 4 \cdot \frac{S_{\mathrm{psnr}} - \mathrm{minpsnr}}{\mathrm{maxpsnr} - \mathrm{minpsnr}} + 1, \tag{6.9}$$

in which S_{psnr} has the same value as in the previous version, and $\mathrm{minpsnr}$ and $\mathrm{maxpsnr}$ are the lowest and highest PSNR value, respectively, that occur in all the sequences used in the test (see Table 6.2). The values of MOSfr were obtained by using the same values as in the improved version of the SoI model, incremented by 1:

$$S_{\mathrm{MOSfr}} = S_{\mathrm{frqual}} + 1. \tag{6.10}$$

S_{MOSres} had a value of 5 for a CIF sequence, and 3 for a QCIF sequence:

$$S_{\mathrm{reslevel}} = \begin{cases} 5 & \text{, when} \quad S_{\mathrm{res}} = \mathrm{CIF} \\ 3 & \text{, when} \quad S_{\mathrm{res}} = \mathrm{QCIF} \end{cases} . \tag{6.11}$$

In Fig. 6.8, we can compare the performance of the $1\mathrm{ARCtemp}$ version of the 1ARC algorithm with its initial version $1\mathrm{ARCinit}$. We observe that there is a significant improvement: for the test set method, there is an improvement between 2.3% and 3.0%. The highest reliability is achieved with a training set of size 8: we achieve an accuracy of 67.9%. In the BiG method, the largest improvement is at the lower training set sizes: for a training set of size 2, the results of the initial version were $\mathcal{B}_U^{1\mathrm{ARCinit}}(2) = 42.4\%$, whereas the improved version scores $\mathcal{B}_U^{1\mathrm{ARCtemp}}(2) = 47.4\%$. This difference decreases when the training set size increases, but for a training set of size 8, the improved version still outperforms the initial version by 2.4%.

6.7.3 Influence of the selection mechanism

In Sect. 5.8.2, we showed that the order of evaluating the different candidate optimal versions can have an influence on which version is selected as the best version. We proposed a modification to the algorithm, by determining the likeliness that a version X is better than another version Y, and using this information to determine which version is most likely to be the best from the set of candidate optimal versions.

We implemented this modification in another version of our evaluation algorithms, $\mathbf{1ARCselect}$, to see if the performance of the algorithm would improve, according to the BiG method. Note that the results of the test set method do not change ($\mathcal{T}_U^{1\mathrm{ARCinit}}(n) = \mathcal{T}_U^{1\mathrm{ARCselect}}(n)$, for any n), as there is no modification on the way two versions are compared, but only on the way one version is selected from a larger set of candidate versions.

As there is no difference in comparing sequences two by two, the performance of the test set method does not change. Therefore, in Fig. 6.9, we only compare the performance of the $1\mathrm{ARCtemp}$ version with the $1\mathrm{ARCselect}$ version regarding the BiG method. We observe that for all training set sizes,

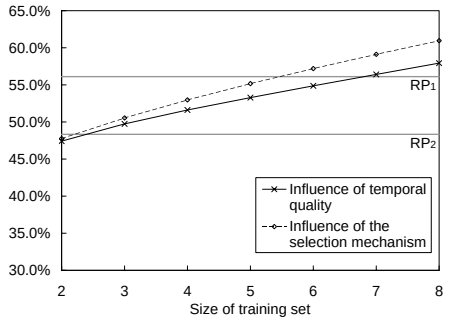


Figure 6.9: Performance of two versions of the 1ARC algorithm, using the best in group method.

there is an improvement, and this improvement becomes larger for larger training sets: with a training set of size 2, there is only 0.3% improvement. For a training set of size 8, the gain increased up to 3.0%. It is interesting to note that the improved version already crosses the RP_1 value when the training set has size 6. It is important to remember that there is an additional cost in using this more reliable algorithm: this version is quadratic rather than linear in terms of the number of candidate solutions, and both algorithms are linear in terms of the size of the training set.

6.8 Comparison between both algorithms

In Fig. 6.10, we have plotted the best performing version of each of both algorithms, $\mathrm{SoIbounds}$ and $\mathrm{1ARCselect}$, in order to compare them. Note that for the SoI algorithm, we only calculated the results up to a training set size of 6, because at that point, the chance of finding a training set that produces a preference model drops below 50%.

In Fig. 6.10 (a), one of the most important observations is that the values of $\mathcal{T}_U^{\mathrm{SoIbounds}}(n)$ increase much slower than those of $\mathcal{T}_U^{\mathrm{1ARCselect}}(n)$ when n is increased. In fact, it even decreases when changing the training set size from 5 to 6: $\mathcal{T}_U^{\mathrm{SoIbounds}}(5) > \mathcal{T}_U^{\mathrm{SoIbounds}}(6)$. The 1ARC algorithm does not show such a behavior: its reliability keeps increasing for larger training set sizes. As a consequence, the difference between the performance of both algorithms increases as well: from 1.3% up to 3.2%.

When looking at Fig. 6.10 (b), we note a similar behavior. Even though the results of the SoI algorithm increase more rapidly than when looking at the test set method, the difference with the 1ARC algorithm becomes bigger for

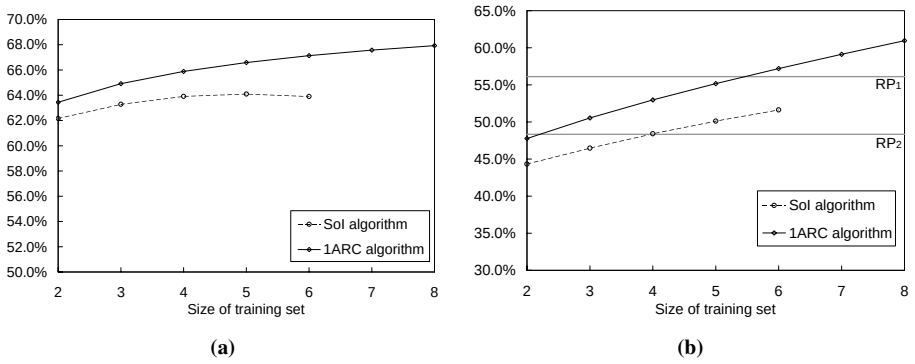


Figure 6.10: Comparison of the performance of the SoI and the 1ARC algorithm, using (a) the test set method and (b) the best in group method.

the larger training sets: from 3.4% up to 5.6% in this case.

Before claiming that the 1ARC algorithm outperforms the SoI algorithm, we should be certain that the difference is not caused by coincidence, but by a systematic trend. This can be checked by means of the *paired t-test*.

In such a test, a so-called t-value is calculated and compared with a critical t-value. The calculated t-value is obtained by taking the ratio of the mean of the differences in the scores and the standard error of the differences in the scores. More formally:

$$t = \frac{\overline{d}}{\sigma_{\overline{d}}}, \tag{6.12}$$

$\overline{d}$ being the average of the differences between two series, and $\sigma_{\overline{d}}$ its standard error.

In our case:

$$\overline{d} = \overline{\mathcal{T}^{\mathrm{1ARCselect}}(u,n) - \mathcal{T}^{\mathrm{SoIbounds}}(u,n)}|_{u:U} \tag{6.13}$$

for the test set method and

$$\overline{d} = \overline{\mathcal{B}^{\mathrm{1ARCselect}}(u,n) - \mathcal{B}^{\mathrm{SoIbounds}}(u,n)}|_{u:U} \tag{6.14}$$

for the BiG method, in which n is a fixed training set size, and $\mathrm{1ARCselect}$ and $\mathrm{SoIbounds}$ are the best versions of the 1ARC algorithm and SoI algorithm presented in this Sect. 6.7.3 and Sect. 6.6.3, respectively.

The actual calculated t-values for the test set method and the BiG method can be found in Table 6.5 and 6.6. In both tables, the first column shows the

Table 6.5: Calculated t-values for the test set method, when comparing the best version of the SoI and the 1ARC algorithm.

k	$\overline{d}$	$\sigma_{\overline{d}}$	t
2	1.3%	1.3%	1.01
3	1.6%	1.3%	1.30
4	2.0%	1.3%	1.51
5	2.5%	1.4%	1.81
6	3.2%	1.5%	2.22

Table 6.6: Calculated t-values for the BiG method, when comparing the best version of the SoI and the 1ARC algorithm.

k	$\overline{d}$	$\sigma_{\overline{d}}$	t
2	3.4%	2.2%	1.57
3	4.1%	2.1%	1.97
4	4.6%	2.0%	2.32
5	5.0%	1.9%	2.66
6	5.6%	1.9%	2.98

size of the training set, the second column the average difference, the third column the standard error of this difference, and the fourth column shows the actual calculated t-value.

When we want a 95% confidence, and we are using the results of 30 test subjects, the critical t-value is 1.70 in the case of a *one-tailed test*, where we want to know if we can safely say that the average of the differences is positive.

When looking at Table 6.5 and 6.6, we see that this value is not exceeded for the smaller training set sizes. As a consequence, we cannot conclude from the experiments we conducted that the 1ARC algorithm outperforms the SoI algorithm for small training set sizes. For training set sizes of at least 5, we are confident that the 1ARC algorithm is more reliable than the SoI algorithm. It may well be possible that this confidence would be increased if the results of more participants would be added.

When considering the BiG method, the difference between the 1ARC algorithm and the SoI algorithm increases. The critical t-value for a 95% confidence is reached as soon as the training set size is at least 3. For training set sizes of 4 or more, the critical t-value of 2.15, corresponding with a 98% confidence, is even reached.

6.9 The impact of noise

The results of the algorithms for predicting user preferences, as presented in this chapter, are rather good, but still far from perfect. When machine learning techniques are applied for predicting preferences in any domain, a high reliability, mostly above 90%, is expected. In the results we presented, the average reliability does not even reach 70%.

There are three reasons for this limited performance. A first reason comes from the requirements of the algorithms used, as introduced in Sect. 5.7: the training sets are very small, building a model should not take too much time, and taking decisions has to be possible in real time. Therefore, complex machine learning approaches such as neural networks cannot be applied in this scenario. As our models are not so complex, we can expect that they will be less accurate.

A second reason is already mentioned: both algorithms start from the assumption that the overall quality of a sequence can be expressed by means of a number of independent quality aspects, that can be described very accurately. Unfortunately, the accuracy of the metrics we used is far from perfect, as we already discussed when talking about the temporal quality. The same goes for PSNR: it is known that PSNR does not correlate very well with the characteristics of the Human Visual System (HVS).

Recent evolutions in the understanding of the Human Visual System resulted in new visual quality metrics that are far more accurate in describing the visual quality of a sequence as experienced by humans. A recent report of the Video Quality Experts Group (VQEG) discusses six different quality metrics that all outperform PSNR significantly when looking at the correlation between the quality estimated by the quality metrics and the judgements entered by a test panel [99]. In [100], Wang et al. present a new distortion measure called the Structural Similarity (SSIM) index, for which they proved it outperforms PSNR in terms of accuracy in predicting the visual quality of both still images and video data.

The third cause of the limited performance is probably the one having the most significant impact: the problem of noise. When we talk about *noisy* statements, we mean statements in the preference set of an end user, for which that particular user would *normally* take the opposite decision. This is mostly caused by a lack of concentration at some point during the test. It is nearly inevitable that this will occur rather frequently.

A single noisy statement in the test of a participant already causes a dramatic decrease in the performance of our model. Suppose the size of the pref-

erence set of a user is k, and we are testing a perfect algorithm[3], for a training set of size n. In this case, the total number of possible training sets is $\binom{n}{k}$. When one noisy statement exists in the set of k statements, the number of corrupted training sets, i.e. the training sets that contain the noisy statement is represented by $\binom{n-1}{k-1}$. This means that the rate of corrupted training sets is

$$\frac{\binom{n-1}{k-1}}{\binom{n}{k}} = \frac{n}{k}.$$

We suppose that for all these training sets, we fail to predict all the $k-n$ statements in the test set. The rate of training sets that should produce a perfect model is $1 - \frac{n}{k} = \frac{k-n}{k}$, but for these models, 1 out of $k-n$ statements will not be predicted correctly. Therefore, the total amount of incorrectly predicted statements will be

$$\frac{n}{k} + \frac{k-n}{k} \cdot \frac{1}{k-n} = \frac{n+1}{k}.$$

Suppose that a user has taken 28 discriminating statements, of which one should be considered noise, and we want to evaluate a perfect algorithm for a training set of size 6 using the test set method. In such a situation, we would fail to predict a statement correctly in 25% of the cases. So, even though we are using a perfect model, we will achieve an accuracy of only 75% when evaluating the test set method!

6.10 Conclusions and original contributions

The main objective of this chapter was to evaluate the prediction accuracy of the two algorithms presented in Chapter 5. We proposed these algorithms as candidates for capturing individual preferences of users regarding visual quality, and for using them in a multimedia content negotiation agent, that would retrieve the optimal version of a scalable video sequence, given the constraints imposed by the environment.

In order to measure the performance in terms of prediction accuracy of both algorithms, we set up a subjective test, in which the participants are asked to compare two versions of the same original sequence, and to select one of them as the best version. In Sect. 6.3, we described all details of the setup of this test.

[3]When talking about a perfect algorithm, we mean an algorithm that can predict all statements from any training set, as long as the statements in this training set are consistent (no noise). As soon as noise is involved, the algorithm fails.

We defined two methods for evaluating the performance of both algorithms. In these evaluation methods, all possible training sets are generated and used for predicting information that is not in the training set. For the *Best in Group* method, we defined three reference points to get a better understanding of the actual performance of the algorithms. One of these reference points corresponds with the maximum performance that can be achieved when no personalization is used in any way, i.e., the same version is offered to all users.

For the *Systems of Inequalities* algorithm, none of the different versions that we implemented crossed all reference points. As a consequence, it may be possible to implement some other method that has a better prediction accuracy, even when it does not take user preferences into account. At the same time, we observed that the way the problem of unbounded regions is handled, has a significant impact on the overall result. The same goes for the metrics used for measuring the individual quality aspects: using a metric for the temporal quality that better reflects the reality, resulted in a better overall performance.

In the *1ARC* algorithm, a similar modification had a positive effect as well. When using a realistic metric for expressing the temporal quality, we noticed that the reliability of the algorithm improved. In the previous chapter, we mentioned the problem of the order of evaluating the different candidate versions, and we proposed a modification that should be able to overcome this problem. We proved that this assumption was correct: the modification caused a significant improvement of the reliability of the algorithm.

When the parameters for the 1ARC algorithms are selected appropriately, and the training set is sufficiently large, the algorithm is more reliable than any algorithm that performs the same task but that does not attempt to model the preferences of the end user. The reliability of the optimal configuration of the 1ARC algorithm seems to be better than the optimal configuration of the SoI algorithm. We validated this by means of a paired t-test. From that test, we concluded that we could only state that the 1ARC algorithm outperforms the SoI algorithm for larger training sets.

Still, the results are less convincing than what we initially hoped to achieve. Several reasons can be found, explaining different causes for the relatively low performance. We showed that even a small amount of *noisy* statements expressed by a participant could already drastically reduce the overall result for that particular user.

One cause of these noisy statements might be the fact that users did not have the opportunity to express how much they preferred one sequence over another. Such gradations would probably give us more information, that is likely to be more reliable as well. On the other hand, we would have to rethink the methods we used for measuring the reliability of the algorithms.

Another reason explaining why the algorithms do not achieve a very high accuracy, is in the simplicity of the underlying models that were used. We assumed that the individual quality aspects are not user-dependent, and furthermore, are perfectly measurable. This certainly does not reflect reality. For example, we showed that a small correction of one particular quality metric can already cause significant changes in the performance of the algorithms.

As a consequence, we can assume that better results are still possible, if we would use better quality metrics. Recently, metrics for measuring the distortion of images and video sequences are developed that have a better correlation with human perception than PSNR, the metric that we used in our experiments. In the case of temporal quality, it should be possible to develop a similar metric, that not only considers the frame rate itself, but also the amount of motion of the video sequence. Unfortunately, no such metric exists to this day as far as we know. Another solution would be to follow the approach of Önür and Alatan [96]. They conduct subjective tests for obtaining more accurate values of the individual quality functions.

Intuitively, we felt that the overall subjective quality or utility as experienced by the end user is not only determined by the user preferences, but also by the content of the video sequence. As an example, a user may find the temporal quality more important when watching an action movie than when watching the news. Such aspects are not incorporated in our model.

In a recent publication by Wang et al. [95], an approach is presented that tries to solve the same problem as what we did in this and the previous chapter: determining which adaptation should be selected, in such a way that the overall utility is maximized. Their approach differs from ours in the source of information that is used: while we try to relate user preferences to the overall utility, they try to relate sequence characteristics to the overall utility, also by means of a learning process. According to the results they describe, their approach is more satisfying than ours. At the same time, they report a significant drop in the performance of their method in the medium bandwidth range (between 200 and 600 kbps). They say that this "comes from the fact that at mid bandwidths human subjects do not show consistent preferences to specific dimensions among different spatio-temporal scales". This phenomenon is exactly what we tried to capture with our algorithms. Note that this medium bandwidth range broadly corresponds to the bit rates we used for the sequences in our test.

As a conclusion, it seems that more successful results should be possible by using a joint approach, in which both user preferences and sequence characteristics are captured and used for predicting the optimal adaptation operation. This could be possible by using the method proposed by Wang et al.

for determining the values of the individual quality aspects, and our method for distilling the overall quality for a particular end user, taking his preferences into account.

The results of the subjective test that is described in this chapter, can also be found in the following publication. A second publication is submitted to a journal and is currently under review.

1. Sam Lerouge, Robbie De Sutter, and Rik Van de Walle. Personalizing quality aspects in scalable video coding. In *IEEE Proceedings of ICME 2005*, Amsterdam, The Netherlands, July 2005.

Chapter 7

Conclusions

The world of multimedia communication is drastically evolving since a few years. Advanced compression formats for audiovisual information arise, new types of wired and wireless networks are developed, and a broad range of different types of devices capable of multimedia communication appear on the market. The era where multimedia applications available on the Internet were the exclusive domain of PC users has passed. The next generation multimedia applications will be characterized by heterogeneity: differences in terms of the networks, devices and user expectations.

This heterogeneity causes some new challenges: transparent consumption of multimedia content is needed in order to be able to reach a broad audience. In Chapter 2, we described two important types of technologies that are both essential for realizing such transparent *Universal Multimedia Access*. In the first place, scalable or layered content representation schemes are needed in order to make it possible that a multimedia stream can be consumed by devices with different capabilities and transmitted over network connections with different characteristics. The second technology does not focus on the content representation itself, but rather on linking information about the content, so-called metadata, to the content itself. One of the possible uses of metadata is in the automatic selection and adaptation of multimedia presentations. This is one of the main goals of the MPEG-21 Multimedia Framework.

Within the MPEG-21 standard, two formats were developed that can be used for bitstream descriptions. Such descriptions can act as an intermediate layer between a scalable bitstream and the adaptation process. This way, format-independent bitstream adaptation engines can be built. Furthermore, it is straightforward to add metadata information to the bitstream description, and use this information later on during the adaptation process. Because of the efforts spent on bitstream descriptions during our research, the entire Chap-

ter 3 is devoted to this topic. After a description of the two frameworks for bitstream descriptions that were standardized by MPEG, we elaborated our own contributions: a number of bitstream schemas and transformation examples for different types of multimedia content.

Our ultimate goal was to describe a content negotiation process that uses scalable bitstreams in a generic way. In order to be able to express such an application, we felt the need for a better understanding of the data structures, in particular scalable bitstreams, on which this content negotiation process operates. Therefore, we developed a formal model describing the fundamental concepts of scalable bitstreams and their relations and dependencies. This formal model can be found in Chapter 4. Apart from the definition of the theoretical model itself, we demonstrated its correctness by applying it to a number of existing formats for scalable bitstreams. We made a first attempt to formulate a content negotiation process as a constrained optimization problem, by means of the notations defined in the abstract model.

In Chapter 5, we explained why the representation of a content negotiation process as a constrained optimization problem does not sufficiently reflect reality, especially when scalable bitstreams with multiple quality dimensions are involved. After introducing Pareto's theory of multi-criteria optimization, we modified our definition of a content negotiation process into a multi-criteria optimization problem. We were the first to do so, and currently this approach is already adopted by others.

One of the most important problems with multi-criteria optimization problems is that multiple candidate optimal solutions may exist. Additional information, e.g. user preferences, is needed if a single optimal solution has to be selected. In Chapter 5, we explained why existing solutions from the domain of multi-criteria optimization are not suitable in a content negotiation scenario. We proposed a scenario in which a so-called content negotiation agent would give some sample video sequences to the end user, asking him to select which sequence he liked the most. This information would be used for training the agent: a model would be built representing the preferences of the end user, and this model can be used later on for selecting one solution from a set of candidate optimal solutions.

We proposed two candidate algorithms that can be used for constructing a model of the user's preferences and for using this model when selecting an optimal version. The first one considers the quality of a video sequence as a weighted sum of a number of independent quality aspects, and derives a system of linear inequalities from the example decisions. The second algorithm, called 1ARC, is a nearest-neighbor approach, where predictions are made based on the similarity with example decisions entered by the user. We analyzed the

strengths and weaknesses of both algorithms from multiple points of view.

The actual performance, the reliability of both algorithms is discussed in Chapter 6. We described how we set up a test in which human subjects had to make a number of pairwise decisions between two versions of the same original video sequence. The two algorithms we proposed could then be tested by selecting a part of these decisions for training a model, and by observing if this model would be able to predict other decisions entered by the same user. Apart from comparing both algorithms, we observed the result of modifying several parameters on both algorithms. Ultimately, we could conclude that the 1ARC algorithm had an acceptable performance, certainly when the training set was sufficiently large. The reliability was better than what would be theoretically achievable by any other algorithm that selects one optimal version from a set of candidate versions, but does not try to capture the user's preferences.

Still, the results that we achieved are not as good as what we hoped. One possible cause may be the fact that we did not take sequence characteristics, such as the amount of motion, into account. Intuitively, we expect that this would improve the reliability of our user-centric approach. Our intuition is confirmed by a recent publication in the literature, where more or less the same problem is handled, but instead of learning user preferences and use this information for predicting the overall quality of a sequence, the relation between sequence characteristics and observed quality is learned and used for predicting the quality of other sequences.

There are some other interesting research topics that are not discussed in this thesis, but that would be useful to investigate as an addition to the framework we presented. As a first topic, it would be interesting to explore the usability of the agent-based approach we proposed: are users prepared to undergo a training session, how long is this training session allowed to be, how can users reject the agent's decision and select an alternative solution in an intuitive way, etc.

A second option for future research would be to explore a more advanced way of taking advantage of the principles of software agents. As an example, it would be useful to see if it would be possible to allow agents to communicate with each other, in order to build more sophisticated models. This way, decisions for which there is a large agreement among most users can be reused for other users.

Another research topic that has not been covered in this thesis is the way constraints are determined. We supposed they were available at any time. In practice, some constraints are not known beforehand, but have to be predicted. Moreover, one should bare in mind that drastically changing constraints, for example regarding the bit rate, may result in an instable visual quality, which

can be very annoying for an end user. As a consequence, it might be necessary to modify certain constraints in order to avoid such fluctuations.

In Chapter 1, we cited a fragment of an article from Ramesh Jain on the concept of Quality of Experience. We repeat it here, because it nicely helps us summarizing some of the important original contributions that can be found in this thesis.

> To do so, we will have to develop measures that will help us capture QoE in a given application and use it. We need to make these measures as applicable to our field as required by our practice, while capturing the subjective nature of experience.

In this thesis, the *given application* we focussed on was a multimedia content negotiation process that takes advantage of scalable bitstreams. By decomposing the global quality or utility of a video sequence into a number of independent quality aspects or features, we *developed measures* that can be used in the content negotiation process. We managed to develop algorithms that are capable of capturing the *Quality of Experience* as observed by a user, and this was done in such a way that it *captures the subjective nature of experience*, by taking into account the preferences of the individual end user.

With this citation, we hope that we have convinced the reader that this thesis, although limited in its performance and its application area, offers one of the first significant steps towards the objective of offering true Quality of Experience in multimedia applications.

Appendix A

Bitstream descriptions for MPEG-4 FGS

A.1 Introduction

In this appendix, we show the entire BSDL Schema that we used for generating bitstream descriptions for MPEG-4 FGS video sequences, as well as a detailed example of a bitstream description generated by means of this Schema. We also show the Java code that we produced for generating a gBSD description from BSDL descriptions of the base and enhancement layers of an FGS sequence.

A.2 BSDL Schema for MPEG-4 FGS

In what follows, the entire BSDL Schema for MPEG-4 FGS is shown, followed by one particular bitstream description generated with this schema.

Listing A.1: BSDL Schema for MPEG-4 FGS.

```
<xsd:schema targetNamespace="MPEG4" xmlns:mp4="MPEG4"
      xmlns:bt="urn:mpeg:mpeg21:2003:01-DIA-BasicDatatypes-01"
      xmlns:bs0="urn:mpeg:mpeg21:2003:01-DIA-BSDL0-NS"
      xmlns:bs1="urn:mpeg:mpeg21:2003:01-DIA-BSDL1-NS"
      xmlns:bs2="urn:mpeg:mpeg21:2003:01-DIA-BSDL2-NS"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      elementFormDefault="qualified"
      bs2:rootElement="mp4:Bitstream">
    <xsd:import
      namespace="urn:mpeg:mpeg21:2003:01-DIA-BasicDatatypes-01"
      schemaLocation="../BasicTypes.xsd"/>
```

```
<!-- **** Root element declaration **** -->
<xsd:element name="Bitstream">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="mp4:VOS" minOccurs="0"/>
            <xsd:element ref="mp4:VO"  minOccurs="0"/>
            <xsd:element ref="mp4:VOL" minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute ref="xml:base"/>
    </xsd:complexType>
</xsd:element>
<!-- **** VOS declaration **** -->
<xsd:element name="VOS" bs2:ifNext="000001B0">
    <xsd:complexType>
        <xsd:sequence>
          <xsd:element
              name="video_object_sequence_start_code"
              type="mp4:StartCodeType" fixed="000001B0"/>
          <xsd:element name="profile_and_level_indication"
              type="bt:b8"/>
          <xsd:element name="visual_object_start_code"
              type="mp4:StartCodeType" fixed="000001B5"/>
          <xsd:element name="visual_object_data"
              type="mp4:PayloadType"/>
          <xsd:element ref="mp4:VO"  minOccurs="0"/>
          <xsd:element ref="mp4:VOL" minOccurs="0"/>
          <xsd:element
              name="video_object_sequence_end_code"
              type="mp4:StartCodeType" fixed="000001B1"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<!-- ***** VO declaration **** -->
<xsd:element name="VO" bs2:ifNext="00000100-0000011F">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="video_object_start_code"
              type="mp4:StartCodeType"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<!-- **** VOL declaration **** -->
<xsd:element name="VOL" bs2:ifNext="00000120-0000012F">
    <xsd:complexType>
        <xsd:sequence>
          <xsd:element
            name="video_object_layer_start_code"
            type="mp4:StartCodeType"/>
          <xsd:element name="random_accessible_vol"
```

```
        type="bt:b1"/>
      <xsd:element
        name="video_object_type_indication"
        type="bt:b8"/>
      <xsd:element name="is_object_layer_identifier"
        type="bt:b1" minOccurs="0"
        bs2:if=
          "mp4:video_object_type_indication = 1"/>
      <xsd:element name="fgs_layer_type"
        type="bt:b2" minOccurs="0"
        bs2:if=
          "mp4:video_object_type_indication = 18"/>
      <xsd:element name="video_object_layer_verid"
        type="bt:b4" minOccurs="0"
        bs2:if="mp4:is_object_layer_identifier = 1"/>
      <xsd:element name="video_object_layer_priority"
        type="bt:b3" minOccurs="0"
        bs2:if="mp4:is_object_layer_identifier = 1
          or mp4:video_object_type_indication = 18"/>
      <xsd:element name="aspect_ratio_info"
        type="bt:b4"/>
      <xsd:element name="par_width"
        type="bt:b4" minOccurs="0"
        bs2:if="mp4:aspect_ratio_info = 15"/>
      <xsd:element name="par_height"
        type="bt:b4" minOccurs="0"
        bs2:if="mp4:aspect_ratio_info = 15"/>
      <xsd:element name="vol_control_parameters"
        type="bt:b1" fixed="0"/>
      <xsd:element name="video_object_layer_shape"
        type="bt:b2" minOccurs="0" fixed="0"
        bs2:if="mp4:video_object_type_indication = 1"/>
      <xsd:element name="marker_bit"
        type="bt:b1" fixed="1"/>
      <xsd:element name="vop_time_increment_resolution"
        type="bt:b16"/>
      <xsd:element name="stuffing"
        type="bs0:fillByte"/>
      <xsd:element name="VOL_data"
        type="mp4:PayloadType" />
      <xsd:element ref="FGSVOP" minOccurs="0"
        maxOccurs="unbounded"
        bs2:if=
          "mp4:video_object_type_indication = 18"/>
      <xsd:element ref="VOP" minOccurs="0"
        maxOccurs="unbounded"
        bs2:if="mp4:video_object_type_indication = 1"/>
  </xsd:sequence>
</xsd:complexType>
```

```
</xsd:element>
<!-- **** VOP declaration **** -->
<xsd:element name="VOP" bs2:ifNext="000001B6">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="VOP_code"
              type="mp4:StartCodeType" fixed="000001B6"/>
            <xsd:element name="vop_coding_type"
              type="bt:b2"/>
            <xsd:element name="modulo_time_base"
              type="bt:b1" fixed="1" minOccurs="0"
              maxOccurs="unbounded" bs2:ifNext="80-FF"/>
            <xsd:element name="modulo_time_base"
              type="bt:b1" fixed="0"/>
            <xsd:element name="marker_bit"
              type="bt:b1" fixed="1"/>
            <xsd:element name="vop_time_increment"
              type="mp4:time_incr"/>
            <xsd:element name="stuffing"
              type="bs0:fillByte"/>
            <xsd:element name="VOP_data"
              type="mp4:PayloadType" />
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<xsd:element name="FGSVOP" bs2:ifNext="000001B9">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="fgs_vop_start_code"
              type="mp4:StartCodeType" fixed="000001B9"/>
            <xsd:element name="vop_coding_type"
              type="bt:b2"/>
            <xsd:element name="modulo_time_base"
              type="bt:b1" fixed="1" minOccurs="0"
              maxOccurs="unbounded" bs2:ifNext="80-FF"/>
            <xsd:element name="modulo_time_base"
              type="bt:b1" fixed="0"/>
            <xsd:element name="marker_bit"
              type="bt:b1" fixed="1"/>
            <xsd:element name="vop_time_increment"
              type="mp4:time_incr"/>
            <xsd:element name="stuffing"
              type="bs0:fillByte"/>
            <xsd:element name="VOP_data"
              type="mp4:PayloadType" />
            <xsd:element ref="mp4:BitPlane"
              maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
```

```
    </xsd:element>
    <xsd:element name="BitPlane"
      bs2:ifNext="00000140-0000015F">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="fgs_bp_start_code"
                  type="mp4:StartCodeType"/>
                <xsd:element name="BP_data"
                  type="mp4:PayloadType" />
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <!-- **** Basic Types **** -->
    <xsd:simpleType name="StartCodeType">
        <xsd:restriction base="xsd:hexBinary">
            <xsd:length value="4"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="PayloadType">
        <xsd:restriction base="bs1:byteRange">
            <xsd:annotation>
                <xsd:appinfo>
                    <bs2:startCode value="000001"/>
                </xsd:appinfo>
            </xsd:annotation>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="time_incr">
        <xsd:union
          memberTypes="bt:b1 bt:b2 bt:b3 bt:b4 bt:b5 bt:b6">
          <xsd:annotation>
            <xsd:appinfo>
              <bs2:ifUnion value="2 &gt;
                //mp4:VOL/mp4:vop_time_increment_resolution"/>
              <bs2:ifUnion value="4 &gt;
                //mp4:VOL/mp4:vop_time_increment_resolution"/>
              <bs2:ifUnion value="8 &gt;
                //mp4:VOL/mp4:vop_time_increment_resolution"/>
              <bs2:ifUnion value="16 &gt;
                //mp4:VOL/mp4:vop_time_increment_resolution"/>
              <bs2:ifUnion value="32 &gt;
                //mp4:VOL/mp4:vop_time_increment_resolution"/>
              <bs2:ifUnion value="64 &gt;
                //mp4:VOL/mp4:vop_time_increment_resolution"/>
            </xsd:appinfo>
          </xsd:annotation>
        </xsd:union>
    </xsd:simpleType>
</xsd:schema>
```

Listing A.2: Bitstream description generated using the BSDL Schema for MPEG-4 FGS.

```
<Bitstream xml:base="stockholm_cif_fgs.cmp"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="MPEG4" xmlns:mp4="MPEG4"
  xsi:schemaLocation="MPEG4 file:/./MPEG4-FGS.xsd"
  xmlns:pref0="urn:mpeg:mpeg21:2003:01-DIA-BasicDatatypes-01">
    <VO>
        <video_object_start_code>
          00000101
        </video_object_start_code>
    </VO>
    <VOL>
        <video_object_layer_start_code>
          00000121
        </video_object_layer_start_code>
        <random_accessible_vol>1</random_accessible_vol>
        <video_object_type_indication>
          18
        </video_object_type_indication>
        <fgs_layer_type>1</fgs_layer_type>
        <video_object_layer_priority>
          2
        </video_object_layer_priority>
        <aspect_ratio_info>1</aspect_ratio_info>
        <vol_control_parameters>0</vol_control_parameters>
        <marker_bit>1</marker_bit>
        <vop_time_increment_resolution>
          30
        </vop_time_increment_resolution>
        <stuffing>12</stuffing>
        <VOL_data>13 5</VOL_data>
        <FGSVOP>
            <fgs_vop_start_code>000001B9</fgs_vop_start_code>
            <vop_coding_type>0</vop_coding_type>
            <modulo_time_base>0</modulo_time_base>
            <marker_bit>1</marker_bit>
            <vop_time_increment xsi:type="pref0:b5">
              0
            </vop_time_increment>
            <stuffing>76</stuffing>
            <VOP_data>24 3</VOP_data>
            <BitPlane>
                <fgs_bp_start_code>00000140</fgs_bp_start_code>
                <BP_data>31 176</BP_data>
            </BitPlane>
            <BitPlane>
                <fgs_bp_start_code>00000141</fgs_bp_start_code>
                <BP_data>211 3945</BP_data>
```

```
            </BitPlane>
            <BitPlane>
                <fgs_bp_start_code>00000142</fgs_bp_start_code>
                <BP_data>4160 9742</BP_data>
            </BitPlane>
            <BitPlane>
                <fgs_bp_start_code>00000143</fgs_bp_start_code>
                <BP_data>13906 14137</BP_data>
            </BitPlane>
        </FGSVOP>
        <FGSVOP>
            <fgs_vop_start_code>000001B9</fgs_vop_start_code>
            <vop_coding_type>0</vop_coding_type>
            <modulo_time_base>0</modulo_time_base>
            <marker_bit>1</marker_bit>
            <vop_time_increment xsi:type="pref0:b5">
              2
            </vop_time_increment>
            <stuffing>76</stuffing>
            <VOP_data>28049 3</VOP_data>
            <BitPlane>
                <fgs_bp_start_code>00000140</fgs_bp_start_code>
                <BP_data>28056 74</BP_data>
            </BitPlane>
            <BitPlane>
                <fgs_bp_start_code>00000141</fgs_bp_start_code>
                <BP_data>28134 3024</BP_data>
            </BitPlane>
            <BitPlane>
                <fgs_bp_start_code>00000142</fgs_bp_start_code>
                <BP_data>31162 9784</BP_data>
            </BitPlane>
            <BitPlane>
                <fgs_bp_start_code>00000143</fgs_bp_start_code>
                <BP_data>40950 14681</BP_data>
            </BitPlane>
        </FGSVOP>
        <!-- and so on -->
    </VOL>
</Bitstream>
```

A.3 Merging BSDL FGS bitstream descriptions into one gBSD bitstream description

For generating a bitstream description in a gBSD format, one can use BSDL descriptions and transform this into gBSD, e.g. using XSLT or a programming language. The following Java code fragment shows how this can be done for the case of MPEG-4 FGS bitstreams. Because multiple streams are available, multiple bitstream descriptions exist and must be merged into a single description.

Listing A.3: Generating a gBSD bitstream description in Java based on existing BSDL descriptions.

```
package be.mmlab.slerouge;

import java.io.*;
import java.lang.*;
import javax.xml.parsers.*;
import javax.xml.transform.*;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import org.w3c.dom.*;

public class BSDMerge {
  public static void usage() {
    System.out.println(
      "BSDMerge - tool for merging FGS-T BSDL descriptions into a gBSD description.");
    System.out.println("\tSam Lerouge, march-april 2004\n");
    System.out.println("Usage: BSDMerge -base b.xml -fgs f.xml -fgst ft.xml [-out gbsd.xml]");
    System.out.println("\tb.xml              : BSDL description of the base layer");
    System.out.println("\tf.xml              : BSDL description of the FGS  enhancement layer");
    System.out.println("\tft.xml             : BSDL description of the FGST enhancement layer");
    System.out.println("\tgbsd.xml (optional): output (merged) gBSD description (default: stdout)");
  }

  public static void main(String args[]) {
    String basename = null;
    String fgsname  = null;
    String fgstname = null;
    String outname  = null;
    InputStream base = null;
    InputStream fgs  = null;
```

```
InputStream fgst = null;
OutputStream out = null;

/*** reading parameters ***/
for (int i = 0; i < args.length; i++) {
  if (args[i].equals("-base"))
    basename = args[++i];
  else if (args[i].equals("-fgs"))
    fgsname = args[++i];
  else if (args[i].equals("-fgst"))
    fgstname = args[++i];
  else if (args[i].equals("-out"))
    outname = args[++i];
  else if (args[i].equals("-help")) {
    usage();
    return;
  }
  else {
    usage();
    System.exit(-1);
  }
}

/*** checking parameter values ***/
if (basename == null) {
  usage();
  System.out.println("\n[ERROR] base layer must be specified");
  System.exit(-1);
}
else {
  try {
```

```
    base = new FileInputStream(basename);
  }
  catch (FileNotFoundException e) {
    System.out.println("\n[ERROR] base layer description not found");
    System.exit(-1);
  }
 }
 if (fgsname == null) {
  usage();
  System.out.println("\n[ERROR] FGS layer must be specified");
  System.exit(-1);
 }
 else {
  try {
    fgs = new FileInputStream(fgsname);
  }
  catch (FileNotFoundException e) {
    System.out.println("\n[ERROR] FGS layer description not found");
    System.exit(-1);
  }
 }
 if (fgstname == null) {
  usage();
  System.out.println("\n[ERROR] FGST layer must be specified");
  System.exit(-1);
 }
 else {
  try {
    fgst = new FileInputStream(fgstname);
  }
  catch (FileNotFoundException e) {
```

```
      System.out.println("\n[ERROR] FGST layer description not found");
      System.exit(-1);
    }
  }
  if (outname == null)
    out = System.out;
  else {
    try {
      out = new FileOutputStream(outname);
    }
    catch (FileNotFoundException e) {
      System.out.println("\n[ERROR] Output file could not be created");
      System.exit(-1);
    }
  }

  /*** create a DOM parser ***/
  Document[] BSD = new Document[3];
  Document outBSD = null;
  try {
    DocumentBuilderFactory fact = DocumentBuilderFactory.newInstance();
    fact.setNamespaceAware(true);
    DocumentBuilder parser = fact.newDocumentBuilder();
    BSD[0] = parser.parse(base);
    BSD[1] = parser.parse(fgs);
    BSD[2] = parser.parse(fgst);
    outBSD = parser.newDocument();
  }
  catch (Exception e) {
    System.out.println ("[ERROR] Something went wrong during XML parsing:\n\n" + e);
    e.printStackTrace();
```

```
}

String[] bitstream = new String[3];
for (int i = 0; i < 3; i++)
  bitstream[i] = BSD[i].getDocumentElement().getAttribute("xml:base");

/*** initialize output bitstream description ***/
Element description = null;
try {
  Element root = outBSD.createElement("dia:DIA");
  root.setAttribute("xmlns:dia","urn:mpeg:mpeg21:2003:01-DIA-NS");
  root.setAttribute("xmlns","urn:mpeg:mpeg21:2003:01-DIA-gBSD-NS");
  root.setAttribute("xmlns:xsi","http://www.w3.org/2001/XMLSchema-instance");
  description = outBSD.createElement("dia:Description");
  description.setAttribute("xsi:type","gBSDType");
  Element header = outBSD.createElement("Header");
  Element defaultvals = outBSD.createElement("DefaultValues");
  defaultvals.setAttribute("addressUnit","byte");
  defaultvals.setAttribute("addressMode","absolute");
  defaultvals.setAttribute("globalAddressInfo",bitstream[0]);
  header.appendChild(defaultvals);
  description.appendChild(header);
  root.appendChild(description);
  outBSD.appendChild(root);
}
catch (DOMException e) {
  System.out.println ("[ERROR]: failed to construct the gBSD description\n" + e);
}

/*** start reading data from the bitstream ***/
int[] cursor = new int[3];
```

```
boolean headers = true;
boolean end = false;
int parcels = 1;
int[] framecount = new int[2];
framecount[0] = framecount[1] = 0;
float prevtime = 0;
long[][] size = new long[2][8];

try {
  /* assign all global headers to label(0,0) in the first parcel */
  Element parcel = outBSD.createElement("gBSDUnit");
  parcel.setAttribute("marker",":parcel:0");
  parcel.setAttribute("start", "0");
  for (int i = 0; i < 3; i++) {
    Element data = outBSD.createElement("gBSDUnit");
    data.setAttribute("marker",":label:(0,0)");
    /* get the length of the header */
    Node VOL_data = BSD[i].getElementsByTagNameNS("MPEG4","VOL_data").item(0).getFirstChild();
    cursor[i] = Integer.parseInt(VOL_data.getNodeValue().split(" ")[0]) +
      Integer.parseInt(VOL_data.getNodeValue().split(" ")[1]);
    data.setAttribute("start", "0");
    data.setAttribute("length", Integer.toString(cursor[i]));
    data.setAttribute("globalAddressInfo",bitstream[i]);
    size[0][0] += cursor[i];
    parcel.appendChild(data);
  }
  NodeList[] VOP = new NodeList[3];
  int clock = 0;
  int clockbase = 0;
  int fgstclock = 0;
  int fgstclockbase = 0;
```

```
int clockunit = Integer.parseInt(BSD[0].getElementsByTagNameNS
   ("MPEG4","vop_time_increment_resolution").item(0).getFirstChild().getNodeValue());
int[] pos = new int[2];
for (int i = 0; i < 3; i++) {
  if (i == 0)
    VOP[i] = BSD[i].getElementsByTagNameNS("MPEG4","VOP");
  else
    VOP[i] = BSD[i].getElementsByTagNameNS("MPEG4","FGSVOP");
  if (i < 2)
    pos[i] = 0;
}
while (true) {
  /* check if a new parcel should be created (if the next frame is an I-frame) */
  if ( end || (!headers && ((Element) VOP[0].item(pos[0])).getElementsByTagNameNS(
      "MPEG4","vop_coding_type").item(0).getFirstChild().getNodeValue().equals("0"))) {
    parcel.setAttribute("length",
      Integer.toString(cursor[0] - Integer.parseInt(parcel.getAttribute("start"))));
    description.appendChild(parcel);
    // insert frame rate
    NodeList units = parcel.getElementsByTagName("gBSDUnit");
    int b = 0; int e = 0;
    float time = ((float) clock) / clockunit - prevtime;
    float[] fps = new float[2];
    fps[0] = framecount[0] / time; fps[1] = framecount[1] / time;
    prevtime = clock / clockunit;
    long[][] totalsize = new long[2][8];
    for (int i = 0; i < 2; i++)
      for (int j = 0; j < 8; j++)
        for (int ii = 0; ii <= i; ii++)
          for (int jj = 0; jj <= j; jj++)
            totalsize[i][j] += size[ii][jj];
```

```
  for (int i = 0; i < units.getLength(); i++) {
    String marker = ((Element) units.item(i)).getAttribute("marker");
    if (marker.indexOf(":label:(0," + b + ")") != -1) {
      ((Element) units.item(i)).getAttributeNode("marker").setValue(marker + " :fps:" +
        fps[0] + " :kbps:" + ((int) (totalsize[0][b] * 8 / time / 1000)));
      b++;
    }
    else if (marker.indexOf(":label:(1," + e + ")") != -1) {
      ((Element) units.item(i)).getAttributeNode("marker").setValue(marker + " :fps:" +
        fps[1] + " :kbps:" + ((int) (totalsize[1][e] * 8 / time / 1000)));
      e++;
    }
  }
  if (end)
    break;
  for (int i = 0; i < 2; i++)
    for (int j = 0; j < 8; j++)
      size[i][j] = 0;
  framecount[0] = framecount[1] = 0;
  parcel = outBSD.createElement("gBSDUnit");
  parcel.setAttribute("marker",":parcel:" + (parcels++));
  parcel.setAttribute("start", Integer.toString(cursor[0]));
}
headers = false;
/* add base layer frame to current parcel */
Element data = outBSD.createElement("gBSDUnit");
data.setAttribute("marker",":label:(0,0)");
data.setAttribute("start", Integer.toString(cursor[0]));
Node VOP_data = ((Element) VOP[0].item(pos[0])).getElementsByTagNameNS(
  "MPEG4","VOP_data").item(0).getFirstChild();
int endpos = Integer.parseInt(VOP_data.getNodeValue().split(" ")[0]) +
```

```
    Integer.parseInt(VOP_data.getNodeValue().split(" ")[1]);
  data.setAttribute("length", Integer.toString(endpos - cursor[0]));
  size[0][0] += endpos - cursor[0];
  cursor[0] = endpos;
  parcel.appendChild(data);
  framecount[0]++; framecount[1]++;
  if (pos[0] + 1 < VOP[0].getLength()) {
    clockbase += (((Element) VOP[0].item(pos[0] + 1)).
      getElementsByTagNameNS("MPEG4","modulo_time_base").getLength() - 1) * clockunit;
    clock = clockbase + Integer.parseInt(((Element) VOP[0].item(pos[0] + 1)).
      getElementsByTagNameNS("MPEG4","vop_time_increment").
      item(0).getFirstChild().getNodeValue());
  }
  else
    clock += 2;

  /* add FGS enhancement layer to current parcel */
  data = outBSD.createElement("gBSDUnit");
  data.setAttribute("marker",":label:(0,1)");
  data.setAttribute("start", Integer.toString(cursor[1]));
  data.setAttribute("globalAddressInfo",bitstream[1]);
  VOP_data = ((Element) VOP[1].item(pos[0])).
    getElementsByTagNameNS("MPEG4","VOP_data").item(0).getFirstChild();
  endpos = Integer.parseInt(VOP_data.getNodeValue().split(" ")[0]) +
    Integer.parseInt(VOP_data.getNodeValue().split(" ")[1]);
  data.setAttribute("length", Integer.toString(endpos - cursor[1]));
  size[0][1] += endpos - cursor[1];
  cursor[1] = endpos;
  parcel.appendChild(data);
  //attach an appropriate label to each bitplane
  NodeList bps = ((Element) VOP[1].item(pos[0])).getElementsByTagNameNS("MPEG4","BitPlane");
```

```
for (int i = 0; i < bps.getLength(); i++) {
  data = outBSD.createElement("gBSDUnit");
  data.setAttribute("marker",":label:(0," + (i + 1) + ")");
  data.setAttribute("start", Integer.toString(cursor[1]));
  data.setAttribute("globalAddressInfo",bitstream[1]);
  VOP_data = ((Element) VOP[1].item(pos[0])).
    getElementsByTagNameNS("MPEG4","BP_data").item(i).getFirstChild();
  endpos = Integer.parseInt(VOP_data.getNodeValue().split(" ")[0]) +
    Integer.parseInt(VOP_data.getNodeValue().split(" ")[1]);
  data.setAttribute("length", Integer.toString(endpos - cursor[1]));
  size[0][i+1] += endpos - cursor[1];
  cursor[1] = endpos;
  parcel.appendChild(data);
}

/* add FGST enhancement layer to current parcel */
while (true) {
  if (pos[1] == VOP[2].getLength())
    break;
  int tempclockbase = fgstclockbase;
  fgstclockbase += (((Element) VOP[2].item(pos[1])).
    getElementsByTagNameNS("MPEG4","modulo_time_base").getLength() - 1) * clockunit;
  fgstclock = fgstclockbase + Integer.parseInt(((Element) VOP[2].item(pos[1])).
    getElementsByTagNameNS("MPEG4","vop_time_increment").item(0).
    getFirstChild().getNodeValue());
  if (fgstclock > clock) {
    fgstclockbase = tempclockbase; // reset clockbase
    break;
  }
  // header data belongs to label (1,0)
  data = outBSD.createElement("gBSDUnit");
```

```
    data.setAttribute("marker",":label:(1,0)");
    data.setAttribute("start", Integer.toString(cursor[2]));
    data.setAttribute("globalAddressInfo",bitstream[2]);
    VOP_data = ((Element) VOP[2].item(pos[1])).
      getElementsByTagNameNS("MPEG4","VOP_data").item(0).getFirstChild();
    endpos = Integer.parseInt(VOP_data.getNodeValue().split(" ")[0]) +
      Integer.parseInt(VOP_data.getNodeValue().split(" ")[1]);
    data.setAttribute("length", Integer.toString(endpos - cursor[2]));
    size[1][0] += endpos - cursor[2];
    cursor[2] = endpos;
    parcel.appendChild(data);
    //attach an appropriate label to each bitplane
    bps = ((Element) VOP[2].item(pos[1])).getElementsByTagNameNS("MPEG4","BitPlane");
    for (int i = 0; i < bps.getLength(); i++) {
      data = outBSD.createElement("gBSDUnit");
      data.setAttribute("marker",":label:(1," + (i + 1) + ")");
      data.setAttribute("start", Integer.toString(cursor[2]));
      data.setAttribute("globalAddressInfo",bitstream[2]);
      VOP_data = ((Element) VOP[2].item(pos[1])).
        getElementsByTagNameNS("MPEG4","BP_data").item(i).getFirstChild();
      endpos = Integer.parseInt(VOP_data.getNodeValue().split(" ")[0]) +
        Integer.parseInt(VOP_data.getNodeValue().split(" ")[1]);
      data.setAttribute("length", Integer.toString(endpos - cursor[2]));
      size[1][i+1] += endpos - cursor[2];
      cursor[2] = endpos;
      parcel.appendChild(data);
    }
    pos[1]++; framecount[1]++;
  }
  if (++pos[0] == VOP[0].getLength())
    end = true;
```

```
  }
  parcel.setAttribute("length",
    Integer.toString(cursor[0] - Integer.parseInt(parcel.getAttribute("start"))));
  description.appendChild(parcel);
  }
  catch (DOMException e) {
    System.out.println ("[ERROR]: failed to construct the gBSD description\n" + e);
  }

  /*** exporting gBSD description ***/
  try {
    Transformer idTransform = TransformerFactory.newInstance().newTransformer();
    idTransform.setOutputProperty("indent", "yes");
    idTransform.setOutputProperty("{http://xml.apache.org/xslt}indent-amount", "2");
    Source input = new DOMSource(outBSD);
    Result output = new StreamResult(out);
    idTransform.transform(input, output);
    //System.out.println ();
  }
  catch (TransformerException e) {
    System.out.println ("[ERROR]: failed to print out the gBSD description\n" + e);
  }
  try {
    out.close();
  }
  catch (IOException e) {
    System.out.println ("[ERROR]:\n" + e);
  }
 }
}
```

Appendix B

Sequences used in the subjective test

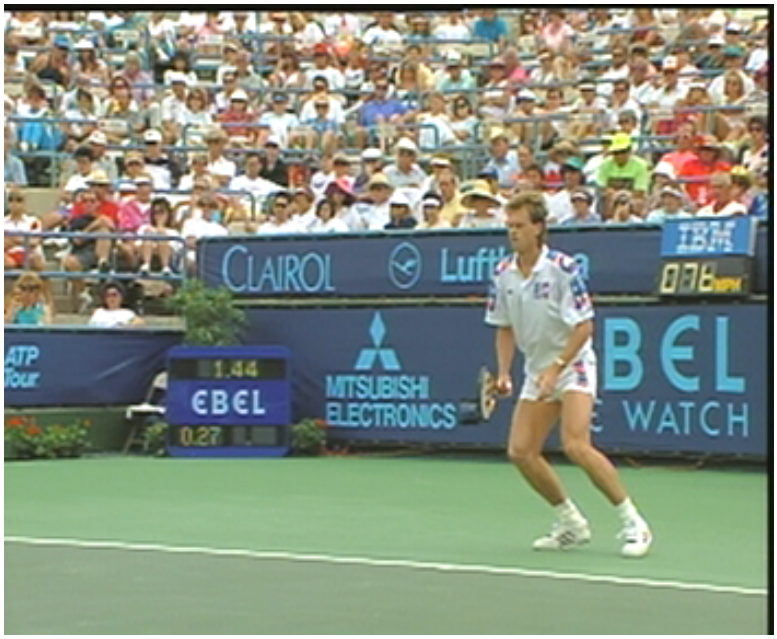
In this appendix, we show some images of the six sequences that were used in the subjective test presented in Chapter 6. For each sequence, we show one frame of the original sequence, and the same frame in the encoded version at 30 frames per second and CIF resolution, which is the version that contains the most severe amount of distortion.

The images that are shown on the left pages come from the original sequences that were used in the subjective test. These sequences were not presented to the participants, but were used for producing the actual sequences that were presented in the test.

On the right pages, the same frames are shown, this time extracted from the encoded sequences. In most images, the distortion introduced by the encoding process is clearly visible. The bit rates that were used for encoding these sequences can be found in Table 6.1; the PSNR values, indicating the amount of distortion, can be found in Table 6.2.

coastguard (original)

stefan (original)



coastguard (encoded)

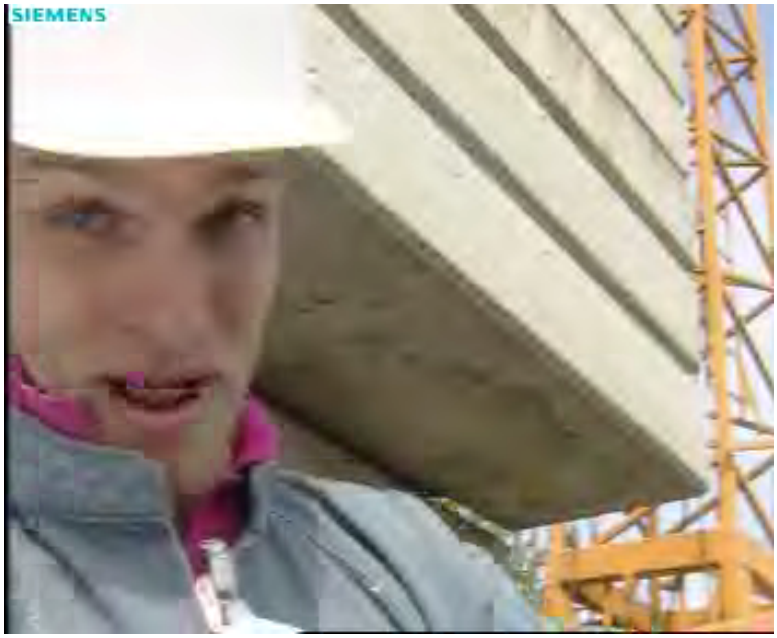
stefan (encoded)

akiyo (original)

foreman (original)

akiyo (encoded)

foreman (encoded)



mother (original)

silent (original)

mother (encoded)

silent (encoded)

References

[1] Ramesh Jain. Quality of experience. *IEEE Multimedia*, 11(1):95–96, January 2004.

[2] Fernando Pereira and Ian Burnett. Universal multimedia experiences for tomorrow. *IEEE Signal Processing Magazine*, 20(2):63–73, March 2003.

[3] Sam Lerouge, Peter Lambert, and Rik Van de Walle. Multi-criteria optimization for scalable bitstreams. In *Visual Content Processing and Representation, 8th International Workshop VLBV 2003*, volume 2849 of *Lecture Notes in Computer Science*, September 2003.

[4] Davy De Schrijver, Chris Poppe, Sam Lerouge, Wesley De Neve, and Rik Van de Walle. MPEG-21 bitstream syntax descriptions for scalable video codecs. *Multimedia Systems*, 2006. To appear.

[5] Sam Lerouge, Boris Rogge, Dimitri Van De Ville, Rik Van de Walle, and Jan Van Campenhout. An XML-based framework for content adaptation. In *Proceedings of Euromedia 2002*, pages 175–179, Modena, Italy, April 2002.

[6] Sam Lerouge, Boris Rogge, Robbie De Sutter, Jeroen Bekaert, Dimitri Van De Ville, and Rik Van de Walle. A generic mapping mechanism between content description metadata and user environments. In *Internet Multimedia Management Systems III*, volume 4862 of *Proceedings of SPIE*, July 2002.

[7] Sam Lerouge, Robbie De Sutter, Peter Lambert, and Rik Van de Walle. Fully scalable video coding in multicast applications. In *Proceedings of SPIE/Electronic Imaging 2004*, volume 5308, pages 555–564, San Jose, California, USA, January 2004.

[8] Sam Lerouge, Robbie De Sutter, and Rik Van de Walle. Personalizing quality aspects in scalable video coding. In *IEEE Proceedings of ICME 2005*, Amsterdam, The Netherlands, July 2005.

[9] Robbie De Sutter, Sam Lerouge, Jeroen Bekaert, Boris Rogge, Dimitri Van De Ville, and Rik Van de Walle. Dynamic adaptation of multimedia data for mobile applications. In *Internet Multimedia Management Systems III*, volume 4862 of *Proc. of SPIE*, July 2002.

[10] Jeroen Bekaert, Dimitri Van De Ville, Boris Rogge, Sam Lerouge, Robbie De Sutter, and Rik Van de Walle. Metadata-based access to multimedia architectural and historical archive collections. In *Internet Multimedia Management Systems III*, volume 4862 of *Proceedings of SPIE*, July 2002.

[11] Robbie De Sutter, Sam Lerouge, Jeroen Bekaert, and Rik Van de Walle. Dynamic adaptation of streaming MPEG-4 video for mobile applications. In *Proceedings of Euromedia 2003*, pages 185–190, Plymouth, UK, April 2003.

[12] Robbie De Sutter, Sam Lerouge, Wesley De Neve, Peter Lambert, and Rik Van de Walle. Advanced mobile multimedia applications: using MPEG-21 and time-dependent metadata. In *Proceedings of SPIE/ITCom 2003*, volume 5241, pages 147–156, Orlando, USA, September 2003.

[13] Wesley De Neve, Peter Lambert, Sam Lerouge, and Rik Van de Walle. Assessment of the compression efficiency of the MPEG-4 AVC specification. In *Proceedings of SPIE/Electronic Imaging 2004*, volume 5308, pages 1082–1093, San Jose, California, USA, January 2004.

[14] Wesley De Neve, Sam Lerouge, Peter Lambert, and Rik Van de Walle. A performance evaluation of MPEG-21 BSDL in the context of H.264/AVC. In *Proceedings of SPIE 2004: Signal and Image Processing and Sensors*, volume 5558, pages 555–566, Denver, Colorado, USA, August 2004.

[15] Robbie De Sutter, Sam Lerouge, Davy De Schrijver, and Rik Van de Walle. Enhancing RSS feeds: Eliminating overhead through binary encoding. In *Proceedings of the 3rd International Conference on Information Technology and Applications*, Sidney, Australia, July 2005.

[16] Fernando Pereira and Touradj Ebrahimi. *The MPEG-4 Book*. Prentice Hall, New Jersey, 2002.

[17] Sungdae Cho and William A. Pearlman. A full-featured, error-resilient, scalable wavelet video codec based on the set partitioning in hierarchical trees (SPIHT) algorithm. *IEEE Transactions on Circuits and Systems for Video Technology*, 12(3):157–171, March 2002.

[18] Susie J. Wee and John G. Apostolopoulos. Secure scalable streaming enabling transcoding without decryption. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, Thessaloniki, Greece, October 2001.

[19] Chun Yuan, Bin B. Zhu, Yidong Wang, Shipeng Li, and Yuzhuo Zhong. Efficient and fully scalable encryption for MPEG-4 FGS. In *Proceedings of the International Symposium on Circuits and Systems (ISCAS)*, Bangkok, Thailand, May 2003.

[20] Barry G. Haskell, Atul Puri, and Arun N. Netravali. *Digital Video: an Introduction to MPEG-2*. Chapman and Hall, 1997.

[21] Weiping Li. Overview of Fine Granularity Scalability in MPEG-4 video standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(3):301–317, March 2001.

[22] Hayder M. Radha, Mihaela van der Schaar, and Yingwei Chen. The MPEG-4 Fine-Grained Scalable video coding method for multimedia streaming over IP. *IEEE Transactions on Multimedia*, 3(1):53–68, March 2001.

[23] Majid Rabbani and Rajan Joshi. An overview of the JPEG 2000 still image compression standard. *Signal Processing: Image Communication*, 17(1):3–48, January 2002.

[24] Jens-Rainer Ohm. Motion-compensated 3-D subband coding with multiresolution representation of motion parameters. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, volume 3, pages 250–254, 1994.

[25] Jens-Rainer Ohm. Three-dimensional subband coding with motion compensation. *IEEE Transactions on Image Processing*, 3(5):559–571, September 1994.

[26] Shih-Ta Hsiang and John W. Woods. Embedded video coding using invertible motion compensated 3-D subband/wavelet filter bank. *Signal Processing: Image Communication*, 16(8):705–724, May 2001.

[27] Seung-Jong Choi and John W. Woods. Motion-compensated 3-D subband coding of video. *IEEE Transactions on Image Processing*, 8(2):155–167, February 1999.

[28] Yiannis Andreopoulos, Mihaela van der Schaar, Adrian Munteanu, Peter Schelkens, and Jan Cornelis. Fully-scalable wavelet video coding using in-band motion compensated temporal filtering. In *Proc. IEEE International Conference on Acoustics Speech, and Signal Processing (ICASSP)*, volume 3, pages 417–420, Hong Kong, China, March 2003.

[29] Yiannis Andreopoulos, Adrian Munteanu, Joeri Barbarien, Mihaela van der Schaar, Jan Cornelis, and Peter Schelkens. In-band motion compensated temporal filtering. *Signal Processing: Image Communication*, 19(5):653–673, August 2004.

[30] Deepak S. Turaga, Mihaela van der Schaar, Yiannis Andreopoulos, Adrian Munteanu, and Peter Schelkens. Unconstrained motion compensated temporal filtering (UMCTF) for efficient and flexible interframe wavelet video coding. *Signal Processing: Image Communication*, 20(1):1–19, January 2005.

[31] Thomas Wiegand, Heiko Schwarz, Anthony Joch, Faouzi Kossentini, and Gary J. Sullivan. Rate-constrained coder control and comparison of video coding standards. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):688– 703, July 2003.

[32] Thomas Wiegand, Gary J. Sullivan, Gisle Bjontegaard, and Ajay Luthra. Overview of the H.264/AVC video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):560–576, July 2003.

[33] Wesley De Neve, Frederik De Keukelaere, Koen De Wolf, and Rik Van de Walle. Applying MPEG-21 BSDL to the JVT H.264/AVC specification in

MPEG-21 session mobility scenarios. In *Proc. 5th Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS)*, pages 4–7, Lisboa, Portugal, April 2004.

[34] Wesley De Neve, Davy Van Deursen, Davy De Schrijver, Koen De Wolf, and Rik Van de Walle. Using bitstream structure descriptions for the exploitation of multi-layer temporal scalability in H.264/MPEG-4 AVC's base specification. In *Proceedings of the Pacific-Rim Conference on Multimedia (PCM)*, Jeju, Korea, November 2005.

[35] Heiko Schwarz, Detlev Marpe, and Thomas Wiegand. MCTF and scalability extension of H.264/AVC. In *Proceedings of PCS'04*, San Francisco, CA, USA, December 2004.

[36] Ralf Schäfer, Heiko Schwarz, Detlev Marpe, Thomas Schierl, and Thomas Wiegand. MCTF and scalability extension of H.264/AVC and its application to video transmission, storage and surveillance. In *Proceedings of Visual Communications and Image Processing (VCIP) 2005*, Beijing, China, July 2005.

[37] ISO/IEC. ISO/IEC 21000-2:2003 Information technology – Multimedia framework (MPEG-21) – Part 1: Vision, Technologies and Strategy, November 2004.

[38] Ian Burnett, Rik Van de Walle, Keith Hill, Jan Bormans, and Fernando Pereira. MPEG-21: Goals and achievements. *IEEE Multimedia*, 10(4):60–70, October 2003.

[39] Jan Bormans, Jean Gelissen, and Andrew Perkis. MPEG-21: The 21st century multimedia framework. *IEEE Signal Processing Magazine*, 20(2):53–62, March 2003.

[40] ISO/IEC. ISO/IEC 21000-2:2003 Information technology – Multimedia framework (MPEG-21) – Part 2: Digital Item Declaration second edition, July 2005.

[41] ISO/IEC. ISO/IEC 21000-7:2004 Information technology – Multimedia framework (MPEG-21) – Part 7: Digital Item Adaptation, October 2004.

[42] Frederik De Keukelaere and Rik Van de Walle. Digital item declaration and identification. In *The MPEG-21 Book*. To appear, 2005.

[43] Ian S. Burnett, Stephen J. Davis, and Gerrard M. Drury. MPEG-21 Digital Item Declaration and Identification – principles and compression. *IEEE Transactions on Multimedia*, 7(3):400–407, June 2005.

[44] B.S. Manjunath, Philippe Salembier, and Thomas Sikora. *Introduction to MPEG-7: Multimedia Content Description Interface*. Wiley, New Jersey, 2003.

[45] Anthony Vetro. MPEG-21 Digital Item Adaptation: Enabling universal multimedia access. *IEEE Multimedia*, 11(1):84–87, January 2004.

[46] Anthony Vetro and Christian Timmerer. Digital Item Adaptation: Overview of standardization and research activities. *IEEE Transactions on Multimedia*, 7(3):418–426, June 2005.

[47] Debargha Mukherjee, Eric Delfosse, Jae-Gon Kim, and Yong Wang. Optimal adaptation decision-taking for terminal and network quality of service. *IEEE Transactions on Multimedia*, 7(3):454–462, June 2005.

[48] Jae-Gon Kim, Yong Wang, and Shih-Fu Chang. Content-adaptive utility-based video adaptation. In *IEEE International Conference on Multimedia and Expo (ICME)*, Baltimore, Maryland, USA, July 2003.

[49] Anthony Vetro, Charilaos Christopoulos, and Huifang Sun. Video transcoding architectures and techniques: an overview. *IEEE Signal Processing Magazine*, 20(2):18–29, March 2003.

[50] Christian Timmerer, Gabriel Panis, Harald Kosch, Jörg Heuer, Hermann Hellwagner, and Andreas Hutter. Coding format independent multimedia content adaptation using XML. In *Internet Multimedia Management Systems IV*, volume 5242 of *Proceedings of SPIE*, September 2003.

[51] Mihaela van der Schaar and Yiannis Andreopoulos. Rate-distortion-complexity modeling for network and receiver aware adaptation. *IEEE Transactions on Multimedia*, 7(3):471–479, June 2005.

[52] Tim Bray, Jean Paoli, C.M. Sperberg-McQueen, Eve Maler, and François Yergeau. Extensible Markup Language (XML) 1.0 (third edition). Technical report, W3C, February 2004.

[53] Kal Ahmed, Danny Ayers, Mark Birbeck, Jay Cousins, David Dodds, Josh Lubell, Miloslav Nic, Daniel Rivers-Moore, Andrew Watt, Robert Worden, and Ann Wrightson. *XML Meta Data*. Wrox, 2001.

[54] Myriam Amielh and Sylvain Devillers. Multimedia content adaptation with XML. In *8th International Conference on Multimedia Modeling MMM2001*, pages 127–145, Amsterdam, The Netherlands, November 2001.

[55] Myriam Amielh and Sylvain Devillers. Bitstream syntax description language: Application of XML schema to multimedia content adaptation. In *Proceedings of the Eleventh International World Wide Web Conference*, Honolulu, Hawaii, USA, May 2002.

[56] Sylvain Devillers. XML and XSLT modeling for multimedia bitstream manipulation. In *Poster Proceedings of the Tenth International World Wide Web Conference*, Hong Kong, China, May 2001.

[57] Gabriel Panis, Andreas Hutter, Jörg Heuer, Hermann Hellwagner, Harald Kosch, Christian Timmerer, Sylvain Devillers, and Myriam Amielh. Bitstream syntax description: a tool for multimedia resource adaptation within MPEG-21. *Signal Processing: Image Communication*, 18(8):699–719, September 2003.

[58] James Clark. XSL Transformations (XSLT) version 1.0. Recommendation, W3C, November 1999.

[59] Davy De Schrijver, Wim Van Lancker, and Rik Van de Walle. Performance of a scalable bitstream adaptation process based on high level XML descriptions. In *Proceedings of the Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS)*, Montreux, Switzerland, April 2005.

[60] David C. Fallside and Priscilla Walmsley. XML Schema part 0: Primer (second edition). Recommendation, W3C, October 2004.

[61] Gauthier Lafruit, Eric Delfosse, Roberto Osorio, Wolfgang van Raemdonck, Vissarion Ferentinos, and Jan Bormans. View-dependent, scalable texture streaming in 3-D QoS with MPEG-4 visual texture coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(7):1021–1031, July 2004.

[62] Thomas Di Giacomo, Chris Joslin, Stéphane Garchery, HyungSeok Kim, and Nadia Magnenat-Thalmann. Adaptation of virtual human animation and representation for MPEG. *Computers & Graphics*, 28(4):65–74, August 2004.

[63] Alexandros Eleftheriadis. Flavor: A language for media representation. In *Proceedings of ACM Multimedia*, pages 1–9, Seattle, WA, USA, November 1997.

[64] Danny Hong and Alexandros Eleftheriadis. XFLAVOR: Bridging bits and objects in media representation. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, Lausanne, Switzerland, August 2002.

[65] Alexandros Eleftheriadis and Danny Hong. Flavor: A formal language for audio-visual object representation. In *Proceedings of ACM International Conference on Multimedia*, New York, NY, USA, October 2004.

[66] Debargha Mukherjee and Amir Said. Structured scalable meta-formats (SSM) for Digital Item Adaptation. In *Internet Imaging IV*, volume 5018 of *Proceedings of SPIE*, January 2003.

[67] Debargha Mukherjee, Amir Said, and Sam Liu. A framework for fully format-independent adaptation of scalable bit-streams. *IEEE Transactions on Circuits and Systems for Video Technology, Special Issue on Video Adaptation*, 15(10):1280–1290, October 2005.

[68] Sylvain Devillers, Christian Timmerer, Jörg Heuer, and Hermann Hellwagner. Bitstream syntax description-based adaptation in streaming and constrained environments. *IEEE Transactions on Multimedia*, 7(3):463–470, June 2005.

[69] Robbie De Sutter, Christian Timmerer, Hermann Hellwagner, and Rik Van de Walle. Multimedia metadata processing: a format independent approach. In *Proceedings of the 9th IASTED International Conference Internet and Multimedia Systems and Applications (EuroIMSA)*, pages 343–348, Grindelwald, Switzerland, February 2005.

[70] Martin Fowler and Kendall Scott. *UML Distilled: A Brief Guide to the Standard Object Modeling Language (2nd Edition)*. Addison-Wesley, 1999.

[71] Shih-Fu Chang and Anthony Vetro. Video adaptation: Concepts, technologies, and open issues. *Proceedings of IEEE, Special Issue on Advances in Video Coding and Delivery*, 93(1):148–158, January 2005.

[72] Debargha Mukherjee, Huisheng Wang, Amir Said, and Sam Liu. Format-agnostic adaptation using the MPEG-21 DIA framework. In *Proceedings of SPIE 2004: Applications of Digital Image Processing*, volume 5558, pages 351–362, Denver, Colorado, USA, November 2004.

[73] Anthony Vetro, Charilaos Christopoulos, and Touradj Ebrahimi. Universal multimedia access. *IEEE Signal Processing Magazine*, 20(2):16, March 2003.

[74] Peter Soetens, Matthias De Geyter, and Stijn Decneut. Multi-step media adaptation with semantic web services. In *Proceedings of the Third International Semantic Web Conference (ISWC 2004)*, Hiroshima, Japan, November 2004.

[75] Ibtissam El Khayat and Guy Leduc. A stable and flexible TCP-friendly congestion control protocol for layered multicast transmission. In *Interactive Distributed Multimedia Systems*, volume 2158 of *Lecture Notes in Computer Science*, September 2001.

[76] Vilfredo Pareto. *Cours d'Economie Politique*. Lausanne, 1896.

[77] Ralph E. Steuer. *Multiple criteria optimization: theory, computation and application*. Krieger Publishing Company, 1986.

[78] Ralph L. Keeney and Howard Raiffa. *Decisions with Multiple Objectives: Preferences and value trade-offs*. Wiley, New York, 1976.

[79] Eric C. Reed and Jae S. Lim. Optimal multidimensional bit-rate control for video communication. *IEEE Transactions on Image Processing*, 11(8):873 – 885, August 2002.

[80] Stefan Winkler and Christof Faller. Maximizing audiovisual quality at low bitrates. In *Proceedings of the Workshop on Video Processing and Quality Metrics*, Scottsdale, Arizona, USA, January 2005.

[81] Paul Bockeck, Andrew T. Campbell, Shih-Fu Chang, and Raymond R.-F. Liao. Utility-based network adaptation for MPEG-4 systems. In *Proceedings of NOSSDAV 1999*, Basking Ridge, New Jersey, USA, June 1999.

[82] Rakesh Mohan, John R. Smith, and Chung-Sheng Li. Adapting multimedia internet content for universal access. *IEEE Transactions on Multimedia*, 1(1):104–114, March 1999.

[83] Carlos E. Luna, Lisimachos P. Kondi, and Aggelos K. Katsaggelos. Maximizing user utility in video streaming applications. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(2):141 – 148, February 2003.

[84] Thomas L. Saaty. *The Analytic Hierarchy Process*. McGraw-Hill, New York, 1980.

[85] Pattie Maes. Agents that reduce work and information overload. *Communications of the ACM*, 37(7):31–40, July 1994.

[86] Bradley J. Rhodes and Pattie Maes. Just-in-time information retrieval agents. *IBM Systems Journal*, 39(3-4):685–704, 2000.

[87] Peter van Beek, John R. Smith, Touradj Ebrahimi, Teruhiko Suzuki, and Joel Askelof. Metadata-driven multimedia access. *IEEE Signal Processing Magazine*, 20(2):40–52, March 2003.

[88] Gustav Theodor Fechner. *Vorschule der Aesthetik*. Breikopf & Härterl, Leipzig, 1876.

[89] George A. Gescheider. *Psychophysics: Method, Theory and Application*. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1985.

[90] Louis Leon Thurstone. A law of comparative judgement. *Psychological Review*, 34:273–286, 1927.

[91] George B. Dantzig and B. Curtis Eaves. Fourier-Motzkin elimination and its dual. *Journal of Combinatorial Theory (A)*, 14(3):288–297, May 1974.

[92] L. Karl Branting and Patrick S. Broos. Automated acquisition of user preferences. *International Journal of Human-Computer Studies*, 46:55–77, 1997.

[93] Jenq-Nenq Hwang, Tzong-Der Wu, and Chia-Wen Lin. Dynamic frame-skipping in video transcoding. In *IEEE Workshop on Multimedia Signal Processing*, pages 616–621, December 1998.

[94] Giovanni Iacovoni, Salvatore Morsa, and Renzo Felice. Quality-temporal transcoder driven by the jerkiness. In *IEEE Proceedings of ICME 2005*, Amsterdam, The Netherlands, July 2005.

[95] Yong Wang, Mihaela van der Schaar, Shih-Fu Chang, and Alexander C. Loui. Classification-based multi-dimensional adaptation prediction for scalable video coding using subjective quality evaluation. *IEEE Transactions on Circuits and Systems for Video Technology, Special Issue on Analysis and Understanding for Video Adaptation*, 15(10):1270–1279, October 2005.

[96] Özgür D. Önür and A. Aydin Alatan. Video adaptation for transmission channels by utility modeling. In *IEEE Proceedings of ICME 2005*, Amsterdam, The Netherlands, July 2005.

[97] ITU. ITU-R recommendation BT.500-11 "methodology for the subjective assessment of the quality of television pictures, 2002.

[98] A. Murat Tekalp. *Digital Video Processing*. Prentice-Hall, Englewood Cliffs, NJ, 1995.

[99] VQEG. Final report from the Video Quality Experts Group on the validation of objective models of video quality assessment, phase II. http://www.vqeg.org, August 2003.

[100] Zhou Wang, Ligang Lu, and Alan C. Bovik. Video quality assessment based on structural distortion measurement. *Signal Processing: Image Communication, Special issue on objective video quality metrics*, 19(2):121–132, February 2004.