MPEG-4:
Next Generation Standard for Interactive Media

Course Organizer

Klaus Diepold
DynaPel Laboratories GmbH
Fraunhoferstr. 9, D-85737 Ismaning, Germany
Tel: +49 89 9624-2814, Fax: +49 89 9624-2890
kldi@computer.org

Presenters

Klaus Diepold, DynaPel Systems, Inc.
Radek Grzeszczuk, Intel Corp.
Igor Pandzic, University of Linköping
Eric Petajan, Face2face animation, inc.
Iraj Sodagar. PacketVideo Corp.
JC Spierer, MITRE Corp.
Gabriel Taubin, IBM

Summary

The graphics and content creation community will get insight into the opportunities offered by MPEG-4 as an attractive target format for the creation and delivery of rich and interactive media.

The media is meant to be delivered to users via a wide range of networks and channels. This inlcudes the Internet, and also broadcast or wireless channels. The list of target receiver devices goes from set-top boxes down to PDAs.

MPEG-4 employs an object oriented approach, addressing the seamless and transparent combination of locally stored and remotely accessed media objects into one presentation. This approach is the central point of MPEG-4 and the global convergence theme, where technical aspects of telecommunications, computer and entertainment are merged into a unifying content delivery framework.

The course provides general information about the MPEG-4 standard, the technology included and supported as well as exmaples of applications. Further, since MPEG-4 provides a standardized framework for the transport of interactive content that includes computer graphics techniques next to video and audio, the course provides information and demonstrations of this framework in action.

This course provides information and demonstrations about the MPEG-4 new media standard, the technology included and supported. It will be emphasized how interactive

media evolved under this standard as well as how computer graphics aspects have become part of an international standard that is traditionally associated with video and audio only.

This session will be heavily grounded in demonstrations and case studies.

Prerequisites

Participants should have a basic understanding of various media types including video, audio, images and computer graphics.

Topics List

- MPEG-4 Overview – Anatomy and Rationale for the standard
- Synthetic and Natural Media Objects – Video, Audio, Graphics
- Compositing of Media Objects – Scene Description
- Compression of Media Objects and Scene Description
- Basic concepts in MPEG-4:
    - Scalability – temporal, spatial, object
    - Compositing/Synchronization – temporal and spatial
    - Stream Management and Streaming
    - Delivery over heterogeneous networks – Multiplexing
    - Interactivity – local and remote interactivity, backchannel
    - Error Resilience
    - File Formats
- Digital Rights Management - Intellectual Property Management and Protection
- Putting it all together – Profiles and Levels in MPEG-4
- Applications – Internet Streaming, Wireless, Broadcast

Course Syllabus

8:30 - 9:00 - Introduction to MPEG-4 (Klaus Diepold):

- Overview of the MPEG-4 standard
- Define media and the role of the standard

9:00 – 9:30 - Streams of Media Objects (JC Spierer):

- Architectural issues of MPEG-4
- Delivery of Media Objects,
- Synchronization of Objects and Streams
- Object Descriptors

9:30 - 10:15 - Delivery of Object Streams (JC Spierer)

- Architecture of a networked media delivery system
- Interactive media
- Delivery of media over heterogeneous networks
- MP4 file format

10:15 - 10:30 - Break

10:30 - 11:15 - Video and Image Coding (Iraj Sodagar)

- Overview of MPEG-4 visual aka video coompression
- MPEG-4 video coding tools including temporal, spatial and object based scalability, error resilience and shape coding;
- Still image coding tools in MPEG-4 (Wavelets)

11:15 - 12:15 - Binary Format for Scene Description (Klaus Diepold)

- Features of the Sceme description framework,
- Relation to Virtual Reality Modelling Language (VRML)
- 2D/3D nodes and Streaming nodes of a tree structured Scene Description
- Encoding and quantization of scene descriptions
- Animation of scene decriptions

12:15 - 1:30 - Lunch

1:30 - 2:15 - Synthetic Media Objects (Gabriel Taubin)

- 3D Mesh Coding Tools
    - Polygon mesh representation / decimation / compression
    - Single Resolution mode
    - Non-Manifold coding
    - Error Resiliency
    - Progressive Transmission mode
- 2D Mesh Coding Tools
    - Representation
    - Animation

2:15 – 3:00 - Light Field Mapping in MPEG-4 (Radek Grzeszczuk)

- Texturing tools: Image-based rendering
- Light field mapping
- multi-texturing framework

3:15 - 3:30 - Break

3:30 – 4:15 - Synthetic Media Objects (Eric Petajan, Igor Pandzic)

- Face and Body Animation
- Text-to-Speech Synthesis
- Applications in Communcations and Broadcast

4:15 – 4:45 - Protecting your assets Intro to (JC Spierer)

- Intelectual Property Managament and Protection
- Digital Rights Management

4:45 - 5:15 - Putting it all together (Klaus Diepold)
- Interoperability
- Conformance points
- Profiles and Levels in MPEG-4

5:15 - 5:30 - Proliferation of MPEG-4 (Iraj)
- Where is the Standard to be applied ?

1. Course History

Course Outlook

The content and technologies presented in this course provide tools and opportunities on how to create interactive content using a blend of media technologies. The list of covered media technologies starts with video and audio. For MPEG-4 this list also includes images, text, graphics, animation, synthetic audio, VRML-like scene descriptions and much more.

The MPEG-family of standards is focused on how content can be transported to users via existing or next-generation transmission channels in an economic and reliable way. Those transmission channels include broadcast channels (terrestrial, satellite, and cable) as well as IP networks (Internet, LAN) along with wireless channels such as UMTS or GPRS. It is of vital importance for the success of the MPEG-4 standard that all stakeholders fully understand the tremendous opportunity to build new applications and to create a ubiquitous infrastructure for delivering new and interactive media.

The list of stakeholders includes, among others, content creators as much as system developers and service providers. Computer graphics concepts have made it into the standard it is a topic of long-term importance to the computer graphics community to either build applications or to support the future standardization efforts.

New Contributions

The MPEG-1/-2 standard has been covered to some extend throughout the past years at SIGGRAPH courses as a means to compress video. MPEG-4 goes beyond the point of only addressing video functionality and compression.

MPEG-4 is novel as it is based on the concept that a media presentation may be seen as the composition of various media types, such as video and graphics, as well as audio and animation, or synthetic and natural media types. Since the compositing of such objects is done in the receiver, this architecture allows for various levels of user interaction. Such a presentation is delivered in a complete framework to enable compelling media applications.

The course highlights how the technical implications of content delivery mechanisms may influence the content creation and representation process. Compression and bandwidth efficiency are among those influences. In summary, the novelty of the course is the integration of video, audio and graphics into one comprehensive standards-based delivery framework.

Relationship to previous SIGGRAPH Courses

Some of the more graphics oriented topics have been covered by previous SIGGRAPH courses, such as the compression of 3D meshes, or the use of VRML as a means for scene description, that is the composition of various of media objects to form a presentation. There were also courses delivered in the past to provide information on video compression technology such as MPEG-1.

There have been courses on technologies that are complemtary to the material presented in the context of this MPEG-4 course, such as Open ML (9/2001), SMIL (22/2001) (46/2001) Video Compression (9/43/44 1998), VRML (18/1998), 3D mesh Compression (21/1998), Web Content Creation (2/1997), Video for Web (20/1997). MPEG-4 provides a framework to combine techniques and concepts inot a unifying standard for interactive media.

Course Presenters' Biography

Klaus Diepold, CEO
DynaPel Systems, Inc./ Munich University of Technology
Email: kldi@ei.tum.de

Klaus Diepold is CTO of DynaPel Systems, Inc. a New York based company in the video processing and compression industry. He is working out of Munich based DynaPel Laboratories the research and development facility of DynaPel. Klaus Diepold is the lead architect of DynaPel's award winning video post-production tools MotionPerfect as well as its successor SteadyHand. His current focus is on developing motion estimation techniques and implementing MPEG-4 video CODECs. Klaus Diepold received a Dipl.-Ing. degree and a Dr.-Ing. degree both in Electrical Engineering from Munich University of Technology in 1987 and 1992, respectively. 1993 he received the best paper award from the German Society of Information Technology (ITG). He has been working for the IRT in Munich, being engaged in research projects contributing to the DVB-T specification for digital television systems. Since 1996 he is an active member of the MPEG group, mainly focusing on video aspects in the MPEG-4 video and requirements subgroups. He is also actively involved with the MPEG-4 Industry Forum to promote the new standard. Recently he got appointed as full professor at Munich University of Technology, Department of Electrical Engineering.

Radek Grzeszczuk
Intel Corporation
Email: radek.Grzeszczuk@intel.com

Radek Grzeszczuk joined Intel in 1998 as a Senior Researcher. He Received his Ph.D. degree (1998) and his M.Sc. degree (1994) in Computer Science from University of Toronto. His PhD thesis research was done under the supervision of Demetri Terzopoulos and Geoffrey Hinton and focused on using neural networks for fast emulation and control of physics-based models. The results of this work were published at SIGGRAPH'98 and NIPS'98. Radek Grzeszczuk's pioneering research with Steven Gortler, Michael Cohen and Richard Szeliski at Microsoft Research Graphics Group on image-based rendering culminated in the publication of ``The Lumigraph'' at SIGGRAPH'96. His recent work on image-based modeling and rendering focuses on methods for efficient representation and visualization of complex shape and reflectance properties of objects. He published a number of important scientific papers, primarily in computer graphics, but also in artificial life, neural networks, and computer vision. He received an award in 1995 from Ars Electronica, the premier competition for creative work with digital media, for his work on artificial animals for computer animation and virtual reality.

Igor Pandzic
University of Linköping
Email: igor@isy.liu.se

Igor S. Pandzic is currently a Visiting Scientist at the University of Linköping, Sweden, as well as a Visiting Professor at the University of Zagreb, Croatia. Formerly he worked as a Senior Assistant at MIRALab, University of Geneva, Switzerland, where he obtained his PhD in 1998. The same year he worked as visiting scientist at AT&T Labs, USA. Igor received his BSc degree in Electrical Engineering from the University of Zagreb in 1993, and MSc degrees from the Swiss Federal Institute of Technology (EPFL) and the University of Geneva in 1994 and 1995, respectively. His current research interests focus on Virtual Characters for the Internet and mobile platforms, and include Networked Collaborative Virtual Environments, facial analysis and synthesis, computer generated film production and parallel computing. He published numerous papers and a book on these topics. Igor is a member of the Moving Pictures Expert Group (MPEG) and contributed to the MPEG-4 International Standard. He is involved in undergraduate and postgraduate teaching activities at Linköping and Zagreb Universities, respectively. He serves in Program Committees of several conferences and a journal.

Eric Petajan
Face2face animation, inc.
Email: eric@f2f-inc.com

Dr. Eric Petajan is Chief Scientist and Cofounder of face2face animation, inc (a Lucent New Venture) and chairs the MPEG-4 Face and Body Animation (FBA) group. Prior to forming face2face, Eric was a Bell Labs researcher where he developed facial motion capture, HD video coding, and interactive graphics systems.
He received a PhD in EE in 1984 and an MS in Physics from the U. of Illinois where he built the first automatic lipreading system. Eric is also associate editor of the IEEE Transactions on Circuits and Systems for Video Technology.

Iraj Sodagar
PacketVideo Corporation
Email: iraj@pv.com

Iraj Sodagar is currently an Executive Member of Technical Staff, leading advanced architecture and technology development at PacketVideo Corporation, San Diego, CA. He received the B.S. degree in electrical engineering from Tehran University, Tehran, Iran and the M.S. degree and the Ph.D. degree in electrical engineering from Georgia Institute of Technology, Atlanta, in 1993 and 1994, respectively. In January 1995, He joined Sarnoff Corporation where he developed low bit rate image and video compression algorithms for multimedia applications. Last, he was the Head of Sarnoff's Interactive Media Group and responsible for the technical research and developments in the areas of MPEG-4, MPEG-7, JPEG-2000, VLBV coding and interactive multimedia. He also represented Sarnoff at ANSI NCITS, ISO MPEG-4 and JPEG-2000 standard bodies from 1995 to 1999. He joined Packetvideo Corp. in Feb 2000. He has served as the chair of MPEG-4 visual texture coding Ad-Hoc group and the co-chair of JPEG-2000 scalability and progressive-to-lossless coding Ad-Hoc group. He is currently the Technical Working Group Chair at Wireless Multimedia Forum (WMF) and has been representing PacketVideo in 3GPP2, ISMA and M4IF. He has authored and co-authored several technical papers. His current interests include interactive media streaming and content-based multimedia representation.

JC Spierer
MITRE Corp.
Email: jcspiere@mitre.org

JC Spierer is a project leader and technical architect at The MITRE Corp. in the Corporate Center for information & Technology (Bedford, MA) in the areas of collaboration and new media. He has led projects ranging from next generation immersion and media convergence to enterprise design/architecture, development and implementation of the VTC Team Rooms. He was formerly principal project leader at OtherVision working on both technical integration efforts as well as international technology strategies for clients such as Yahoo!, Inc and AT&T. Previously, he served as project analyst at Cambridge Technology Partners, a leading technology integrator. Educated at Washington University (St. Louis) and Harvard University, JC has published in the areas of collaboration and media technologies, international technology, technology policy and communications media markets.

Gabriel Taubin
IBM, T. J. Watson Research Center
Email: taubin@computer.org

Gabriel Taubin is a Research Staff Member at the IBM T.J. Watson Research Center. He joined IBM in 1990 as member of the Exploratory Computer Vision group, from 1996 to 2000 he was Manager of the Visual and Geometric Computing Group, and during the

2000-2001 academic year he was on sabbatical at CalTech as Visiting Professor of Electrical Engineering. He was named IEEE Fellow for his contributions to the development of three-dimensional geometry compression technology and multimedia standards. He earned a Ph.D. degree in Electrical Engineering from Brown University, and a Licenciado en Ciencias Matematicas degree from the University of Buenos Aires, Argentina. His main research interests fall into the following disciplines: Applied Computational Geometry, Computer Graphics, Geometric Modeling, 3D Photography, and Computer Vision. He made significant contributions to 3D capturing and surface reconstruction, modeling, compression, progressive transmission, and display of polygonal meshes. The 3D geometry compression technology that he developed is now part of the MPEG-4 standard and the IBM HotMedia product.

Overview of Course

- Speakers
  - JC Spierer – MITRE Corp.
  - Radek Radek Grzeszczuk – Intel
  - Gabriel Taubin – IBM
  - Iraj Sodagar – PacketVideo
  - Igor Pandzic – Linköping University
  - Eric Petajan – Face2Face
  - Klaus Diepold – DynaPel Systems

Summary

- Overview (Klaus)
- Part 1: Systems
  - Scene Description (Klaus)
  - Face and Body Animation (Igor, Eric)
  - Delivery of Streams (JC)
  - Content Protection (JC)
- Part 2: Visual
  - Video and Image Compression (Iraj)
  - Mesh Compression (Gabriel)
  - Light Field Rendering/AFX (Radek)
- Profiling (Klaus)

Introduction to MPEG-4

Klaus Diepold
Munich University of Technology

What is MPEG ?

- Official ISO work group title
  - ISO/IEC Joint Technical Committee 1, Sub Committee 29, Work Group 11
- MPEG stands for **M**oving **P**icture **E**xperts **G**roup
- Topic: Codiing of Video and Audio
- Related groups under the ISO umbrella
  - JPEG/JPEG2000 (WG 1)
  - JBIG (WG 1)
  - VRML (WG 24)
  - MHEG (WG 12)

Problems of Standardistion

- Frozen Technology
- Obsolete Technology
- Overspecification
- Underspecifiaction
- Delayed Delivery
- Patents and IPRs

Principles of MPEG Standards

- A-priori Standardisation
- Guaranteed delivery on time
- Don’t specify systems – only tools
- Specify only the minimum
  - Specify Bitstream Syntax and Decoder Semantics
  - Encoder is never specified !
- One functionality – one Tool
- Modularity of tools
- Verification of the Standard

Track Record: MPEG Standards

1990	**JPEG**	"*Coding of still images*"
1992	**MPEG-1**	"*Coding for digital Storage media* "
1994	**MPEG-2**	"*Coding for digital TV and DVD*"
1998	**MPEG-4**	"*Interactive Multimedia A/V Objects*"
2001	**MPEG-7**	"*Interface for Content Description*"
2003	**MPEG-21**	"*Multimedia Framwork*"

What happened to **MPEG-3**, **MPEG-5** und **MPEG-6** **?**

The most important objectives

- Common technology for many types of services:
  - interactive, broadcast, conversational
- Allowing more & different interactivity - not just stop/play/slow, but interactivity involving elements within the 'scene'
- Integrating natural and synthetic content
- Covering a wide range of access conditions
  - Includes low bitrates, error resilience, scalable coding
- Helping to manage and protect Intellectual Property

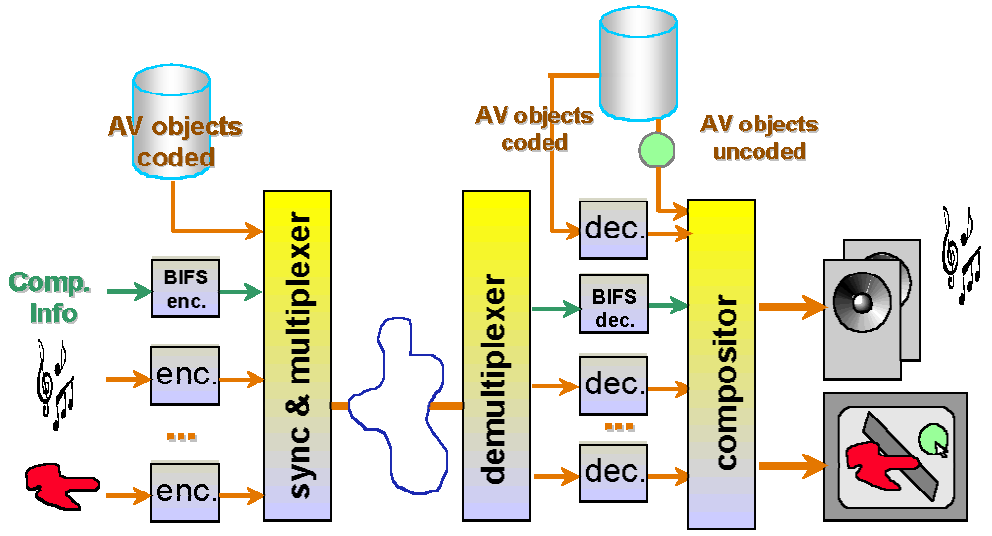
MPEG-4 Standard

- Generic Toolbox of Technologies to cover a wide range of applications
- MPEG-4 applications are transport agnostic
- Support for a new way of interactivity based on content and semantics
- Audio/video data compression is no longer the sole motivator for coding
- MPEG-4 enables new functionalities
- MPEG-4: 'Coding of Audio-Visual Objects'

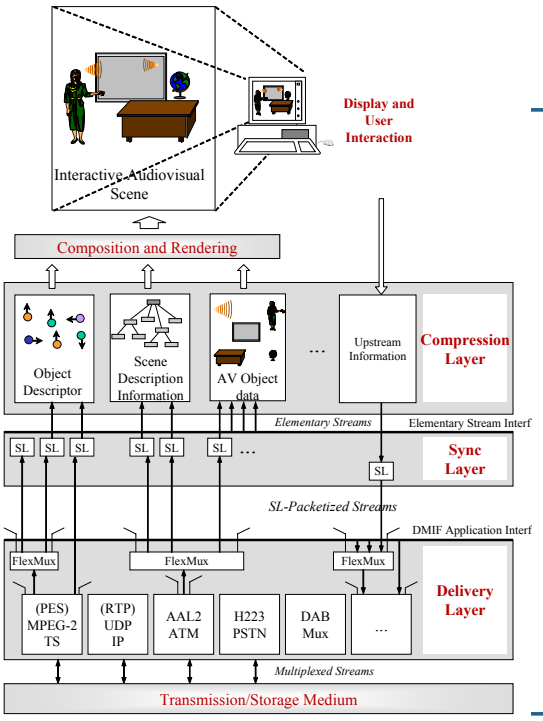
MPEG-4: Coding of Audio-Visual Objects

- Audio-Visual Scenes are composed of 'Objects'
- 'Compositor' in the decoder places objects into scene
- Each media object gets assigned its optimum encoder
- Transparent use of remotely accessed and locally stored objects
- The principle is independent of bit rate
- Combination of natural and synthetic media objects
  - Video, Images, Textures
  - Computer Graphics and Animations
  - Natural and synthetic speech
  - Synthetic audio, generic (natural) audio
  - Text

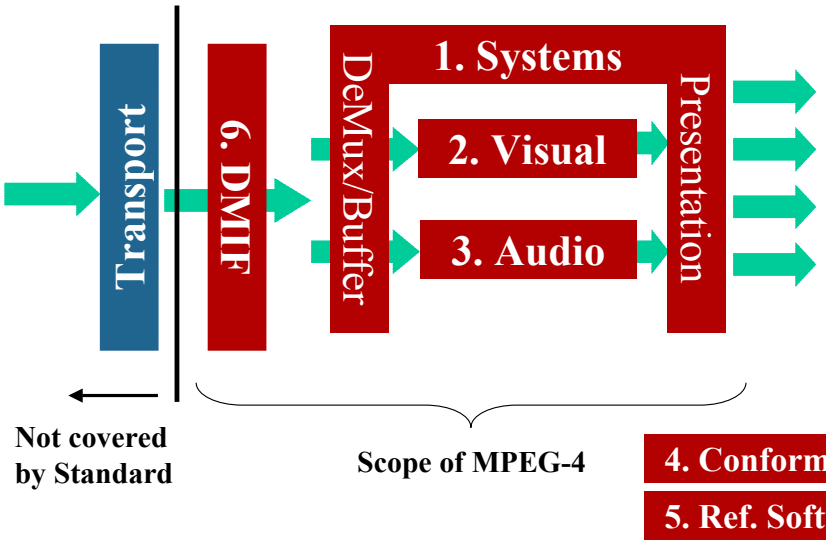
MPEG-4 Architecture



MPEG-4 Architecture



Anatomy of the MPEG-4 Standard



More than an elevator pitch

- Myths and Reality about MPEG-4 (FUD)
- MPEG-4 Industry Forum

Good News

- MPEG-4 has great potential
- The MPEG-4 specification is available now !
  - You can buy it from ISO
- There is free and openly available reference software
  - open source
- MPEG-4 has got the industry's attention
- Product development is proceeding
- Industry support for MPEG-4 is growing

Bad News

- Some of the media do not understand MPEG-4 very well
- The misinformation campaign has begun
  - "Standards are an obstacle to progress"
  - Apparently, MPEG-4 has some vendors scared.

Myths and Reality

Myth #1

- Standards impede innovation

Reality

- Standards Build Markets
  - TV Broadcast
  - Telecommunications
  - Consumer electronics
  - the Internet and the World Wide Web

Myths and Reality

Myth # 2

- Standards are obsolete by the time they are done

Reality

- Standards Survive the Test of Time
  - NTSC – 1953
  - VHS – 1976
  - CD – 1980
  - TCP/IP – 1981
  - JPEG – 1990
  - MPEG-1, MP3 - 1992
  - MPEG-2 – 1994

Myths and Reality

Myth # 3

- Standards don't deliver high quality
  - Corollary: Standards are frozen for all time

Reality

- Standards Achieve Quality Without Churn
  - Specification captures best core technologies
  - Standard provides many degrees of freedom
  - Well-defined process balances interests of entire community
  - Cast of thousands working in the open to improve quality

Myths and Reality

Myth # 3

- Standards don't deliver high quality
  - Corollary: Standards are frozen for all time

Reality (cont'd)

- MPEG-4 is much more than Simple Visual Profile or DivX
  - Quality comparisons are invariably based on Simple Visual Profile
  - This is MPEG-4's lowest complexity conformance point
  - There are more Profiles, that give significantly better quality

Myths and Reality

Myth # 4

- Standards bear onerous patent royalties, so costs will be high

Reality

- Standards Minimize Lifecycle Costs
  - ISO requires licensing on reasonable & non-discriminatory terms and conditions from contributors of technology
  - Standards support “encode once” philosophy
  - Standards support “best practices” workflow

Myths and Reality

Myth # 5

- MPEG-4 is still not ready

Reality

- MPEG-4 has been ready since 1998
  - MPEG-4 Version 1 was ready in 1998
  - MPEG-4 Version 2 was ready in 1999
  - Several additions were completed after 1999
  - None of these additions invalidated any earlier version
  - The Standard is here and ready to be used

Myths and Reality

Myth # 6

- MPEG-4 is unstable

Reality

- MPEG-4 is being enhanced to suit market needs
  - Further functionalities are added
  - There is ongoing work in extensions on
    - Textual Format XMT (interoperability with SMIL and X3D)
    - Advanced 3D graphics and animation (AFX)
    - Intellectual Property Management and Protection (DRM)
    - Multi-user worlds
    - Advanced Video Coding

Standards Fuel Innovation

- Portable media devices
  - Music players, cameras, camcorders
- Wired technologies link portable devices to PC
  - USB, Firewire
- Optical disc storage bridges PC to CE
  - DVD-R, CD-RW
- Wireless networking
  - Handhelds, mobile phones

Standards Help Markets Mature

- Support compatibility over time
  - How long do you use a TV set ?
- Support interoperability between vendors
  - A different TV set for every channel you watch ?
- Minimize over-reliance on a single vendor
  - No one company can service all the needs
  - No one implementation is best for every use

And MPEG-4 is Not Alone

- Internet Streaming Media Alliance (ISMA)
- 3GPP and 3GPP-2 (Wireless devices and handphones)
- Wireless Multimedia Forum
- JPEG 2000
- DRM (Digital Radio Mondiale) Consortium
- Digital Video Broadcast (DVB) (?)
- Web3D
- W3C
- IETF
- And probably several others…

Yet There Are Challenges

- Standards aren't a quick fix
  - Patience and persistence required
- Interoperability between vendors MUST be achieved
- Patent licenses must be simple to obtain
  - Royalties must reflect market realities

So, what's the bottom line ?

MPEG-4 Industry Forum Charter:

- To further the adoption of the MPEG-4 Standard, by establishing MPEG-4 as an accepted and widely used standard among application developers, service providers, content creators and end users.
- The purpose of M4IF shall be pursued by: promoting MPEG-4, making available information on MPEG-4, making available MPEG-4 tools or giving information on where to obtain these, creating a single point for information about MPEG-4, creating industrial focus around the usage of MPEG-4.
- Go check: www.m4if.org

Streams of Media Objects

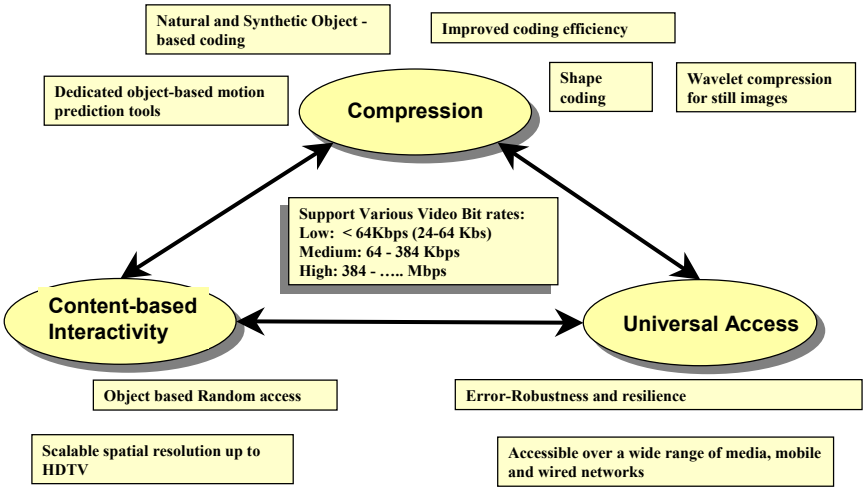
JC Spierer
MITRE Corp.

Course Overview

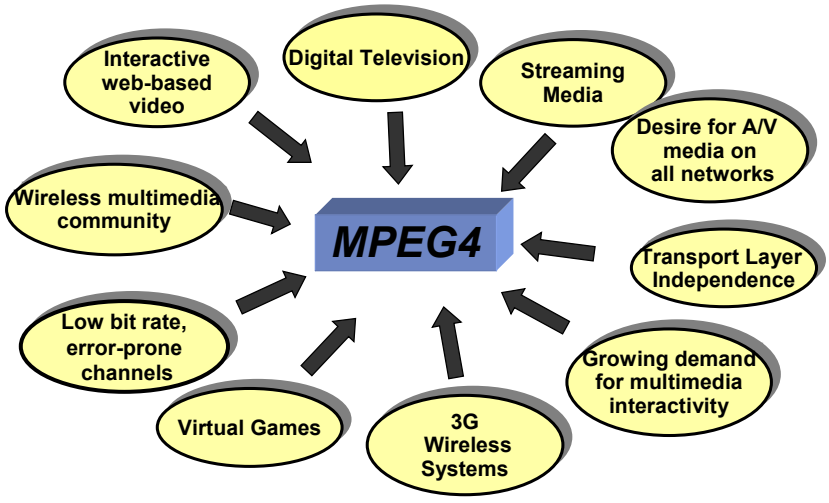
- Architectural issues of MPEG-4
- Delivery of media objects
- Synchronization of objects and streams
- Object descriptors

- Case Study

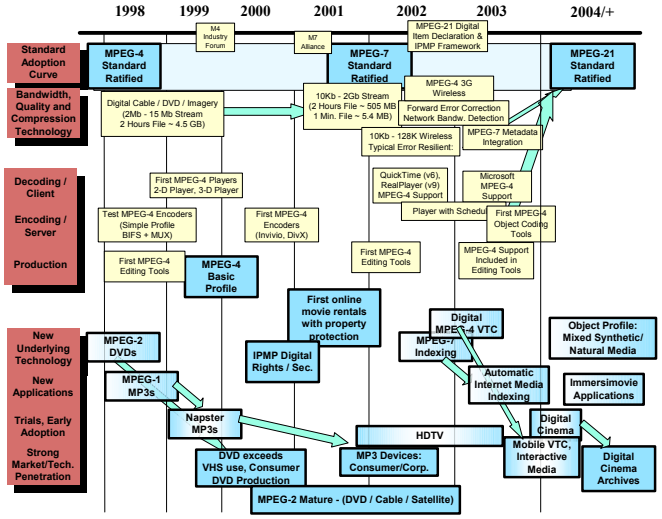
MPEG-4 Functionality and Characteristics



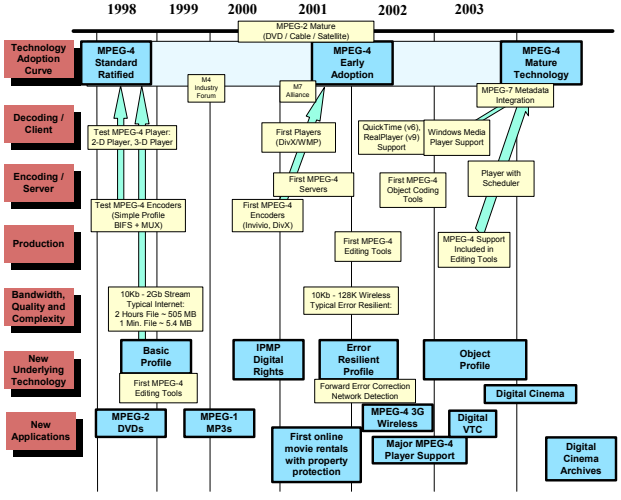
Motivation for MPEG-4 Requirements



MPEG Technology Roadmap



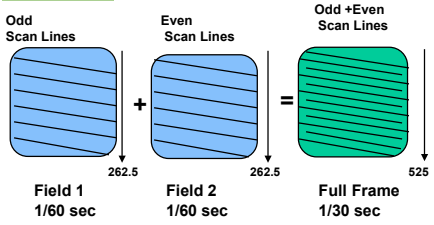
MPEG-4 Technology Roadmap



Digital Video Compression and Display Basics

Analog Television

NTSC Video



Why the Need for Compression??

- 480 visible scan lines per frame
- 480 pels per scan line in the red, green and blue components
- 8 bits per color

= **168Mbits/s**

CCIR-601 is the ISO/IEC standard that defines the Digital Image Format.
-It is Central to MPEG and VTC Compression

	Video Resoulution (pels x lines x frames/s)	Uncompressed Bitrate (RGB)	Compressed Bitrate
NTSC video	(480 x 480 x29.97Hz)	168 Mbits/s	4 to 8 Mbits/s
HDTV video	(1920 x 1080 x 30Hz)	1493 Mbits/s	18 to 30 Mbits/s
HDTV video	(1280 x 720 x 60 Hz)	1327 Mbits/s	18 to 30 Mbits/s
videophone (CIF)	(352 x 288 x 29.97Hz	73 Mbits/s	64 to 1920 kbits/s
videophone (QCIF)	(176 x 144 x 29.97Hz)	18 Mbits/s	30 kbits/s
Two-channel stereo audio		1.4 Mbits/s	128 to 384 kbits/s
Five-channel stereo audio		3.5 Mbits/s	384 to 968 kbits/s

Raw and Compressed Bitrates

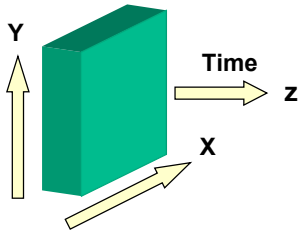
QCIF 176 x 144
Quarter CIF

CIF 352 x 288
Common
Intermediate Format

7.5,10,15or30 Fps

VTC Display resolution

Digital Video Compression Basics Redundancy Removal



Remove Redundancy on Three Axis

Remove Spatial Redundancy

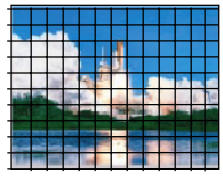
Transform Techniques

Quantization

Sub-sampling

Entropy Coding

Intra Frame Coding



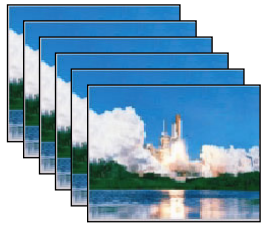
Remove Temporal Redundancy

Frame Reduction

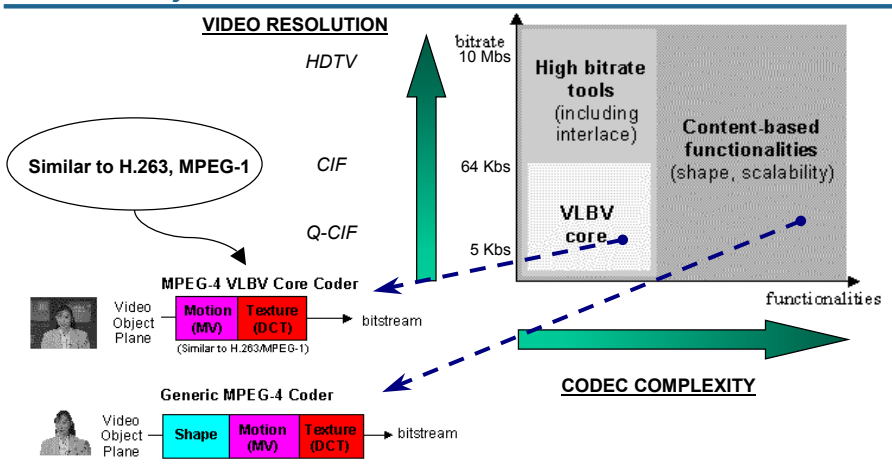
Motion Compensation

Motion Estimation
Inter-frame Predictive Coding

Inter Frame Coding



Tools for Representing Natural Video/Objects

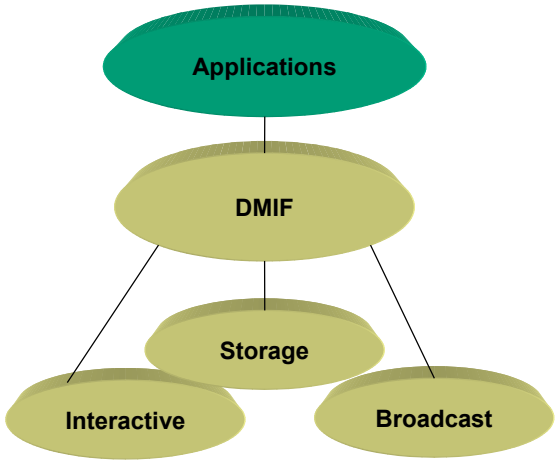


Basic coding structure involves shape coding and motion compensation as well as DCT-based texture coding

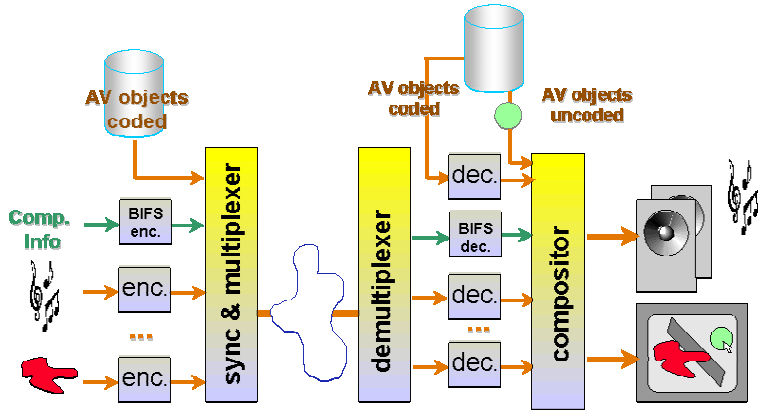
MPEG-4 Delivery Multimedia Integration Framework (DMIF)

Session Layer protocol

DMIF allows content providers to develop applications irrespective of the delivery technology
- applications can access content from storage devices or remote servers or broadcast channels
- Different delivery technologies would be hidden



MPEG-4 High Level System



Multiplexed streams are transmitted containing compressed AVO's and the associated scene description, these streams are demultiplexed, and the objects decompressed, according to the scene description information

MPEG-4 Systems Layer Model

Coding of AVO's, texture and scene information

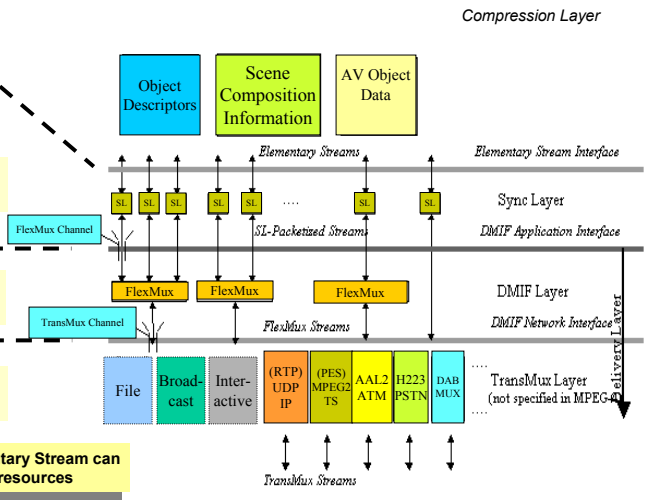
Synchronization

ID of Access Unit type

Recovery of media object or scene description time base

Flex Mux is core multiplexer in MPEG-4

Only interface to this layer specified



The required QoS for each Elementary Stream can be translated into actual network resources

Evolution of Standards-Based Compression

Blocked-Based Hybrid DCT/Motion Compensation Video Coding

MPEG-1

- MP-3
- Interactive CD ROM
- Storage and Playback

H.261 (ITU-T)

- Circuit Switched VTC
- ISDN Networks

MPEG-2

- Broadcast Entertainment Video
- DVD
- DBS
- HDTV

H.263 (ITU-T)

- IP-Based Video Conferencing
- IP/ATM Networks

Object based coding

Error Resilience Coding

MPEG-4

- Mobile Video Phones
- 2.5/3G Cellular
- Wireless Streamed Video
- Video Email
- Interactive TV/Multimedia, Games
- Wireless Security Surveillance
- Video Conferencing

One Way / Broadcast

Real Time VTC

MPEG-3

MPEG-3 Initial intended to support HDTV but was dropped during development when MPEG-2 video proved adequate when scaled up to meet HDTV requirement. Some of MPEG-3 was merged into MPEG-2

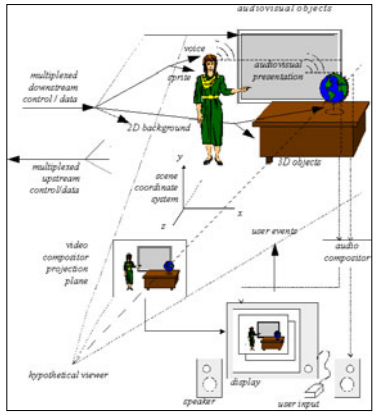
MPEG-4 Audio Visual Scene Description

MPEG-4 views scene as a composition of Audio Visual Objects (AVO's)

Individual objects coded separately and composed to form scene

Primitive AVO's grouped together to form compound AVO's (e.g. talking person + voice = compound AVO)

Scene can contain both natural and synthetic content

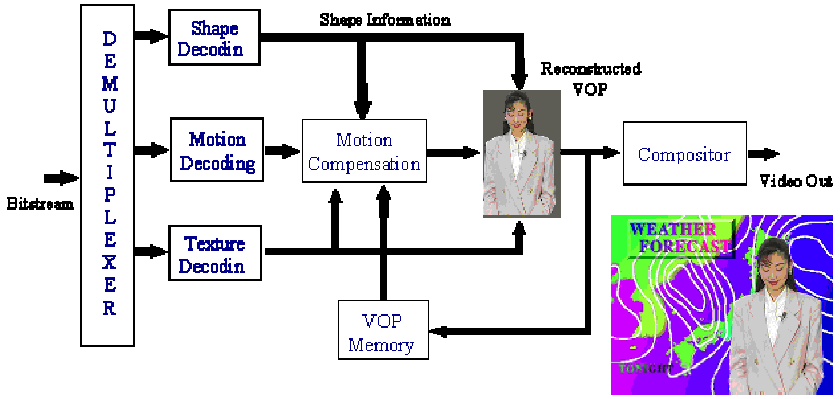


MPEG-4 Audio/Visual scene

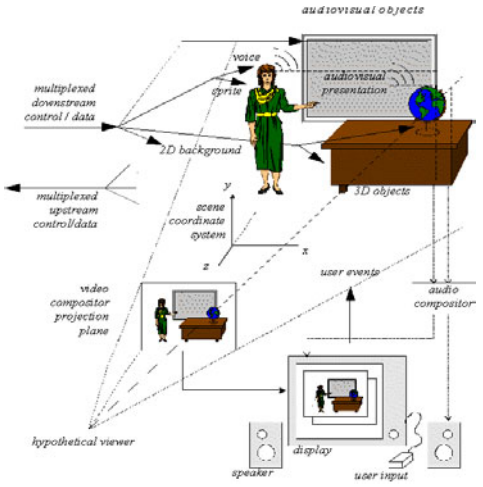
MPEG-4 provides standardized ways to describe a scene

Virtual Reality Modeling Language

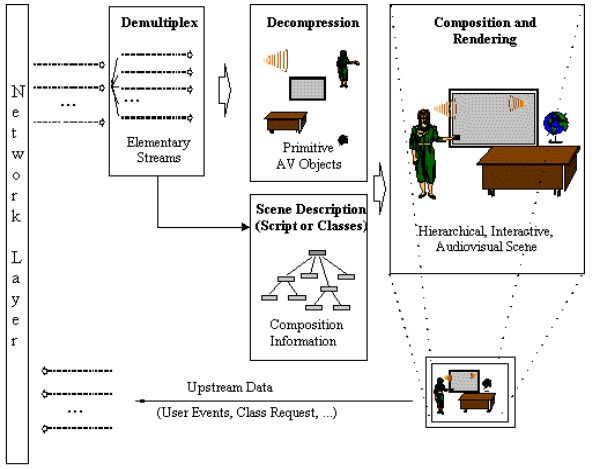
Decoding of Video Object Planes



MPEG-4 Scene (Source: Main Site Tutorial)

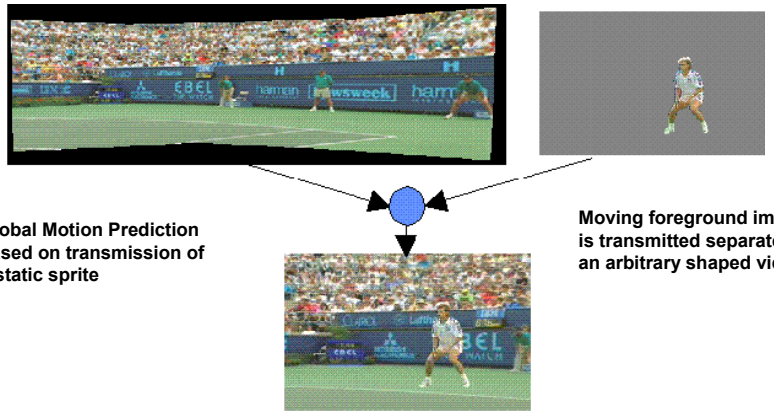


MPEG-4 Object Interactivity (Source: Main Site Tutorial)



Sprite Coding a Video Sequence

Sprite Panoramic Image

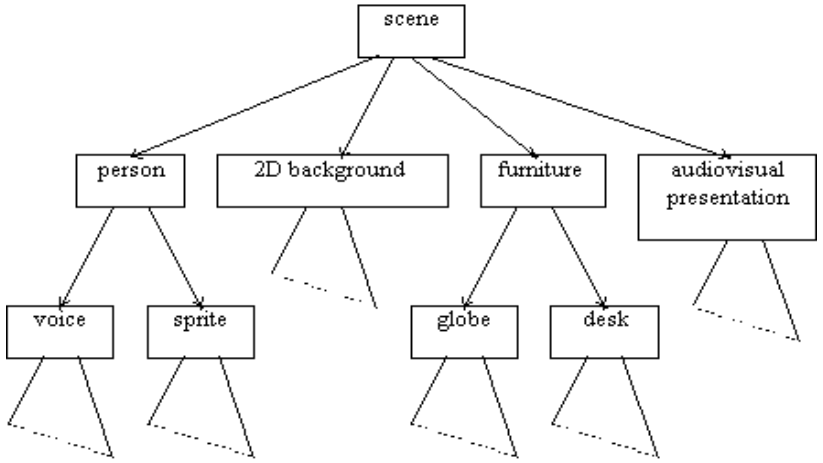


Global Motion Prediction based on transmission of a static sprite

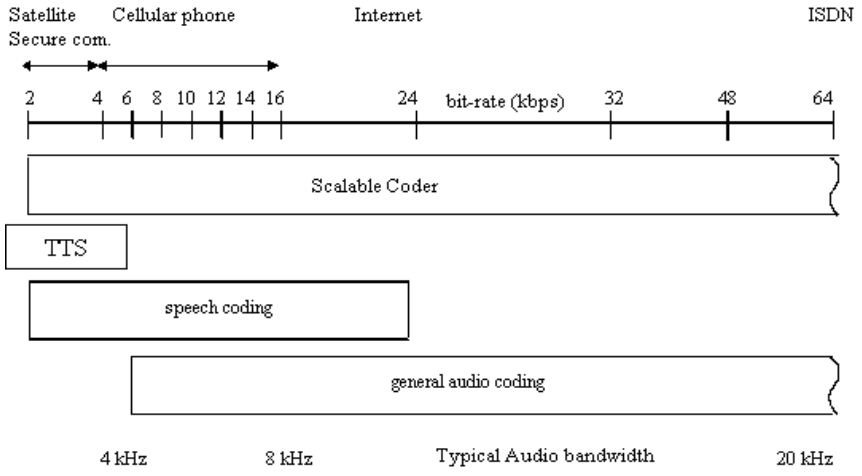
Moving foreground image is transmitted separately as an arbitrary shaped video object

For consecutive images in a sequence, only 8 global motion parameters are coded and need to be sent

MPEG-4 Scene Hierarchical Structure (Source: Main Site Tutorial)



MPEG-4 Audio Compression



MPEG-4 Audio Profiles

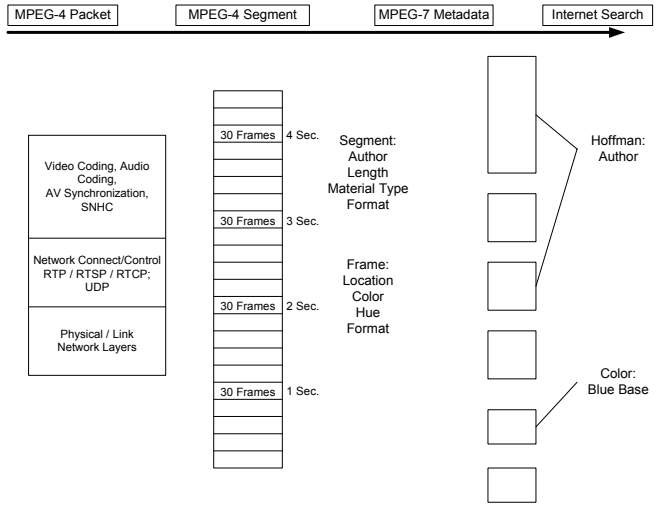
Combination Profile	Hierarchy	Audio Object Profiles Supported
Main	Contains Scalable, speech and low rate synthetic	AAC Main, LC, SSR T/F, T/F Main Scalable, T/F LC Scalable Twin VQ core CELP HVXC HILN Main Synthetic TTSI
Scalable	Contains Speech	T/F LC Scalable AAC-LC or/and TF CELP HVXC TwinVQ core HILN Wavetable Synthesis TTSI
Speech		CELP HVXC TTSI
Low Rate Synthesis		Wavetable Synthesis TTSI

Conclusion and Summary

- Coding efficiency of MPEG-4 has proven to outperform existing compression algorithms in use today
- Error resiliency has proved to be very effective in field trials
- Leading codec of choice for 3G wireless multimedia services
- Many vendors will claim to be MPEG-4 compliant, but may only implement the simple profile
- Streaming, asymmetric web-access to handheld mobiles have recently been introduced in foreign markets
- Video on the battlefield will be subjected to error-prone channels and error robustness techniques will be a critical enabling technology
- Key barriers to widespread adoption are licensing terms and education on benefits of various profiles
- MPEG4 thinking is just in its infancy and will influence audio and video compression for at least the next 20 years.
- Battery and power technologies must be improved

Integrating MPEG-4 with MPEG-7 and MPEG-21 Provides Significant Object Improvements

MPEG 4 & 7: Relationship



MPEG-7 Standard Scope

standardization

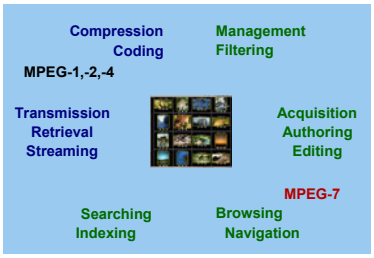
Feature Extraction → MPEG-7 Description → Search Engine

Industry competition

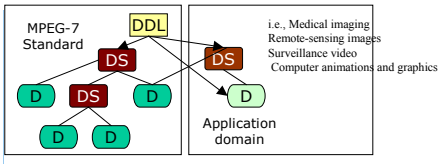
Feature Extraction:	MPEG-7 Scope:	Search Engine:
Content analysis (D, DS)	Description Schemes (DSs)	Searching & filtering
Feature extraction (D, DS)	Descriptors (Ds)	Classification
Annotation tools (DS)	Language (DDL)	Complex querying
Authoring (DS)	Coding Schemes (CS)	Indexing
	MPEG-7 Concepts	Personalization

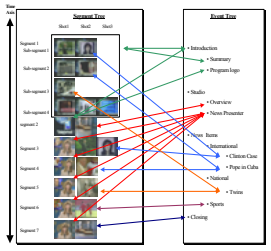
MPEG-7 Overview

(XML for Multimedia Content Description)

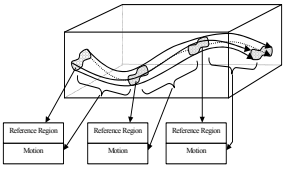


- **MPEG-7 Normative elements:**
  - Descriptors and Description Schemes
  - DDL for defining Description Schemes
  - Extensible for application domains



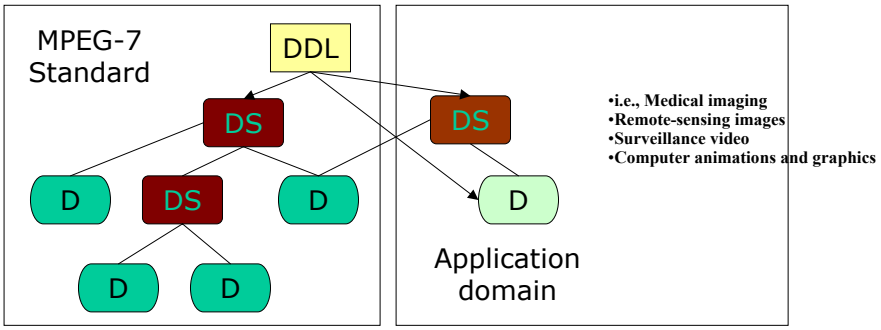


- **Example MPEG-7 Descriptions:**
  - Video segments (shots, key-frames, text and semantics)
  - Moving regions (spatio-temporal locators)
  - Audio-visual features, object tracking, scene description

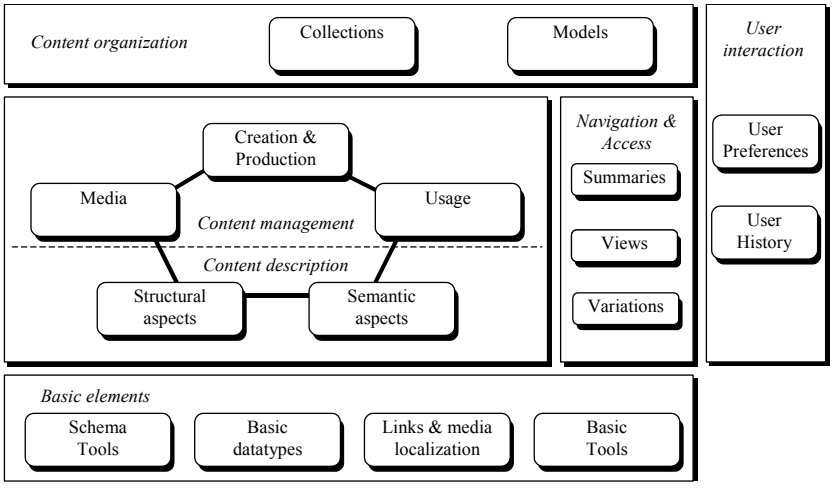


MPEG-7 Normative Elements

- **Descriptors** and **Description Schemes**
- **DDL** for defining Description Schemes
- **Coding Schemes** for coding Descriptions
- **Extensible** for application domains



MPEG-7 Multimedia Description Schemes (MDS Overview)



MPEG-21 Elements

- Element 1: Digital Item Declaration
- Element 2: Digital Item Identification and Description Framework
- Element 3: Content Handling and Usage
- Element 4: Intellectual Property Mgt./Protection
- Element 5: Terminals and Networks
- Element 6: Content Representation
- Element 7: Event Reporting

Thanks For Materials, Ideas and Support To...

- MPEG-4:
- Vinnie Mosera
- MITRE - NJ
- email: vmosera@mitre.org

- MPEG-7:
- John Smith
- IBM Corp.
- email: jsmith@us.ibm.com

- Demonstrations:
- Wo Chang
- National Institute for Standards in Technology (NIST)
- email: wchang@nist.gov

…For More Information

- JC Spierer
- The MITRE Corporation
- Systems Engineering (R105)
- Project Leader, Media and Collaboration
- Chair, Media Cluster Group
- (781) 271 - 7726
- jcspiere@mitre.org

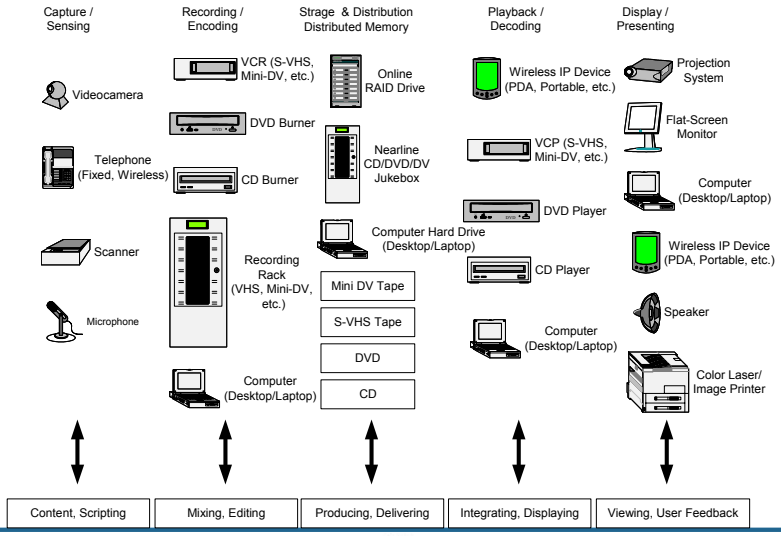
Delivery of Object Streams

JC Spierer
MITRE Corp.

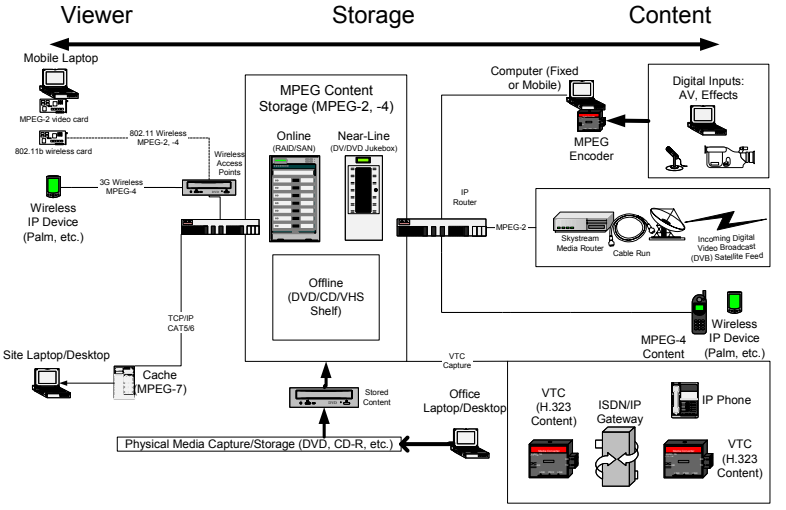
Course Overview

- Architecture of a networked media delivery system
- Interactive media
- Delivery of media over heterogeneous networks
- MP4 file format
- Case Study

Media Content Delivery Products



Next-Generation Networked Media Delivery



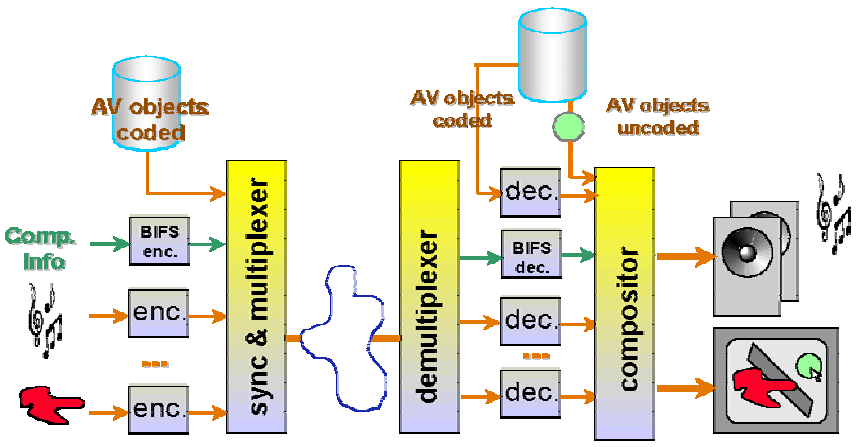
Comparison of Multimedia Compression Standards

	MPEG-1	MPEG-2	MPEG-4	H.261	H.263
Dates of Standardization	11.92 11.94	05.96 Version 1 01.98 Version 2	1.99 Version 1 1.00 Version 2	12.90 Version 1 05.94 Version 2	1.98
Primary Applications	Digital Storage Media, CD-ROM	Broadcast, DVD, HDTV	Web Authoring, Multimedia, Compression, Wireless Mediaphone	Wireline (ISDN) Video Conferencing	Desktop/ Wireless Media Conferencing
Typical Video Birates	1.5 Mbps	4.6 Mbps	20 Kbps - 6Mbps	20-384 Kbps	128-384 Kbps
Typical Video Frame Size	352 x 240 (SIF) 720 x 480 (Rec. 601)	176 x 144 (QCIF) 352 x 288 (CIF) 1280 x 720(HDTV)	176 x 144 (QCIF) 352 x 288 (CIF) 720 x 480 (601) 1280 x 720(HDTV)	176 x 144 (QCIF) 352 x 288 (CIF)	176 x 144 (QCIF) 352 x 288 (CIF)
Typical Associated Audio Quality	Stereo CD Quality	Surround Sound	Speech, Music, Stereo CD,	Surround Sound	Speech
Error Resiliance & Scalability	Basic	Basic	Advanced	Adequate	Basic

Serial Video vs IP Video

Error Performance

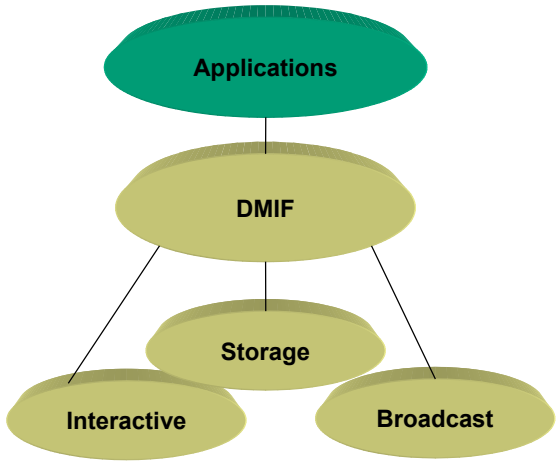
MPEG-4 Architecture (Source: Main Site Tutorial)



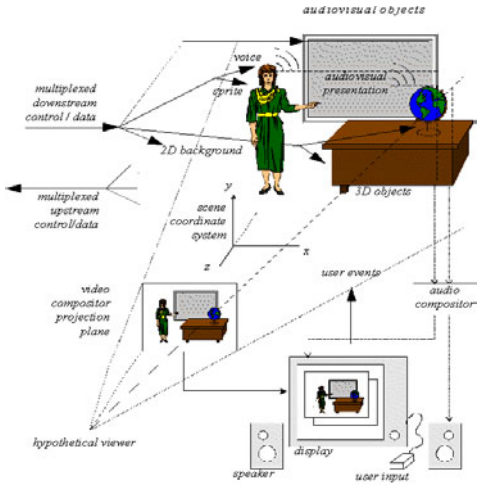
MPEG-4 Delivery Multimedia Integration Framework

Session Layer protocol

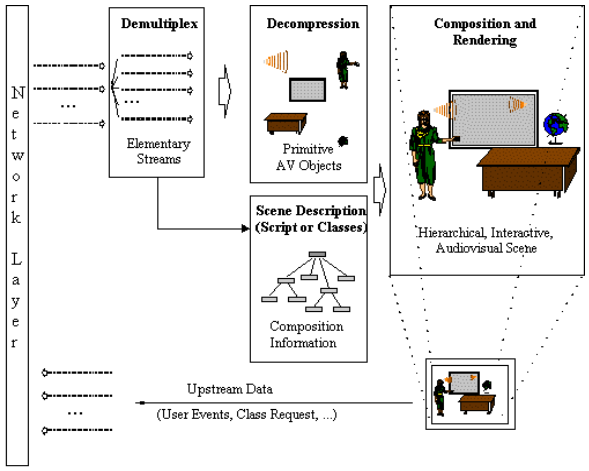
DMIF allows content providers to develop applications irrespective of the delivery technology
- **applications can access content from storage devices or remote servers or broadcast channels**
- **Different delivery technologies would be hidden**



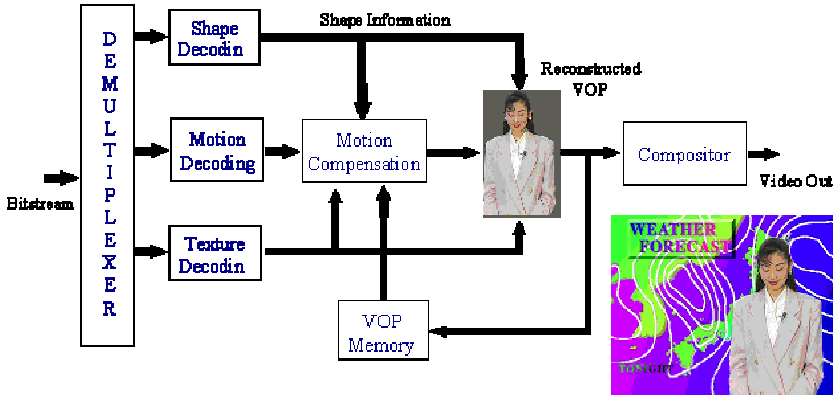
MPEG-4 Scene (Source: Main Site Tutorial)



MPEG-4 Object Interactivity (Source: Main Site Tutorial)



Decoding of Video Object Planes



MPEG-4 Systems Layer Model

Coding of AVO's, texture and scene information

Synchronization

ID of Access Unit type

Recovery of media object or scene description time base

Flex Mux is core multiplexer in MPEG-4

Only interface to this layer specified

The required QoS for each Elementary Stream can be translated into actual network resources

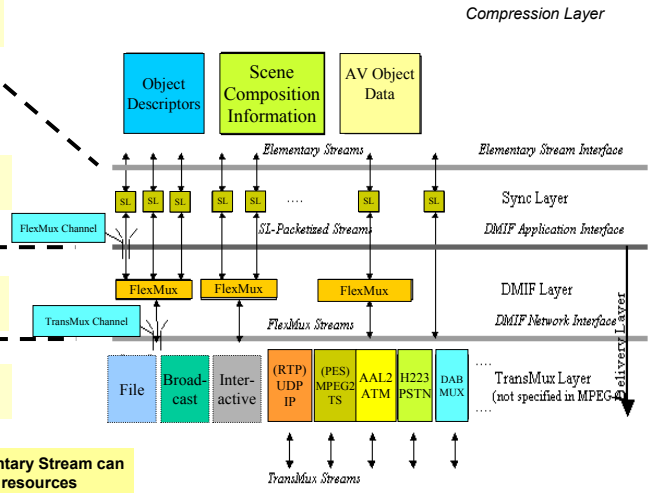
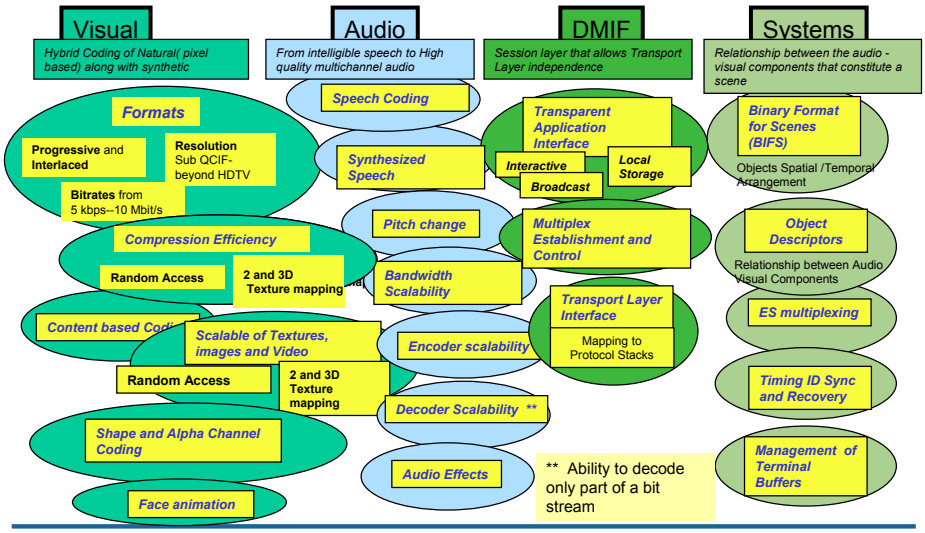
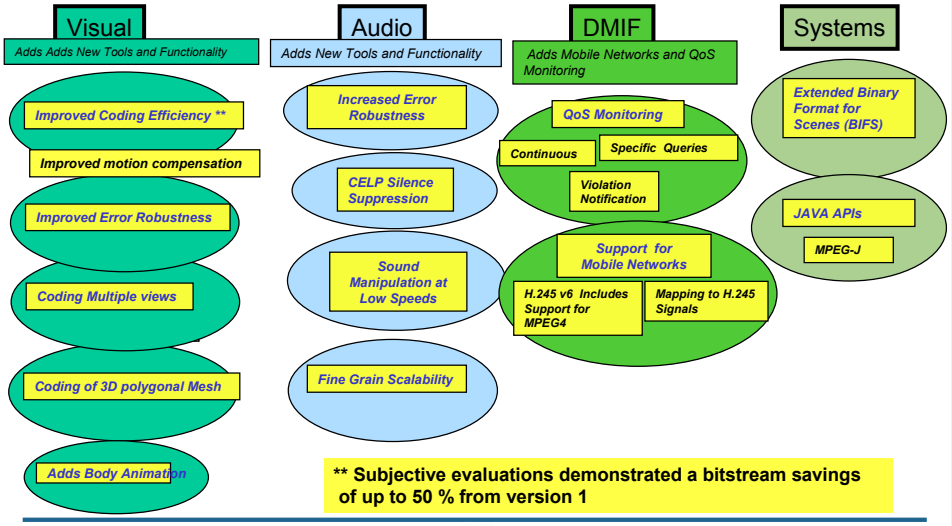


Figure - an MPEG-4 System Layer Model

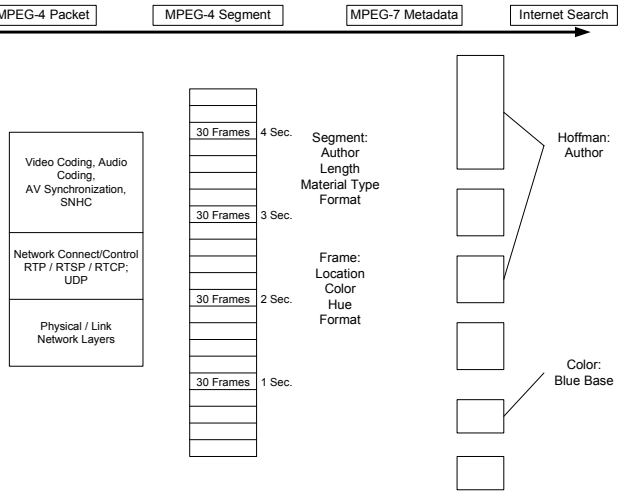
Major Functionalities in MPEG-4 v1



Major Functionalities Extended in v2



MPEG 4 & 7: Relationship



UMA for Mobile Applications

- **Mobile Access of Multimedia**:
  - Growth of mobile phones, PDAs, hand-held devices
  - WAP (24,000 WAP-accessible sites) – WAP MM group
  - NTT DoCoMo I-Mode (cHTML, color displays, video downloading, audio capabilities)
  - IMT-2000 3G Wireless - 144 kbit/s to 2 Mbit/s indoors

- **Multimedia for Mobile Commerce**:
  - **Shopping**: mobile access of e-Commerce catalogs (rich multimedia content), searching, filtering, browsing
  - **Push**: multimedia rich promotions, advertisements (spatial and temporal awareness), impulse buys for mobile users
  - **Media**: e-Commerce of multimedia content (MP3, news, sports, video summaries)

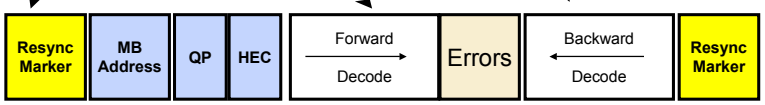
MPEG-4 Error Resilience Tools

- **Resynchronization**
  - Provides periodic resynchronization markers throughout the bit stream
  - Markers determines the start of a new video packet

Three approaches to providing Robustness In Error Prone Environments

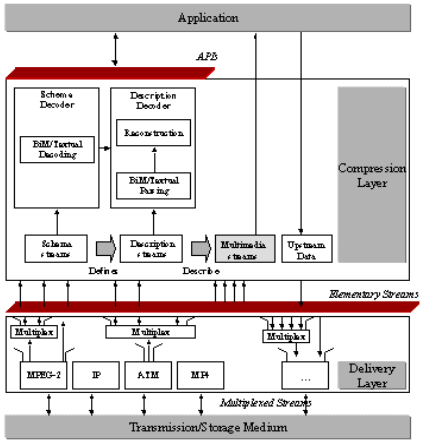
- **Error Concealment**
  - **Copy blocks from previous frames**
  - **Add second resynchronization marker**

- **Data Recovery**
  - **Reverse Variable Length Codes (RVLC)**

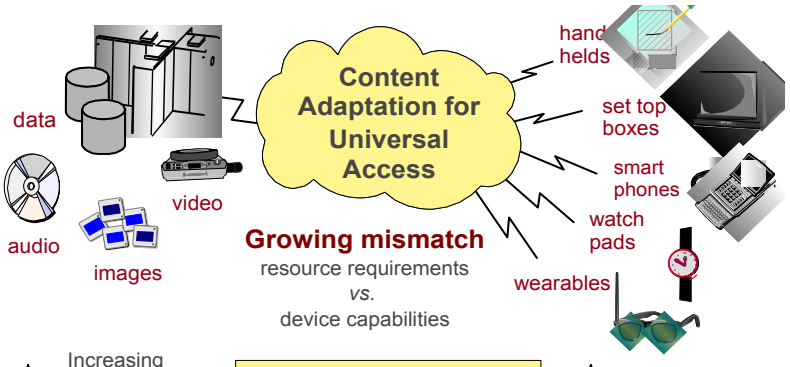


Video Packet

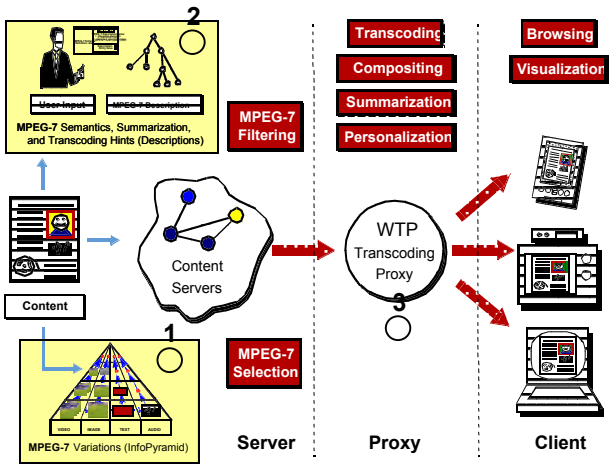
MPEG-7 Architecture



MPEG-7 Universal Multimedia Access (UMA)



MPEG-7 Universal Multimedia Access



- On-line processing at server:
  - Variation selection (optimization)
- Off-line processing at server (MPEG-7):
  - Automatic processing (Variations)
  - Human annotation (Transcoding Hints)
  - Semi-automatic feature extraction (Semantics and Summarization)
- On-line processing at proxy (MPEG-7):
  - Video transcoding (Transcoding Hints)
  - Video summarization and personalization

There are Several Existing Applications

Example: Image Transcoding Hint Annotation Framework

- **Constraint optimization:**
  - Constraint on image size from device
  - Benefit gained from content value of transcoded image
- **Problem statement:**
  - Maximize total content value given constraints

- **MPEG-7 region annotation (R_i):**
  - **Importance Hint (I_i):** indicates relative importance of region ($0 < I_i < 1$)
  - **Spatial Resolution Hint (S_i):** indicates minimum resolution of region for preserving details ($0 < S_i < 1$)
  - **Content Value (V_i):** indicates value of the transcoded region $V_i = f(I_i, S_i)$

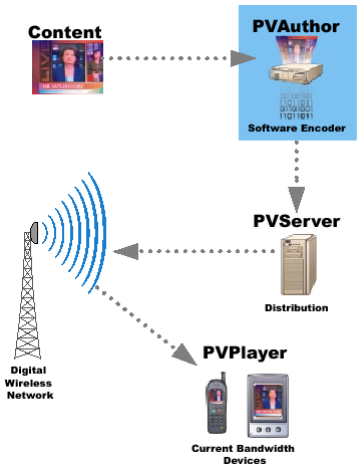
Example: Image Transcoding (with MPEG-7 Hints)

			
1280 x 960	640 x 480	320 x 240	160 x 120
PC (SVGA)	TV Browser	Handheld	PDA/Phone

- **MPEG-7 Transcoding hint adaptation:**
  - Combination of cropping and scale reduction to adapt to device screen size
  - Preserves details of important regions (cost-based optimization)
  - Results in higher content value for given data size

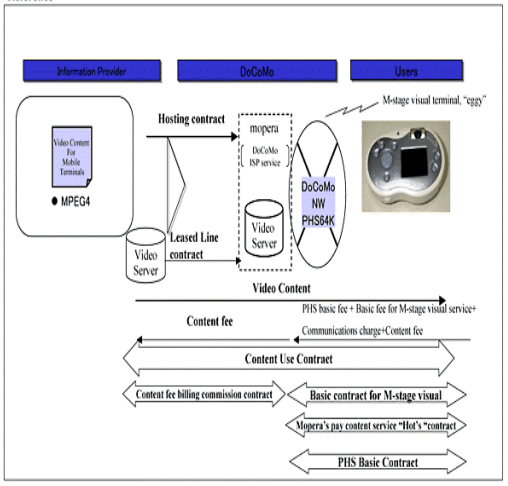
Packet Video MPEG-4 Streaming Media Solution

Input formats:AVI,MPEG1, WAV
•Output formats:MP4, GSM-AMR
•Stream live content or archive to disk
•Resize video and optimize encoded content for target display devices
•Variable bit-rate encoding
•Network bandwidth 9.6 kbps to 384 kbps
•Frame rates less than 1fps to 30fps
•Frame sizes 128 x96 to 352 x288
Content PVAuthor
Software Encoder
PVServer
Distribution



NTT DoCoMo world's first wireless video content distribution service using MPEG4* technology.

Reference



*The service will enable users to download or stream video content such as movies and music as well as news,sports and other content, via NTT DoCoMo's "mopera" mobile Internet service.

The menu includes some 47 channels of providers offering more than 130 sub-menus of content (as of December 8, 2000).

Initially, the service will be accessible only with "eggy" terminals, which are connected to PHS phones.

The service will start on December 8, 2000 and will be available throughout Japan wherever PHS 64/32K service is offered.

1: Users may experience trouble when attempting to view web sites that are incompatible with this terminal browser.

DoCoMos First Color-Display I-Mode Cell Phone

DoCoMo's First Color-Display i-mode Cell Phone

NTT DoCoMo introduces the F502i and D502i, its first i-mode compatible cellular phones to be equipped with vivid color display.
Features:
* Color liquid-crystal display capable of displaying up to 256 colors.
* Downloading popular cartoon characters from the Internet for use as moving figures on the display, and downloading hit tunes for use as ringing patterns (with up to three-note chords).
* Downloading and displaying video content during standby, talk and mail send/receive.
* Hypertalk (advanced voice processing for improved speech quality)

Mobile Multimedia Systems (MoMuSys) Project

Finland

•MoMuSys project has an objective of contributing to the MPEG4 standardization process by carrying out field trials on mobile multimedia terminals that use the MPEG-4 standard
•Two scenarios used for field trials:
- •Travelers information system
- •Surveillance

Results from early tests (1998):
• MPEG4 provided significant improvements upon systems that are currently being used.
The current obstacles for MPEG-4 are :
- • Mainly related to complexity of implementation.
- • The very tedious means to produce contents for MPEG-4 applications.

Despite all technical obstacles for MPEG-4 we mentioned above, this standard will succeed because all competing technologies face the same obstacles. Which leaves us with the question no one can answer yet:

"When will the real breakthrough of MPEG-4 take place?"$

MoMuSys Surveillance Application Using Tomas/ISDN Satellite Facilities

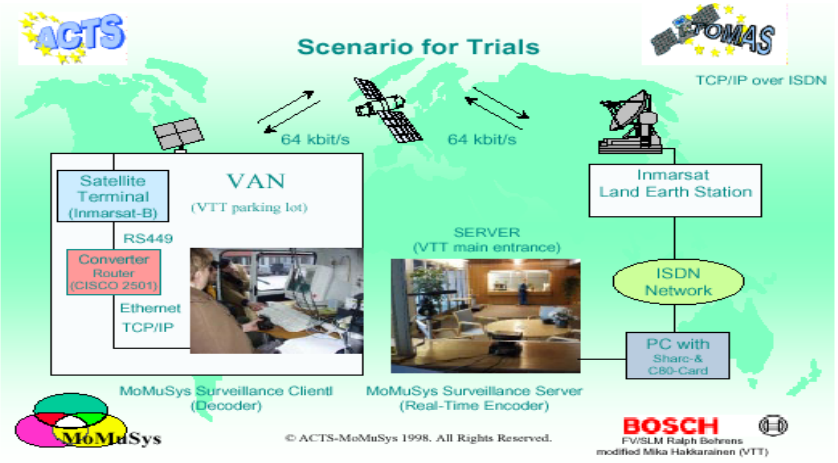


Figure A2.1 Surveillance trials setup using TOMAS ISDN/Satellite facilities

Mitsubishi's, January 9, 2001 Trium and Microsoft today introduce Mondo, the latest GSM/GPRS Phone-enabled Pocket PC

Mondo will be Trium's flagship product.

Mondo will work on both GSM and GPRS (General Packet Radio Service) networks.

The GPRS connection will allow permanent Internet connectivity and access to a wide range of new applications and services with higher bandwidth capabilities. These have been developed in partnership with a number of leading application providers which will provide several advanced services so users can have full Web browsing, secure e-commerce facilities and be able to view video clips.

(**Packet Video Corporation will provide the MPEG4 applications)**

MPEG-7 Delivery Applications

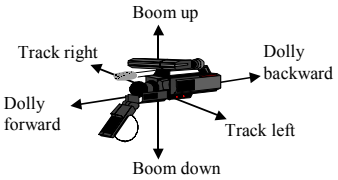
- MPEG-7 applications cover a broad range, including media industry, technical, commercial, and consumer uses.
  - Media Industry:
    - Film, video and radio archives (e.g. sports clips)
    - Journalism (e.g. searching for speeches by voice or face, summaries)
  - Technical, Scientific and Security:
    - Geographical information systems (GIS)
    - Remote sensing (cartography, ecology, natural resources management, etc.)
    - Surveillance (traffic control, surface transportation)
    - Investigation services (human characteristics recognition, forensics)
    - Bio-medical applications
  - Commercial:
    - Education (e.g., distance learning)
    - E-commerce and shopping (e.g. searching for clothes/patterns)
    - Architecture, real estate, and interior design
  - Consumer:
    - Cultural services (history museums, art galleries, etc.)
    - Entertainment (e.g. searching a game, karaoke)
    - Social (e.g. dating services)

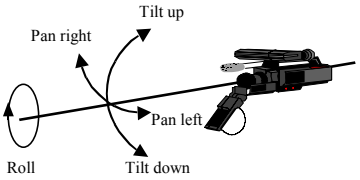
Industry Focus on MPEG-7 Delivery

- Industry Participation:
  - Sony, Philips, IBM, Sharp, Avid, Mitsubishi, Toshiba, Canon, Ricoh, Siemens, NTT DoCoMo, Singing Fish, etc.
- Pull Applications (Searching):
  - Internet search engines and multimedia databases
  - **Advantages**: queries based on standardized descriptions, interoperability
- Push Applications (Filtering):
  - Broadcast video and interactive television
  - **Advantages**: intelligent software agents filter content / channels based on standardized descriptions
- Navigation and Browsing (Summarization):
  - Summarization and personalization of content according to user preferences
  - **Advantages**: digital television set-top boxes operate on standardized meta-data
- Access (Mobile):
  - Universal Multimedia Access (Perceptual QoS), transcoding, content adaptation
  - **Advantages**: content providers reach wider audience

MPEG-7 Visual: Motion Descriptors

- **Camera motion description:**
  - Camera parameters

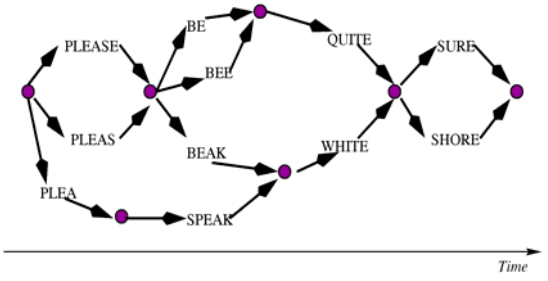




- Motion trajectory
- Parametric motion
- Motion activity
  - Intensity, Direction, Spatial and Temporal

MPEG-7 Audio: Spoken Content

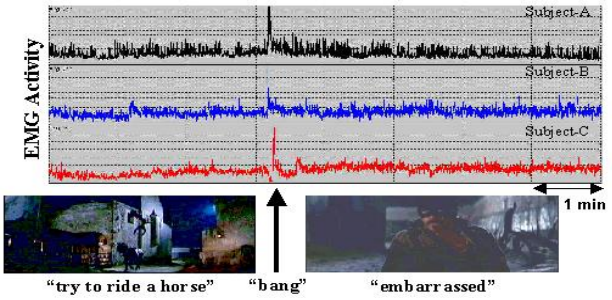
- Describes word and phone (sub-word units) lattices for speakers (audio)
- i.e., hypothetical decoding of the phrase “Please be quite sure":



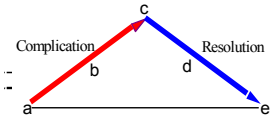
MPEG-7 MDS: Affective DS

- **Describes**: audience's affective (analytical, emotional or mood) response to multimedia content
- **Example**: pattern of emotional tension

- **Automatic Extraction**: via physiological measurements and analysis (real-time)
- **Manual Extraction**: Audience survey (high temporal-resolution)

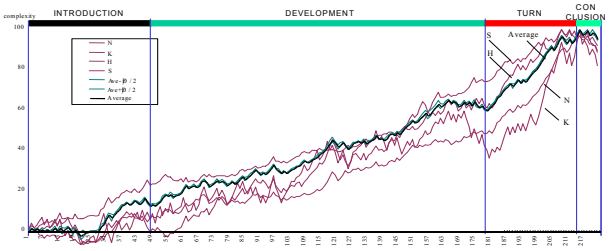


MPEG-7 MDS: Affective DS Example (Story Description)

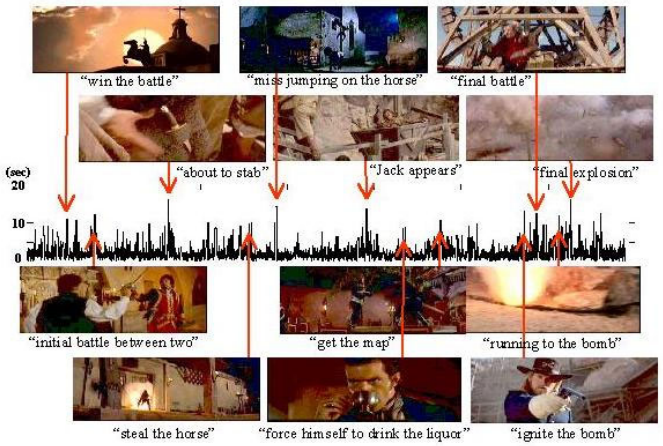


a. Introduction
b. Rise
c. Climax
d. Return or fall
e. Catastrophe

- **Freytag's Triangle**
- Characterizes each dramatic incident
- Cumulative complication describes net intensity



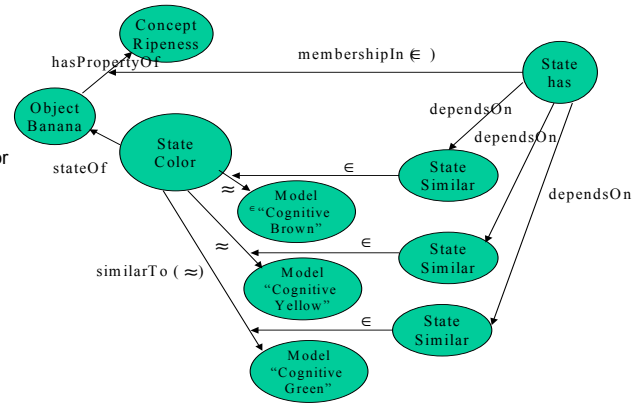
Example: Affective DS (Excitement)



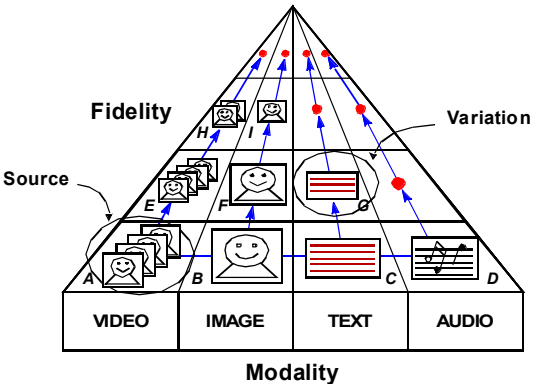
- **Electromyogram (EMG):**
  - Measures voltage difference in forehead muscles
  - Detects smiles and periods of non-blinking
- ← ***The Mask of Zorro***
- **Applications**:
  - Automatic creation of movie trailers

MPEG-7 MDS: Example Semantics Description (Ripe Banana)

- **Object:** banana
- **Concept:** ripeness
- **States:** color (green, yellow, brown)
- **Analytic models:** color histograms
- **Memberships:** based on color similarity



Example: MPEG-7 MDS Variation Description




```
<VariationSet>
  <Source><Video>
   <MediaInformation><MediaInstance>
    <MediaLocator><MediaURI> A
      </MediaURI></MediaLocator>
   </MediaInstance></MediaInformation>
   <CreationInformation><Creation>
    <Title><TitleText> Soccer video </TitleText></Title>
   </Creation></CreationInformation>
  </Video></Source>

  <Variation fidelity="0.85" priority="1">
   <VariationProgram>
    <MediaLocator><MediaURI> A
      </MediaURI></MediaLocator>
   </VariationProgram>
   <VariationRelationship> extract </VariationRelationship>
  </Variation>

  <Variation fidelity="0.75" priority="3">
   <VariationProgram>
    <MediaLocator><MediaURI> C
      </MediaURI></MediaLocator>
   </VariationProgram>
   <VariationRelationship> extract </VariationRelationship>
   <VariationRelationship> languageTranslation
      </VariationRelationship>
  </Variation>
</VariationSet>
```

MPEG-4 Video and Image Coding Tools

Iraj Sodagar
PacketVideo

Outline

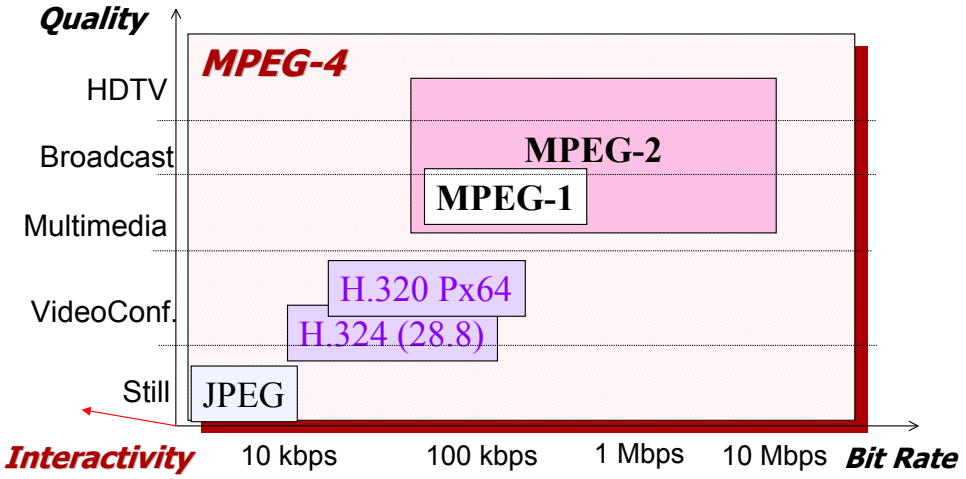
- Overview of visual coding tools
- Video coding tools
- Visual Texture coding
- Advanced Video Coding (Part 10)

Previous Video standards

- What previous standards could do:
  - Frame-based coding techniques for specific applications
  - ITU H.261: Low bit rate video conferencing
  - ITU H.263: Very low bit rates video conferencing
  - MPEG-2: Digital television
- What the previous standard could not do:
  - Representing the video content in a meaningful way
  - Representing multimedia content, not just video frames
  - Addressing a broad spectrum of multimedia applications
  - Enabling wide range of wired and wireless networks for streaming

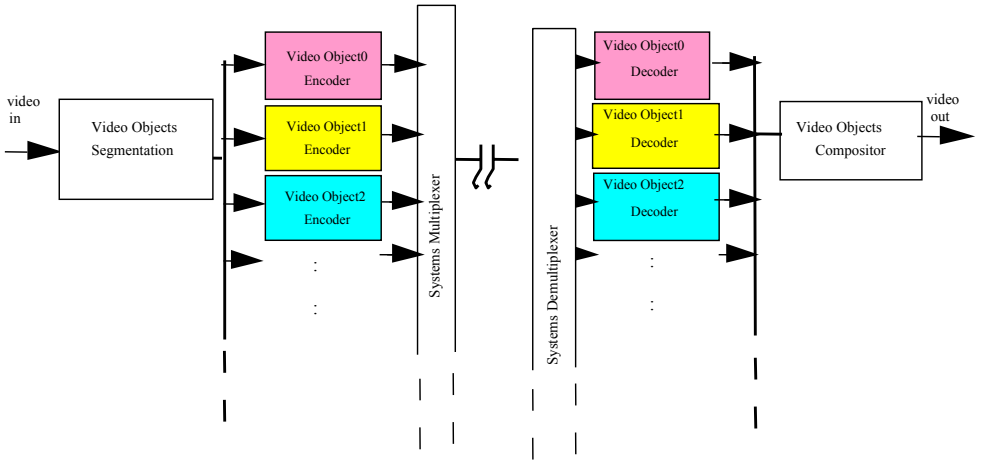
MPEG-4 Visual vs. Other Visual Standards

MPEG-4 allows high compression in all bit rates as well as interactivity



Visual Object-based Coding

Each Visual Object in a Scene is Coded and Transmitted Separately



Coding of Natural Visual Objects

- Natural Textures, Images and Video
  - Video Frames (Progressive and Interlaced)
  - Arbitrarily Shaped Objects
  - Still Image Texture Maps
  - Sprites
  - Error Resilience
  - Scalability

Data Structure For Visual Data

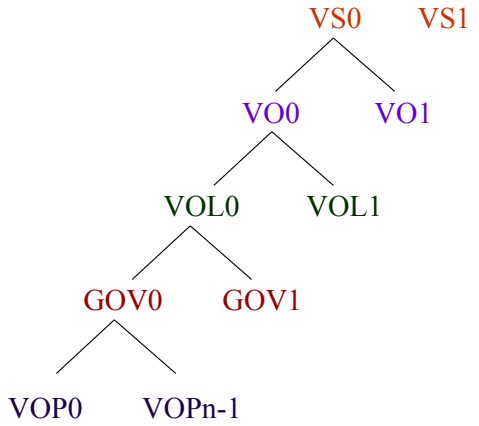
Visual Object Sequence (VS)

Video Object (VO)

Video Object Layer (VOL)

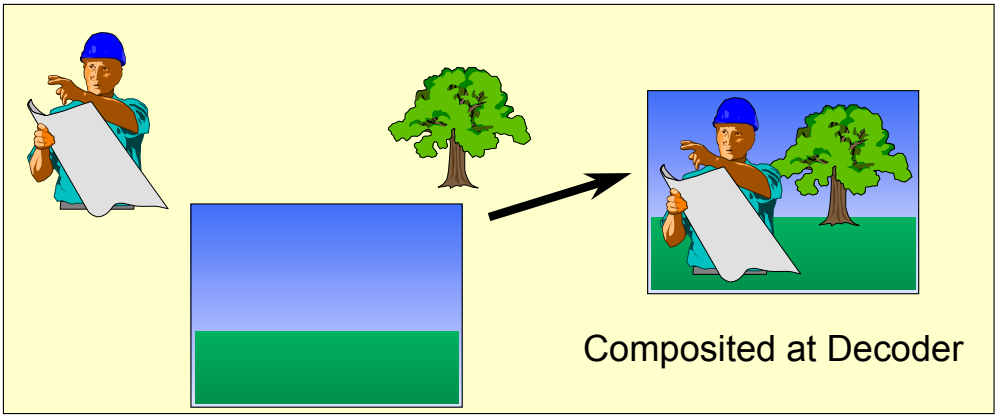
Group of Video Object Plane (GOV)

Video Object Plane (VOP)

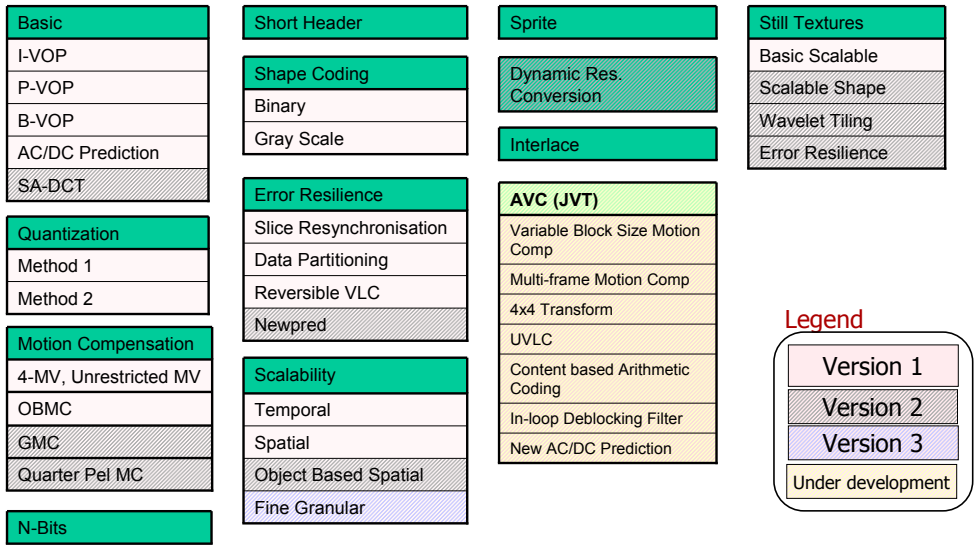


Visual Objects

- Arbitrarily shaped objects
  - Each encoded with DCT syntax + shape
  - Each object plane is YUV + α channel



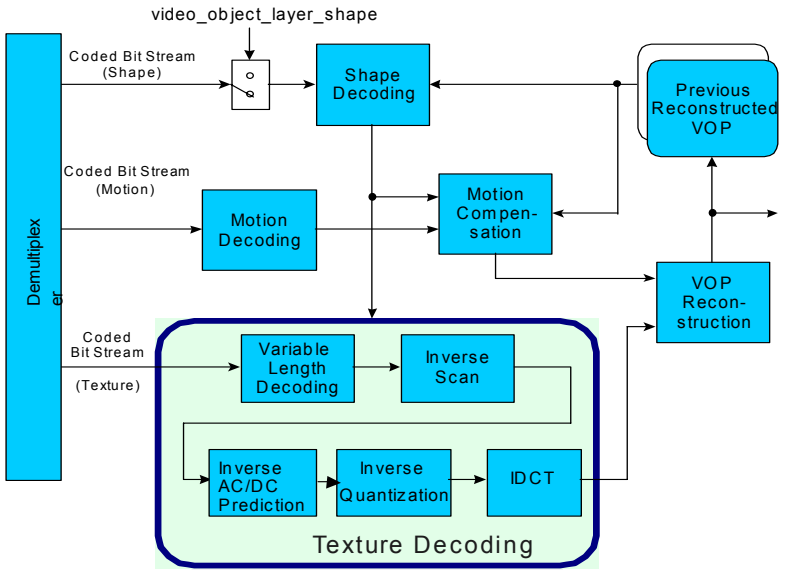
Visual Tools



MPEG-4 Natural Video Coding

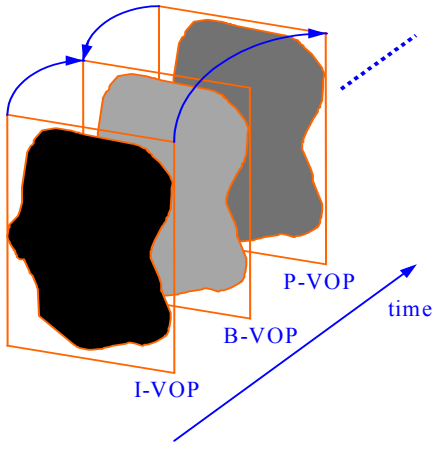
- Coding Efficiency Parameters:
  - 5 kbits/s - 10 Mb/s
  - Resolution: Very Small - DTV
  - Progressive/Interlace
- Error Resilience/Robustness
  - Mobile Environments
- Scalability
  - Spatial and Temporal

MPEG-4 Video Decoder



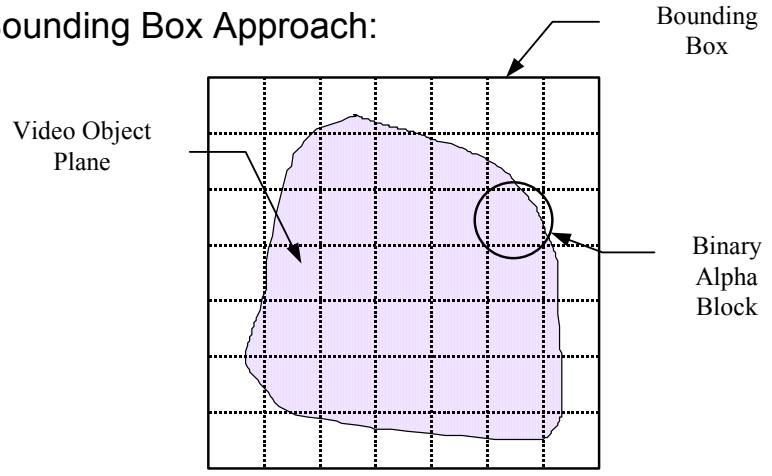
Motion Compensation Tools

Motion compensated coding modes (I, B, P)



Shape Coding

- Bounding Box Approach:

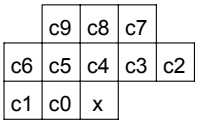


- Texture is Padded Block DCT Coded

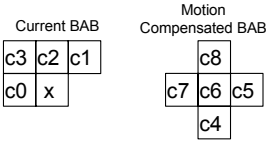
Shape Coding

- Context-Based Arithmetic Encoding
  - Binary Shape Pixels in Binary Alpha Blocks (BAB)

Inter Shape



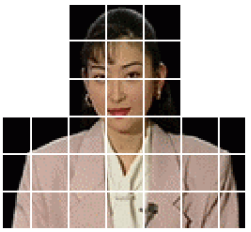
Intra Shape



10-Apr-02

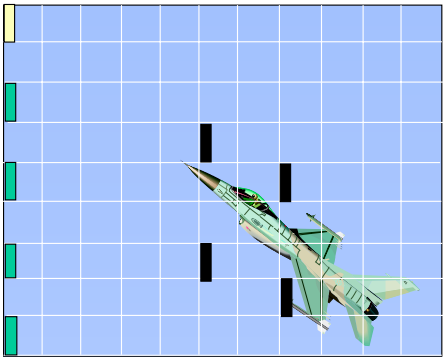
Texture Coding

- Pad Macroblocks - Extract Shape



Error Resilience

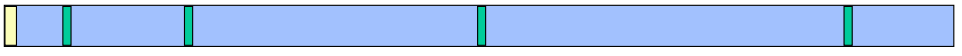
Resynchronization



Picture Start Code

H.263 GOB Header

MPEG-4 Resync Marker

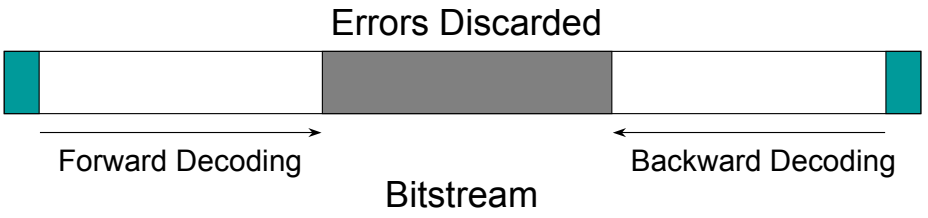


Old H.263 Bitstream

MPEG-4 Bitstream

Error Handling

- Data Partitioning
  - Separate Motion and Texture
- Reversible Variable Length Codes
  - Decode More of a Video Packet



Effects of Error Resilience

- Test Case
  - Errors: 10^{-3} Avg. BER, 10 ms Bursts

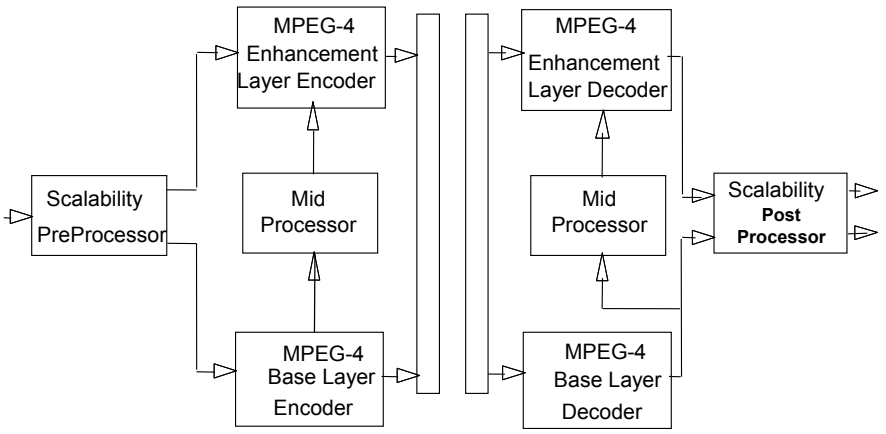
No Sub-Frame Resynchronization

Resynchronization Every 512 Bits

Scalability

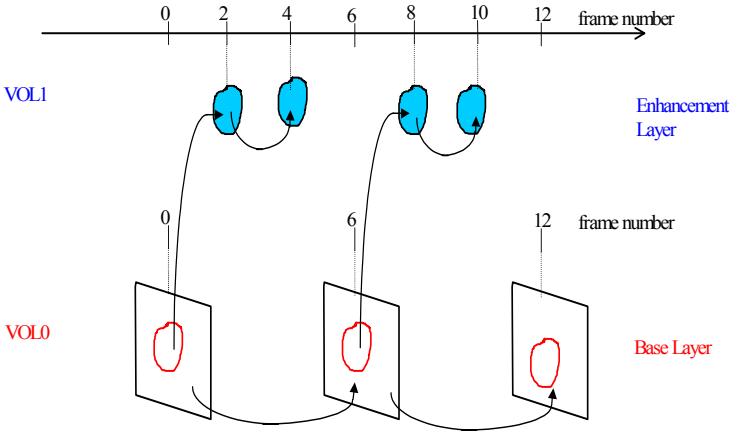
- Object scalability
  - Achieved by the data structure used and the shape coding
- Temporal scalability
  - Achieved by generalized scalability mechanism
- Spatial scalability
  - Achieved by generalized scalable mechanism
- Fine granular scalability
  - Achieved by DCT bit plane coding

Scalable Coding General Scheme



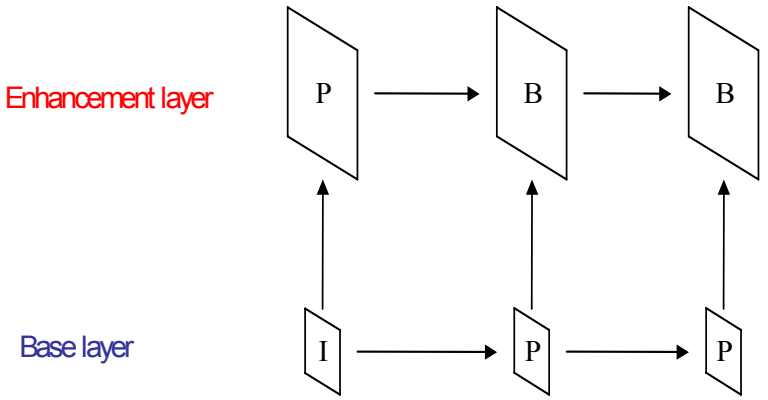
Temporal Scalability

- Enhancing the temporal resolution
- Two type: entire frame, object based



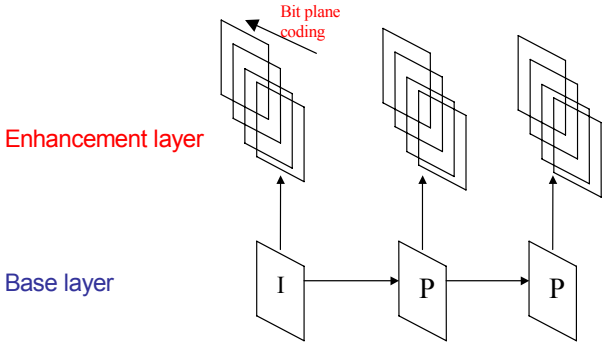
Spatial Scalability

- Base layer: conventional MPEG-4 video
- Enhancement layer: predicted mechanism from the base layer



Fine Granular Scalability

- DCT bit plane coding in the enhancement layer



- The enhancement bit planes can arbitrary dropped on fly.
- Several enhancement tools

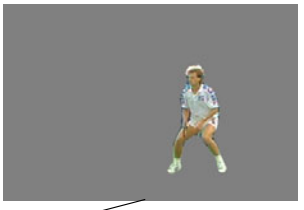
N-Bit Video Coding Tool

- Allows compression of video data with precision of up to 12-bits/pixel
- The syntax, semantics, and coding tools are extended:
  - bit-precision
  - extended DC VLC tables
  - extended quantization mechanism
  - Insertion of marker bits to avoid start code emulations

Interlaced Coding Tool

- Allows all options in progressive to be used for interlaced.
- Motion compensation for field or frames similar to that of MPEG-2
- Modified AC/DC prediction
- Field DCT
- Interlaced I, P, and B VOP coding
- Modified prediction for motion coding
- Modified scan rules
- 10% more efficient in compression efficiency compared to MPEG-2

Sprites

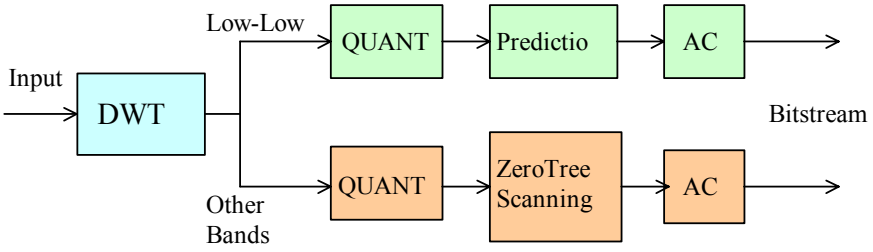


Mosaic Background

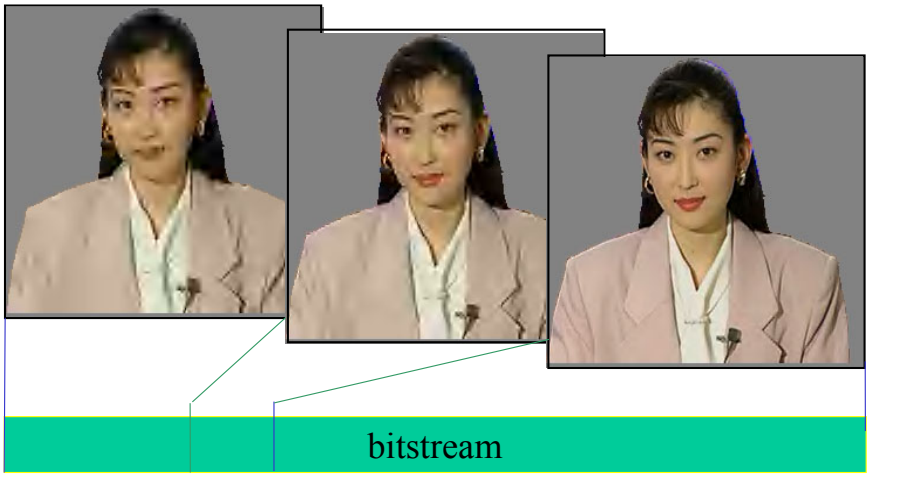
Manipulated by
Motion Parameters
(Pan, Rotation, Zoom,..)

Texture Coding (Still Image)

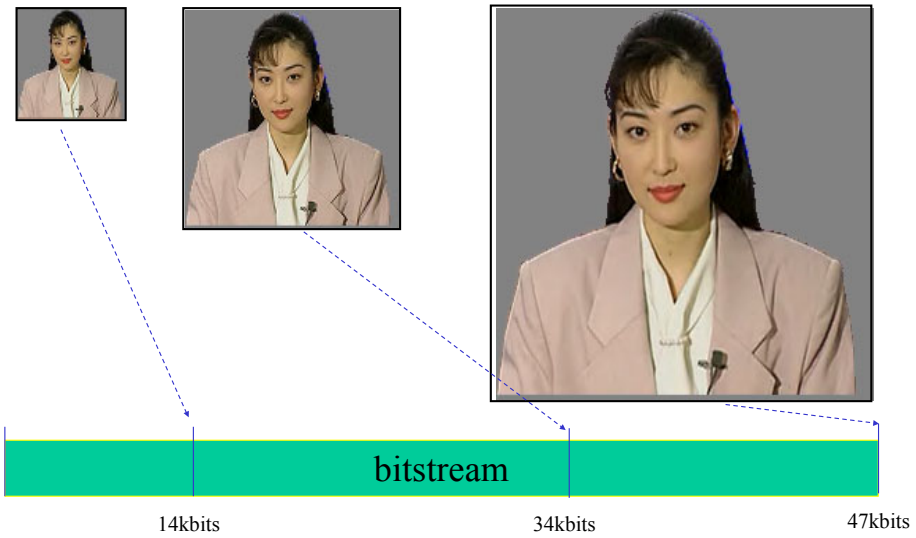
- Discrete Wavelet Transform (DWT)
- Separate DC band Coding
- Zero-Tree Scanning (ZTS) and Multiscale Zero-Tree Entropy (MZTE) coding



Texture Coding: SNR Scalability



Texture Coding: Spatial Scalability



Advanced Texture Coding Tools

- Error Resilience
  - Resynchronization Markets
  - Segment Markers
- Shape Coding Tools
  - Similar to Shape coding
  - Arbitrary Shaped Wavelet transform
- Wavelet Tiling
  - Coding images as tiles, i.e. for random access on spatial domain

Advanced Video Coding

- A new work item, in collaboration with ITU-T
  - Joint Video Team
  - H.26L as MPEG-4 Part 10
- Goal: High coding efficiency
  - Twice compression ratio than MPEG-4 simple/advanced simple profiles
  - Higher complexity is allowed
- Workplan:

Description	WD	CD	FCD	FDIS
Advanced Video Coding	01/12	02/05	02/07	02/12

Advanced Video Coding Tools

- Several New Coding Tools
  - Advanced Motion Compensation:
    - Multiple block size : 16x16, 8,16,8x8,4x8,8x4, 4x4
    - 1/4 and 1/8 sample motion accuracy
    - Multiple frame memory
  - New Predictive Frames: SP frames for predictive switching
  - 4x4 Exact Integer DCT transform
  - New Entropy Coding tools
    - Universal VLC
    - Context adaptive binary arithmetic coder
  - In-loop de-blocking filter
  - Intra block prediction
- New Profiles for part 10

Summary

- MPEG-4 provides an extensive tool set for coding of natural video and images
  - Wide range of bit rates
  - High compression tools such as sprites, wavelet coding, video coding tools
  - Extensive functionalities such as scalability, object based coding, error resilience
- Tools are grouped in profiles for conformance and interoperability for different applications
- MPEG-4 has some work under development which adds extra functionality or additional performance

Conclusion: Why use MPEG-4 Video?

- There are several video and image coding format in market.
- Multiple coding format:
  - Reduced interoperability and reach
  - Most complex content management
  - Different QOS for different users
  - More cost to content creator and distributor
  - More cost to device manufacture
- Result: The need of industry wide standard/dominate format
  - Extensive in features
  - Expandable
  - Supports wide rang of applications, platforms and delivery networks
  - Open format

Conclusion: Why MPEG-4 Video

- MPEG-4 provides the latest standardized technology for multimedia
  - Superior coding compared to other standards
  - Comparable to the best commercial technologies in the market
  - Extendable: not just video, but multimedia including graphics, animation, audio, scene description and interactivity
  - Industry wide support
    - Multi-vendor support
    - Wide range of devices and equipments
    - Lower cost devices and equipments

MPEG-4 Scene Description and Interactivity

Klaus Diepold
DynaPel Systems, Inc.
MPEG-4 Evangelist

Overview

- Scene Description in MPEG-4
- BIFS Components and Composition
  - Composition
  - BIFS - Update
  - BIFS – Animation
  - Systems-Related Features
  - BIFS - Compression
- Sample Compression Results
- Coming up next: XMT

Introduction

- **Bi**nary **F**ormat for **S**cene Description (BIFS)
  - Scene composition in a synchronised environment
- BIFS has been designed
  - With broadcast delivery in mind
  - With streaming media over IP in mind
  - With interactive media in mind
  - With all possible media types in mind
- BIFS inherits its basic structure from VRML 2.0
- However:
  - VRML does not support 2D graphics
  - VRML is not well suited for streaming or broadcast

Introduction

- BIFS inheritage from VRML 2.0:
  - set of nodes to represent the primitive scene objects to be composed
  - the scene graph constructs
  - the behavior
  - the interactivity

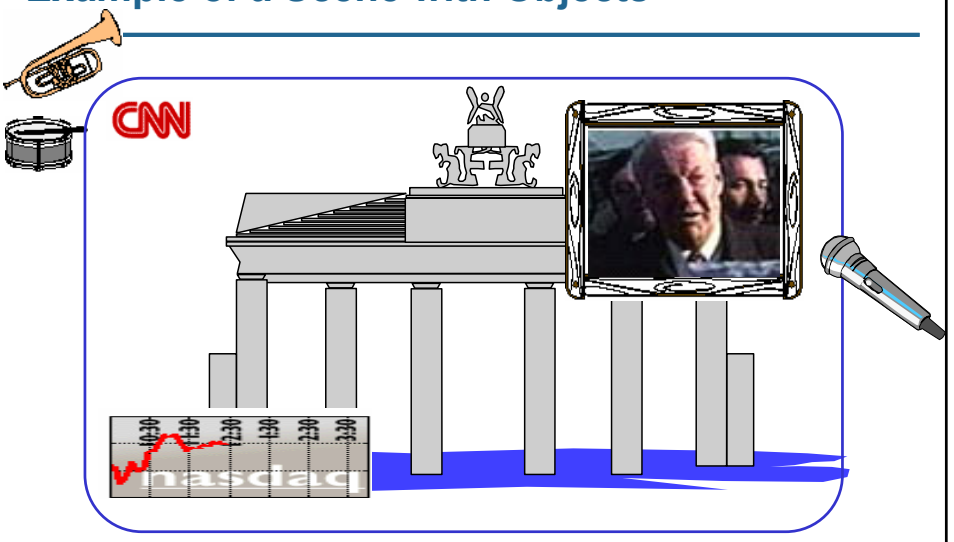
Introduction

- BIFS adds
  - 2D capabilities
  - Integration of 2D and 3D
  - Advanced Audio Features
  - Timing Model
  - Update protocol to modify the scene in time
  - Animation protocol to animate the scene in time
  - Binary Encoding for the scene
  - Face and Body, Mesh, Media Control ...

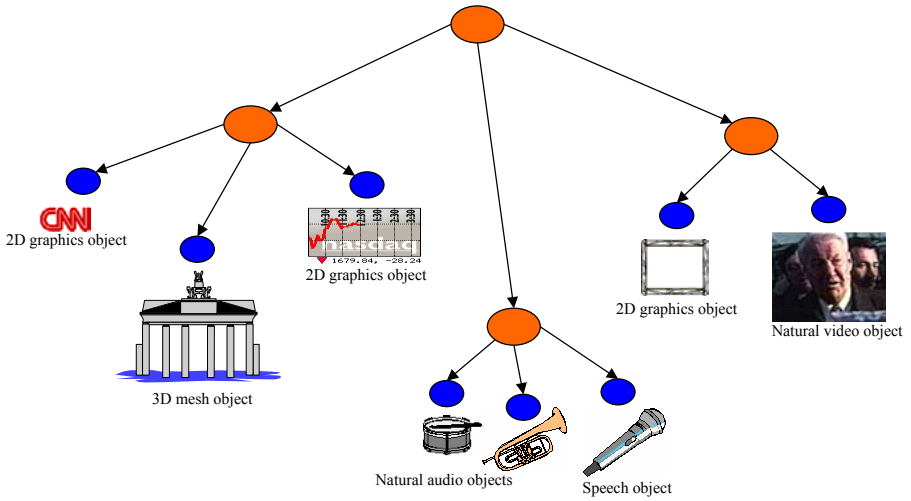
Design Goals for BIFS

- Composition and mixing of all possible media data types
- Local and remote interactivity
- Support for streaming and broadcasting environments
- Highly efficient representation (compression)
  - Scene description takes > 1MB data for complex scenes
  - Streaming: downloading 1MB of data takes more than 5 minutes on a 28.8 kbps modem
  - Broadcast: retransmit of scene description every 0.5 seconds takes transmission bit rate of
    + 1.6 Mbps for scene description file of 100 kB
    + 16 Mbps for 1 MB for scene description of 1MB

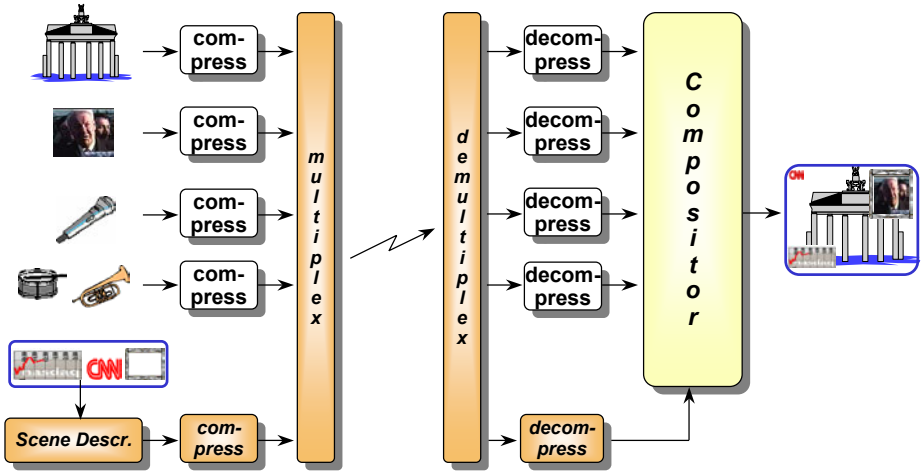
Example of a Scene with Objects



Corresponding scene description



MPEG-4 Architecture



Overview

- Scene Description in MPEG-4
- BIFS Components and Composition
  - Composition
  - BIFS - Update
  - BIFS – Animation
  - Systems-Related Features
  - BIFS - Compression
- Sample Compression Results
- Coming up next: XMT

BIFS Scene Description Features

- Scene Components
  - BIFS uses VRML nodes, fields and events types
  - Description is structured as an acyclic, directed graph
  - Nodes typically represent media objects
  - MPEG-4 defines about 100 nodes in seven categories
- Examples
  - Sphere node: radius field
  - Transform node: translation field, rotation field
  - Node field types: eventIn, evenOut, exposedField
- 11 basic field data types
  - Boolean, integer, float, color, ...

Hierarchy / Structure

- Group
- OrderedGroup: display order
- Transform, Transform2D: position, scale, rotation
- Switch: choice between variants
- Automatical choice depending on distance/framerate
- Grouping nodes
- Children of grouping nodes represent media objects in the scene

Overview

- Scene Description in MPEG-4
- BIFS Components and Composition
  - Composition
  - BIFS - Update
  - BIFS – Animation
  - Systems-Related Features
  - BIFS - Compression
- Sample Compression Results
- Coming up next: XMT

Composition

- 3D Scene Graph
  - Uses VRML scene graph structure
  - Hierarchical transforms using 4x4 matrix transforms
  - Spatialized Sound
  - DEF mechanism to tag objects for future reference
  - USE mechanism to point to DEFed objects
  - BIFS is binary → DEFs are node identifiers (ID integer)

Composition

- Media Objects in 3D
  - Still and Moving Images as textures
  - Basic 3D graphic primitives
  - 3D Meshes
  - texture mapping
  - Face and Body animations
  - Natural and synthetic audio

Composition

- 2D Scene Graph
  - Special case of 3D scene
  - MPEG applications often will use only 2D composition
  - 3D scenes not efficient for representing 2D scenes
  - Requires extra projection on 2D
  - Direct 2D description leads to more efficient representation
  - 2D is based on 2D translation, rotation and scaling
    - \+ → Transform 2D node + depth ordering
  - Hierarchical 2D scene graph with depth
  - Layout manager

Composition

- Media Objects in 2D
  - Still and Moving Images
  - Simple Audio Primitives
  - Basic graphic primitives
  - 2D Mesh
  - Natural and syntheric audio

Mixed Scenes

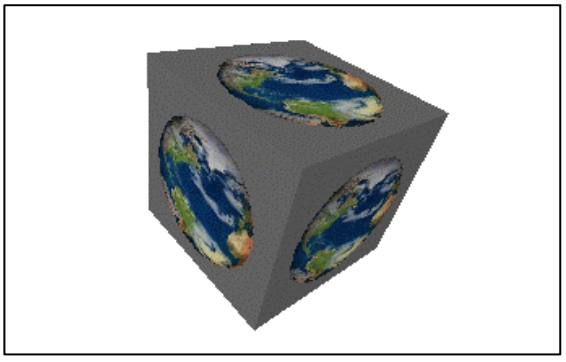
- Scene Description Features:
  - Ability to layer rendered 2D or 3D scenes in a 2D space
  - 2D interfaces for a 3D scene
  - 3D Objects on top of a 2D scene
  - Inclusion of a 2D scene in a 3D scene Graph
  - 2D or 3D scenes as textures maps

Texture Mapping

2D or 3D scene as a texture map on 3D

2D inside a 3D plane

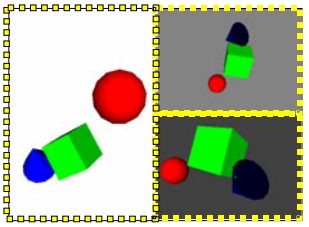
3D Object projected on 2D planes



Mixing of 2D and 3D Objects

3D on top of 2D

2D on top of 3D



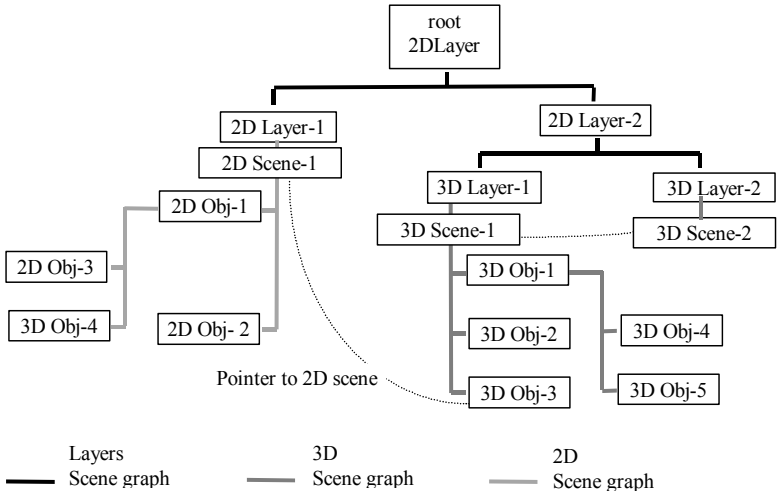
Multi View Point 3D

Mixing of 2D and 3D Objects

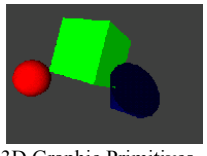
- Scene is 2D or 3D depending on top node
- Layer2D/3D: window on sub-scene
- CompositeTexture2D/3D: sub-scene as texture

- Layer2D in 2D: clipping region

Layering of Scene Descriptions

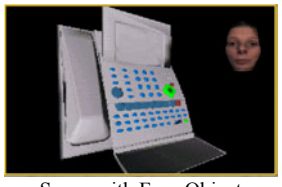


Mixing 2D and 3D Objects



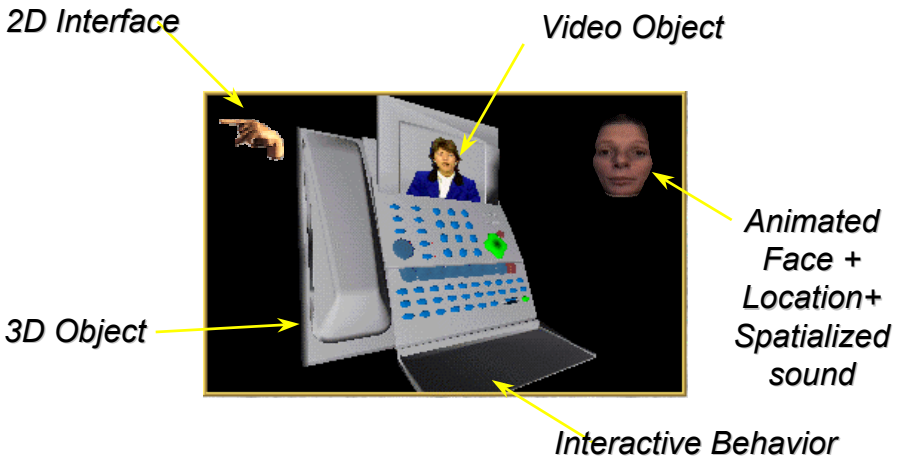
3D Graphic Primitives

Complex 3D Mesh



Scene with Face Object

Mixing 2D and 3D Objects



Sound Composition

- Scene Description Features
  - Audio scene graph
  - Mixing of Sound (Natural / Synthetic)
  - Special Effects
  - Room Effects: physical and perceptual

Audio Nodes

- Sound, Sound2D, AudioSource, AudioBuffer, AudioSwitch, AudioClip, AudioFX, AudioDelay, AudioMix
- Physical:
  - AcousticScene
  - AcousticMaterial
  - DirectiveSound
- Perceptual:
  - PerceptualParameters

Synthetic and Natural Objects

- 2D Graphics
  - 2D Geometry primitives: rectangle, circle, face set, line set, point set, curve, bitmap, text
  - 2D meshes
- 3D Graphics
  - 3D Geometry primitives: box, cone, cylinder, sphere, extrusion, elevation map, face set, line set
  - 3D meshes
- Appearance
  - Material, Material2D
  - Texture

Synthetic and Natural Objects

- Video and texture
  - Bit map mode
  - only translation permitted
  - Texturing mechanism for still and moving pictures
- Sound
  - Synthetic generated audio
  - natural audio
  - speech

Face and Body

- Nodes allowing to include
  - Animated faces into scenes
  - Animated bodies into scenes
- See extra presentation

Timing Model

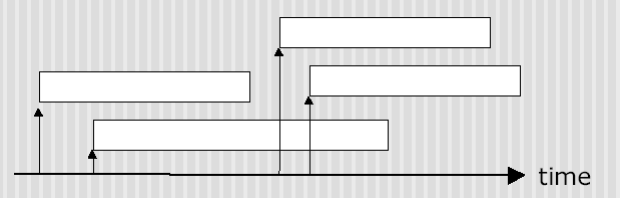
- BIFS: Access Units and notion of time inherited from MPEG-4 Synchronization Layer and Object Descriptor Framework
- Timing in the BIFS Scene is conveyed by
  - Reference “Clock” through an elementary stream
  - Recovers composition time stamps from object clock reference to determine when a BIFS-command is to be executed
- FlexTime deals with unknown durations (starts together with, ends together with, …)

FlexTime

- What it does:
  - Provides temporal grouping (or temporal snapping).
- Allows a segment of stream to stretch/shrink.

Traditional "Hard" Object Time

- Absolute start time
  - No temporal grouping
  - Fixed duration

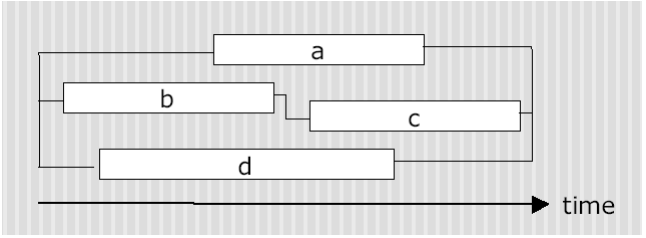


Traditional "Hard" Object Time

- Designed for “push” broadcast streams.
- Not adequate for interactive applications that require:
  - Synchronization of multimedia objects that originate from different hosts and/or different channels with different time bases.
- Difficulties in dealing with missed time stamps due to uncertainties in delivery systems.

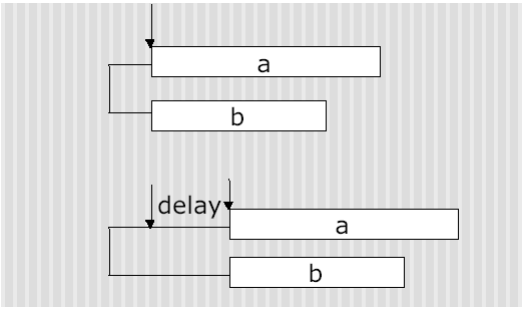
Flexible Object Time

- Relative start/end time
  - Duration need not be fixed ("springs")
  - Resolved at run-time



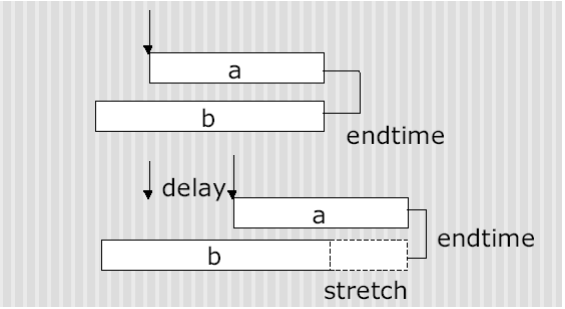
Example

- Delay of start



Example

- Duration stretching



FlexTime Model

- Media objects are linked to one another in a time graph using relationship constraints such as:
  - "CoStart, CoEnd or Meet."
- Each object may have a flexible duration:
  - Spring metaphor: min, opt, max durations
  - Author may specify specific stretch and shrink mode preferences for each object
- Supports synchronization of objects (or segments) from multiple sources (with possibly different time bases).

A FlexTime Player Can

- Support a timed wait for the arrival of a stream segment before the player starts rendering/playing the node tied to it:
- To compensate for the various network delays
- Slow down or speed up the rendering/playback speed of portions of streams:
- To re-adjust out-of-sync situations (caused by unknown length, uncontrolled arrival time or variation).

A FlexTime Player Can

- Synchronize insertion/deletion of multiple media/ BIFS nodes with some of the media stream (of unknown length or uncontrolled arrival time).
- Synchronize BIFS updates (e.g. events such as field updates) and OD deletes among multiple nodes/streams (with some of the streams of unknown length or uncontrolled arrival time).
- Have timeouts/escapes for all above constraints.

Interactivity and Behaviour

- VRML Based event model
  - Sensors generate events according to some user or scene events
  - Interpolators enable to generate time varying attributes for animation
  - ROUTEs pass events between nodes
- Extension: devices other than mouse with InputSensor

Interactivity and Behaviour

- Sensors:
  TimeSensor, TouchSensor, PlaneSensor(2D), ProximitySensor(2D), SphereSensor, DiscSensor, CylinderSensor, VisibilitySensor, InputSensor
- Interpolators:
  PositionInterpolator(2D), ScalarInterpolator, ColorInterpolator, CoordinateInterpolator(2D), NormalInterpolator, OrientationInterpolator
- ROUTEs …

Local Interaction and Behaviours

- Event model of BIFS uses VRML concept of ROUTEs
- ROUTE connects values of one node field with values of other node fields
- ROUTE can be combined with interpolators to cause animation
  - e.g. value of an interpolator is ROUTEd to the rotation field of a transform node
- Interpolators are inherited from VRML
- Sensors pick up user interaction

Overview

- Scene Description in MPEG-4
- BIFS Components and Composition
  - Composition
  - BIFS - Update
  - BIFS – Animation
  - Systems-Related Features
  - BIFS - Compression
- Sample Compression Results
- Coming up next: XMT

Changing Scenes: BIFS-Update

- A presentation/the scene has a temporal dimension
- On the Web:
  - a file with html or VRML content is downloaded and played/rendered locally
- In MPEG-4:
  - a BIFS presentation is delivered over time
  - An initial scene is downloaded
  - Further updates are delivered over time
    - → **BIFS-Update stream**
    - → **BIFS-Command protocol**
    - → **BiFS-Command stream**

Changing Scenes: BIFS-Update

- BIFS – Update protocol to modify the scene
- Supported commands :
  - Replace the entire current scene with a new one
  - Insert command for
    - + Node, indexed field, ROUTE
  - Delete command for
    - + Node, indexed value, ROUTE
  - Replace command for
    - + Node, field, indexed value, ROUTE

Overview

- Scene Description in MPEG-4
- BIFS Components and Composition
  - Composition
  - BIFS - Update
  - BIFS – Animation
  - Systems-Related Features
  - BIFS - Compression
- Sample Compression Results
- Coming up next: XMT

Streaming Animations: BIFS - Anim

- Animation:
  - Key frame animation by downloading interpolator data along with TimeSensor
  - modify fields in a 2D transform node
  - alternatively use BIFS-Update commands to update fields in the 2D transform node
- Both schemes are ok for small animations
- Won't work well if animations are dependent on live events
- Won't work well if high level of compression is needed (interpolators require a lot of data)

Streaming Animations: BIFS - Anim

- BIFS-Anim Protocol is designed to allow for high compression and to continuously animate objects
- Support continuous animation of :
  - Position, size, colors of objects
  - Face and Body parameters
  - 2D and 3D Mesh parameters
- Animation at work
  - BIFS scene is loaded – some objects are DEFed
  - Animation mask is loaded – contains list of nodes and fields to be animated
  - Animation stream is streamed – animation of frames in time stamped access units

Overview

- Scene Description in MPEG-4
- BIFS Components and Composition
  - Composition
  - BIFS - Update
  - BIFS – Animation
  - Systems-Related Features
  - BIFS - Compression
- Sample Compression Results
- Coming up next: XMT

Systems-Related Features

- Programmability
- Scalability and Scene Control

Programmability

- MPEG-J, access to resources, network and decoder interfaces
- ECMA-Script
  - Java-Scri
  - VRML-Script
- Script node, with eventIn/Out interface
- Example: emulating QTVR Panorama

Programmability

- Complex Actions
  - Conditionals: user-triggerable set of updates
  - Valuators: event casting
  - Scripts: parametric updates
  - MPEG-J:
    - full programs
    - more activation scenarios

Scalability and Scene Control

- Scalability and Scene Control
  - Varying capabilities of client platform
  - No normative composition specified
  - Hints for „best“ presentation
  - TermCap node – degradation of scene can be controlled through script or Java code

Scalability and Scene Control

- TermCap node
  - Ping client for capabilities
  - Frame rate, color depth, graphics hardware, audio outpur format, max audio sampling rate, spatial audio capabilities (e.g. reverb), CPU load, memory load
  - Each capability has 5 levels
  - Scene Graph can be adjusted according to client capabilities
- Content creator can assign global priority to objects

Overview

- Scene Description in MPEG-4
- BIFS Components and Composition
  - Composition
  - BIFS - Update
  - BIFS – Animation
  - Systems-Related Features
  - BIFS - Compression
- Sample Compression Results
- Coming up next: XMT

BIFS representation of VRML

- BIFS is a superset VRML
- BIFS is a binary representation for VRML worlds
- Automatic conversion of VRML worlds into MPEG-4 BIFS is possible

BIFS – Binary Coding

- Bit efficient representation of Scene Descriptions is a major requirement
- Binary encoding of BIFS – commands
  - Binary representation is done dependent on context
  - E.g. 18 different components for geometry of scene
  - → 5 bits needed
  - Provides some compression
- Quantization of node fields leads to further compression
- Different quantization schemes for BIFS – Updates and BIFS – Animation (intra/predictive coding plus arithmetic coding)

BIFS – Binary Coding

- Binary code for a node is an index into a list
- A node may have more than one type – it's binary ID depends on where the node appears in the scene tree
- DEFs:
  - 1 bit to indicate if a node has an ID;
  - Followed by ID bits; number of ID bits is conveyed through header coming with decoder configuration
- ROUTEs:
  - Basically the same as for DEFs
  - To distinguish between input the output fields, four different modes are specified
    - Field, ExposeField, EventIn, EventOut

BIFS – Binary Coding

- BIFS uses concept of Node Data Type (NDT)
- ≈ 30 NDTs, each node belongs to one or more NDTs
- NDT table is a list of nodes for each NDT
- Fixed length binary value indicates position of node in table

BIFS – Binary Coding

- E.g. Shape node exists as SF2DNode and SF3DNode
- 3D:
  - Shape exists as a child node of the Transform node, a MF3DNode accepting SF3DNodes as children
  - Shape node number 36 out of 48 possible SF3DNodes
  - → 6 bit binary code: ID 100100;
- 2D:
  - Shape exists as a child node of the 2D Transform node, a MF2DNode accepting SF3DNodes as children
  - Shape is number 23 out of 20 possible SF2DNodes
  - → 5 bit binary code: ID 10111;

BIFS – Binary Coding

- Node representation almost optimal
- No probabilistic model for nodes → no entropy coding

- At field level, BIFS defines new data types
  - SURL: for URLs, string or reference to streams
  - SFScript: use URLs to hold Scripts
  - SFBuffer: encode buffers to trigger BIFS-Update command using the ROUTE mechanism

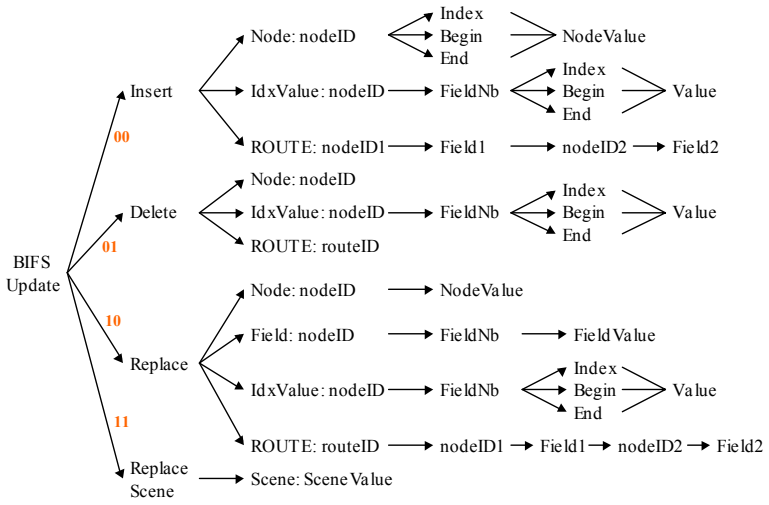
Quantization of Node Fields

- BIFS numeric types are int, float, double and aggregates
- QuantizationParameter is a pseudo-node within scene tree
- Every numerical field is classified into one of 14 semantic quantization categories:
  - 2D/3D positions/sizes
  - Color, Angle, Scale…
- Each category has a QuantizationNode specifying Parameters for a static, linear quantizer
- Quantizer is specified by value range (min,max) and number of bits used
- Encoding: QP Node is looked up in Node Coding Table

Quantization of Node Fields

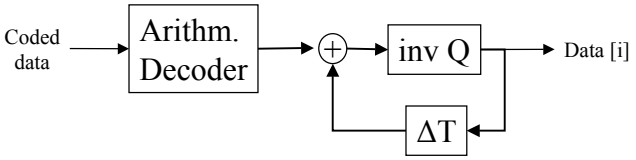
- EfficientFloat
  - Float representation different from IEEE 755
  - Up to 15 bits mantissa and up to 7 bits exponent
  - Exact format is specified in the header
- isLocal
  - Defines QP to be used only for next node in tree
  - Allows for local adaptation of quantization

Coding BIFS - Update Commands



Coding of BIFS - Animation

- Intra – Mode (I-frame):
  - Field values are quantized using QP
  - Entropy coding of quantized value
- Predictive – Mode (P-frame):
  - Difference from previous field value is computed
  - Difference is entropy coded



BIFS Scene Usage

- Usage of BIFS is not unique
- Allows for numerous desing choices and tricks
- Naive method to use Quantization Parameter
  - Determine max, min of filed values
  - Choose acceptable distortion
  - Compute QP and use in top node
- Better: Grouping of nodes with similar characteristics
  - Determine QP for grouping using naive method
  - Use QP in top node of grouping
- Trade off between declaration overhead and coding gain

Overview

- Scene Description in MPEG-4
- BIFS Components and Composition
  - Composition
  - BIFS - Update
  - BIFS – Animation
  - Systems-Related Features
  - BIFS - Compression
- Sample Compression Results
- Coming up next: XMT

Sample Compression Results

- Sample VRML files for comparison with BIFS
- For some files, Scene description is seperated from animation
- Movie and Floops are both animated cartoons, Floops contains the complete animation information

File	Description	VRML File [kB]	VRML Scene [kB]	VRML Animation	Duration [sec]
Intro	3D cartoon	1077	347	730	42.5
Movie	3D cartoon	2370	968	1402	113
Floops	3D cartoon	1324	1324		
Flight	2D interactive content	62	62		
Tree	3D simple geometry	80	80		
Channel	2D interactive content	100	100		

Sample Compression Results

- Sample results from compression with BIFS and BIFS-Commands
- The used quantisation parameters resulted in no apparent error
- Floats for „no QP“ are represented with „Efficient Float“ using 16 bits
- Quantized values for „1 QP“ are represented by 8 to 10 bits
- Quantized values for „many QP“ are represented by 5 to 10 bits

File	VRML Scene	BIFS Scene (no QP)		BIFS Scene (1 QP)		BIFS Scene (Many QPs)	
	File Size [kB]	File Size [kB]	Ratio	File Size [kB]	Ratio	File Size [kB]	Ratio
Intro	347	101	3.4	48	7.3	45	7.7
Movie	968	222	4.4	110	8.8	103	9.4
Floops	1324	284	4.7	182	7.3	95	13.9
Flight	62	5.5	11.3	4.7	13.2	4.7	13.2
Tree	80	4.9	16.3	4.9	16.3	4.9	16.3
Channel	100	3.2	31.3	2.9	34.5	2.9	34.5

Sample Compression Results

- Sample results from combined BIFS and BIFS- Animation compression

File	VRML File	BIFS		Winzip		BIFS vs. Winzip
	File Size [kB]	File Size [kB]	Ratio	File Size [kB]	Ratio	Gain (%)
Intro	1077	75	14.4	179	6.0	+140
Movie	2370	187	12.7	422	5.6	+80
Floops	1324	95	13.9	124	10.7	+30
Flight	62	4.6	13.5	3.0	20.7	-30
Tree	80	4.9	16.3	6.2	12.9	+30
Channel	100	2.9	34.5	4.9	20.4	+70

Sample Compression Results

- Sample results from BIFS- Animation compression

File	Duration	VRML Animation	BIFS Animation		Bit Rate
	[sec]	File Size [kB]	File Size [kB]	Ratio	[kbps]
Intro	42.5	730	30	24.3	5.6
Movie	113.0	1402	84	16.7	6.2

Overview

- Scene Description in MPEG-4
- BIFS Components and Composition
  - Composition
  - BIFS - Update
  - BIFS – Animation
  - Systems-Related Features
  - BIFS - Compression
- Sample Compression Results
- Coming up next: XMT

Coming up next: XMT

- XMT: eXtensible MPEG-4 Textual Format
- framework for representing MPEG-4 scene description using a textual syntax.
- Exchange of conent between content authors
- facilitates interoperability with both the Extensible 3D (X3D) and the Synchronized Multimedia Integration Language (SMIL)
- The XMT format can be interchanged between SMIL players, VRML players, and MPEG-4 players.

Coming up next: XMT

- The format can be parsed and played directly by a W3C SMIL player, preprocessed to Web3D X3D and played back by a VRML player, or compiled to an MPEG-4 representation such as mp4.
- XMT-A is an XML-based version of MPEG-4 content, which contains a subset of the X3D. one-to-one mapping between the textual and binary formats.
- XMT-O is a high-level abstraction of MPEG-4 features based on the W3C SMIL. XMT provides a default mapping from O to A

Coming up next: XMT

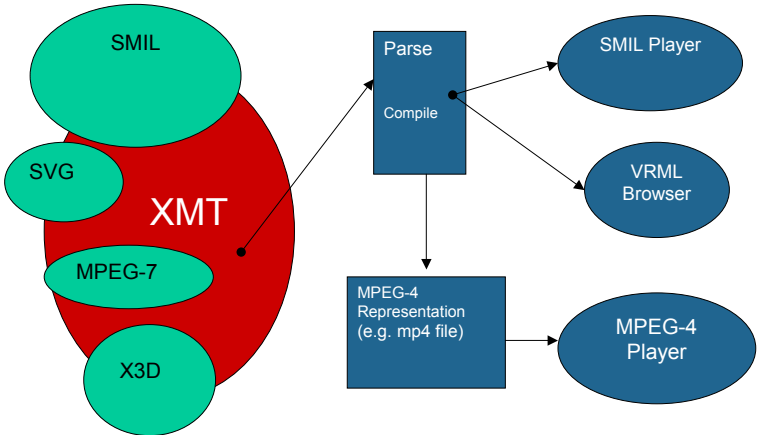


Figure - the eXtensible MPEG-4 Textual format

XMT

- Joint development with Web3D in collaboration with SMIL
- A textual format for MPEG-4 binary representation
  - X3D (an XML representation of VRML) compatible.
  - Preserves authors' intentions.
- A high-level abstraction for MPEG-4 technologies reusing SMIL modules.
- Facilitates interoperability between MPEG-4, SMIL, and X3D.

VRML vs. SMIL

- VRML
  - Designed primarily for 3D spatial manipulation for expert authors
  - There is no time transform
  - Needs an extension to cover ODs, BIFS updates
- SMIL
  - Designed primarily for temporal synchronization (and user interactive events) for web authors
  - Representation of objects captures non programmer authors' point of view
  - Needs to be augmented to cover the notions of nodes, (sensors, routes), ODs, BIFS updates, and 3D.

X3D

- An XML-based version of BIFS based on the XML conversion of VRML97, which is being developed for VRML 200x.
- X3D provides a straightforward, one-to-one mapping between the textual and binary formats.
- BIFS is a rather low-level scene description facility.

Links and References

- MPEG-4 Industry Forum: www.m4if.org
- MPEG Home page: www.Cselt.it/mpeg
- MPEG Systems Home Page:
- Envivio home page: www.envivio.tv

- Aaron Walsh, Mikael Bourges-Sevenier: *MPEG-4 Jump Start*. Prentice Hall PTR. 2002
- Atul Puri, Tsuhan Chen (ed.): *Multimedia Systems, Standards and Networks*. Marcel Dekker. 2000.

Credit and Acknowledgment

- Thanks to for Demo Materials and Technical Information
  - Shawn Ambwani (Envivio)
  - Jean-Claude Duford (ENST)
  - Carsten Herpel (Thomson Multimedia)
  - Marty Stouffer (Wildlife)
  - Rob Koenen and the MPEG community

3D MESH CODING IN MPEG-4

Gabriel Taubin
IBM T.J. Watson Research Center

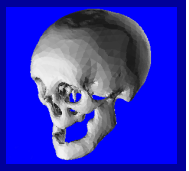
Siggraph 2002
July 2002

3D Geometry Compression

- The goal of 3D Geometry Compression is to do for 3D data what JPEG, MPEG, MP3, and other standards do for images, video, and music:
  - Fast and cheap transmission and download
  - Invisible to users
  - An enabling technology other applications
- Other challenges for 3D data include user interfaces, model acquisition, and software/standardization.

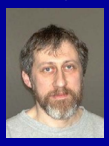
3D Geometry Compression

- POLYGONAL MESH
  - CONNECTIVITY
  - GEOMETRY
  - PROPERTIES
- HOW TO REPRESENT WITH MINIMUM NUMBER OF BITS
  - FOR COMPACT STORAGE
  - INCREMENTAL TRANSMISSION
  - PROGRESSIVE TRANSMISSION



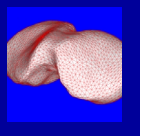
Main differences between 2D and 3D

- 2D Image data :
  3D signals (RGB) defined on regular grids
  - Grid described by two numbers W x H
  - Can exploit implicit hierarchical structure
  - Can use Fourier analysis (DCT)



Main differences between 2D and 3D

- 3D Mesh data :
  3D signals (XYZ) defined on irregular grids
  - Need to encode the connectivity
  - The topology may be complex
    - Handles, holes, non-manifold
  - No implicit hierarchical structure
  - Fourier Analysis may not be feasible without remeshing



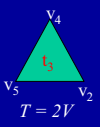
Large T-mesh needs O(V log(V)) bits

Vertices and values:
3xB bits/vertex

vertex 1	x	y	z	c
vertex 2	x	y	z	c
vertex 3	x	y	z	c

Triangle/vertex incidence:
3xlog$_2$(V) bits/triangle

Triangle 1	1	2	3
Triangle 2	3	2	4
Triangle 3	4	2	5
Triangle 4	7	5	6
Triangle 5	6	5	8
Triangle 6	8	5	1

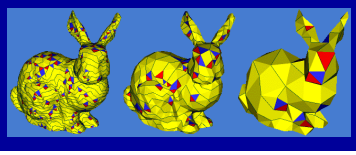


$T = 2V$

- Triangle mesh often stored as **(VRML)**
  - table of V vertices (geometry)
  - table of T triangles (connectivity)
- Vertex Index log(V)) bits
- Connectivity 6V log(V) bits

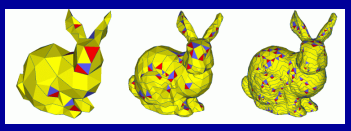
Simplification

- SIZE REDUCTION (VERTICES AND FACES)
- MINIMIZING GEOMETRIC ERROR TO ACCELERATE INTERACTION WITH LODs
- TO SUB-SAMPLE OVER-SAMPLED MESHES
- LOSSY COMPRESSION OF CONNECTIVITY
- LOSSY COMPRESSION OF GEOMETRY
- ENCODING & TRANSMISION ?



Progressive Transmission : Naïve Solution

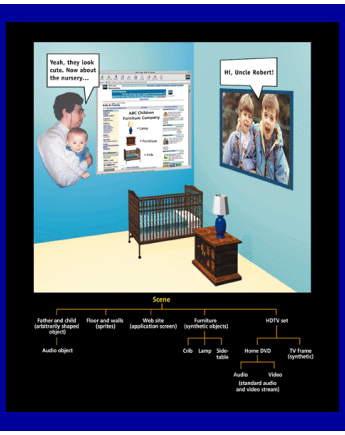
- TRANSMIT LEVELS OF DETAIL FROM COARSE TO FINE RESOLUTION
- COMPRESS LEVELS INDEPENDENTLY OF EACH OTHER



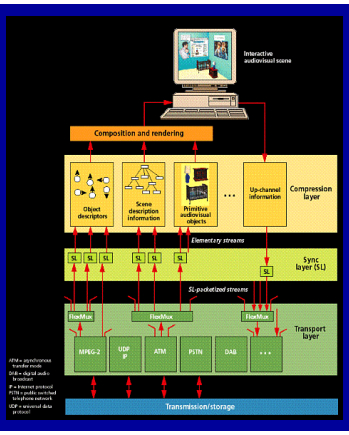
Progressive Transmission : Better Solution

- TRANSMIT BASE MESH + SEQUENCE OF REFINEMENT STEPS
- IDEALLY ALL COMPRESSED
- SIZE-GRANULARITY TRADEOFF
- BETTER THAN NON-PROGRESSIVE COMPRESSED LODs

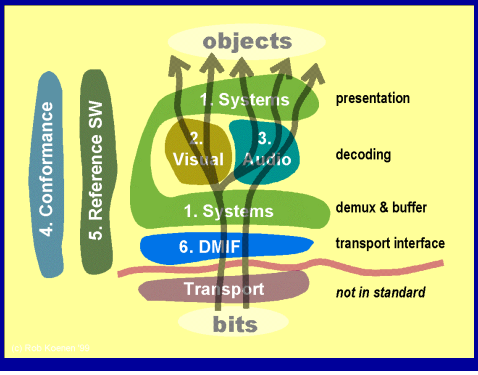
MPEG-4



MPEG-4



MPEG-4 PARTS, PROFILES, LEVELS



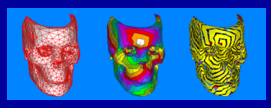
MPEG-4 PROCESS

- PROPOSALS
- CORE EXPERIMENTS
- 2 INDEPENDENT IMPLEMENTATIONS
- BITSTREAM EXCHANGE
- SOURCE CODE RELEASE
- REPORTED RESULTS VERIFIED
- REFERENCE IMPLEMENTATION IS INTEGRATED

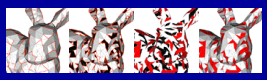
MPEG-4 3D MESH CODING

- INCREMENTAL SINGLE RESOLUTION MODE BASED ON TOPOLOGICAL SURGERY (IBM/EPFL)
- PROGRESSIVE MODE BASED ON PROGRESSIVE FOREST SPLIT (IBM)
- NON-MANIFOLD ENCODING BASED ON CUTTING AND STITCHING (IBM)
- ERROR RESILIENCY BASED ON COMPONENT DATA PARTITIONING (SAMSUNG)
- OTHER PROPOSED FEATURES DID NOT SATISFY REQUIREMENTS

3D Geometry Compression @ IBM



•Topological Surgery (96)



•Progressive Forest Split (98)

ISO/IEC JTC1/SC29 WG11

MPEG

MOVING PICTURE EXPERTS GROUP

•3D Mesh Coding Tools

•HotMedia

Very Active Research Area

- Papers on connectivity encoding (circa June 2001)
  - Itai,Rodeh: Representation of graphs, Acta Informatica, 82
  - Turan: On the succinct representation of graphs, Discrete Applied Math, 84
  - Naor: Succinct representation of general unlabeled graphs, Discrete Applied Math, 90
  - Keeler,Westbrook: Short encoding of planar graphs and maps, Discrete Applied Math, 93
  - He, Kao,Lu: Linear time succinct encodings of planar graphs, Siam J. Discrete Math, 99
  - Chuang, Garg, He, et al: Compact encodings of planar graphs, ICALP, 98
  - Deering: Geometry Compression, Siggraph, 95
  - Taubin,Rossignac: Geometric compression through topological surgery, ACM ToG, 98
  - Taubin,Horn,Lazarus,Rossignac: Geometry coding and VRML, Proc. IEEE, 98
  - Touma,Gotsman: Triangle Mesh Compression, GI, 98
  - Gumbold,Straßer: Realtime compression of triangle mesh connectivity, Siggraph, 98
  - Rossignac: Edgebreaker: Compressing the incidence graph of triangle meshes, TVCG, 99
  - Rossignac,Szymczak: Wrap&Zip: Linear decompression of triangle meshes, CGTA, 99
  - King&Rossignac: Guaranteed 3.67V Bit Encoding of Planar Triangle Graphs, CCCG, 99
  - Bajaj et al.: Single resolution compression of arbitrary triangle meshes, DCC 99
  - Cohen-Or: Progressive compression of arbitrary triangle meshes, Visualization 99
  - Isenburg&Snoeyink, Mesh Collapse Compression, SIBGRAPI 99
  - Snoeyink, VanKreveld: Linear-time reconstruction of Delaunay triangulations, ESA 99
  - Denny,Sohler: Encoding a triangulation as a permutation of its point set, CCCG, 97
  - King, Symczak, &Rossignac: Connectivity Compression for Irregular Quadrilateral Meshes, submitted
  - Isenburg&Snoeyink: Face-Fixer, Siggraph 2000
- Above list does not include the progressive methods

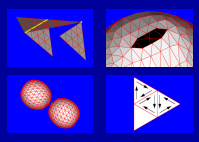
Intelectual Property (circa June 2001)

- 6,167,159 -- Touma&Gotsman's valence-based mesh compression
- 6,046,744 -- Hoppe's Selective Refinement of Progressive Meshes
- 5,930,248 -- Zarge&Schroeder's Mesh Decimation/Simplification Algorithm
- 5,793,371 -- Deering's Compression used in Java3D
- 5,825,369 -- Rossignac&Taubin's Topological Surgery
- 6,047,088 -- vanBeek&Tekalp -- dynamic 2D meshes
- 5,736,987 -- Drucker&Mitchell's compression method for normals
- 5,818,463 -- Tao et al. -- a method for animated 3D model compression using quads, 1998
- 5,905,507 -- Rossignac&Taubin's Topological Surgery for generalized models
- 5,929,860 -- Hoppe's Progressive Meshes
- 5,963,209 -- Hoppe's Progressive Meshes -- Encoding and transmission
- 5,966,133 -- Hoppe's Progressive Meshes -- Geomorphs and variable resolution
- 5,966,140 -- Popovic&Hoppe's Progressive Simplicial Complices
- 6,009,435 -- Taubin -- level of detail method that sends the highest LOD connectivity first
- 6,031,548 -- Gueziec,Lazarus,Taubin -- Progressive Multi-level transmission
- 6,016,152 -- Gueziec&Taubin -- Cutting&Stitching
- 6,175,368 -- Dyer (HP) -- Fast decompression of surface normals
- 6,169,819 -- Dyer (HP) -- Fast compression of surface normals
- 6,160,848 -- Burr, 2001, method for continuous LOD control
- 6,088,034 -- Deering's Java3D -- surface normal decompression
- 5,870,094 -- Deering's Java3D -- system for transferring compressed 3D
- 5,867,167 -- Deering's Java3D compression
- 5,842,004 -- Deering's Java3D decompression
- 6,028,610 -- Deering's Java3D -- geometry instructions for decompression
- 5,793,153 -- Deering's Java3D -- mesh buffer for decompression
- 5,905,502 -- Deering's Generalized Triangle Mesh compression

Surface Representations / Meshes

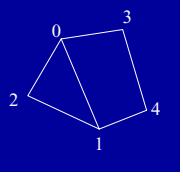
Polygon Meshes

- Components
  - Connectivity (Vertices, Edges, Faces)
  - Geometry (Vertex Coordinates)
  - Properties (Normals, Colors, Texture Coordinates)
- Connectivity (combinatorial algorithms)
  - Boundary / Regular / Singular Edges and Vertices
  - Connected Components
  - Manifold / Non-manifold
  - Orientable / Oriented
  - Topology



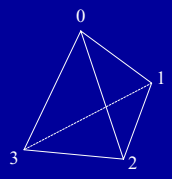
IndexedFaceSet (VRML)

- Array of vertex coordinates
- Each 3D vertex has an associated vertex index in {0,…,V-1}
- A triangle is defined by three vertex indices (i,j,k)
- A polygonal face without holes is defined by more indices
- coordIndex [0,1,2,-1,0,3,4,1,-1]
- VRML'97 file format
- Adjacency information can be recovered with no ambiguity



Tetrahedron.wrl

```
#VRML V2.0 utf8
Shape {
  geometry IndexedFaceSet {
    coord Coordinate {
      point [
         1.633 -0.943 -0.667
         0.000  0.000  2.000
        -1.633 -0.943 -0.667
         0.000  1.886 -0.667
      ]
    }
    coordIndex [
      0 1 2 -1 3 1 0 -1 2 1 3 -1 2 3 0 -1
    ]
  }
}
```



Polygonal Mesh Components in VRML

- Connectivity
  - coordIndex (faces)
- Geometry
  - coord (vertex coordinates)
- Properties
  - color/colorIndex/colorPerVertex
  - normal/normalIndex/normalPerVertex
  - texCoord/texCoordIndex

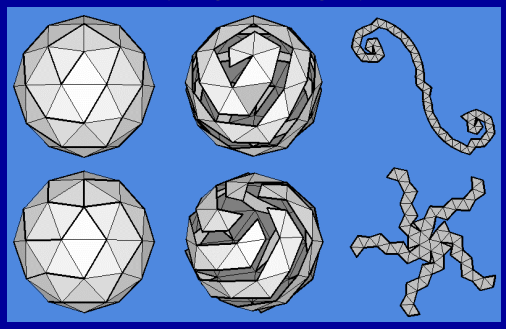
Manifold Polygonal Mesh

- No singular edges
  - Boundary
  - Regular
- No singular vertices
  - Regular

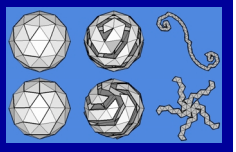
Single Resolution Mesh Compression

Geometric Compression through Topological Surgery,
by G. Taubin and J. Rossignac,
in ACM Transactions on Graphics, Vol. 17, No. 2, 1998; and IBM Technical Report RC-20340, January 1996

Topological Surgery



Topological Surgery



- First for a simple T-mesh
  - Sphere topology
  - Euler formula V-E+F=2
- Cut connected manifold mesh through a maximal set of edges such that new mesh
  - has no internal vertices
  - is connected
- Set of cut edges is a spanning tree (vtree)
  - Touches all the mesh vertices
  - Has no loops

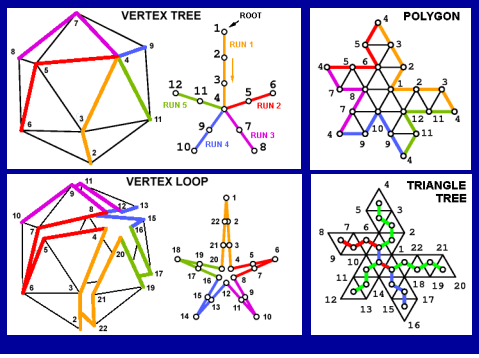
Topological Surgery

- Set of remaining edges is a spanning tree in the dual graph (ftree) because
  - It is a connected spanning sub-graph
  - By Euler formula : F-1 remaining edges
    - V-E+F=2 or E=(V-1)+(F-1)
  - Connected graph with F nodes and F-1 edges is a tree
- Triangle mesh has a tree as dual graph if and only if it is a simple polygon (with one boundary loop)

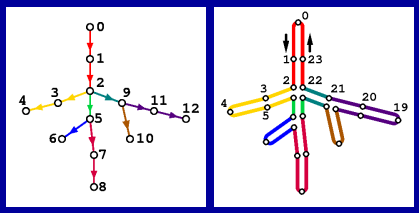
Topological Surgery Representation

- For a simple mesh
- Vertex tree (VT)
  - Root node and explicit order of traversal
- Simple polygon (SP)
  - Root edge and implicit order of traversal
- Vertex coordinates
  - Associated to nodes of VT
- Implicit stitching information

Topological Surgery



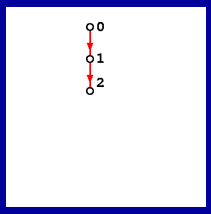
Vertex Tree



1. ORDER OF TRAVERSAL
2. RUN-LENGTH ENCODING
3. VERTEX LOOP DECODING

Taubin / 3DMC MPEG-4

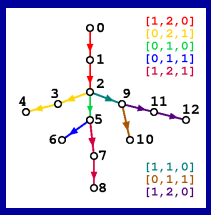
Vertex Tree



- COMPOSED OF RUNS
- TRAVERSAL ORDER DETERMINED BY ROOT AND ORIENTATION
- REPRESENTED AS TABLE OF RUNS (last,length,leaf)

Taubin / 3DMC MPEG-4

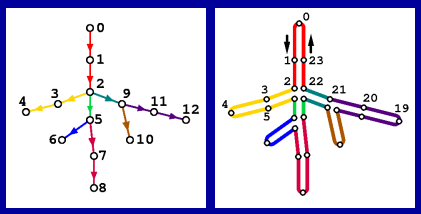
Vertex Tree



- REPRESENTED AS TABLE OF RUNS (last,length,leaf)
  - last
    LAST STARTING AT CURRENT BRANCHING NODE
  - length
    NUMBER OF EDGES
  - leaf
    ENDS IN LEAF OR BRANCHING

Taubin / 3DMC MPEG-4

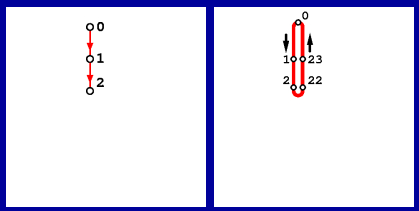
Vertex Tree



1. ORDER OF TRAVERSAL
2. RUN-LENGTH ENCODING
3. VERTEX LOOP DECODING

Taubin / 3DMC MPEG-4

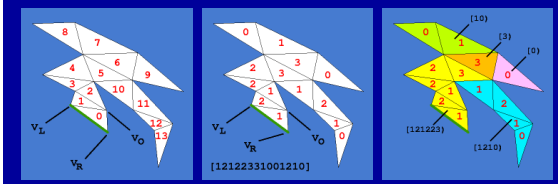
Vertex Loop Decoding



- LOOK-UP TABLE OF LENGTH 2V-2 CONTAINING VERTEX TREE INDICES
- DETERMINES HOW TO STITCH BOUNDARY EDGES OF SIMPLE POLYGON
- CONSTRUCTED BY DECODER TRAVERSING VERTEX TREE

Taubin / 3DMC MPEG-4

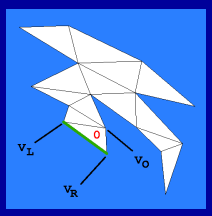
Simple Polygon



1. ORDER OF TRAVERSAL
2. CONSTANT LENGTH ENCODING
3. RUN-LENGTH ENCODING
4. POLYGON LOOP DECODING

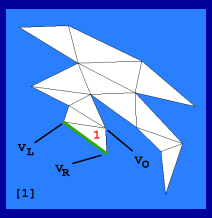
Taubin / 3DMC MPEG-4

Order of Traversal



- START AT ROOT EDGE
  WITH RIGHT VERTEX = ROOT VERTEX
- AT BRANCH : PUSH RIGHT TRAVERSE LEFT
- AT LEAF : POP OR FINISH
- AT REGULAR: ADVANCE

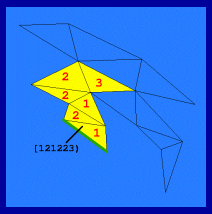
Constant Length Encoding



0 - LEAF
1 - LEFT BDRY
2 - RIGHT BDRY
3 - BRANCH

- 2-BIT CODE PER TRIANGLE

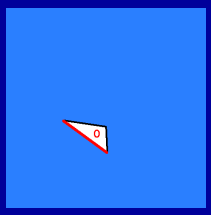
Run-Length Encoding



[121223)=(5,0)+[01011]
[3) =(1,0)+[]
[10) =(2,1)+[0]
[0) =(1,1)+[]
[1210) =(3,1)+[010]

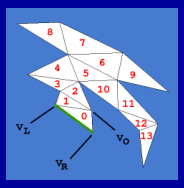
- END IN LEAF OR BRANCHING
- TABLE OF RUNS (length,leaf) + MARCHING BITS

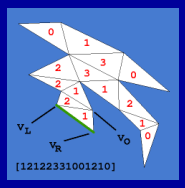
Polygon Loop Decoding

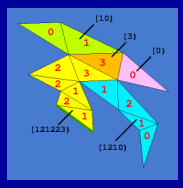


- POLYGON VERTICES ARE NOT VISITED IN LOOP ORDER DURING TRAVERSAL
- MATCHING WITH VERTEX LOOP DETERMINES HOW TO STITCH BOUNDARY EDGES

Simple Polygon







1. ORDER OF TRAVERSAL
2. CONSTANT LENGTH ENCODING
3. RUN-LENGTH ENCODING
4. POLYGON LOOP DECODING

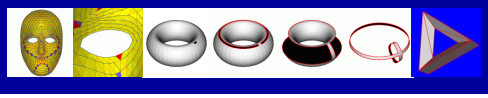
Complex Mesh Algorithm

FACE TREE
- CONSTRUCT SPANNING TREE IN DUAL GRAPH

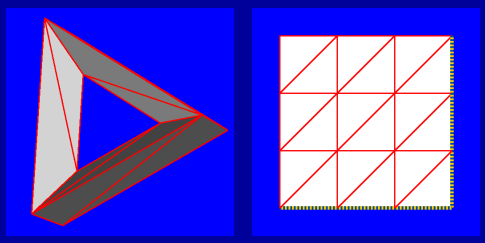
VERTEX GRAPH
- GRAPH COMPOSED OF REMAINING EDGES

VERTEX TREE
- CONSTRUCT SPANNING TREE IN VERTEX GRAPH

JUMP EDGES
- REMAINING EDGES OF VERTEX GRAPH

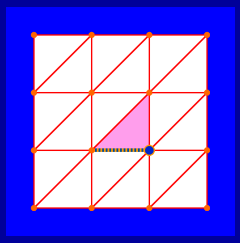


Complex Mesh Example



Taubin / 3DMC MPEG-4

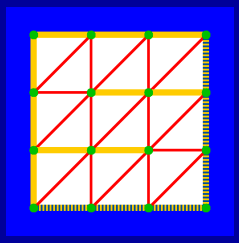
FACE TREE



CONSTRUCT DUAL SPANNING TREE

Taubin / 3DMC MPEG-4

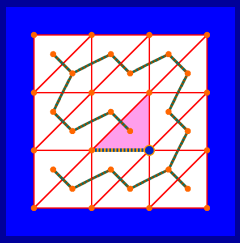
VERTEX GRAPH



COMPOSED OF REMAINING EDGES

Taubin / 3DMC MPEG-4

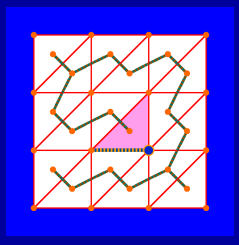
VERTEX TREE



SPANNING TREE IN VERTEX GRAPH

Taubin / 3DMC MPEG-4

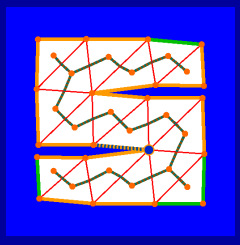
JUMP EDGES



REMAINING EDGES

Taubin / 3DMC MPEG-4

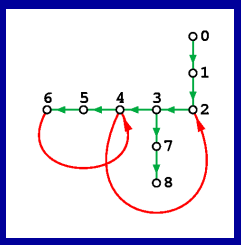
SIMPLE POLYGON



CUT THROUGH VERTEX GRAPH

Taubin / 3DMC MPEG-4

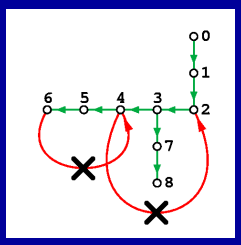
VERTEX GRAPH



VERTEX TREE + JUMP EDGES

Taubin / 3DMC MPEG-4

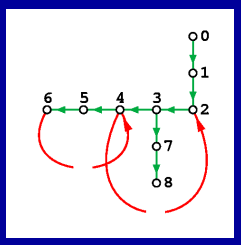
EXTENDED VERTEX TREE



CUT THROUGH JUMP EDGES

Taubin / 3DMC MPEG-4

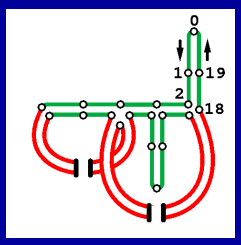
JUMP EDGES



ATTACHED TO LEAF-ENDING RUNS

Taubin / 3DMC MPEG-4

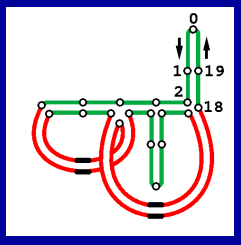
VERTEX LOOP



DECODE ALMOST AS USUAL

Taubin / 3DMC MPEG-4

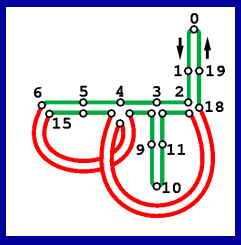
GRAPH LOOP



THEN CONNECT START AND END OF EACH JUMP EDGE

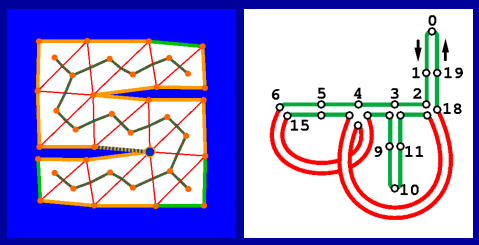
Taubin / 3DMC MPEG-4

GRAPH LOOP



Taubin / 3DMC MPEG-4

CONNECTIVITY



SIMPLE POLYGON + VERTEX LOOP

VERTEX GRAPH ENCODING

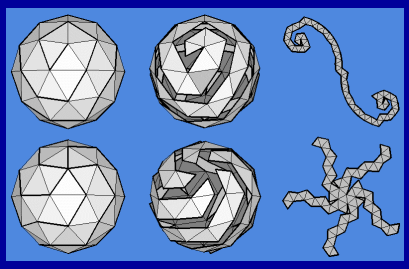
- VERTEX TREE
- TREE SPANNING VERTEX GRAPH
- JUMP EDGES
- CUT IN HALF AND REGARDED AS PART OF THE TREE
- START AND END OF JUMP ASSOCIATED WITH LEAF-ENDING RUNS
- EXTENDED VERTEX TREE
- TABLE OF VERTEX RUNS --- (last,length,leaf)
- APPEND JUMP DATA IF RUN ENDS IN LEAF

ENCODING OF CONNECTIVITY

- SIMPLE POLYGON
- TABLE OF TRIANGLE RUNS --- (length,leaf)
- MARCHING BITS
- POLYGON BITS
- VERTEX GRAPH
- TABLE OF VERTEX RUNS --- (last,length,leaf)
- JUMP EDGES
- FURTHER ENTROPY ENCODING

Multiple Ways of Constructing Trees

Affect encoding efficiency



Encoding of Polygon Meshes

- Topological Surgery
  - Construct face tree in the dual mesh
  - Construct vertex graph as remaining edges
  - While traversing face tree, triangulate faces and record one additional bit per marching edge

Encoding of non-manifold meshes

- Convert to manifold by "cutting" through singular edges and vertices
- Minimize number of connected components by stitching pairs of singular boundaries
- Encode stitching information (vertex clustering) along traversal

Compresion of Geometry and Properties

Geometry prediction

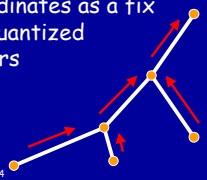
- Connectivity is encoded independently of geometry, but geometry is predicted as a function of the connectivity
- Each value is predicted as a function of previously received values and connectivity

Quantization / Prediction

- Enclose vertices in bounding cube
- Quantize to B bits per coordinate
- Encode quantized vertex coordinates without loss of information
- Entropy of quantized values is high
- Need better modeling
- Use order of traversal to predict quantized values from previous values
- Encode errors

Tree Prediction

- Vertices are nodes of a rooted tree
- Encode absolute quantized coordinates of root node
- In depth order
  - Predict quantized coordinates as a fix linear combination of quantized coordinates of ancestors
  - Encode correction

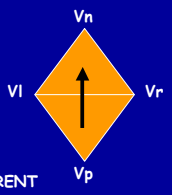


Tree Prediction

- Special case : delta coding
  - Predictor is ancestor value
  - Error is difference
  - Used by Deering

PARALLELOGRAM PREDICTION

En=Vn-P(Wl,Wr,Wp)
Fn=Q(En)
Wn=Fn+P(Wl,Wr,Wp)



- PREDICT (P)
  - VERTEX COORDINATES --- Vn
  - PREDICT FROM 3 VERTICES OF PARENT TRIANGLE IN TRIANGLE TREE Wl,Wr,Wp
- QUANTIZE (Q)
  - CORRECTION VECTORS --- En
- ENTROPY ENCODE
  - QUANTIZED CORRECTION VECTORS --- Fn

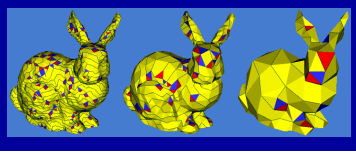
TS - ENCODING OF CONNECTIVITY

- SIMPLE POLYGON
- TABLE OF TRIANGLE RUNS --- (length,leaf)
- MARCHING BITS
- POLYGON BITS
- VERTEX GRAPH
- TABLE OF VERTEX RUNS --- (last,length,leaf)
- JUMP EDGES
- FURTHER ENTROPY ENCODING

Progressive Transmission

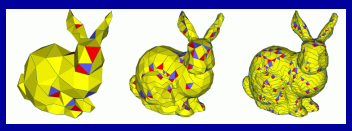
Simplification

- SIZE REDUCTION (VERTICES AND FACES)
- MINIMIZING GEOMETRIC ERROR TO ACCELERATE INTERACTION WITH LODs
- TO SUB-SAMPLE OVER-SAMPLED MESHES
- LOSSY COMPRESSION OF CONNECTIVITY
- LOSSY COMPRESSION OF GEOMETRY
- ENCODING & TRANSMISION ?



Progressive Transmission : Naïve Solution

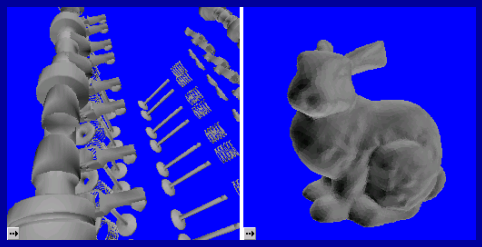
- TRANSMIT LEVELS OF DETAIL FROM COARSE TO FINE RESOLUTION
- COMPRESS LEVELS INDEPENDENTLY OF EACH OTHER



Progressive Transmission : Better Solution

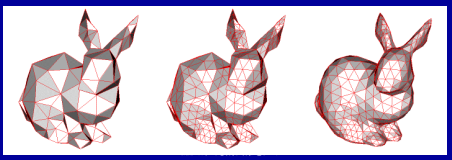
- TRANSMIT BASE MESH + SEQUENCE OF REFINEMENT STEPS
- IDEALLY ALL COMPRESSED
- SIZE-GRANULARITY TRADEOFF
- BETTER THAN NON-PROGRESSIVE COMPRESSED LODs

Topological Complexity



Recursive Subdivision

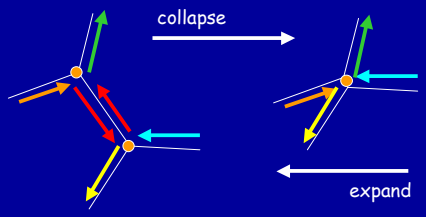
- CONNECTIVITY SUBDIVISION + SMOOTHING
- IDEAL PROGRESSIVE SCHEME
- CONSTANT COST PER REFINEMENT STEP
- RESTRICTED TO MESHES WITH SUBDIVISION CONNECTIVITY



Progressive Meshes
by H. Hoppe,
in Proceedings of Siggraph'96

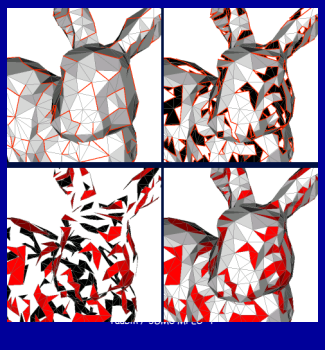
Edge Collapse / Split

- Remove/insert pair of twin half-edges and fix 2 next and 2 prev edges

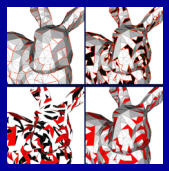


Progressive Forest Split Compression
by G. Taubin, A. Gueziec,
W. Horn, and F. Lazarus,
Siggraph'98, Orlando, FL, July 1998

Progressive Forest Split



Progressive Forest Split

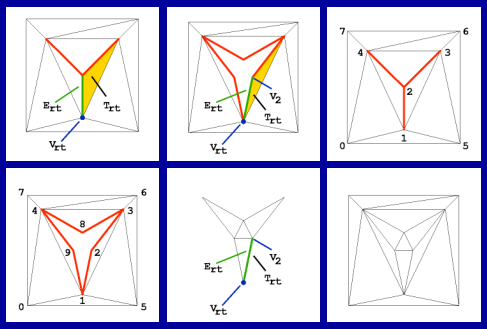


- BASE MESH + SEQUENCE OF REFINEMENT STEPS
- BASE MESH TOPOLOGICAL SURGERY
- REFINEMENT STEP
- FOREST SPLIT OPERATION
- EFFICIENT ENCODING OF PROGRESSIVE MESHES (Hoppe SG'96)

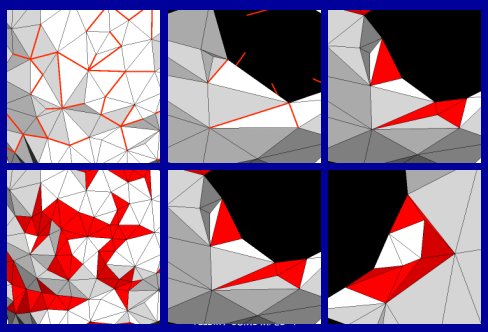
The Forest Split Operation

- REPRESENTATION
  - FOREST OF EDGES
  - SEQUENCE OF SIMPLE POLYGONS
- NAIVE ENCODING
  - IMPLICIT ENUMERATION CONVENTIONS
  - FOREST = 1 BIT PER EDGE
  - POLYGONS = 2 BITS PER TRIANGLE
- COST OF ADDING aT TRIANGLES
  - 3.5 BITS PER TRIANGLE FOR a=1.00
  - 5.0 BITS PER TRIANGLE FOR a=0.50
  - 5+log(T) BITS PER TRIANGLE FOR PM

Tree Split



Mesh Boundaries

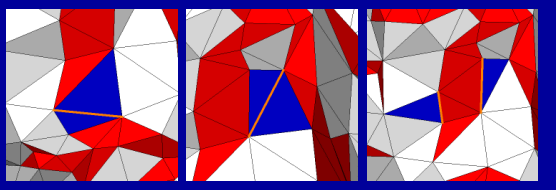


Edge collapse based simplification

- For each edge compute an error value and putt all errors in a priority queue
- While the queue is not empty
  - Delete edge e corresponding to the minimum error from the queue
  - If e is collapsible
    - Collapse e
    - Remove other edges incident to e from the queue, recompute error, reinsert in the queue

Automatic Conversion to PFS Format

- ANY EDGE-COLLAPSED BASED SIMPLIFICATION
- WITH TWO EXTRA COLLAPSIBILITY TESTS

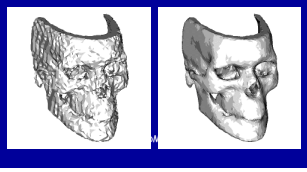


GOOD BAD BAD

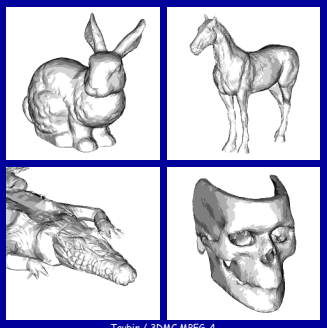
Smoothing in PFS

PRE-SMOOTHING AS PREDICTOR
- QUANTIZE AND ENCODE DISPLACEMENTS
- REDUCE DISPLACEMENTS BY PRE-SMOOTHING

POST-SMOOTHING TO REMOVE ARTIFACTS

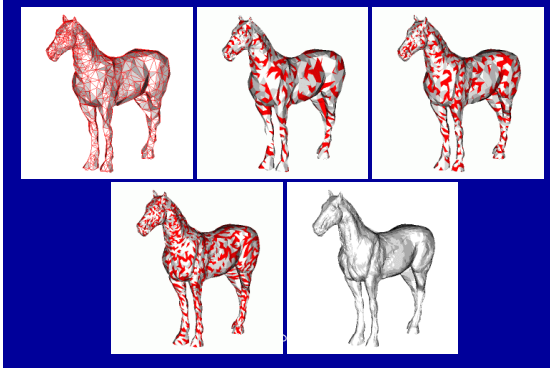


EXAMPLES

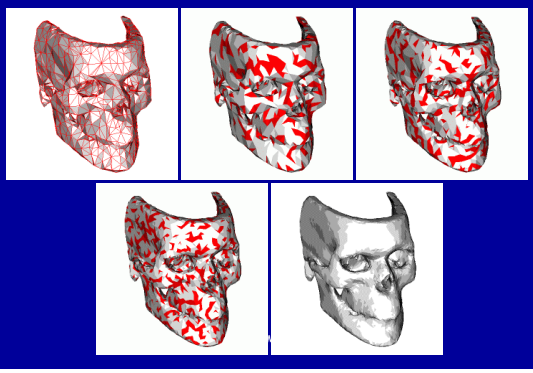


Taubin / 3DMC MPEG-4

Horse



Skull



Statistics / Results

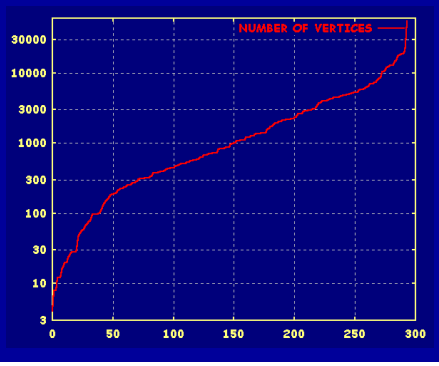
Taubin / 3DMC MPEG-4

MPEG-4 INCREMENTAL ENCODING

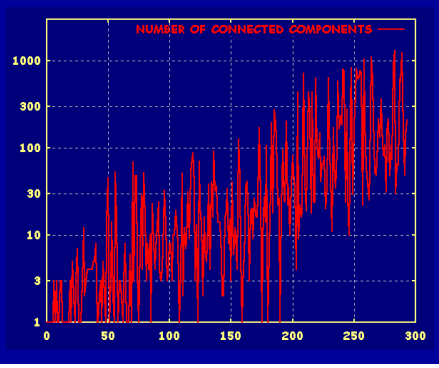
- VERTEX GRAPH
  - TABLE OF VERTEX RUNS (last,length,leaf)
  - JUMP EDGES
- TRIANGLE TREE
  - TABLE OF TRIANGLE RUNS (length,leaf)
- PER TRIANGLE DATA
  - POLYGON+MARCHING BITS
  - VERTEX COORDINATES
  - NORMAL COORDINATES
  - COLOR COORDINATES
  - TEXTURE COORDINATES
- FURTHER ENTROPY ENCODING

Taubin / 3DMC MPEG-4

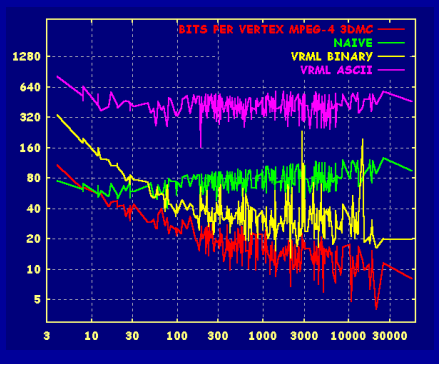
294 SIMPLE VRML MODELS



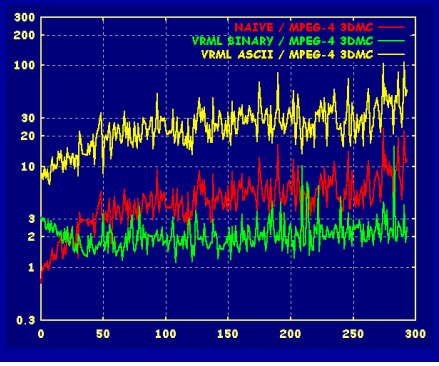
NUMBER OF CONNECTED COMPONENTS



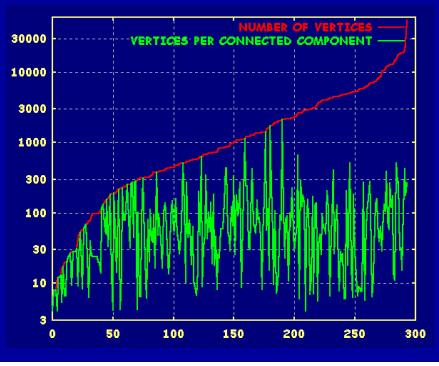
COMPRESSION EFFICIENCY



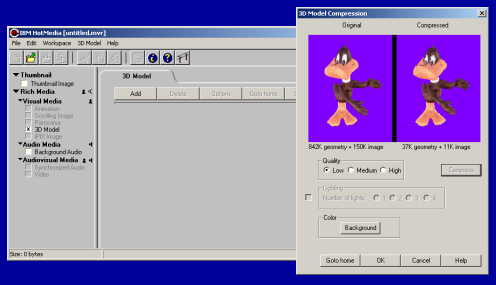
COMPRESSION RATIOS



NUMBER OF VERTICES PER CC



IBM HotMedia 3.0 Assembly Tool



http://www.ibm.com/hotmedia

Taubin / 3DMC MPEG-4

IBM HotMedia 3D Model Java Player

http://www.ibm.com/hotmedia

Taubin / 3DMC MPEG-4

3D Graphics on Handheld Computers

Taubin / 3DMC MPEG-4


SAN ANTONIO
SIGGRAPH
2002


SAN ANTONIO
SIGGRAPH
2002
Light Field Mapping
MPEG-4 Implementation
Radek Grzeszczuk

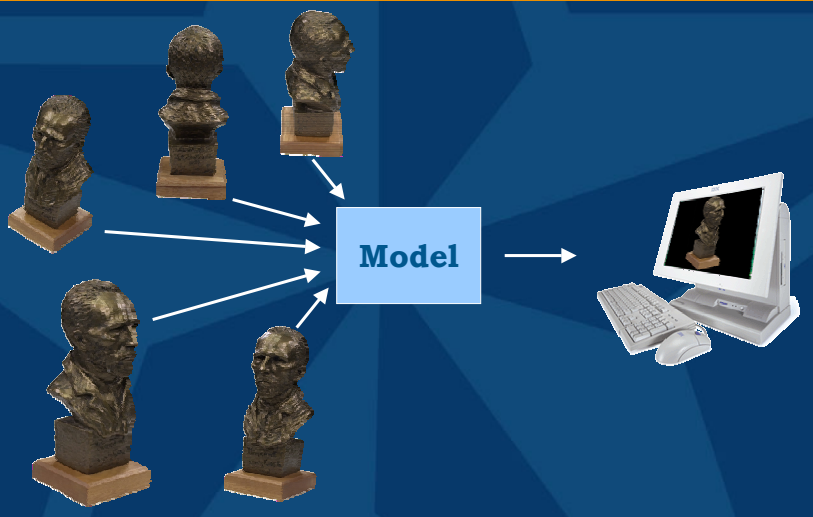
Collaborations

Wei-Chao Chen
University of North Carolina at Chapel Hill

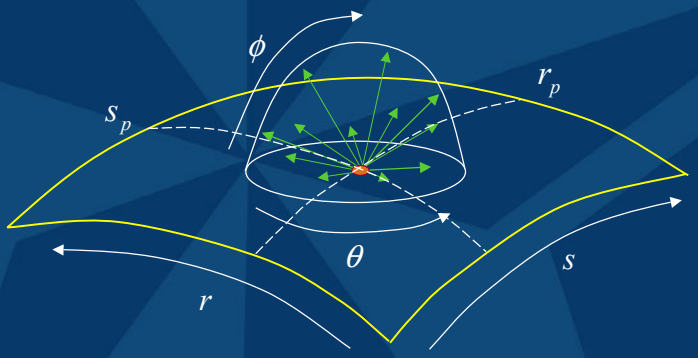
Alexei Smirnov
Intel Corporation

Sergei Molinov
Intel Corporation

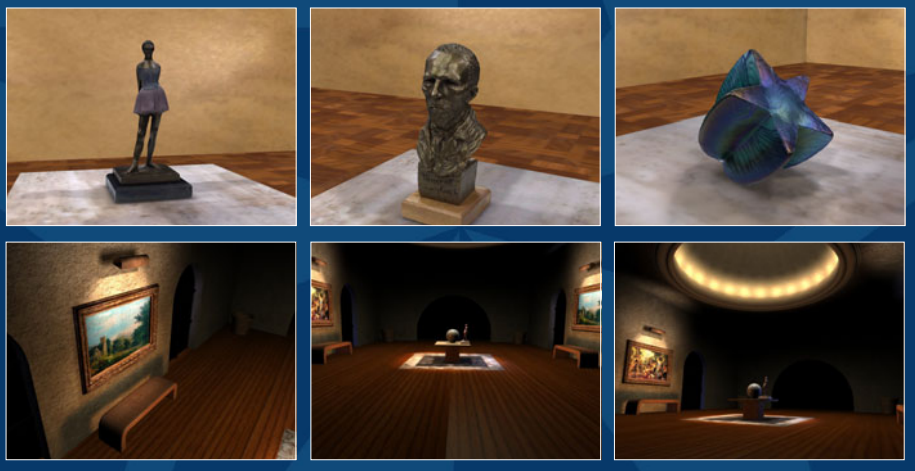
Image-based Modeling and Rendering



Parameterization of Surface Light Field



Examples of Surface Light Fields



Properties of Image-based Modeling and Rendering

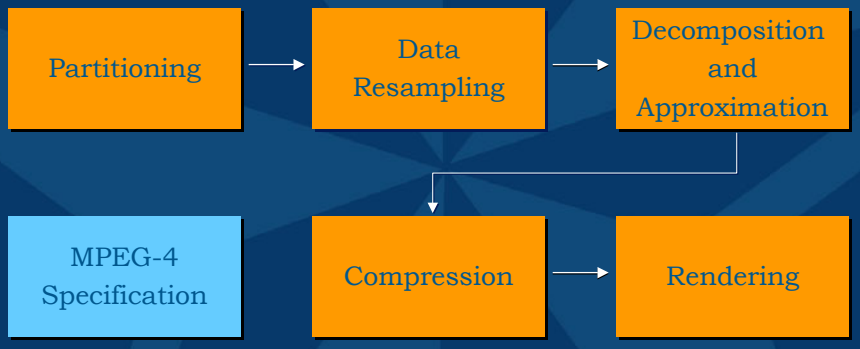
- **Positive**
  - Image-based representation offers photorealism
  - Data acquisition holds promise of simple modeling
- **Negative**
  - Poor rendering quality
  - Large datasets are difficult to represent and render
- **Does it have to be this way?**

Light Field Mapping Method

- **Efficient Representation**
  - Very high compression ratio (~10000:1)
  - Easy to implement
  - Compressed format easy to handle
- **Hardware Rendering**
  - On-the-fly decompression on hardware
  - Progressive support
  - Very efficient rendering performance
  - Easy to implement

Demo

Talk Overview



Partitioning of Surface Light Fields

- **Partition SLF across surface primitives P_i**

$$f^{P_i}(r,s,\theta,\varphi) = \Pi^{P_i}(r,s) f(r,s,\theta,\varphi)$$

- **Individual parts must add up to the original SLF**

$$f(r,s,\theta,\varphi) = \sum_i f^{P_i}(r,s,\theta,\varphi)$$

- **Partitioning must result in continuous approximation**

Resampling of Data

- **Acquired surface light field data is irregular**
  - Views of the same surface patch differ in size
  - Viewing directions are non-uniform
- **Resampling of data is a two step process**
  - Step 1 normalizes texture sizes
  - Step 2 resamples uniformly viewing direction
- **Dense sampling of viewing directions is critical for decomposition and rendering**

Decomposition and Approximation

- Decompose SLF for each P_i individually into sum of products of 2D functions

$$f^{P_i}(r,s,\theta,\varphi) = \sum_{k=1}^{N} g_k^{P_i}(r,s)\, h_k^{P_i}(\theta,\phi)$$

- Approximate SLF for each P_i by a small number of summation terms

$$f^{P_i}(r,s,\theta,\varphi) = \sum_{k=1}^{K} g_k^{P_i}(r,s)\, h_k^{P_i}(\theta,\phi),\ K \ll N$$

- Functions $g_k^{P_i}$ and $h_k^{P_i}$ are stored as 2D texture maps and called light field maps

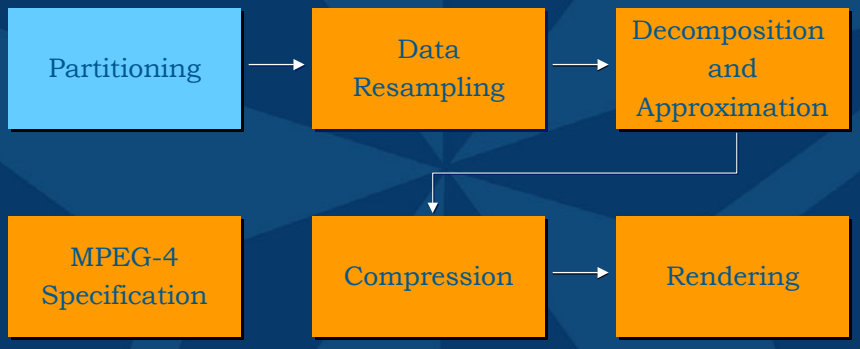
Compression and Rendering

- Approximation produces compact and accurate representation (100:1 compression)
- Light field maps can be further compressed (additional 100:1 compression)
- Rendering routine extremely simple
  - Uses multitexturing and blending only
- Represenation ideally suited for streaming data over the Internet
  - Data is compact, quality improves progressively

Related Work

- **Sample-based approx. of analytic reflectance models**
  - [Heidrich99,Kautz99]
- **Plenoptic modeling/Light field rendering/Lumigraph**
  - [McMillan95,Levoy96, Gortler96]
- **View-dependent texture mapping**
  - [Debevec96, Pulli97, Debevec98]
- **Eigen-texture approach**
  - [Nishino99]
- **Surface light fields for 3D photography**
  - [Wood00]

Talk Overview

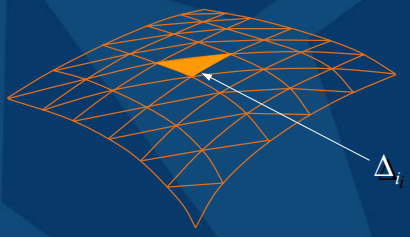


Triangle-centered Partitioning

- **Partition surface light field across triangles Δ_i**

$$f^{\Delta_i}(r,s,\theta,\varphi) = \Pi^{\Delta_i}(r,s) f(r,s,\theta,\varphi)$$

where $\Pi^{\Delta_i}(r,s) = \begin{cases} 1 & inside \quad \Delta_i \\ 0 & outside \quad \Delta_i \end{cases}$



Triangle-centered Partitioning

- **Partition surface light field across triangles Δ_i**

$$f^{\Delta_i}(r,s,\theta,\varphi) = \Pi^{\Delta_i}(r,s) f(r,s,\theta,\varphi)$$

where $\Pi^{\Delta_i}(r,s) = \begin{cases} 1 & inside \quad \Delta_i \\ 0 & outside \quad \Delta_i \end{cases}$

Discontinuity!

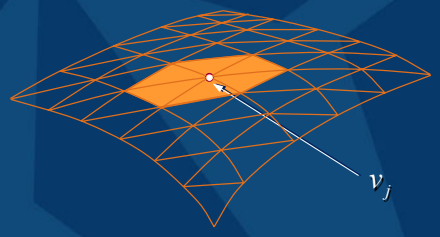
Vertex-centered Partitioning

- **Partition surface light field across triangle rings around each vertex v_j**

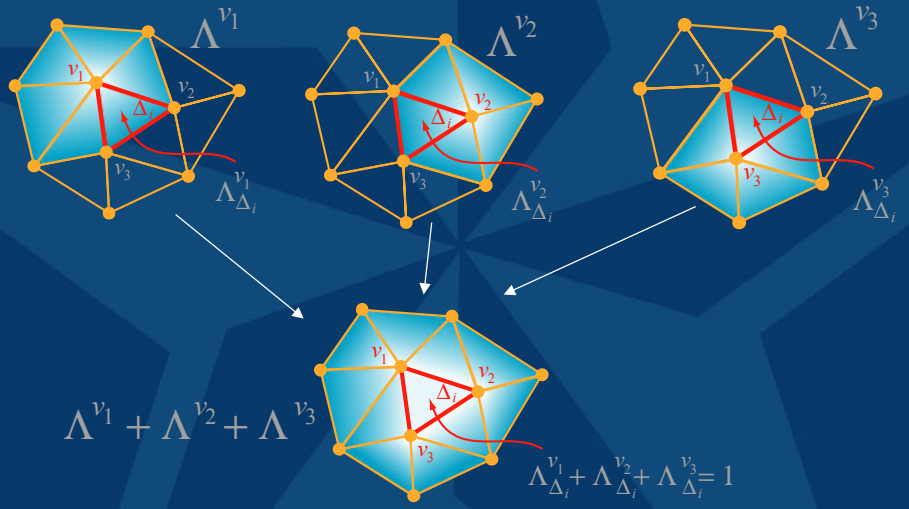
$$f^{v_j}(r,s,\theta,\varphi) = \Lambda^{v_j}(r,s) f(r,s,\theta,\varphi)$$

where $\Lambda^{v_j}(r,s) = \begin{cases} a & inside\ ring, 0 < a \le 1 \\ 0 & outside\ ring \end{cases}$

Hat Function



Partitioning with Hat Functions

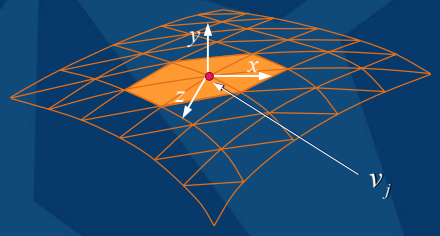


Vertex-centered Partitioning

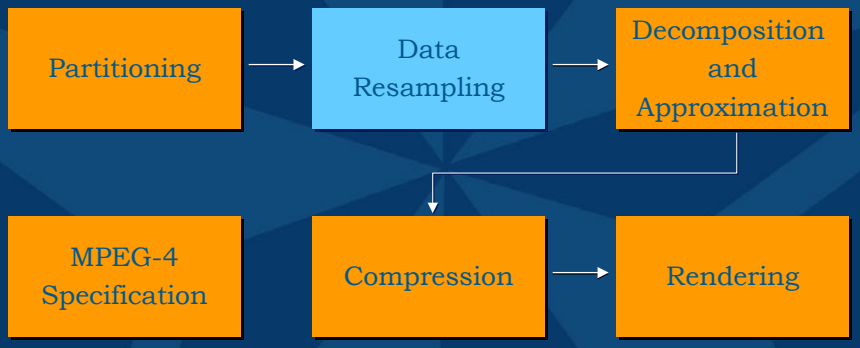
- Define local reference frame for vertex v_j
- Convert surface light field for vertex v_j to its local reference frame

$$f^{v_j}(r,s,\theta^{v_j},\varphi^{v_j}) \leftarrow f^{v_j}(r,s,\theta,\varphi)$$

Vertex Light Field



Talk Overview



Data Acquisition

- **Our Custom-built 3D Scanner:**
  - Geometry scanned with structured-light
  - 200-400 image captured with hand-held camera
  - Images registered to geometry
  - Precise & inexpensive
- **Surface Light Fields of synthetic objects can be generated directly.**

Visibility Computation and Data Resampling

- **Visibility computation determines un-occluded views for each triangle**
- **Resampling consists of two steps:**
- **Normalization of texture sizes**
  - Produces triangle views of uniform size
- **Resampling of viewing directions**
  - Produces uniform sampling of viewing directions

Resampling Step 1:
Normalization of Texture Sizes

- Normalizes number of samples representing each view of triangle Δ_i

$$\mathbf{I}^{\Delta_i} = [\, \mathbf{I}_1^{\Delta_i} \; \mathbf{I}_2^{\Delta_i} \; \ldots \mathbf{I}_{V_i}^{\Delta_i} \,]$$

1^{st} *view* 2^{nd} *view* V_i*-th view*

- Each matrix column corresponds to a different view
- Each matrix row corresponds to all visible views of one surface sample

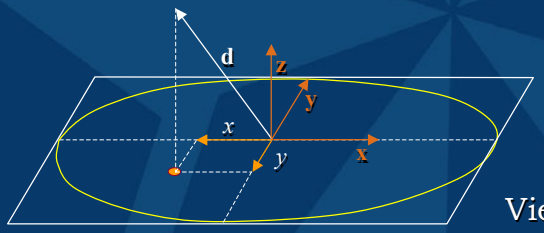
Why Resample Viewing Directions?

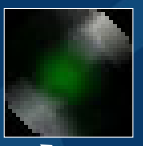
- Simplifies synthesis of novel views
- Enables per-sample interpolation of viewing directions when rendering a triangle
- Results in representation suitable for hardware-accelerated implementation
- Results in approximation with minimal root mean square error
- See the course notes for more details

Parameterization of Viewing Directions

- **Orthographic projection of d on *xy*-plane**
- **Simple and accurate parameterization**

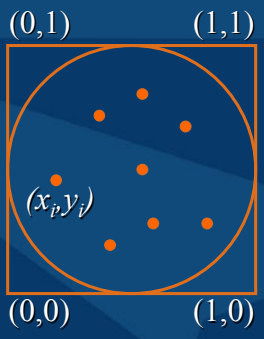
$$x = (\mathbf{d}\,\mathbf{x} + 1)/2, \qquad \mathbf{x} = [1\ 0\ 0]$$
$$y = (\mathbf{d}\,\mathbf{y} + 1)/2, \qquad \mathbf{y} = [0\ 1\ 0]$$

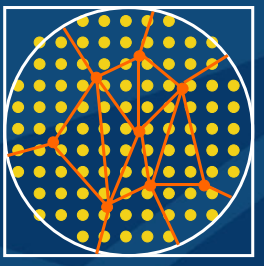




Viewing directions represented by 2D map

Resampling Step 2: Viewing Directions





- project viewing directions for visible triangle views $\mathbf{d}_i$
- compute Delaunay triangulation of original views
- compute regular grid of views by blending original views

Resampling Step 2: Viewing Directions

- **Radiance data after resampling of texture sizes**

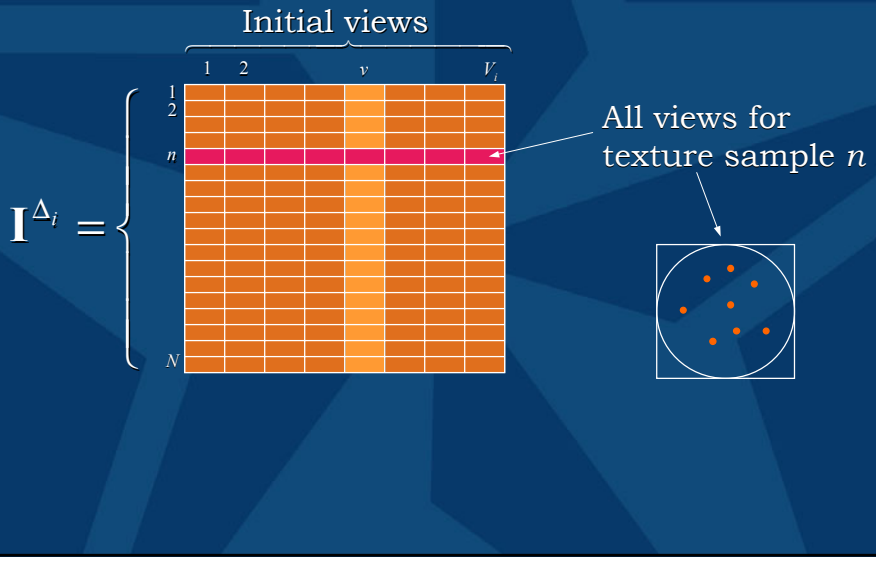
$$\mathbf{I}^{\Delta_i} = [\mathbf{I}_1^{\Delta_i}\ \mathbf{I}_2^{\Delta_i} \ldots \mathbf{I}_{V_i}^{\Delta_i}]$$

- **Radiance data after resampling of texture sizes and viewing directions**

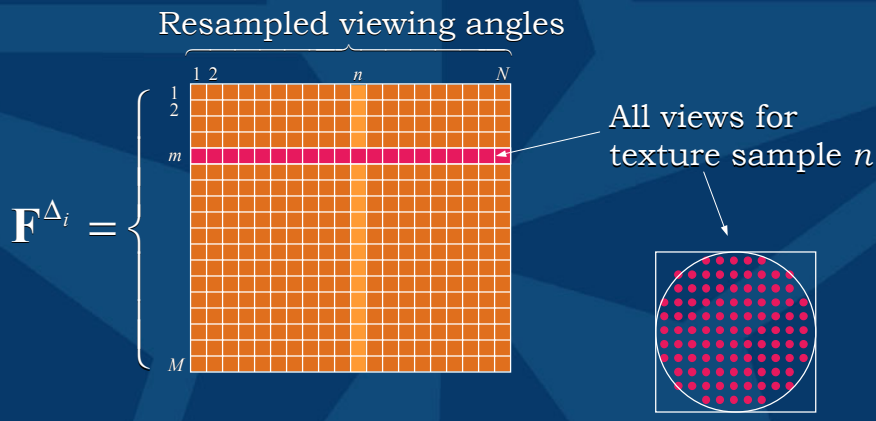
$$\mathbf{F}^{\Delta_i} = [\mathbf{F}_1^{\Delta_i}\ \mathbf{F}_2^{\Delta_i} \ldots \mathbf{F}_N^{\Delta_i}],\ N >> V_i$$

- ***N* views densely sample parameters θ and ϕ**

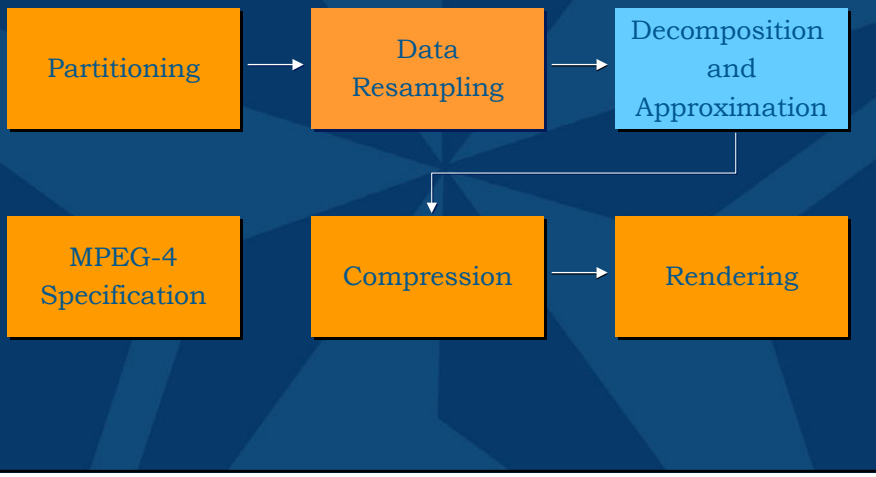
Resampling Step 1: Normalization of Texture Sizes



Resampling Step 2:
Viewing Directions



Talk Overview



Surface Light Field Decomposition

- **Consider surface light field for element P_i**
  - can be either triangle or vertex light fields

$$f^{P_i}(r_p, s_p, \theta_q, \varphi_q)$$

- **Resample 4-dimensional f^{P_i} into *M* x *N* matrix**

$$\mathbf{F}^{P_i} = [\, \mathbf{f}_1^{P_i} \; \mathbf{f}_2^{P_i} \cdots \mathbf{f}_N^{P_i} \,]$$

- **Decompose $\mathbf{F}^{P_i}$ using matrix factorization**

$$\mathbf{F}^{P_i} = \sum_{k=1}^{N} \sigma_k \mathbf{u}_k \mathbf{v}_k = \sum_{k=1}^{N} \mathbf{u}'_k \mathbf{v}'_k$$

Surface Light Field Approximation

- **Rearrange vectors $(\mathbf{u}'_k, \mathbf{v}'_k)$ into 2D textures**

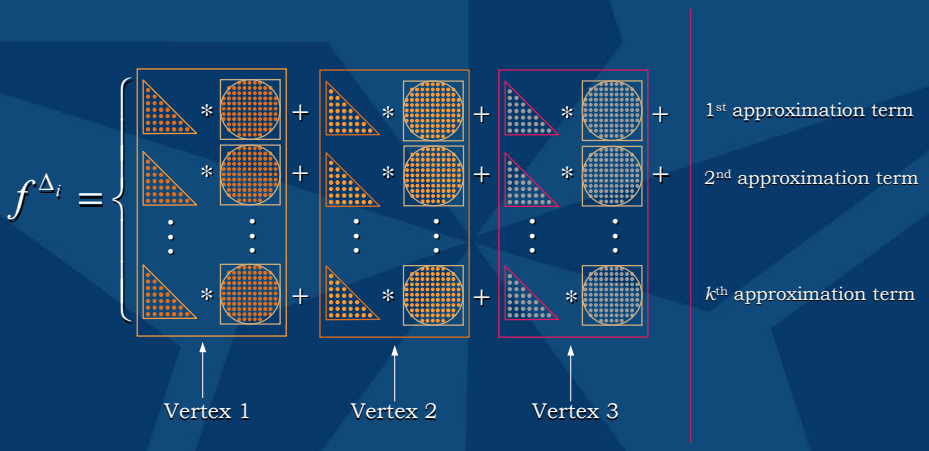
$$f^{P_i}(r_p, s_p, \theta_q, \varphi_q) = \sum_{k=1}^{N} g_k^{P_i}(r_p, s_p)\, h_k^{P_i}(\theta_q, \phi_q)$$

- **Truncate sum after *K* terms (typically, *K*=3)**

$$f^{P_i}(r_p, s_p, \theta_q, \varphi_q) \approx \sum_{k=1}^{K} g_k^{P_i}(r_p, s_p)\, h_k^{P_i}(\theta_q, \phi_q)$$

Surface maps View maps

Vertex-centered Approximation



1 approximation term = 3 surface maps + 3 view maps

Decomposition Algorithms

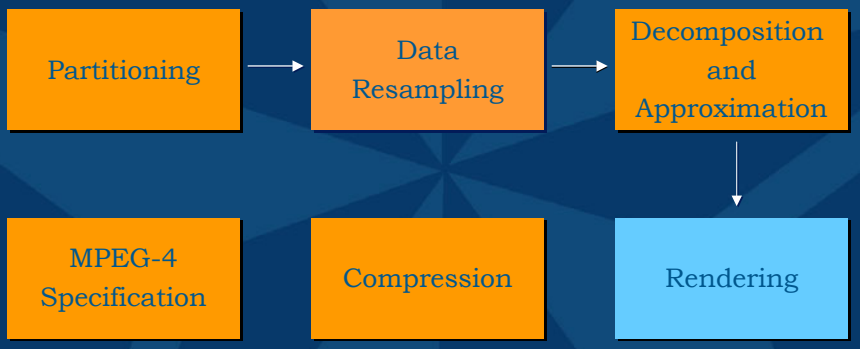
- Principal Component Analysis (PCA)

- Rotates matrix using matrix eigenvectors
- Uses subspace spanned by principal eigenvectors
- RMS Optimal
- Values can have arbitrary sign

- Non-Negative Matrix Factorization (NMF)

- Computes parts-based matrix approximation
- Finds only local optimal solution
- Only positive values allowed
- Easier to render on commodity hardware

Talk Overview

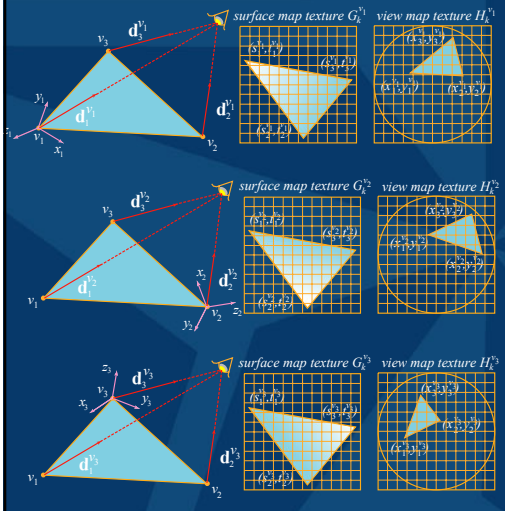


Rendering Algorithm

- **For each triangle Δ_k :**
- **For each approximation term k :**
- **For each vertex v_j in (v_1, v_2, v_3) :**
  - compute texture coordinates for light maps
  - texture map Δ_k using surface map $g_k^{v_j}$
  - texture map Δ_k using view map $h_k^{v_j}$
  - perform pixel-by-pixel multiplication of 2 images

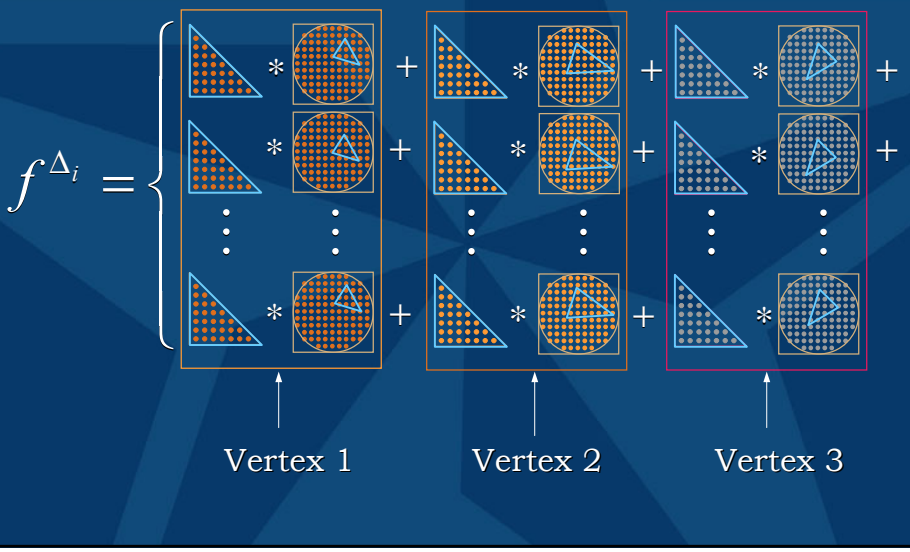
- Completely Hardware-Accelerated

Computation of Texture Coordinates



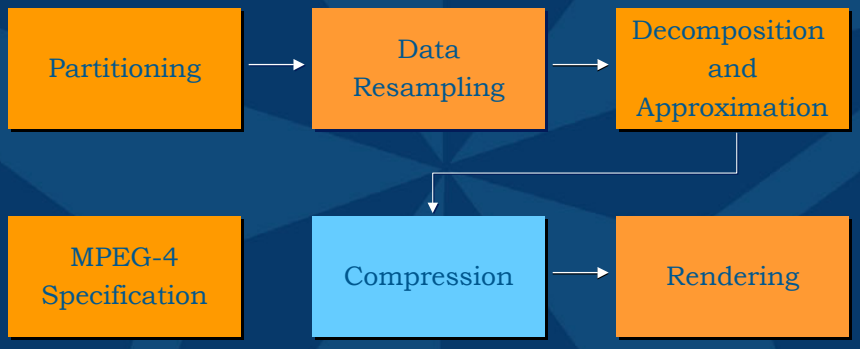
- Each approximation term has 3 pairs of light field maps
- Each pair has its own coordinate system
- View map texcoords differ between pairs
- Surface map texcoords are the same for each pair

Rendering Algorithm



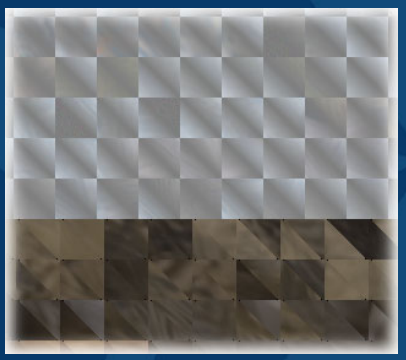
Demo

Talk Overview

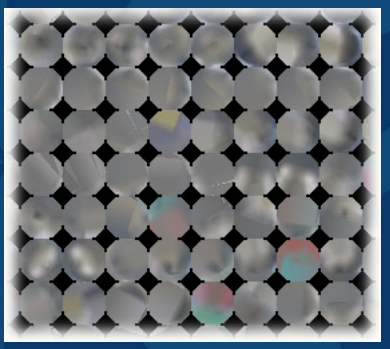


Compression of Light Field Maps

Light field maps are redundant

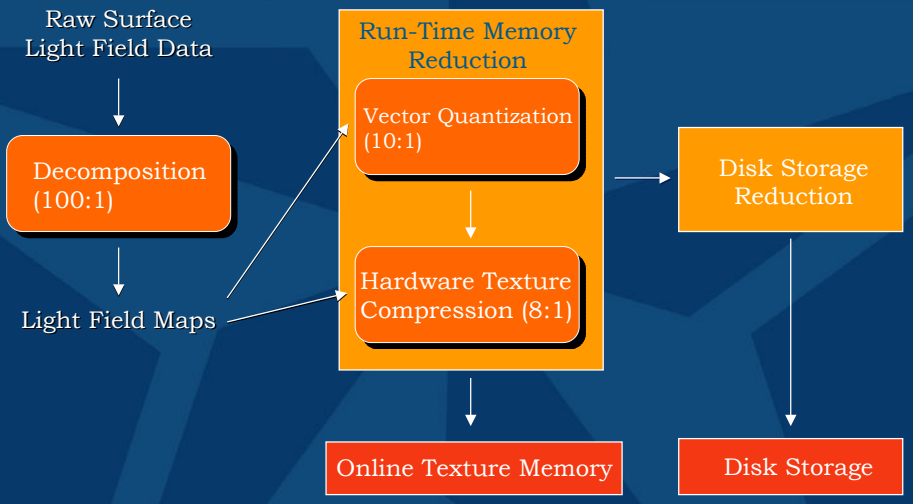


Tiled Surface Maps



Tiled View Maps

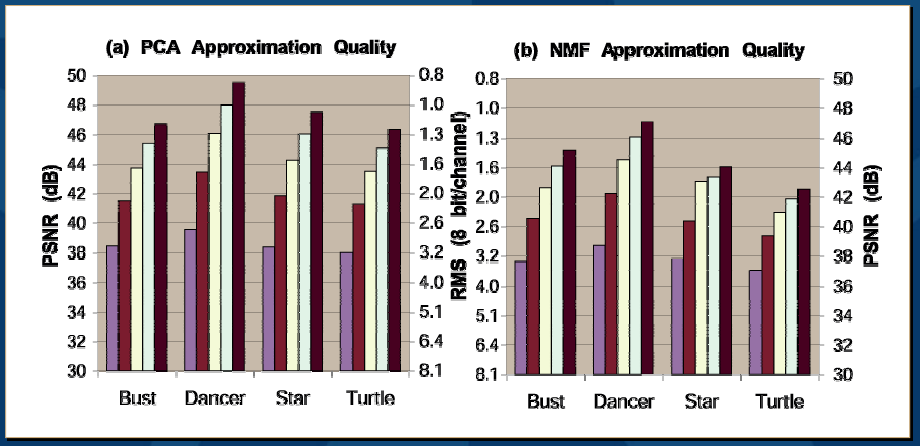
Compression Pipeline



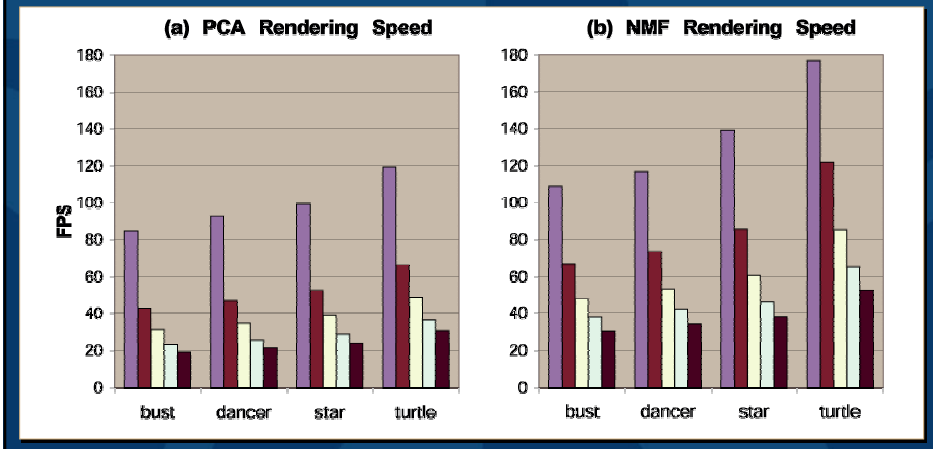
Results: Compression

Models	Raw Light Field Data	Light Field Maps	Compression of Light Fields		
			VQ	S3TC	VQ+S3TC
Bust	5.1GB	48MB (106:1)	5.7MB (885:1)	8.0MB (636:1:1)	951KB (5475:1)
Dancer	2.9GB	36MB (82:1)	5.4MB (542:1)	5.9MB (490:1)	904KB (3303:1)
Star	5.2GB	42MB (122:1)	7.2MB (716:1)	7.0MB (737:1)	1.2MB (4301:1)
Turtle	3.8GB	32MB (121:1)	4.5MB (847:1)	5.3MB (726:1)	683KB (5084:1)

Results: Approximation Quality

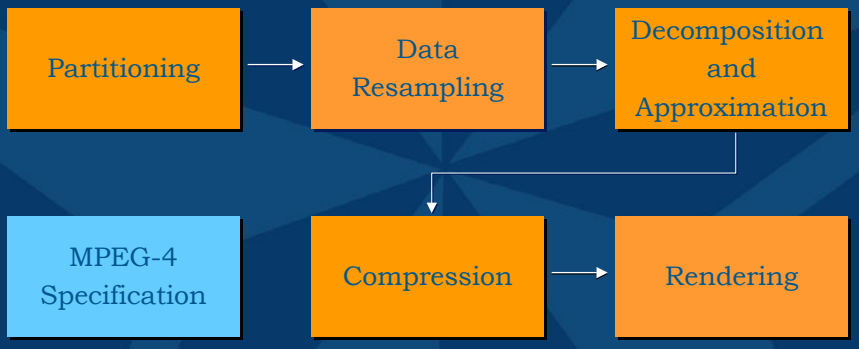


Results: Rendering Speed



* On a 2GHz P4 PC with nVidia GeForce3

Talk Overview



Root Node

The root node that describes the Light Field is LFM_Appearance. This node should be present in the appearance field of a Shape node, while its geometry field should contain an IndexedFaceSet.

```
LFM_Appearance {
        exposedField    MFNode          tileList        []
        exposedField    MFNode          lightMapList    []
        exposedField    SFNode          blendList       NULL
        exposedField    LFM_FrameList   vertexFrameList NULL
}
```

Vertex reference frames can be either computed automatically or can be specified explicitly through **LFM_FrameList** node.

```
LFM_FrameList {
        exposedField    MFInt32         index           [ -1 ]
        exposedField    MFVec3f         frame           [ 1 0 0, 0 1 0, 0 0 1 ]
}
```

Light Map Node

The field **lightMapList,** used in the definition of the **LFM_Appearance** node, describes how to access individual light maps from within the image tiles.

```
LFM_LightMap {
        exposedField    SFNode          surfaceMapList  NULL
        exposedField    SFNode          viewMapList     NULL
        exposedField    SFVec3f         scaleRGB        1 1 1
        exposedField    SFVec3f         biasRGB         0 0 0
        exposedField    SFInt32         priorityLevel   0
}
```

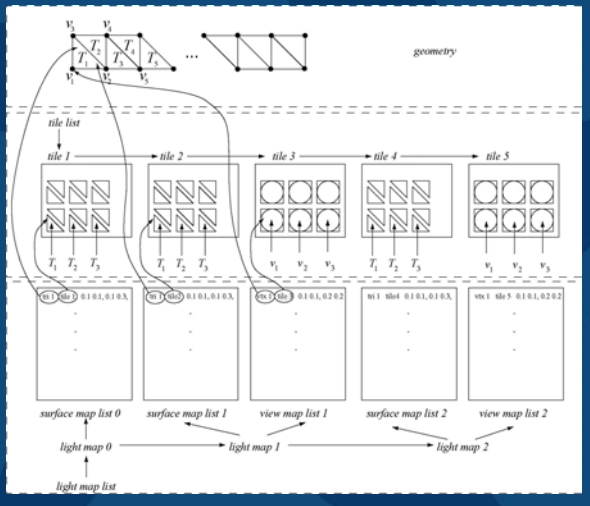
Surface/View Map Nodes

The lists **surfaceMapList** and **viewMapList** containing information about the surface maps and the view maps are defined as follows

```
LFM_SurfaceMapList {
  exposedField    MFInt32    triangleIndex       []
  exposedField    MFInt32    tileIndex           []
  exposedField    MFInt32    viewMapIndex        []
  exposedField    SFNode     triangleCoordinate  NULL
}

LFM_ViewMapList {
  exposedField    MFInt32    vertexIndex         []
  exposedField    MFInt32    tileIndex           []
  exposedField    SFNode     textureOrigin       NULL
  exposedField    SFNode     textureSize         NULL
}
```

Diagram of Nodes



Blend Node

The field **blendList** describes how to combine the individual light maps together during rendering.

It may contain a **LFM_blendList** node.

```
LFM_blendList {
exposedField        MFInt32        lightMapIndex
    exposedField    MFInt32        blendMode
}
```

where **lightMapIndex** refers to the index of the lightMap in **lightMapList** and **tileIndex** refers to the index of the tile as defined by the list of tiles and the field **blendMode** specifies what type of blending operation to use when combining the given **LightMapList** node with the data in the framebuffer. This field can take on the following values: LFM_ADD or LFM_SUBTRACT.

MPEG-4 Face and Body Animation Tools and Applications

Igor Pandzic, PhD
Department of Electrical Engineering
Linkoping University
SWEDEN
http://www.bk.isy.liu.se/staff/igor
igor@isy.liu.se

Dr. Eric Petajan
Chief Scientist and CoFounder
face2face animation, inc.
www.f2f-inc.com
eric@f2f-inc.com

What MPEG-4 enables

- Human-centered communication
  - Mobile terminal/thin client
  - Standards drive costs down
  - low bandwidth (wireless/dialup)
  - Rich interactivity with content and each other
- Boosts related standards
  - Web3D/VRML/H-Anim
- Efficient content repurposing
- Worldwide content exchange

MPEG-4 Object Composition

- Objects are organized in a scene graph
- Scene graphs are specified using a binary format called BIFS (based on VRML)
- Both 2D and 3D objects, properties and transforms are specified in BIFS
- BIFS allows objects to be transmitted once and instanced repeatedly in the scene after transformations

MPEG-4 Face and Body Animation Coding

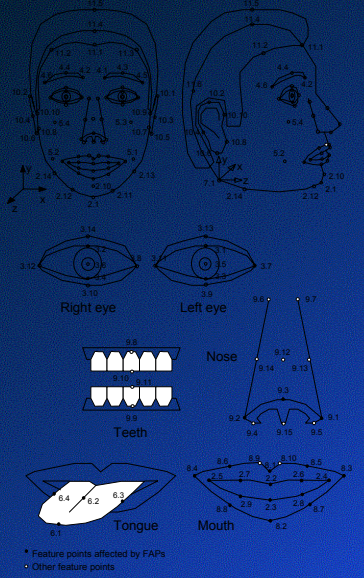
- Face animation is in MPEG-4 version 1
- Body animation is in MPEG-4 version 2
- Face animation parameters displace feature points from neutral position
- Body animation parameters are joint angles
- Face and body animation parameter sequences are compressed to low bitrates

Neutral Face Definition

- Head axes parallel to the world axes
- Gaze is in direction of Z axis
- Eyelids tangent to the iris
- Pupil diameter is one third of iris diameter
- Mouth is closed and the upper and lower teeth are touching
- Tongue is flat, horizontal with the tip of tongue touching the boundary between upper and lower teeth

face2face

Face Feature Points



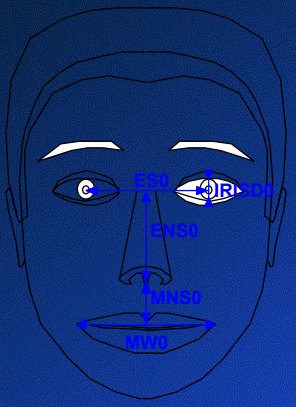
face2face

Face Animation Parameter Normalization

- Face Animation Parameters (FAPs) are normalized to facial dimensions
- Each FAP is measured as a fraction of neutral face mouth width, mouth-nose distance, eye separation, or iris diameter
- 3 Head and 2 eyeball rotation FAPs are Euler angles

face2face

Neutral Face Dimensions for FAP Normalization



face2face

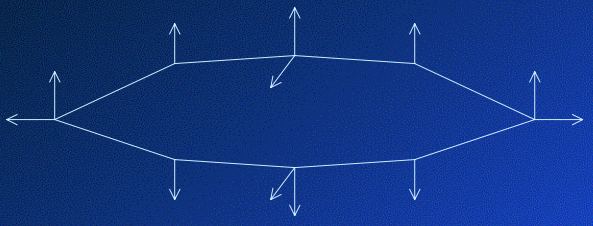
FAP Groups

Group	Number of FAPs
1: visemes and expressions	2
2: jaw, chin, inner lowerlip, cornerlips, midlip	16
3: eyeballs, pupils, eyelids	12
4: eyebrow	8
5: cheeks	4
6: tongue	5
7: head rotation	3
8: outer lip positions	10
9: nose	4
10: ears	4

face2face

Lip FAPs

Mouth closed if sum of upper and lower lip FAPs = 0



face2face

Face Model Independence

- FAPs are always normalized for model independence
- FAPs (and BAPs) can be used without MPEG-4 systems/BIFS
- Private face models can be accurately animated with FAPs
- Face models can be simple or complex depending on terminal resources

MPEG-4 BIFS Face Node

- Face node contains FAP node, Face scene graph, Face Definition Parameters (FDP), FIT,and FAT
- FIT (Face Interpolation Table) specifies interpolation of FAPs in terminal
- FAT (Face Animation Table) maps FAPs to Face model deformation
- FDP information included face feature points positions and texture map

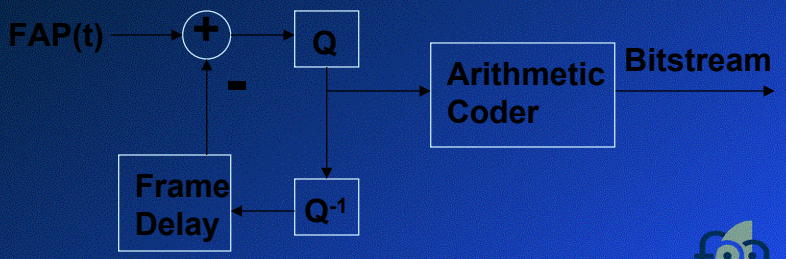
Face Model Download

- 3D graphical models (e.g. faces) can be downloaded to the terminal with MPEG-4
- 3D model specification is based on VRML
- Face Animation Table(FAT) maps FAPs to face model vertex displacements
- Appearance and animation of downloaded face models is exactly predictable

FAP Compression

- FAPs are adaptively quantized to desired quality level
- Quantized FAPs are differentially coded
- Adaptive arithmetic coding further reduces bitrate
- Typical compressed FAP bitrate is less than 2 kilobits/second

FAP Predictive Coding



face2face

Sources of FAPs

- Facial motion capture (with or without facial markers)
- Existing animation (if properly instrumented)
- Text-to-speech system

face2face

MPEG-4 Visemes and Expressions

- A weighted combination of 2 visemes and 2 facial expressions for each frame
- Decoder is free to interpret effect of visemes and expressions after FAPs are applied
- Definitions of visemes and expressions using FAPs can also be downloaded

face2face

Visemes

viseme_select	phonemes	example
0	none	na
1	p, b, m	put, bed, mill
2	f, v	far, voice
3	T,D	think, that
4	t, d	tip, doll
5	k, g	call, gas
6	tS, dZ, S	chair, join, she
7	s, z	sir, zeal
8	n, l	lot, not
9	r	red
10	A:	car
11	e	bed
12	I	tip
13	Q	top
14	U	book

face2face

Facial Expressions

expression_select	expression name	textual description
0	na	na
1	joy	The eyebrows are relaxed. The mouth is open and the mouth corners pulled back toward the ears.
2	sadness	The inner eyebrows are bent upward. The eyes are slightly closed. The mouth is relaxed.
3	anger	The inner eyebrows are pulled downward and together. The eyes are wide open. The lips are pressed against each other or opened to expose the teeth.
4	fear	The eyebrows are raised and pulled together. The inner eyebrows are bent upward. The eyes are tense and alert.
5	disgust	The eyebrows and eyelids are relaxed. The upper lip is raised and curled, often asymmetrically.
6	surprise	The eyebrows are raised. The upper eyelids are wide open, the lower relaxed. The jaw is opened.

Free Face Model Software

- Wireface is an openGL-based, MPEG-4 compliant face model
- Good starting point for building high quality face models for web applications
- Reads FAP file and raw audio file
- Renders face and audio in real time
- Wireface source is freely available

Computer-generated Face Animation Methods

- Morph targets/key frames (traditional)
- Speech articulation model (TTS)
- Facial Action Coding System (FACS)
- Physics-based (skin and muscle models)
- Marker-based (dots glued to face)
- Video-based (surface features)

Morph targets/key frames

- Advantages
  - Complete manual control of each frame
  - Good for exaggerated expressions
- Disadvantages
  - Hard to achieve good lipsync without manual tweeking
  - Morph targets must be downloaded to terminal for streaming animation (delay)

Speech articulation model

- Advantages
  - High level control of face
  - Enables TTS
- Disadvantages
  - Robotic character
  - Hard to sync with real voice

Facial Action Coding System

- Advantages
  - Very high level control of face
  - Maps to morph targets
  - Explicit specification of emotional states
- Disadvantages
  - Not good for speech
  - Not quantified

Physics-based

- Advantages
  - Good for realistic skin, muscle and fat
  - Collision detection
- Disadvantages
  - High complexity
  - Must be driven by high level articulation parameters (TTS)
  - Hard to drive with motion capture data

Marker-based

- Advantages
  - Can provide accurate motion data from most of the face
  - Face models can be animated directly from surface feature point motion
- Disadvantages
  - Dots glued to face
  - Dots must be manually registered
  - Not good for accurate inner lip contour or eyelid tracking

Video-based

- Advantages
  - Simple to capture video of face
  - Face models can be animated directly from surface feature motion
- Disadvantages
  - Must have good view of face

Facial Motion Capture

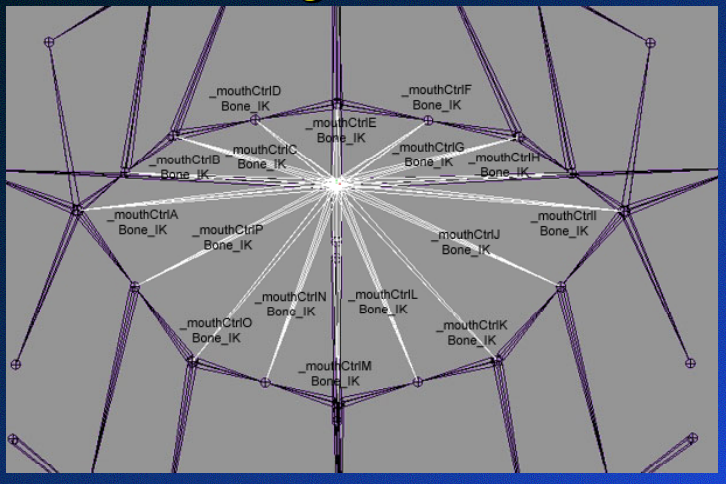
- Track/analyze points or features on face
- Captured face feature motion easily converted to FAPs
- Face model is “puppeteered” by FAPs
- MPEG-4 FAPs only specify motion of feature points (not surrounding surface)

Types of Puppeteered Face Models

- Facial bones rig
  - Arbitrarily complexity
- Fixed mesh deformation algorithm
  - Lightweight
- FAT table

face2face

Bones rig for mouth area



face2face

Body Animation

- Harmonized with VRML Hanim spec
- Body Animation Parameters (BAPs) are humanoid skeleton joint Euler angles
- Body Animation Table (BAT) can be downloaded to map BAPs to skin deformation
- BAPs can be highly compressed for streaming

Body Animation Parameters (BAPs)

- 186 humanoid skeleton euler angles
- 110 free parameters for use with downloaded body surface mesh
- Coded using same codecs as FAPs
- Typical bitrates for coded BAPs is 5-10kbps

Body Definition Parameters (BDPs)

- Humanoid joint center positions
- Names and hierarchy harmonized with VRML/Web3D H-Anim working group
- Default positions in standard for broadcast applications
- Download just BDPs to accurately animate unknown body model

Body Animation Table (BAT)

- Contains mapping between BAPs and deformation of downloaded body surface mesh
- 110 free BAPs can animate body surface features (fat, clothes,hair) independent of joint BAPs if BAT is downloaded with surface mesh
- Compressed for transmission

Sources of BAPs

- Body motion capture (optical, EM)
- Procedural animation (robotic)
- Text to sign language
- Single video to BAPs is a good research topic

MPEG-4 3D Model Coding Applied to FBA

- Face and body models are the most complex
- MPEG-4 provides static 3D mesh compression for neutral face and body surface
- FBA provides efficient animated (deformable) meshes for skin

MPEG-4 Applications

Bringing the right tools to the job

Faces Enhance the User Experience

- Virtual call center agents
- News readers (e.g. Ananova)
- Story tellers for the child in all of us
- eLearning
- Program guide
- Multilingual (same face different voice)
- Entertainment animation
- Multiplayer games

Visual Content for the Practical Internet

- Broadband deployment is happening slowly
- DSL availability is limited and cable is shared
- Talking heads need high frame-rate
- Consumer graphics hardware is cheap and powerful
- MPEG-4 SNHC/FBA tools are matched to available bandwidth and terminals

Visual Speech Processing

- FAPs can be used to improve speech recognition accuracy
- Text-to-speech systems can use FAPs to animate face models
- FAPs can be used in computer-human dialogue systems to communicate emotions, intentions and speech especially in noisy environments

Conclusions

- MPEG-4 is an international standard for true multimedia coding
- MPEG-4 provides very low bitrate and error resilience for internet and wireless applications
- MPEG-4 Face and body animation coding is a new type of content which is practical and entertaining
- For more information see
  - www.cselt.it/mpeg/
  - www.m4if.org
  - www.f2f-inc.com

Video-driven Face Animation

- Facial expressions, lip movements and head motion transferred to face model
- FAPs extracted from talking head video with special computer vision system
- No face markers or lipstick is required
- Normal lighting is used
- Communicates lip movements and facial expressions with visual anonymity

Automatic Face Animation Demonstration

- FAPs extracted from camcorder video
- Inner lip FAPs and head rotation FAPs compressed to less than 2 kbits/sec
- 30 frames/sec animation generated automatically
- Face models developed with face2face plugin for Softimage, 3DStudioMax, or Maya, or textured fixed mesh

QuickTime™ and a
decompressor
are needed to see this picture.

Protecting Your Assets

JC Spierer
MITRE Corp.

Course Overview

- **Intellectual property management and protection**
- **Digital rights management**
- **Security for content and delivery**

Issues in Security and Property Protection

- Who is the owner?
  - Name
  - Address
  - Verify who is the original owner
  - Verify how ownership can be transferred
    - One time view vs. movie sale (unlimited viewing)
    - Temporary viewing license (get the rental for 3 days then it self-destructs (automatic movie rental return)
- Level of rights for streaming media or media presentations
  - Internal
  - External
  - Web / External

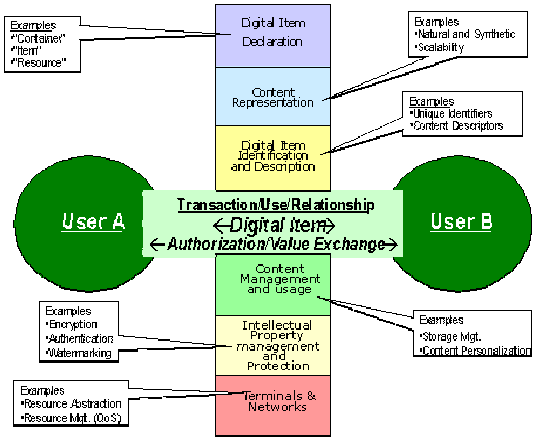
MPEG Standards Comparison

Standard	Bandwidth (Range)	Optimal Uses	Security Measures	Implementation Difficulty/Cost	Adoption Curve
MPEG-1	500Kb - 2 Mb	MP3 music, VCDs	External encryption	Well defined / low	Numerous Vendors
MPEG-2	2Mb - 15Mb	Digital cable, DVDs, wideband satellite downlink	External encryption	Well defined / medium-high	Numerous Vendors
MPEG-4	6Kb - 768Kb	Web media, mobility, interactivity	Watermarking, external encryption	Fairly well-defined / low-high	Few Vendors
MPEG-7	N/A	Metadata, media content description interface	?	Newly defined / low-high	Few - No Vendors
MPEG-21	Undefined	Multimedia architecture framework	IP rights / watermarking	Starting definition / ?	No Vendors

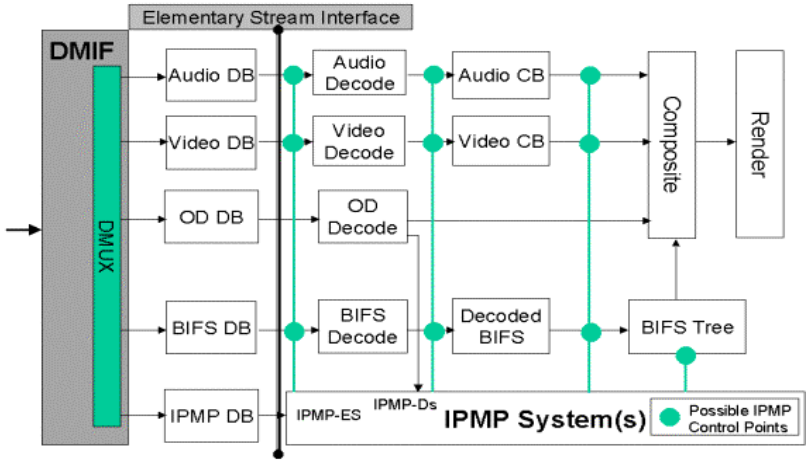
MPEG-21 Elements

- Element 1: Digital Item Declaration
- Element 2: Digital Item Identification and Description Framework
- Element 3: Content Handling and Usage
- Element 4: Intellectual Property Management/Protection
- Element 5: Terminals and Networks
- Element 6: Content Representation
- Element 7: Event Reporting

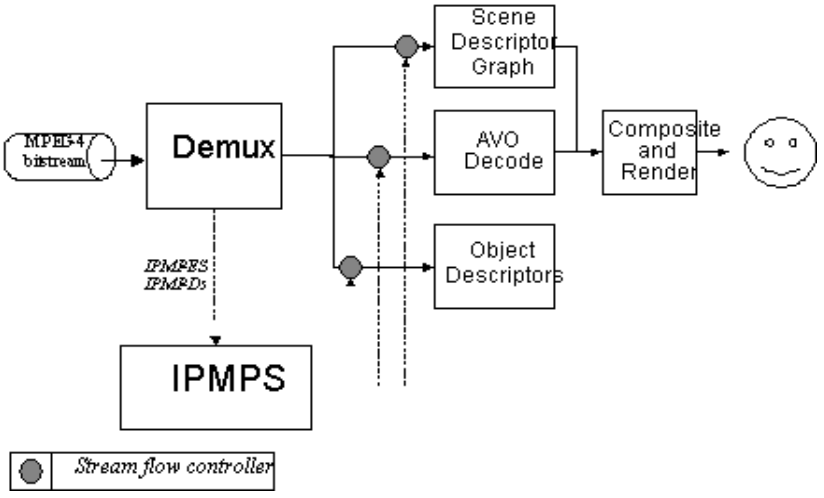
MPEG-21 Framework



Tight integration of IPMP and MPEG-4 Frameworks



MPEG-4 Intellectual Property (IPMP) (Source: Main Site Tutorial)



Basic Security Aspects

	Security Issue	Explanation	Example
1	Unauthorized Disclosure	Protect the data from being seen	Not allowing someone to watch the latest movie unless they paid for it
2	Data Integrity	Makes sure the data is what you think it is	Wrote/produced documentary; don't want someone to change part of it and yet claim that it's the original; adding mustache to the face example
3	Non-Repudiation	Closely associates creator/signature with content	Provided electronic signature on your taxes, an't go back 2 years later and say (repudiate) that you didn't sign it
4	Denial of Service	Signal goes bad or stops because someone hacked in	Movie stream stopped because someone hacked into the system - movie goes bad due to hacking, jamming, etc.

MPEG Media Security Elements

- Encryption - security technique involved in preventing many types of security issues
- Authentication - prevents unauthorized disclosure
- Watermarking - involved with data integrity, authentication and non-repudiation (signature)

Putting it all together – MPEG Profiles

Klaus Diepold
Munich University of Technology

Why Profiles ?

- Restriction of Syntax
- Restriction on used Tools
- Interoperability
  - Interworking of MPEG-4 systems supplied by different vendors
- Conformance testing

Wide spectrum of MPEG-4 devices

Problem

Various devices and applications

have different demands on the permissible complexity of the implementation of the MPEG-4 standard

Solution

Segmentation of the standard into different

„MPEG-4 Profiles“ and „Levels“

with varying extent and stepped complexity.

Concept of Profiles and Levels

MPEG-4 Profiles

Represent defined subsets of all available tools

(qualitative)

MPEG-4 Level

Represent ranks within the profiles, which define the complexity

(quantitative)

Profiling Principles

- Object Type
  - Which tools can be used to define meaningful objects
- Audio & Visual Profiles
  - Defines which objects can be used to build a presentation/scene
- Level
  - For a given Profile, a level defines quantitative bounds on technical parameters in order to bound implementation complexity and cost
- Profiles apply to DECODER specifications

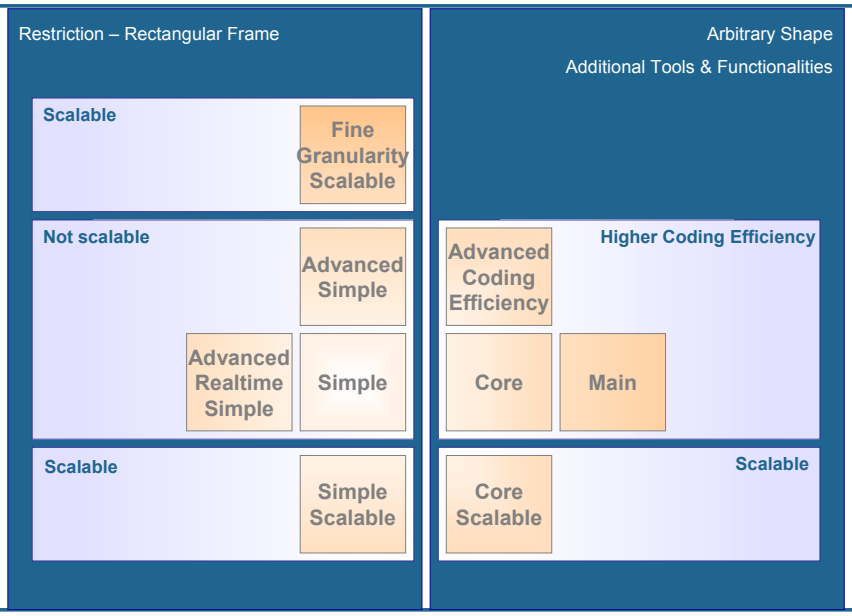
Profiling Principles

- MPEG-4's Conformance points are Profiles@Levels
  - A bit like in MPEG-2
- Profiles determine tool set
  - E.g. B frames, ¼ pel Motion Compensation
- Levels limit complexity
  - E.g. MBs/sec, max # objects, Complexity Units (Audio)
- Profiles will be convergence point for Industry Standards built on MPEG-4
  - They will be the vehicle for market decisions and uptake

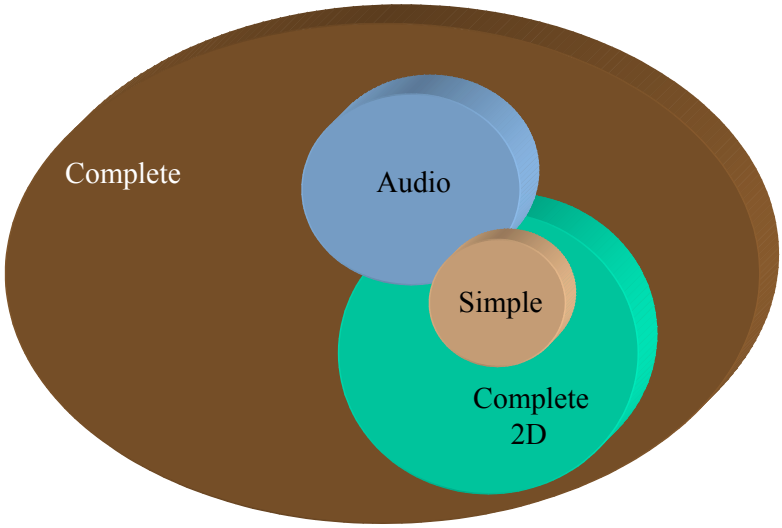
Profile Dimensions

- Visual (natural, synthetic, natural + synthetic)
- Audio (natural, synthetic, natural + synthetic)
- Graphics
- Scene Description (Scene Graph)
- Tools to describe an manipulate scene
- MPEG-J (Main and Personal)
- Object Descriptor (Synch and Buffers)
- MPEG does not prescribe how to combine these
- Media
- Profiles

Visual Profiles



BIFS Profiles



Graphics Profiles

- Simple 2-D Graphics Profile
  - only those graphics elements of the BIFS tool that are necessary to place one or more visual objects in a scene.
- Complete 2-D Graphics Profile
  - two-dimensional graphics functionalities and features such as arbitrary two-dimensional graphics and text;
- Complete Graphics Profile
  - provides advanced graphical elements such as elevation grids and extrusions and allows creating content with sophisticated lighting.

Graphics Profiles

- The *3D Audio* Graphics Profile
  - does not propose visual rendering, but graphics tools are provided to define the acoustical properties of the scene (geometry, acoustics absorption, diffusion, transparency of the material).

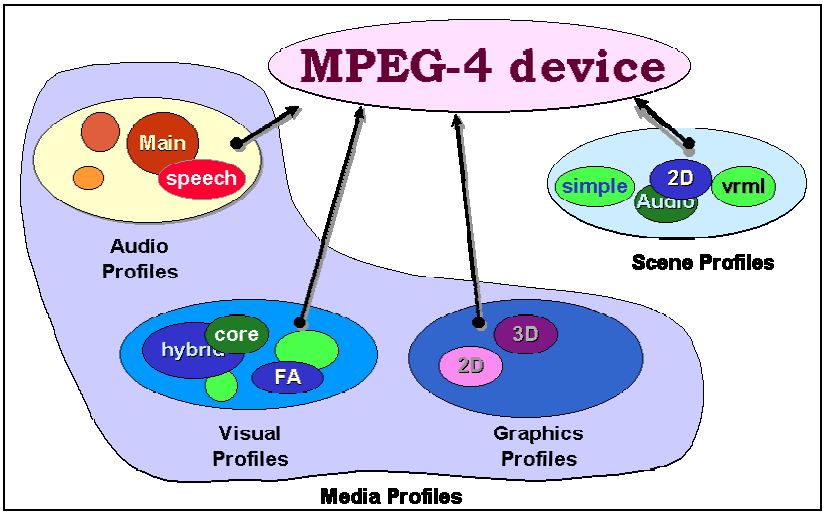
Scene Graph Profiles

- Audio Scene Graph Profile
  - BIFS scene graph elements for usage in audio only applications.
- The 3D Audio Scene Graph Profile
  - three-dimensional sound positioning in relation either with acoustic parameters of the scene or its perceptual attributes. The user can interact with the scene by changing the position of the sound source, by changing the room effect or moving the listening point.

Scene Graph Profiles (cont'd)

- The Simple 2-D Scene Graph Profile
  - presentation of audio-visual content with potential update of the complete scene but no interaction capabilities.
- The Complete 2-D Scene Graph Profile
  - Complete 2-D transformations and alpha blending.
- The Complete Scene Graph profile
  - complete scene graph elements BIFS; applications like dynamic virtual 3-D world and games.

Profiles



Proliferation of MPEG-4 in Multimedia Industry

Where is MPEG-4 being used?

Iraj Sodagar
PacketVideo

Introduction

- MPEG-4 provides extensive tools for multimedia coding and efficient representation
- Other standard bodies have provided some complementary tools to MPEG-4
  - IETF
- The meta standard bodies and forum has taken MPEG-4 to define new system level specifications and standards
  - 3GPP
  - 3GPP2
  - WMF
  - ISMA

IETF

- Objective: Internet Engineering Task Force
  - addressing transport/session protocols for streaming
- MPEG-4 relevant work:
  - RTP Payload formats for MPEG-4 content
    - RFC 3016: audio-video elementary stream payloads
    - Internet Draft: generic MPEG-4 payload formats
- IETF related work on streaming:
  - RFC1889, RTP: A Transport Protocol for Real-Time Applications
  - RFC1890, RTP Profile for Audio and Video Conferences with Minimal Control
  - RFC2326, Real Time Streaming Protocol (RTSP)

3GPP

- 3 Generation Partner Project
  - 3G wireless infrastructures and services for GSM and GPRS
- Packet Switch Streaming Services Spec:
  - Wireless terminal specification for video streaming
  - Mandates
    - MPEG-4 simple visual profile for video
    - MPEG-4 file formant for multimedia messaging
    - RTP, RTSP for streaming protocols and control
- IMTC has completed phase 1 of interoperability tests on 3GPP Release 4 spec.

3GPP2

- 3 Generation Partner Project 2
  - 3G wireless infrastructures and services (CDMA)
- Multimedia Streaming Services Spec:
  - Wireless terminal specification for video streaming
  - Mandates
    - MPEG-4 simple visual profile for video
    - MPEG-4 simple scalable profile as optional
    - RTP, RTSP for streaming protocols and control

3G 3rd Generation Partnership Project 2 "3GPP2"

WMF

- Wireless Multimedia Forum
  - Recommends video streaming protocols for wireless
- Recommended Technical Framework:
  - specification for multimedia content creation, streaming servers and terminals
  - Recommends
    - MPEG-4 simple visual profile for video
    - MPEG-4 simple scalable profile as optional
    - MPEG-4 file format for content creation and playback
    - RTP, RTSP for streaming protocols and cor
- Completed Interoperability tests

WIRELESS MULTIMEDIA FORUM

ISMA

- Internet Streaming Multimedia Alliance
  - Recommends streaming protocols for internet

- ISMA Specification version 1.0:
  - MPEG-4 simple visual profile for video
  - MPEG-4 advanced simple profile for video
  - MPEG-4 CLEP for speech coding
  - MPEG-4 AAC for audio coding
  - MPEG-4 file format for file storage
  - RTP, RTSP for streaming protocols and control

M4IF

- MPEG-4 Industry forum
  - to educate, facilitate the deployment of MPEG-4.
- Accomplishments:
  - Facilitated formation of patent pools
    - MPEG-4 simple and core visual profiles
    - MPEG-4 AAC
  - Interoperability tests
    - MPEG-4 video

MPEG-4 Proliferation Summary

	WMF	3GPP	3GPP2	ISMA
Domain	Wireless Streaming (overall)	Wireless Streaming (GSM and GPRS)	Wireless Streaming (CDMA)	Internet Streaming
Current Status	V 1.2	Release 4 and 5 Working on release 6	Release B	V1.0 Working on Version 2
Specification Scope	Terminal Server Content creation File format	Terminal only	Terminal Only	Terminal Server Content creation File format
Supported Technology video codec speech codec audio codec MM file storage format streaming: session desc/setup media control transport	MPEG-4 SP (M), H.263P3L10 (O) AMR (M)/EVRC (O) MP4FF (M) SDP RTSP RTP/RTCP	H.263 Baseline (M), MPEG-4 SP (O) and H.263P3L10(O) AMR (M), AAC (O) MP4FF (O) SDP RTSP RTP/RTCP SMIL 2.0 basic+ Profile /XHTML (O)	MPEG-4 SP (M), H.263P3L10 (O) EVRC (O), GSM-AMR (O), PureVoice (O), SMV (O) FF: not defined yet SDP RTSP RTP/RTCP	MPEG-4 SP (M), Core (O) CLEP/AAC (M) MP4 FF (M) SDP RTSP RTP/RTCP

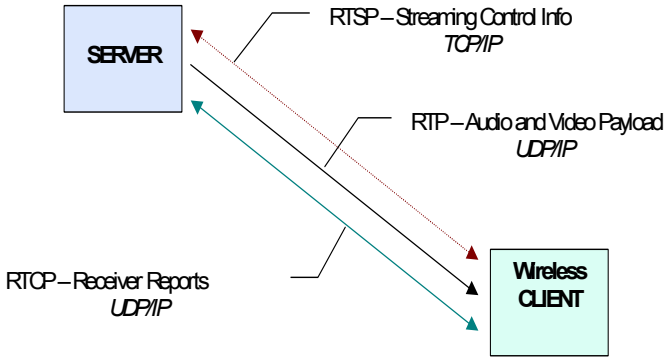
M= Mandatory, O =Optional

Demos

- Here goes the list of Demos:
  - 1
  - 2
  - MPEG-4 Streaming over wireless
  - 3
  - 4

Multimedia Streaming Over Wireless

- Streaming audio visual content using 3GPP/WMF recommendations:



Application Control Commands	MPEG-4 Video GSM-AMR Audio
RTSP	RTP/RTCP
TCP	UDP
IP	
Radio Link/Data Link	
Physical Layer	

INTERNATIONAL ORGANISATION FOR STANDARDISATION
ORGANISATION INTERNATIONALE DE NORMALISATION
ISO/IEC JTC1/SC29/WG11
CODING OF MOVING PICTURES AND AUDIO

ISO/IEC JTC1/SC29/WG11 N4030

March 2001

Source: **WG11 (MPEG)**
Status: **Final**
Title: **MPEG-4 Overview - (V.18 – Singapore Version)**
Editor: **Rob Koenen (rob@intertrust.com)**

Overview of the MPEG-4 Standard

Executive Overview

MPEG-4 is an ISO/IEC standard developed by MPEG (Moving Picture Experts Group), the committee that also developed the Emmy Award winning standards known as MPEG-1 and MPEG-2. These standards made interactive video on CD-ROM and Digital Television possible. MPEG-4 is the result of another international effort involving hundreds of researchers and engineers from all over the world. MPEG-4, whose formal ISO/IEC designation is ISO/IEC 14496, was finalized in October 1998 and became an International Standard in the first months of 1999. The fully backward compatible extensions under the title of MPEG-4 Version 2 were frozen at the end of 1999, to acquire the formal International Standard Status early in 2000. Some work, on extensions in specific domains, is still in progress.

MPEG-4 builds on the proven success of three fields:

- Digital television;
- Interactive graphics applications (synthetic content);
- Interactive multimedia (World Wide Web, distribution of and access to content)

MPEG-4 provides the standardized technological elements enabling the integration of the production, distribution and content access paradigms of the three fields.

More information about MPEG-4 can be found at MPEG's home page (case sensitive): http://mpeg.telecomitalialab.com/. This web page contains links to a wealth of information about MPEG, including much about MPEG-4, many publicly available documents, several lists of 'Frequently Asked Questions' and links to other MPEG-4 web pages. The standard can be bought from ISO, send mail to sales@iso.ch. Notably, the complete software for MPEG-4 version 1 can be bought on a CD ROM, for 56 Swiss Francs. It can also be downloaded for free from ISO's website: www.iso.ch/ittf - look under publicly available standards and then for "14496-5". This software is free of copyright restrictions when used for implementing MPEG-4 compliant technology. (This does not mean that the software is fee of patents. Also see section 7, The MPEG-4 Industry Forum.)

This document gives an overview of the MPEG-4 standard, explaining which pieces of technology it includes and what sort of applications are supported by this technology.

Table of Contents

1. Scope and features of the MPEG-4 standard

The MPEG-4 standard provides a set of technologies to satisfy the needs of authors, service providers and end users alike.

- For *authors*, MPEG-4 enables the production of content that has far greater reusability, has greater flexibility than is possible today with individual technologies such as digital television, animated graphics, World Wide Web (WWW) pages and their extensions. Also, it is now possible to better manage and protect content owner rights.
- For *network service providers* MPEG-4 offers transparent information, which can be interpreted and translated into the appropriate native signaling messages of each network with the help of relevant standards bodies. The foregoing, however, excludes Quality of Service considerations, for which MPEG-4 provides a generic QoS descriptor for different MPEG-4 media. The exact translations from the QoS parameters set for each media to the network QoS are beyond the scope of MPEG-4 and are left to network providers. Signaling of the MPEG-4 media QoS descriptors end-to-end enables transport optimization in heterogeneous networks.
- For *end users*, MPEG-4 brings higher levels of interaction with content, within the limits set by the author. It also brings multimedia to new networks, including those employing relatively low bitrate, and mobile ones. An MPEG-4 applications document exists on the MPEG Home page (www.cselt.it/mpeg), which describes many end user applications, including interactive multimedia broadcast and mobile communications.

For all parties involved, MPEG seeks to avoid a multitude of proprietary, non-interworking formats and players.

MPEG-4 achieves these goals by providing standardized ways to:

1. represent units of aural, visual or audiovisual content, called “media objects”. These media objects can be of natural or synthetic origin; this means they could be recorded with a camera or microphone, or generated with a computer;
2. describe the composition of these objects to create compound media objects that form audiovisual scenes;

3. multiplex and synchronize the data associated with media objects, so that they can be transported over network channels providing a QoS appropriate for the nature of the specific media objects; and
4. interact with the audiovisual scene generated at the receiver's end.

The following sections illustrate the MPEG-4 functionalities described above, using the audiovisual scene depicted in Figure 1.

1.1 Coded representation of media objects

MPEG-4 audiovisual scenes are composed of several media objects, organized in a hierarchical fashion. At the leaves of the hierarchy, we find primitive media objects, such as:

- still images (e.g. as a fixed background),
- video objects (e.g. a talking person - without the background)
- audio objects (e.g. the voice associated with that person);
- etc.

MPEG-4 standardizes a number of such primitive media objects, capable of representing both natural and synthetic content types, which can be either 2- or 3-dimensional. In addition to the media objects mentioned above and shown in Figure 1, MPEG-4 defines the coded representation of objects such as:

- text and graphics;
- talking synthetic heads and associated text used to synthesize the speech and animate the head;
- synthetic sound

A media object in its coded form consists of descriptive elements that allow handling the object in an audiovisual scene as well as of associated streaming data, if needed. It is important to note that in its coded form, each media object can be represented independent of its surroundings or background.
The coded representation of media objects is as efficient as possible while taking into account the desired functionalities. Examples of such functionalities are error robustness, easy extraction and editing of an object, or having an object available in a scaleable form.

1.2 Composition of media objects

Figure 1 explains the way in which an audiovisual scene in MPEG-4 is described as composed of individual objects. The figure contains compound media objects that group primitive media objects together. Primitive media objects correspond to leaves in the descriptive tree while compound media objects encompass entire sub-trees. As an example: the visual object corresponding to the talking person and the corresponding voice are tied together to form a new compound media object, containing both the aural and visual components of that talking person.

Such grouping allows authors to construct complex scenes, and enables consumers to manipulate meaningful (sets of) objects.

More generally, MPEG-4 provides a standardized way to describe a scene, allowing for example to:

- place media objects anywhere in a given coordinate system;
- apply transforms to change the geometrical or acoustical appearance of a media object;
- group primitive media objects in order to form compound media objects;
- apply streamed data to media objects, in order to modify their attributes (e.g. a sound, a moving texture belonging to an object; animation parameters driving a synthetic face);

- change, interactively, the user's viewing and listening points anywhere in the scene.

The scene description builds on several concepts from the Virtual Reality Modeling language (VRML) in terms of both its structure and the functionality of object composition nodes and extends it to fully enable the aforementioned features.

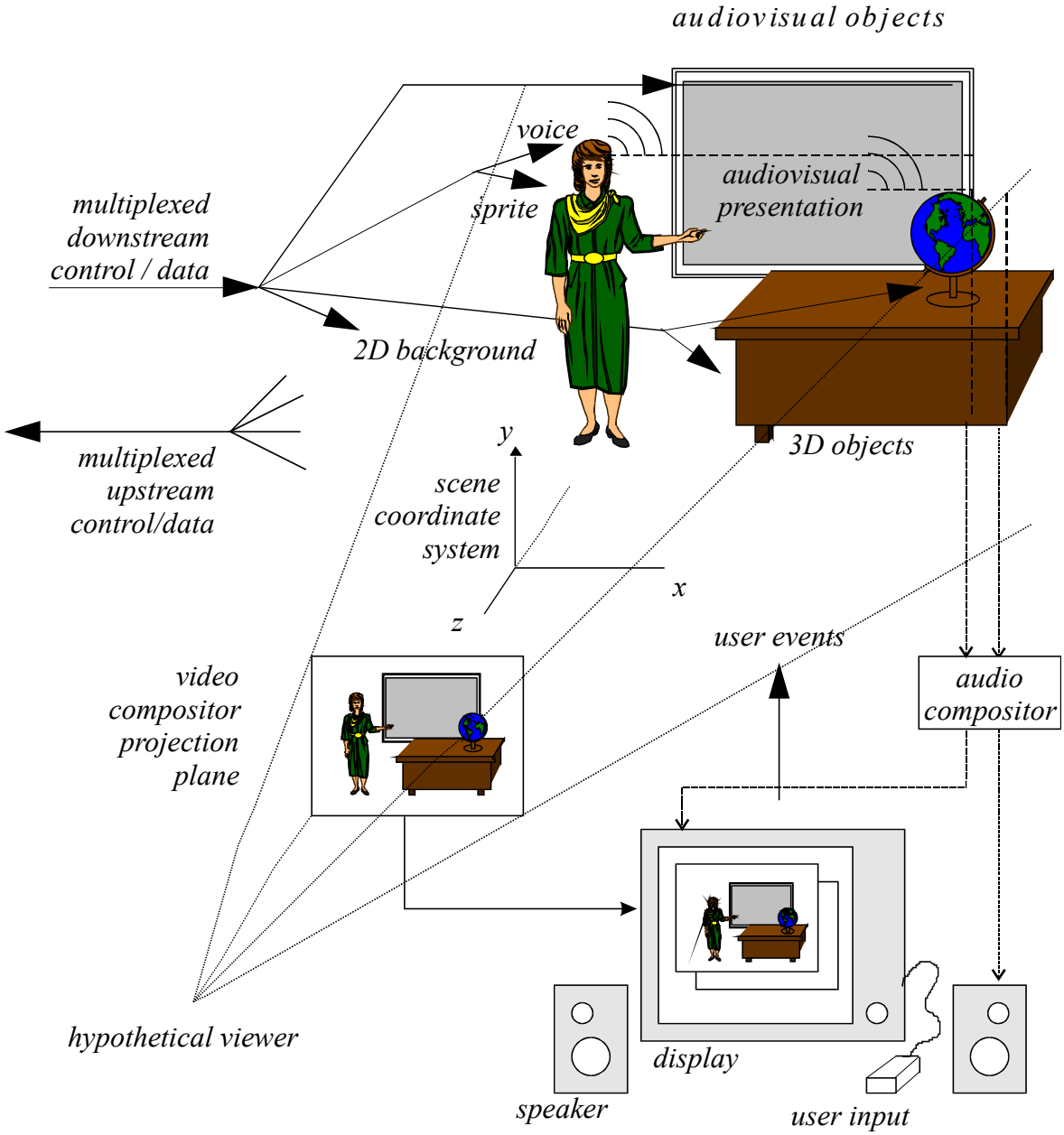


Figure 1 - an example of an MPEG-4 Scene

1.3 Description and synchronization of streaming data for media objects

Media objects may need streaming data, which is conveyed in one or more elementary streams. An object descriptor identifies all streams associated to one media object. This allows handling hierarchically encoded data as well as the association of meta-information about the content (called 'object content information') and the intellectual property rights associated with it. Each stream itself is characterized by a set of descriptors for configuration information, e.g., to determine the required decoder resources and the precision of encoded timing information.

Furthermore the descriptors may carry hints to the Quality of Service (QoS) it requests for transmission (e.g., maximum bit rate, bit error rate, priority, etc.)

Synchronization of elementary streams is achieved through time stamping of individual access units within elementary streams. The synchronization layer manages the identification of such access units and the time stamping. Independent of the media type, this layer allows identification of the type of access unit (e.g., video or audio frames, scene description commands) in elementary streams, recovery of the media object's or scene description's time base, and it enables synchronization among them. The syntax of this layer is configurable in a large number of ways, allowing use in a broad spectrum of systems.

1.4 Delivery of streaming data

The synchronized delivery of streaming information from source to destination, exploiting different QoS as available from the network, is specified in terms of the synchronization layer and a delivery layer containing a two-layer multiplexer, as depicted in Figure 2.
The first multiplexing layer is managed according to the DMIF specification, part 6 of the MPEG-4 standard. (DMIF stands for Delivery Multimedia Integration Framework) This multiplex may be embodied by the MPEG-defined FlexMux tool, which allows grouping of Elementary Streams (ESs) with a low multiplexing overhead. Multiplexing at this layer may be used, for example, to group ES with similar QoS requirements, reduce the number of network connections or the end to end delay.

The "TransMux" (Transport Multiplexing) layer in Figure 2 models the layer that offers transport services matching the requested QoS. Only the interface to this layer is specified by MPEG-4 while the concrete mapping of the data packets and control signaling must be done in collaboration with the bodies that have jurisdiction over the respective transport protocol. Any suitable existing transport protocol stack such as (RTP)/UDP/IP, (AAL5)/ATM, or MPEG-2's Transport Stream over a suitable link layer may become a specific TransMux instance. The choice is left to the end user/service provider, and allows MPEG-4 to be used in a wide variety of operation environments.

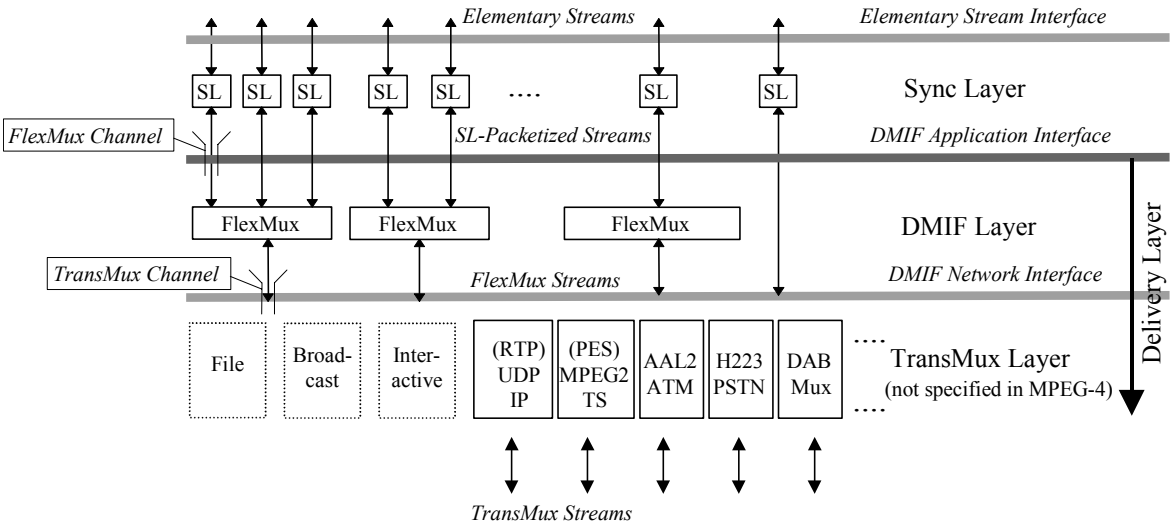


Figure 2 - The MPEG-4 System Layer Model

Use of the FlexMux multiplexing tool is optional and, as shown in Figure 2, this layer may be empty if the underlying TransMux instance provides all the required functionality. The synchronization layer, however, is always present.

With regard to Figure 2, it is possible to:

- identify access units, transport timestamps and clock reference information and identify data loss.
- optionally interleave data from different elementary streams into FlexMux streams
- convey control information to:
- indicate the required QoS for each elementary stream and FlexMux stream;
- translate such QoS requirements into actual network resources;
- associate elementary streams to media objects
- convey the mapping of elementary streams to FlexMux and TransMux channels

Parts of the control functionalities are available only in conjunction with a transport control entity like the DMIF framework.

1.5 Interaction with media objects

In general, the user observes a scene that is composed following the design of the scene's author. Depending on the degree of freedom allowed by the author, however, the user has the possibility to interact with the scene. Operations a user may be allowed to perform include:

- change the viewing/listening point of the scene, e.g. by navigation through a scene;
- drag objects in the scene to a different position;
- trigger a cascade of events by clicking on a specific object, e.g. starting or stopping a video stream ;
- select the desired language when multiple language tracks are available;

More complex kinds of behavior can also be triggered, e.g. a virtual phone rings, the user answers and a communication link is established.

1.6 Management and Identification of Intellectual Property

It is important to have the possibility to identify intellectual property in MPEG-4 media objects. Therefore, MPEG has worked with representatives of different creative industries in the definition of syntax and tools to support this. A full elaboration of the requirements for the identification of intellectual property can be found in 'Management and Protection of Intellectual Property in MPEG-4, which is publicly available from the MPEG home page.

MPEG-4 incorporates identification the intellectual property by storing unique identifiers, that are issued by international numbering systems (e.g. ISAN, ISRC, etc.1). These numbers can be applied to identify a current rights holder of a media object. Since not all content is identified by such a number, MPEG-4 Version 1 offers the possibility to identify intellectual property by a key-value pair (e.g.:»composer«/»John Smith«). Also, MPEG-4 offers a standardized interface that is integrated tightly into the Systems layer to people who want to use systems that control access to intellectual property. With this interface, proprietary control systems can be easily amalgamated with the standardized part of the decoder.

2. Major Functionalities in MPEG-4 Version 1

This section contains, in an itemized fashion, the major functionalities that the different parts of the MPEG-4 Standard offers in the finalized MPEG-4 Version 1. Description of the functionalities can be found in the following sections.

1 ISAN: International Audio-Visual Number, ISRC: International Standard Recording Code

2.1 DMIF

DMIF supports the following functionalities:

- A transparent MPEG-4 DMIF-application interface irrespective of whether the peer is a remote interactive peer, broadcast or local storage media.
- Control of the establishment of FlexMux channels
- Use of homogeneous networks between interactive peers: IP, ATM, mobile, PSTN, Narrowband ISDN.

2.2 Systems

As explained above, MPEG-4 defines a toolbox of advanced compression algorithms for audio and visual information. The data streams (Elementary Streams, ES) that result from the coding process can be transmitted or stored separately, and need to be composed so as to create the actual multimedia presentation at the receiver side.

The systems part of the MPEG-4 addresses the description of the relationship between the audio-visual components that constitute a scene. The relationship is described at two main levels.

- The Binary Format for Scenes (BIFS) describes the spatio-temporal arrangements of the objects in the scene. Viewers may have the possibility of interacting with the objects, e.g. by rearranging them on the scene or by changing their own point of view in a 3D virtual environment. The scene description provides a rich set of nodes for 2-D and 3-D composition operators and graphics primitives.
- At a lower level, Object Descriptors (ODs) define the relationship between the Elementary Streams pertinent to each object (e.g the audio and the video stream of a participant to a videoconference) ODs also provide additional information such as the URL needed to access the Elementary Steams, the characteristics of the decoders needed to parse them, intellectual property and others.

Other issues addressed by MPEG-4 Systems:

- Interactivity, including: client and server-based interaction; a general event model for triggering events or routing user actions; general event handling and routing between objects in the scene, upon user or scene triggered events.
- A tool for interleaving of multiple streams into a single stream, including timing information (FlexMux tool).
- A tool for storing MPEG-4 data in a file (the MPEG-4 File Format, 'MP4')
- Interfaces to various aspects of the terminal and networks, in the form of Java API's (MPEG-J)
- Transport layer independence. Mappings to relevant transport protocol stacks, like (RTP)/UDP/IP or MPEG-2 transport stream can be or are being defined jointly with the responsible standardization bodies.
- Text representation with international language support, font and font style selection, timing and synchronization.
- The initialization and continuous management of the receiving terminal's buffers.
- Timing identification, synchronization and recovery mechanisms.
- Datasets covering identification of Intellectual Property Rights relating to media objects.

2.3 Audio

MPEG-4 Audio facilitates a wide variety of applications which could range from intelligible speech to high quality multichannel audio, and from natural sounds to synthesized sounds. In particular, it supports the highly efficient representation of audio objects consisting of:

- *Speech signals*: Speech coding can be done using bitrates from 2 kbit/s up to 24 kbit/s using the speech coding tools. Lower bitrates, such as an average of 1.2 kbit/s, are also possible

when variable rate coding is allowed. Low delay is possible for communications applications. When using the HVXC tools, speed and pitch can be modified under user control during playback. If the CELP tools are used, a change of the playback speed can be achieved by using and additional tool for effects processing.

- *Synthesized Speech*: Scalable TTS coders bitrate range from 200 bit/s to 1.2 Kbit/s which allows a text, or a text with prosodic parameters (pitch contour, phoneme duration, and so on), as its inputs to generate intelligible synthetic speech. It includes the following functionalities.
  - Speech synthesis using the prosody of the original speech
  - Lip synchronization control with phoneme information.
  - Trick mode functionality: pause, resume, jump forward/backward.
  - International language and dialect support for text. (i.e. it can be signaled in the bitstream which language and dialect should be used)
  - International symbol support for phonemes.
  - support for specifying age, gender, speech rate of the speaker
  - support for conveying facial animation parameter(FAP) bookmarks.
- *General audio signals*: Support for coding general audio ranging from very low bitrates up to high quality is provided by transform coding techniques. With this functionality, a wide range of bitrates and bandwidths is covered. It starts at a bitrate of 6 kbit/s and a bandwidth below 4 kHz but also includes broadcast quality audio from mono up to multichannel.
- *Synthesized Audio*: Synthetic Audio support is provided by a Structured Audio Decoder implementation that allows the application of score-based control information to musical instruments described in a special language.
- *Bounded-complexity Synthetic Audio*: This is provided by a Structured Audio Decoder implementation that allows the processing of a standardized wavetable format.

Examples of additional functionality are speed control and pitch change for speech signals and scalability in terms of bitrate, bandwidth, error robustness, complexity, etc. as defined below.

- The *speed change* functionality allows the change of the time scale without altering the pitch during the decoding process. This can, for example, be used to implement a “fast forward” function (data base search) or to adapt the length of an audio sequence to a given video sequence, or for practicing dance steps at slower play back speed.
- The *pitch change* functionality allows the change of the pitch without altering the time scale during the encoding or decoding process. This can be used, for example, for voice alteration or Karaoke type applications. This technique only applies to parametric and structured audio coding methods.
- *Bitrate scalability* allows a bitstream to be parsed into a bitstream of lower bitrate such that the combination can still be decoded into a meaningful signal. The bitstream parsing can occur either during transmission or in the decoder.
- *Bandwidth scalability* is a particular case of bitrate scalability, whereby part of a bitstream representing a part of the frequency spectrum can be discarded during transmission or decoding.
- *Encoder complexity scalability* allows encoders of different complexity to generate valid and meaningful bitstreams.
- *Decoder complexity scalability* allows a given bitstream to be decoded by decoders of different levels of complexity. The audio quality, in general, is related to the complexity of the encoder and decoder used.
- *Audio Effects* provide the ability to process decoded audio signals with complete timing accuracy to achieve functions for mixing , reverberation, spatialization, etc.

2.4 Visual

The MPEG-4 Visual standard will allow the hybrid coding of natural (pixel based) images and video together with synthetic (computer generated) scenes. This will, for example, allow the virtual presence of videoconferencing participants. To this end, the Visual standard will comprise tools and algorithms supporting the coding of natural (pixel based) still images and video sequences as well as tools to support the compression of synthetic 2-D and 3-D graphic geometry parameters (i.e. compression of wire grid parameters, synthetic text).
The subsections below give an itemized overview of functionalities that the tools and algorithms of the MPEG-4 visual standard will support.

2.4.1 Formats Supported

The following formats and bitrates will be supported by MPEG-4 Version 1:

- bitrates: typically between 5 kbit/s and 10 Mbit/s
- Formats: progressive as well as interlaced video
- Resolutions: typically from sub-QCIF to beyond HDTV

2.4.2 Compression Efficiency

- Efficient compression of video will be supported for all bit rates addressed. This includes the compact coding of textures with a quality adjustable between "acceptable" for very high compression ratios up to "near lossless".
- Efficient compression of textures for texture mapping on 2-D and 3-D meshes.
- Random access of video to allow functionalities such as pause, fast forward and fast reverse of stored video.

2.4.3 Content-Based Functionalities

- *Content-based coding* of images and video to allow separate decoding and reconstruction of arbitrarily shaped video objects.
- *Random access* of content in video sequences to allow functionalities such as pause, fast forward and fast reverse of stored video objects.
- *Extended manipulation of content* in video sequences to allow functionalities such as warping of synthetic or natural text, textures, image and video overlays on reconstructed video content. An example is the mapping of text in front of a moving video object where the text moves coherently with the object.

2.4.4 Scalability of Textures, Images and Video

- *Complexity scalability in the encoder* allows encoders of different complexity to generate valid and meaningful bitstreams for a given texture, image or video.
- *Complexity scalability in the decoder* allows a given texture, image or video bitstream to be decoded by decoders of different levels of complexity. The reconstructed quality, in general, is related to the complexity of the decoder used. This may entail that less powerful decoders decode only a part of the bitstream.
- *Spatial scalability* allows decoders to decode a subset of the total bitstream generated by the encoder to reconstruct and display textures, images and video objects at reduced spatial resolution. For textures and still images, a maximum of 11 levels of spatial scalability will be supported. For video sequences, a maximum of three levels will be supported.
- *Temporal scalability* allows decoders to decode a subset of the total bitstream generated by the encoder to reconstruct and display video at reduced temporal resolution. A maximum of three levels will be supported.
- *Quality scalability* allows a bitstream to be parsed into a number of bitstream layers of different bitrate such that the combination of a subset of the layers can still be decoded into a

meaningful signal. The bitstream parsing can occur either during transmission or in the decoder. The reconstructed quality, in general, is related to the number of layers used for decoding and reconstruction.

2.4.5 Shape and Alpha Channel Coding

- *Shape coding* will be supported to assist the description and composition of conventional images and video as well as arbitrarily shaped video objects. Applications that benefit from binary shape maps with images are content based image representations for image data bases, interactive games, surveillance, and animation. Efficient techniques are provided that allow efficient coding of binary shape. A binary alpha map defines whether or not a pixel belongs to an object. It can be 'on' or 'off'.
- *'Gray Scale' or 'alpha' Shape Coding*
  An alpha plane defines the 'transparency' of an object, which is not necessarily uniform. Multilevel alpha maps are frequently used to blend different layers of image sequences. Other applications that benefit from associated binary alpha maps with images are content based image representations for image databases, interactive games, surveillance, and animation. Efficient techniques are provided, that allow efficient coding of binary as well as gray scale alpha planes. A binary alpha map defines whether or not a pixel belongs to an object. It can be 'on' or 'off'. A gray scale map offers the possibility to define the exact transparency of each pixel.

2.4.6 Robustness in Error Prone Environments

Error resilience will be supported to assist the access of image and video over a wide range of storage and transmission media. This includes the useful operation of image and video compression algorithms in error-prone environments at low bit-rates (i.e., less than 64 Kbps). Tools are provided which address both the band limited nature and error resiliency aspects for access over wireless networks.

2.4.7 Face Animation

The 'Face Animation' part of the standard allow sending parameters that calibrate and animate synthetic faces. These models themselves are not standardized by MPEG-4, only the parameters are.

- Definition and coding of face animation parameters (model independent):
  - Feature point positions and orientations to animate the face definition meshes
  - Visemes, or visual lip configurations equivalent to speech phonemes
- Definition and coding of face definition parameters (for model calibration):
  - 3-D feature point positions
  - 3-D head calibration meshes for animation
  - Texture map of face
  - Personal characteristics
- Facial texture coding

2.4.8 Coding of 2-D Meshes with Implicit Structure

- Mesh-based prediction and animated texture transfiguration
  - 2-D Delaunay or regular mesh formalism with motion tracking of animated objects
  - Motion prediction and suspended texture transmission with dynamic meshes.
- Geometry compression for motion vectors:
  - 2-D mesh compression with implicit structure & decoder reconstruction

3. Major Functionalities in MPEG-4 Version 2

Version 2 was frozen in December 1999. Existing tools and profiles from Version 1 are not replaced in Version 2; technology will be added to MPEG-4 in the form of new profiles. Figure 3 below depicts the relationship between the two versions. Version 2 is a backward compatible extension of Version 1.

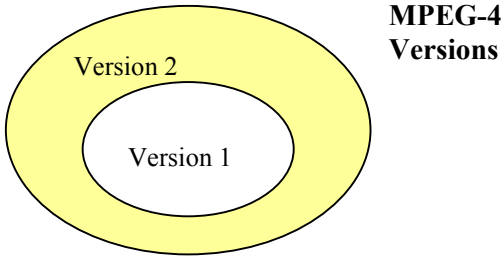


Figure 3 - relation between MPEG-4 Versions

Version 2 builds on Version 1 of MPEG-4. The Systems layer of Version 2 is backward compatible with Version 1. In the area of Audio and Visual, Version 2 will add Profiles to Version 1. The work on MPEG-4 does not stop after MPEG-4; more functionality will be added, albeit in particular, well-defined areas. The same principle applies, and new tools will find their way in the standard in the form of new Profiles. This means that existing systems will always remain compliant, because Profiles will not be changed in retrospect.

3.1 Systems

Version 2 of the MPEG-4 systems extends Version 1 to cover issues like extended BIFS functionalities, and Java (MPEG-J) support. Version 2 also specifies a file format to store MPEG-4 content. In Section 8, these new elements will be introduced in more detail.

3.2 Visual

3.2.1 Natural Video

MPEG-4 Visual Version 2 adds technology in the following areas:

- increased flexibility in object-based scalable coding,
- improved coding efficiency,
- improved temporal resolution stability with the low buffering delay,
- improved error robustness,
- coding of multiple views: Intermediate views or stereoscopic views will be supported based on the efficient coding of multiple images or video sequences. A particular example is the coding of stereoscopic images or video by redundancy reduction of information contained between the images of different views.

See Section 9 for more detail.

3.2.2 Body animation

Version 2 adds Body Animation to the Face Animation already present in V.1

3.2.3 Coding of 3-D Polygonal Meshes

Version 2 MPEG-4 provides a suite of tools for coding 3-D polygonal meshes. Polygonal meshes are widely used as a generic representation of 3-D objects. The underlying technologies compress the connectivity, geometry, and properties such as shading normals, colors and texture coordinates of 3-D polygonal meshes.

3.3 Audio

MPEG-4 Audio Version 2 is an extension to MPEG-4 Audio Version 1. It adds new tools and functionalities to the MPEG-4 Standard, while none of the existing tools of Version 1 is replaced. The following additional functionalities are provided by MPEG-4 Audio Version 2:

- Increased error robustness
- Audio coding that couples high quality to low delay
- Fine Grain scalability (scalability resolution down to 1 kbit/s per channel)
- Parametric Audio Coding to allow sound manipulation at low speeds
- CELP Silence compression, to further lower bitrates in speech conding
- Error resilient parametric speech coding
- Environmental spatialization – the possibility to recreate sound environment using perceptual and/or physical modeling techniques
- A back channel that is helpful to adjust encoding or scalable play out in real time
- A low overhead, MPEG-4caudio-specific transport mechanism

See Section 10, Detailed technical description of MPEG-4 Audio

3.4 DMIF

The major features introduced by DMIF Version 2 cover (limited) support to mobile networks and QoS monitoring. A few minor additions were introduced as well.

3.4.1 Support to mobile networks

In conjunction with ITU-T, the H.245 specification has been extended (H.245v6) to include support to MPEG-4 Systems; the DMIF specification provides the appropriate walkthrough and mapping to H.245 signals. Mobile terminals can now use MPEG-4 Systems features such as BIFS and OD streams, although with some limitation (the MPEG-4 presentation is uniquely selected by the target peer)

3.4.2 Qos Monitoring

DMIF V.2 introduces the concept of monitoring the Quality of Service actually delivered by a network. The DMIF-Application Interface has been extended accordingly. The model allows for three different modes of QoS monitoring: continuous monitoring, specific queries, and QoS violation notification.

3.4.3 UserCommands with ACK

The DMIF model allows peer applications to exchange user messages of any kind (included stream control messages). DMIF V2 adds to V.1 the support for acknowledgment messages.

3.4.4 Management of MPEG-4 Sync Layer information

V.2 enhances the DMIF model to allow applications to exchange application-specific data with the DMIF layer. This addition was introduced to enable, within the model, the exchange of Sync Layer Protocol Data Units as a combination of pure media data (PDU) and logical Sync Layer information. The model acknowledges that within the existing transport stacks there are features that overlap with the MPEG-4 Systems Sync Layer. This is the case of RTP and MPEG-2 PES (Packetized Elementary Steams) as well as MP4 atoms in the file format: in all such cases the obvious implementation of a DMIF instance is to map the Sync Layer information extracted from those structures, as well as from a true SL-PDU, into a uniform logical representation of the Sync Layer Packet Header. As a consequence, the appropriate parameters have been introduced at the DAI, taking care, as usual, to make their semantic independent of both transport stack and application.

3.4.5 DAI syntax in C language

DMIF V.2 includes an informative annex that gives a C/C++ syntax for the DMIF Application Interface, as a recommended API syntax.

4. Extensions to MPEG-4 beyond Version 2

MPEG is currently working on a number of extensions to Version 2., in the Visual and Systems areas. There is no work on extending MPEG-4 DMIF or Audio beyond Version 2.

4.1 Visual

In the visual area, the following technologies are in the process of being added:

- Fine Grain scalability is in balloting phase, with proposed 'Streaming Video Profiles' ('Advanced Simple' and 'Fine Grain Scalability'). Fine Grain Scalability is a tool that allows small quality steps by adding or deleting layers of extra information. It is useful in a number of environments, notably for streaming purposes but also for dynamic ('statistical') multiplexing of pre-encoded content in broadcast environments.
- Tools for usage of MPEG-4 in the Studio. For these tools, care has been taken to preserve some form of compatibility with MPEG-2 profiles. Currently, the Simple Studio Profile is in a balloting phase, this is a profile with I-frame only coding at very high bitrates (several hundred Mbits/s) which employs shape coding. Addition of a Core Studio Profile (with I and P frames) is expected.
- Digital Cinema is under study. This application will require truly lossless coding, and not just the visually lossless that MPEG-4 has provided so far. A Preliminary Call for Proposals was issued in October 2000.

4.2 Systems

4.2.1 Advanced BIFS

Provides new nodes to be used in the scene graph for monitoring available media and managing media, such as sending commands to a server, advanced control of media playback, and the so-called EXTERNPROTO, a node that provides further compatibility with VRML, and that allows writing macros that define behavior of objects. Also, advanced compression of BIFS data is covered, and in particular optimal compression for mesh and for arrays of data.

4.2.2 Textual Format

The Extensible MPEG-4 Textual format (XMT) is a framework for representing MPEG-4 scene description using a textual syntax. The XMT allows the content authors to exchange their content with other authors, tools or service providers, and facilitates interoperability with both the Extensible 3D (X3D) being developed by the Web3D Consortium, and the Synchronized Multimedia Integration Language (SMIL) from the W3C consortium.

The XMT format can be interchanged between SMIL players, VRML players, and MPEG-4 players. The format can be parsed and played directly by a W3C SMIL player, preprocessed to Web3D X3D and played back by a VRML player, or compiled to an MPEG-4 representation such as mp4, which can then be played by an MPEG-4 player. See below for a graphical description of interoperability of the XMT. It encompasses MPEG-4, a large part of SMIL, Scalable Vector Graphics, X3D and also gives a textual representation for MPEG-7 Descrptions (see www.cselt.it/mpeg for documentation on the MPEG-7 Content Description Standard)

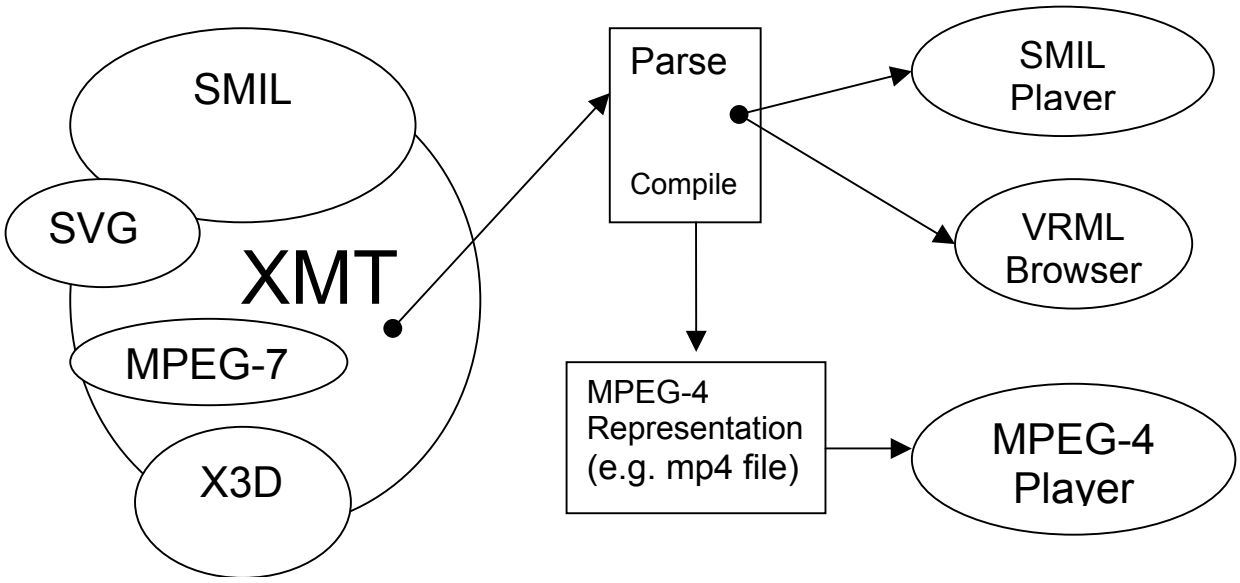


Figure 3 - the eXtensible MPEG-4 Textual format

The XMT framework consists of two levels of textual syntax and semantics: the XMT-A format and the XMT-Ω format.
The XMT-A is an XML-based version of MPEG-4 content, which contains a subset of the X3D. Also contained in XMT-A is an MPEG-4 extension to the X3D to represent MPEG-4 specific features. The XMT-A provides a straightforward, one-to-one mapping between the textual and binary formats.

The XMT-Ω is a high-level abstraction of MPEG-4 features based on the W3C SMIL. The XMT provides a default mapping from Ω to A, for there is no deterministic mapping between the two, and it also provides content authors with an escape mechanism from Ω to A.

4.2.3 Advanced Synchronization Model

The Advanced Synchronization Model (usually called so-called 'FlexTime') supports synchronization of objects from multiple sources with possibly different time bases. The FlexTime Model specifies timing using a flexible, constraint-based timing model. In this model, media objects can be linked to one another in a time graph using relationship constraints such as "CoStart", "CoEnd", or "Meet". And, in addition, to allow some flexibility to meet these constraints, each object may have a flexible duration with specific stretch and shrink mode preferences that may be applied.

The FlexTime model is based upon a so-called "spring" metaphor. A spring has a set of three constants: the minimum length below which it will not shrink, the maximum length beyond which it will break, and the optimal length at which it rests comfortably being neither compressed nor extended. Following this spring model, the temporal playback of media objects

can be viewed as springs, with a set of playback durations corresponding to these three spring constants. The optimal playback duration (optimal spring length) can be viewed as the author's preferred choice of playback duration for the media object. A player should, where possible, keep the playback length as close to the optimal duration as the presentation allows but may choose any duration between the minimum and maximum durations as specified by the author. Note, that whereas stretching or shrinking the duration continuous media, e.g. for video, implies respectively slowing down or speeding up playback, for discrete media such as a still image, shrinking or stretching is merely adjusting the rendering period shorter or longer.

4.2.4 2D and 3D animation

- 2D and 3D animation coding are under study following fgood responses received after a Call for Proposals that was evaluated in October 2000. The work is being progressed together with the Web3D Consortium.

5. Profiles in MPEG-4

MPEG-4 provides a large and rich set of tools for the coding of audio-visual objects. In order to allow effective implementations of the standard, subsets of the MPEG-4 Systems, Visual, and Audio tool sets have been identified, that can be used for specific applications. These subsets, called 'Profiles', limit the tool set a decoder has to implement. For each of these Profiles, one or more Levels have been set, restricting the computational complexity. The approach is similar to MPEG-2, where the most well known Profile/Level combination is 'Main Profile @ Main Level'. A Profile@Level combination allows:

- a codec builder to implement only the subset of the standard he needs, while maintaining interworking with other MPEG-4 devices built to the same combination, and
- checking whether MPEG-4 devices comply with the standard ('conformance testing').

Profiles exist for various types of media content (audio, visual, and graphics) and for scene descriptions. MPEG does not prescribe or advise combinations of these Profiles, but care has been taken that good matches exist between the different areas.

5.1 Visual Profiles

The visual part of the standard provides profiles for the coding of natural, synthetic, and synthetic/natural hybrid visual content. There are five profiles for natural video content:

1. The *Simple* Visual Profile provides efficient, error resilient coding of rectangular video objects, suitable for applications on mobile networks, such as PCS and IMT2000.
2. The *Simple Scalable* Visual Profile adds support for coding of temporal and spatial scalable objects to the Simple Visual Profile, It is useful for applications which provide services at more than one level of quality due to bit-rate or decoder resource limitations, such as Internet use and software decoding.
3. The *Core* Visual Profile adds support for coding of arbitrary-shaped and temporally scalable objects to the Simple Visual Profile. It is useful for applications such as those providing relatively simple content-interactivity (Internet multimedia applications).
4. The *Main* Visual Profile adds support for coding of interlaced, semi-transparent, and sprite objects to the Core Visual Profile. It is useful for interactive and entertainment-quality broadcast and DVD applications.
5. The *N-Bit* Visual Profile adds support for coding video objects having pixel-depths ranging from 4 to 12 bits to the Core Visual Profile. It is suitable for use in surveillance applications.

The profiles for synthetic and synthetic/natural hybrid visual content are:

6. The *Simple Facial Animation* Visual Profile provides a simple means to animate a face model, suitable for applications such as audio/video presentation for the hearing impaired.
7. The *Scalable Texture Visual* Profile provides spatial scalable coding of still image (texture) objects useful for applications needing multiple scalability levels, such as mapping texture onto objects in games, and high-resolution digital still cameras.
8. The *Basic Animated 2-D Texture* Visual Profile provides spatial scalability, SNR scalability, and mesh-based animation for still image (textures) objects and also simple face object animation.
9. The *Hybrid Visual* Profile combines the ability to decode arbitrary-shaped and temporally scalable natural video objects (as in the Core Visual Profile) with the ability to decode several synthetic and hybrid objects, including simple face and animated still image objects. It is suitable for various content-rich multimedia applications.

Version 2 adds the following Profiles for natural video:

10. The *Advanced Real-Time Simple* (ARTS) Profile provides advanced error resilient coding techniques of rectangular video objects using a back channel and improved temporal resolution stability with the low buffering delay. It is suitable for real time coding applications; such as the videophone, tele-conferencing and the remote observation.
11. The *Core Scalable* Profile adds support for coding of temporal and spatial scalable arbitrarily shaped objects to the Core Profile. The main functionality of this profile is object based SNR and spatial/temporal scalability for regions or objects of interest. It is useful for applications such as the Internet, mobile and broadcast.
12. The *Advanced Coding Efficiency* (ACE) Profile improves the coding efficiency for both rectangular and arbitrary shaped objects. It is suitable for applications such as mobile broadcast reception, the acquisition of image sequences (camcorders) and other applications where high coding efficiency is requested and small footprint is not the prime concern.

The Version 2 profiles for synthetic and synthetic/natural hybrid visual content are:

13. The *Advanced Scaleable Texture* Profile supports decoding of arbitrary-shaped texture and still images including scalable shape coding, wavelet tiling and error-resilience. It is useful for applications the require fast random access as well as multiple scalability levels and arbitrary-shaped coding of still objects. Examples are fast content-based still image browsing on the Internet, multimedia-enabled PDA's, and Internet-ready high-resolution digital still cameras.
14. The *Advanced Core* Profile combines the ability to decode arbitrary-shaped video objects (as in the Core Visual Profile) with the ability to decode arbitrary-shaped scalable still image objects (as in the Advanced Scaleable Texture Profile.) It is suitable for various content-rich multimedia applications such as interactive multimedia streaming over Internet.
15. The *Simple Face and Body Animation* Profile is a superset of the Simple Face Animation Profile, adding - obviously - body animation.

In subsequent Versions, the following Profiles were added

16. The *Advanced Simple* Profile looks much like Simple in that it has only rectangular objects, but it has a few extra tools that make it more efficient: B-frames, ¼ pel motion compensation and global motion compensation.
17. The *Fine Granular Scalability* Profile allows many scalable layers – up to 8 – so that delivery quality can easily adapt to transmission and decoding circumstances. It can be used with *Simple* or *Advanced Simple* as a base layer.
18. The *Simple Studio* Profile is a profile with very high quality for usage in Studio editing applications. It only has I frames, but it does support arbitrary shape and in fact multiple alpha channels. Bitrates go up to almost 2 Gigabit per second.

19. The *Core Studio* Profile adds P frames to *Simple Studio,* making it more efficient but also requiring more complex implementations.

5.2 Audio Profiles

Four Audio Profiles have been defined in MPEG-4 V.1:

1. The *Speech Profile* provides HVXC, which is a very-low bit-rate parametric speech coder, a CELP narrowband/wideband speech coder, and a Text-To-Speech interface.
2. The *Synthesis Profile* provides score driven synthesis using SAOL and wavetables and a Text-to-Speech Interface to generate sound and speech at very low bitrates.
3. The *Scalable Profile*, a superset of the Speech Profile, is suitable for scalable coding of speech and music for networks, such as Internet and Narrow band Audio DIgital Broadcasting (NADIB). The bitrates range from 6 kbit/s and 24 kbit/s, with bandwidths between 3.5 and 9 kHz.
4. The *Main Profile* is a rich superset of all the other Profiles, containing tools for natural and synthetic Audio.

Another four Profiles were added in MPEG-4 V.2:

5. The *High Quality Audio* Profile contains the CELP speech coder and the Low Complexity AAC coder including Long Term Prediction. Scalable coding coding can be performed by the AAC Scalable object type. Optionally, the new error resilient (ER) bitstream syntax may be used.
6. The *Low Delay Audio* Profile contains the HVXC and CELP speech coders (optionally using the ER bitstream syntax), the low-delay AAC coder and the Text-to-Speech interface TTSI.
7. The *Natural Audio* Profile contains all natural audio coding tools available in MPEG-4, but not the synthetic ones.
8. The *Mobile Audio Internetworking* Profile contains the low-delay and scalable AAC object types including TwinVQ and BSAC. This profile is intended to extend communication applications using non-MPEG speech coding algorithms with high quality audio coding capabilities.

5.3 Graphics Profiles

Graphics Profiles define which graphical and textual elements can be used in a scene. These profiles are defined in the Systems part of the standard:

1. Simple 2-D Graphics Profile The Simple 2-D Graphics profile provides for only those graphics elements of the BIFS tool that are necessary to place one or more visual objects in a scene.
2. Complete 2-D Graphics Profile The Complete 2-D Graphics profile provides two-dimensional graphics functionalities and supports features such as arbitrary two-dimensional graphics and text, possibly in conjunction with visual objects.
3. Complete Graphics Profile The Complete Graphics profile provides advanced graphical elements such as elevation grids and extrusions and allows creating content with sophisticated lighting. The Complete Graphics profile enables applications such as complex virtual worlds that exhibit a high degree of realism.
4. The *3D Audio* Graphics Profile sounds like a contradictory in terms, but really isn't. This profile does not propose visual rendering, but graphics tools are provided to define the acoustical properties of the scene (geometry, acoustics absorption, diffusion, transparency of the material). This profile is used for applications that do environmental spatialization of audio signals. (See Section 10.2.7)

5.4 Scene Graph Profiles

Scene Graph Profiles (or Scene Description Profiles), defined in the Systems part of the standard, allow audiovisual scenes with audio-only, 2-dimensional, 3-dimensional or mixed 2-D/3-D content.

1. The *Audio* Scene Graph Profile provides for a set of BIFS scene graph elements for usage in audio only applications. The Audio Scene Graph profile supports applications like broadcast radio.
2. The *Simple 2-D* Scene Graph Profile provides for only those BIFS scene graph elements necessary to place one or more audio-visual objects in a scene. The Simple 2-D Scene Graph profile allows presentation of audio-visual content with potential update of the complete scene but no interaction capabilities. The Simple 2-D Scene Graph profile supports applications like broadcast television.
3. The *Complete 2-D* Scene Graph Profile provides for all the 2-D scene description elements of the BIFS tool. It supports features such as 2-D transformations and alpha blending. The Complete 2-D Scene Graph profile enables 2-D applications that require extensive and customized interactivity.
4. The *Complete* Scene Graph profile provides the complete set of scene graph elements of the BIFS tool. The Complete Scene Graph profile will enable applications like dynamic virtual 3-D world and games.
5. The *3D Audio* Scene Graph Profile provides the tools three-dimensional sound positioning in relation either with acoustic parameters of the scene or its perceptual attributes. The user can interact with the scene by changing the position of the sound source, by changing the room effect or moving the listening point. This Profile is intended for usage in audio-only applications.

5.5 MPEG-J Profiles

Two MPEG-J Profiles exist: Personal and Main:

1. *Personal* - a lightweight package for personal devices.
   The personal profile addresses a range of constrained devices including mobile and portable devices. Examples of such devices are cell video phones, PDAs, personal gaming devices. This profile includes the following packages of MPEG-J APIs:
   a) Network
   b) Scene
   c) Resource

2. *Main* - includes all the MPEG-J API's.
   The Main profile addresses a range of consumer devices including entertainment devices. Examples of such devices are set top boxes, computer based multimedia systems etc. It is a superset of the Personal profile. Apart from the packages in the Personal profile, this profile includes the following packages of the MPEG-J APIs:
   a) 1. Decoder
   b) 2. Decoder Functionality
   c) 3. Section Filter and Service Information

5.6 Object Descriptor Profile

The Object Descriptor Profile includes the following tools:

- Object Descriptor (OD) tool
- Sync Layer (SL) tool
- Object Content Information (OCI) tool
- Intellectual Property Management and Protection (IPMP) tool

Currently, only one profile is defined that includes all these tools. The main reason for defining this profile is not subsetting the tools, but rather defining levels for them. This applies especially to the Sync Layer tool, as MPEG-4 allows multiple time bases to exist. In the context of Levels for this Profile, restrictions can be defined, e.g. to allow only a single time base.

6. Verification Testing: checking MPEG's performance

MPEG carries out verification tests to check whether the standard delivers what it promises. The test results can be found on MPEG's home page, http://www.cselt.it/mpeg/quality_tests.htm The main results are described below; more verification tests are planned.

6.1 Video

A number of MPEG-4's capabilities have been formally evaluated using subjective tests. Coding efficiency, although not the only MPEG-4 functionality, is an important selling point of MPEG-4, and one that has been tested more thoroughly. Also error robustness has been put to rigorous tests. Furthermore, scalability tests were done and for one specific profile the temporal resolution stability was examined. Many of these tests address a specific profile.

6.1.1 Coding Efficiency Tests

6.1.1.1 Low and Medium Bit rates (version 1)

In this Low and Medium Bitrates Test, frame-based sequences were examined, with MPEG-1 as a reference. (MPEG-2 would be identical for the progressive sequences used, except that MPEG-1 is a bit more efficient as it uses less overhead for header information). The test uses typical test sequences for CIF and QCIF resolutions, encoded with the same rate control for both MPEG-1 and MPEG-4 to compare the coding algorithms without the impact of different rate control schemes. The test was performed for low bit rates starting at 40 kbps to medium bit rate up to 768 kbps.

The tests of the Coding Efficiency functionality show a clear superiority of MPEG-4 toward MPEG-1 at both the low and medium bit rate coding conditions whatever the criticality of the scene. The human subjects have consistently chose MPEG-4 as statistically significantly superior by one point difference for a full scale of five points.

6.1.1.2 Content Based Coding (version 1)

The verification tests for Content Based Coding compare the visual quality of object-based versus frame-based coding. The major objective was to ensure that object-based coding can be supported without impacting the visual quality. Test content was chosen to cover a wide variety of simulation conditions, including video segments with various types of motions and encoding complexities. Additionally, test conditions were established to cover low bit rates ranging from 256kb/s to 384kb/s, as well as high bit-rates ranging from 512kb/s to 1.15Mb/s. The results of the tests clearly demonstrated that object-based functionality is provided by MPEG-4 with no overhead or loss in terms of visual quality, when compared to frame-based coding. There is no statistically significant difference among any object-based case and the relevant frame-based ones. Hence the conclusion: MPEG-4 is able to provide content-based functionality without introducing any loss in terms of visual quality.

6.1.1.3 Advanced Coding Efficiency (ACE) Profile (version 2)

The formal verification tests on Advanced Coding Efficiency (ACE) Profile were performed to check whether three new Version 2 tools, as included the MPEG-4 Visual Version 2 ACE Profile (Global Motion Compensation, Quarter Pel Motion Compensation and Shape-adaptive DCT)

enhance the coding efficiency compared with MPEG-4 Visual Version 1. The tests explored the performance of the ACE Profile and the MPEG-4 Visual Version 1 Main Profile in the object-based low bit rate case, the frame-based low bit rate case and the frame-based high bit rate case. The results obtained show a clear superiority of the ACE Profile compared with the Main Profile; more in detail:

- For the object based case, the quality provided by the ACE Profile at 256 kb/s is equal to the quality provided by Main Profile at 384 kb/s.
- For the frame based at low bit rate case, the quality provided by the ACE Profile at 128 kb/s and 256 kb/s is equal to the quality provided by Main Profile at 256 kb/s and 384 kb/s respectively.
- For the frame based at high bit rate case, the quality provided by the ACE Profile at 768 kb/s is equal to the quality provided by Main Profile at 1024 kb/s.

When interpreting these results, it must be noted that the MPEG-4 Main Profile is already more efficient than MPEG-1 and MPEG-2.

6.1.2 Error Robustness Tests

6.1.2.1 Simple Profile (version 1)

The performance of error resilient video in the MPEG-4 Simple Profile was evaluated in subjective tests simulating MPEG-4 video carried in a realistic multiplex and over ditto radio channels, at bitrates between 32 kbit/s and 384 kbit/s. The test used a simulation of the residual errors after channel coding at bit error rates up to 10^{-3}, and the average length of the burst errors was about 10ms. The test methodology was based on a continuous quality evaluation over a period of three minutes. In such a test, subjects constantly score the degradation they experience. The results show that the average video quality achieved on the mobile channel is high, that the impact of errors is effectively kept local by the tools in MPEG-4 video, and that the video quality recovers quickly at the end of periods of error. These excellent results were achieved with very low overheads, less than those typically associated with the GOP structure used in MPEG-1 and MPEG-2 video.

6.1.2.2 Advanced Real-Time Simple (ARTS) Profile (version 2)

The performance of error resilient video in MPEG-4 ARTS Profile was checked in subjective tests similar to those mentioned in the previous section, at bitrates between 32 kbit/s and 128 kbit/s. In this case, the residual errors after channel coding was up to 10^{-3}, and the average length of the burst errors was about 10 ms (called “critical”) or 1 ms (called “very critical” - this one is more critical because the same amount of errors is more spread over the bitstream than in the “critical” case).

The results show a clear superiority of the ARTS Profile over the Simple Profile for both the error cases (“critical” and “very critical”). More in detail the ARTS Profile outperforms Simple Profile in the recovery time from transmission errors. Furthermore ARTS Profile in the “critical” error condition provides results that for most of the test time are close to a complete transparency, while Simple Profile is still severely affected by errors. These excellent results were achieved with very low overheads and very fast error recovery provided the NEWPRED, and under low delay conditions.

6.1.3 Temporal Resolution Stability Test

6.1.3.1 Advanced Real-Time Simple (ARTS) Profile (version 2)

This test explored the performance of a video codec using the Dynamic Resolution Conversion technique that adapts the resolution to the video content and to circumstances in real-time. Active scene content was coded at 64 kb/s, 96 kb/s and 128 kb/s datarates. The results show that

at 64 kbit/s, it outperforms the already effective Simple Profile operating at 96 kbit/s, and at 96 kb/s, the visual quality is equally to that of the Simple profile at 128 kbit/s. (The Simple profile already compares well to other, existing systems.)

6.1.4 Scalability Tests

6.1.4.1 Simple Scalable Profile (version 1)

The scalability test for the Simple Scalable Profile was designed to verify that the quality provided by Temporal Scalability tool in Simple Scalable Profile compared to the quality provided by Single Layer coding in Simple Profile, and to the quality provided by Simulcast coding in Simple Profile.
In this test, 5 sequences with 4 combinations of bitrates were used:
a) 24 kbps for base layer and 40 kbps for enhancement layer.
b) 32 kbps for both layers.
c) 64 kbps for the base layer and 64 kbps for the enhancement layer.
d) 128 kbps for both layers.
The formal verification tests showed that in all the given conditions, the Temporal Scalability coding in Simple Scalable Profile exhibits the same or slightly lower quality than can be achieved using Single layer coding in Simple Profile. Furthermore it is evident that the Temporal Scalability coding in Simple Scalable Profile provides better quality than the simulcast coding in Simple Profile for that condition. (Simulcast entails simultaneously broadcasting or streaming at multiple bitrates.)

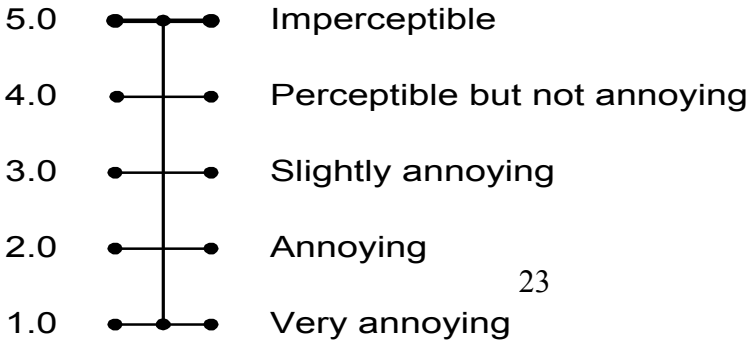
6.1.4.2 Core Profile (version 1)

The verification test was designed to evaluate the performance of MPEG-4 video Temporal Scalability tool in the Core Profile.
The test was performed using the "Single Stimulus" method. The subjects were to evaluate how annoying the impairments of compressed sequences were, with and without use of temporal scalability. The test was conducted using a total of 45 subjects in two different laboratories and the results showed that the quality of sequences encoded using MPEG-4 temporal scalability tools are comparable to the quality of sequences encoded without temporal scalability. Furthermore it is evident that the Temporal Scalability tool in Core Profile provides better quality than the simulcast coding in Core Profile for that condition.

6.2 Audio

MPEG-4 audio technology is composed of many coding tools. Verification tests have focused on small sets of coding tools that are appropriate in one application arena, and hence can be effectively compared. Since compression is a critical capability in MPEG, the verification tests have for the most part compared coding tools operating at similar bit rates. The results of these tests will be presented progressing from higher bit rate to lower bit rates. The exception to this is the error robustness tools, whose performance will be noted at the end of this section.

The primary purpose of verification tests is to report the subjective quality of a coding tool operating at a specified bit rate. Most audio tests report this on the subjective impairment scale.



This is a continuous 5-point scale with subjective anchors as shown here.

The performance of the various MPEG-4 coding tools are summarized in the following table. To better enable the evaluation of MPEG-4 technology, several coders from MPEG-2 and the ITU-T were included in the tests and their evaluation has also been included in the table. In the table results from the same test are delimited by heavy lines. These results can be directly compared. Results taken from different tests should not be compared, but nevertheless give an indication of the expected quality of a coding tool operating at a specific bit rate.

Coding tool	Number of channels	Total bit rate	Typical subjective quality
AAC	5	320 kb/s	4.6
1995 Backward Compatible MPEG-2 Layer II	5	640 kb/s	4.6
AAC	2	128 kb/s	4.8
AAC	2	96 kb/s	4.4
MPEG-2 Layer II	2	192 kb/s	4.3
MPEG-2 Layer III	2	128 kb/s	4.1
AAC	1	24 kb/s	4.2
Scalable: CELP base and AAC enhancement	1	6 kb/s base, 18 kb/s enh.	3.7
Scalable: Twin VQ base and AAC enhancement	1	6 kb/s base, 18 kb/s enh.	3.6
AAC	1	18 kb/s	3.2
G.723	1	6.3 kb/s	2.8
Wideband CELP	1	18.2 kb/s	2.3
BSAC	2	96 kb/s	4.4
BSAC	2	80 kb/s	3.7
BSAC	2	64 kb/s	3.0
AAC – LD (20 ms one-way delay)	1	64 kb/s	4.4
G.722	1	32 kb/s	4.2
AAC – LD (30 ms one-way delay)	1	32 kb/s	3.4
Narrowband CELP	1	6 kb/s	2.5
Twin VQ	1	6 kb/s	1.8
HILN	1	16 kb/s	2.8
HILN	1	6 kb/s	1.8

Coding tools were tested under circumstances that assessed their strengths. The salient features of the MPEG-4 audio coding tools are briefly noted here.

When coding 5-channel material at 64 kb/s/channel (320 kbit/s) Advanced Audio Coding (AAC) Main Profile was judged to have “indistinguishable quality” (relative to the original) according to the EBU definition. When coding 2-channel material at 128 kbps both AAC Main Profile and AAC Low Complexity Profile were judged to have “indistinguishable quality” (relative to the original) according to the EBU definition.

The two scaleable coders, CELP base with AAC enhancement, and TwinVQ base wth AAC enhancement both performed better than an AAC "multicast" operating at the enhancement layer bitrate, but not as good as an AAC coder operating at the total bitrate.

The wideband CELP coding tool showed excellent performance for speech-only signals. (The verification test result shown is for both speech and music signals.)

Bit Slice Arithmetic Coding (BSAC) provides a very fine step bitrate scalability. At the top of the scalability range it has no penalty relative to single-rate AAC, however at the bottom of the scale it has a slight penalty relative to single-rate AAC.

Relative to normal AAC, Low Delay AAC (AAC LD) provides equivalent subjective quality, but with very low on-way delay and only a slight increase in bit rate.

Narrowband CELP, TwinVQ and Harmonic Individual Lines and Noise (HILN) all have the ability to provide very high signal compression.

The Error Robustness (ER) tools provide equivalently good error robustness over a wide range of channel error conditions, and does so with only a modest overhead in bit rate. Verification test results suggest that the ER tools used with an audio coding system provide performance in error-prone channels that is "nearly as good" as the same coding system operating over a clear channel.

7. The MPEG-4 Industry Forum

The MPEG-4 Industry Forum is a not-for-profit organization with the following goal: *To further the adoption of the MPEG-4 Standard, by establishing MPEG-4 as an accepted and widely used standard among application developers, service providers, content creators and end users.*

The following is a non-exhaustive excerpt from M4IF's Statutes about the way of operation:

- The purpose of M4IF shall be pursued by: promoting MPEG-4, making available information on MPEG-4, making available MPEG-4 tools or giving information on where to obtain these, creating a single point for information about MPEG-4, creating industrial focus around the usage of MPEG-4
- The goals are realized through the open international collaboration of all interested parties, on reasonable terms applied uniformly and openly. M4IF will contribute the results of its activities to appropriate formal standards bodies if applicable.
- The business of M4IF shall not be conducted for the financial profits of its Members but for their mutual benefits.
- Any corporation and individual firm, partnership, governmental body or international organization supporting the purpose of M4IF may apply for Membership.
- Members are not bound to implement or use specific technology standards, or recommendations by virtue of participation in M4IF.
- There is no licensing requirement attached to M4IF membership, and M4IF will not set any licensing terms for MPEG-4 technology. (There can, however, be studies about what licensing models would suite MPEG-4's operational environments)
- The initial membership fee is set at US$ 2,000 per year.

M4IF has its homepage: http://www.m4if.org

Interested parties can visit that page for details, and they can also sign up to public mail lists (an information list and a technical list). The site also has all the information about formally joining the Forum.

The Forum currently has close to 90 members and hundreds of people subscribed to its mail lists, with a very broad, worldwide representation from the following industries: Consumer Electronics, Computer, Telecommunications, Research Institutions . Also, some of the members are 'business users' of MPEG-4. Among the list of current participants, there are many large and small companies that develop or deploy MPEG-4 technology. Membership goes beyond the MPEG constituency, partly because some of the smaller companies find it hard to comply with the requirements for participation in MPEG (which vary from one country to another) and also because some companies just don't need to be involved in the development phase.

The activities of M4IF generally start where MPEG stops. This includes issues that MPEG cannot deal with, e.g. because of ISO rules, such as clearance of patents. This following is a list of M4IF's current activities:

- Promoting the standard, and serving as a single point of information on MPEG-4 technology, products and services;
- Initiating discussions leading to the potential establishment of patent pools outside of M4IF, that should grant a license to an unlimited number of applicants throughout the world under reasonable terms and conditions that are demonstrably free of any unfair competition; this work includes studying licensing models for downloadable software decoders, such as internet players;
- Organization of MPEG-4 exhibitions and tutorials. The first event was a successful exhibition in Geneva from May 28 - May 30 2000. The exhibition space show MPEG-4 products from established companies as well as start-ups, including audio and video players, authoring tools, facial animation, and more;
- Interoperability testing, potentially leading to certification of MPEG-4 products.

M4IF anticipates holding some 3 physical meetings per year, with a slightly higher frequency in the current start-up phase. About a hundred people from all over the world have attended these meetings. The focus has been on initiating the patent pools, but it will now shift towards interoperability tests. See the home page for meeting details.

8. Detailed technical description of MPEG-4 DMIF and Systems

This Section contains a detailed overview of the MPEG-4 Standard. It first described the entire system, and then describes all parts of the system in subsections.

Figure 4 shows how streams coming from the network (or a storage device), as TransMux Streams, are demultiplexed into FlexMux Streams and passed to appropriate FlexMux demultiplexers that retrieve Elementary Streams. How this works is described in Section 8.2. The Elementary Streams (ESs) are parsed and passed to the appropriate decoders. Decoding recovers the data in an AV object from its encoded form and performs the necessary operations to reconstruct the original AV object ready for rendering on the appropriate device. Audio and visual objects are represented in their coded form, which is described in sections 10 and 9 respectively. The reconstructed AV object is made available to the composition layer for potential use during scene rendering. Decoded AVOs, along with scene description information, are used to compose the scene as described by the author. Scene description and Composition are explained in Section 8.5. The user can, to the extent allowed by the author, interact with the scene which is eventually rendered and presented. Section 8.5.1 describes this interaction. Sections 8.7 and 8.8 discuss the ‘Intellectual Property Management and Protection’ and Object

Content Information respectively. Sections 8.9 and 8.10 describe Version 2 additions: the File Format and MPEG-J.

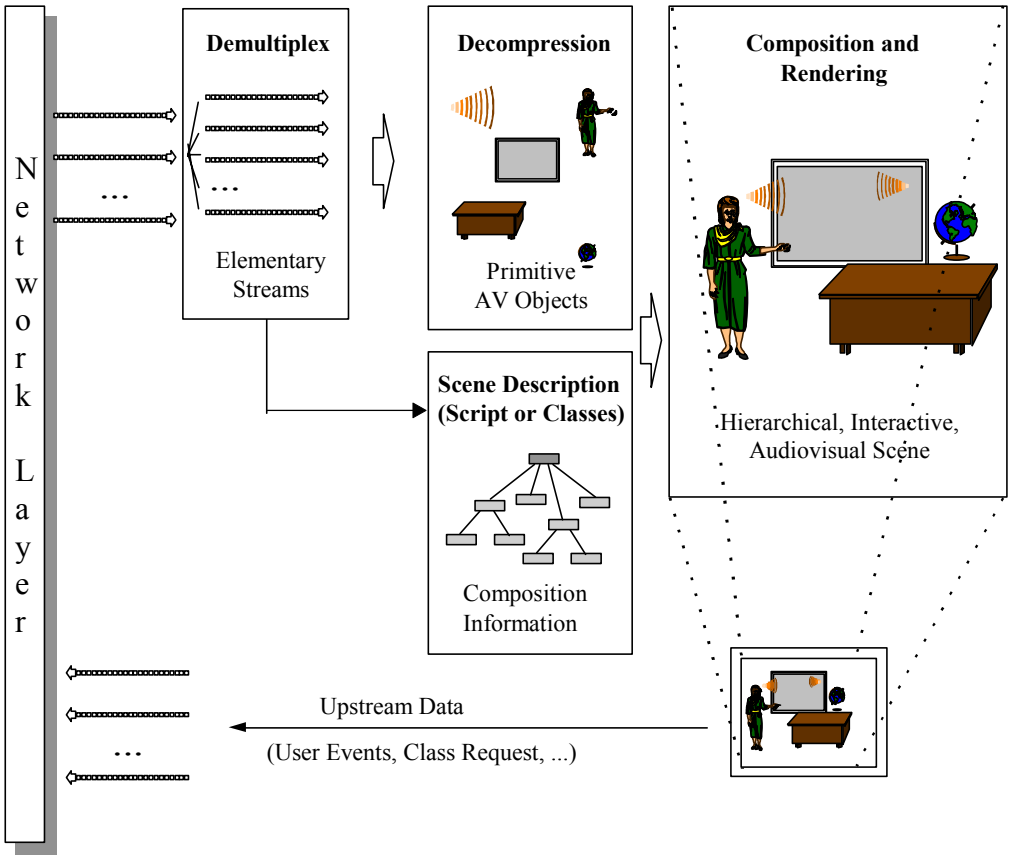


Figure 4- Major components of an MPEG-4 terminal (receiver side)

8.1 DMIF

DMIF (Delivery Multimedia Integration Framework) is a session protocol for the management of multimedia streaming over generic delivery technologies. In principle it is similar to FTP. The only (essential!) difference is that FTP returns data, DMIF returns pointers to where to get (streamed) data

When FTP is run, the very first action it performs is the setup of a session with the remote side. Later, files are selected and FTP sends a request to download them, the FTP peer will return the files in a separate connection.

Similarly, when DMIF is run, the very first action it performs is the setup of a session with the remote side. Later, streams are selected and DMIF sends a request to stream them, the DMIF peer will return the pointers to the connections where the streams will be streamed, and then also establishes the connection themselves.

Compared to FTP, DMIF is both a framework and a protocol. The functionality provided by DMIF is expressed by an interface called DMIF-Application Interface (DAI), and translated into protocol messages. These protocol messages may differ based on the network on which they operate.

The Quality of Service is also considered in the DMIF design, and the DAI allows the DMIF user to specify the requirements for the desired stream. It is then up to the DMIF implementation to make sure that the requirements are fulfilled. The DMIF specification provides hints on how to perform such tasks on a few network types, such as the Internet.

The DAI is also used for accessing broadcast material and local files, this means that a single, uniform interface is defined to access multimedia contents on a multitude of delivery technologies.
As a consequence, it is appropriate to state that the integration framework of DMIF covers three major technologies, interactive network technology, broadcast technology and the disk technology; this is shown in the Figure 5 below.

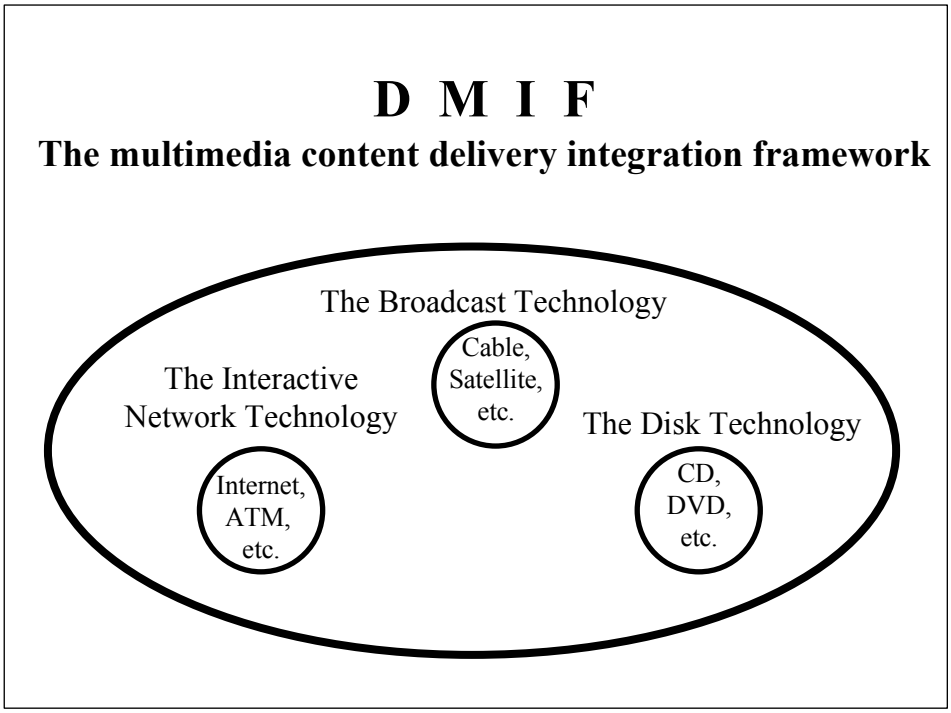


Figure 5 - DMIF addresses the delivery integration of three major technologies

The DMIF architecture is such that applications that rely on DMIF for communication do not have to be concerned with the underlying communication method. The implementation of DMIF takes care of the delivery technology details presenting a simple interface to the application.

Figure 5 represents the above concept. An application accesses data through the DMIF-Application Interface, irrespective of whether such data comes from a broadcast source, from local storage or from a remote server. In all scenarios the Local Application only interacts through a uniform interface (DAI). Different DMIF instances will then translate the Local Application requests into specific messages to be delivered to the Remote Application, taking care of the peculiarities of the involved delivery technology. Similarly, data entering the terminal (from remote servers, broadcast networks or local files) is uniformly delivered to the Local Application through the DAI.

Different, specialized DMIF instances are indirectly invoked by the Application to manage the various specific delivery technologies, this is however transparent to the Application, that only interacts with a single “DMIF filter”. This filter is in charge of directing the particular DAI primitive to the right instance. DMIF does not specify this mechanism, just assumes it is implemented. This is further emphasized by the shaded boxes in the figure, whose aim is to clarify what are the borders of a DMIF implementation, while the DMIF communication architecture defines a number of modules, actual DMIF implementations only need to preserve their appearance at those borders.

Conceptually, a “real” remote application accessed through a network e.g., IP- or ATM-based, is no different than an emulated remote producer application getting content from a broadcast source or from a disk. In the former case, however, the messages exchanged between the two

entities have to be normatively defined to ensure interoperability (these are the DMIF Signaling messages). In the latter case, on the other hand, the interfaces between the two DMIF peers and the emulated Remote Application are internal to a single implementation and need not be considered in this specification. Note that for the broadcast and local storage scenarios, the figure shows a chain of "Local DMIF", "Remote DMIF (emulated)" and "Remote Application (emulated)". This chain only represents a conceptual model and need not be reflected in actual implementations (it is shown in the figure totally internal to a shaded box).

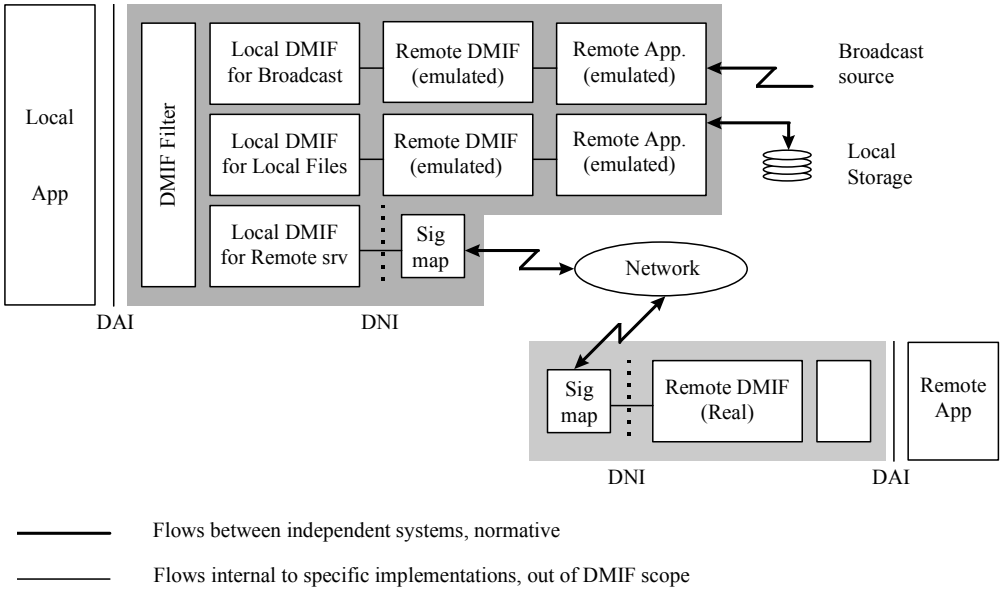


Figure 6 - DMIF communication architecture

When considering the Broadcast and Local Storage scenarios, it is assumed that the (emulated) Remote Application has knowledge on how the data is delivered/stored. This implies knowledge of the kind of application it is dealing with. In the case of MPEG-4, this actually means knowledge of concepts like Elementary Stream ID, First Object Descriptor, ServiceName. Thus, while the DMIF Layer is conceptually unaware of the application it is providing support to, in the particular case of DMIF instances for Broadcast and Local Storage this assumption is not completely true due to the presence of the (emulated) Remote Application (which, from the Local Application perspective, is still part of the DMIF Layer).

It is worth noting that since the (emulated) Remote Application has knowledge on how the data is delivered/stored, the specification of how data is delivered/stored is crucial for such a DMIF implementation, which is thus "MPEG-4 systems aware".

When considering the Remote Interactive scenario instead, the DMIF Layer is totally application-unaware. An additional interface -the DMIF-Network Interface (DNI)- is introduced to emphasize what kind of information DMIF peers need to exchange; an additional module ("Signaling mapping" in the figure) takes care of mapping the DNI primitives into signaling messages used on the specific Network. Note that DNI primitives are only specified for information purposes, and a DNI interface need not be present in an actual implementation, Figure 6 also clearly represents the DNI as internal to the shaded box. Instead, the syntax of the messages flowing in the Network is fully specified for each specific network supported.

DMIF allows the concurrent presence of one or more DMIF instances, each one targeted for a particular delivery technology, in order to support in the same terminal multiple delivery technologies and even multiple scenarios (broadcast, local storage, remote interactive). Multiple

delivery technologies may be activated by the same application, that could therefore seamlessly manage data sent by broadcast networks, local file systems and remote interactive peers

8.1.1 The DMIF Computational Model

When an application requests the activation of a service, it uses the Service primitives of the DAI, and creates a service session; the DMIF implementation then contacts its corresponding peer (that conceptually can be either a remote peer, or a local emulated peer) and creates a network session with it. Network sessions have network-wide significance, service sessions have instead local meaning. The association between them is maintained by the DMIF Layer. In the case of Broadcast and Local Storage scenarios, the way the network session is created and then managed is out of the scope of this specification. In the case of a remote interactive scenario instead, DMIF uses the native signalling mechanism for that network to create and then manage the network session e.g., ATM signalling. The application peers then use this session to create connections which are used to transport application data e.g., MPEG-4 Elementary Streams.

When an application needs a Channel, it uses the Channel primitives of the DAI, DMIF translates these requests into connection requests which are specific to the particular network implementation. In the case of Broadcast and Local Storage scenarios, the way the connections are created and then managed is out of the scope of this specification. In the case of a networked scenario instead, DMIF uses the native signalling mechanism for that network to create those connections. The application then uses these connections to deliver the service.

Figure 7 provides a high level view of a service activation and of the beginning of data exchange; the high level walk-through consists of the following steps:

The Originating Application request the activation of a service to its local DMIF Layer -- a communication path between the Originating Application and its local DMIF peer is established in the control plane (1)

The Originating DMIF peer establishes a network session with the Target DMIF peer -- a communication path between the Originating DMIF peer and the Target DMIF Peer is established in the control plane (2)

The Target DMIF peer identifies the Target Application and forwards the service activation request -- a communication path between the Target DMIF peer and the Target Application is established in the control plane (3)

The peer Applications create channels (requests flowing through communication paths 1, 2 and 3). The resulting channels in the user plane (4) will carry the actual data exchanged by the Applications.

DMIF is involved in all four steps above.

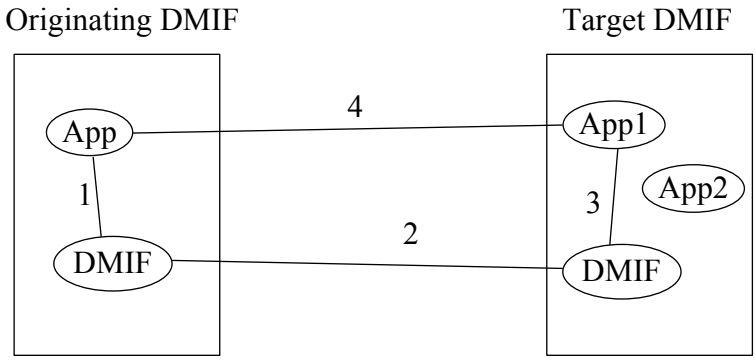


Figure 7 — DMIF Computational Model

The DMIF Layer automatically determines whether a particular service is supposed to be provided by a remote server on a particular network e.g., IP based, or ATM based, by a broadcast network, or resides in a local storage device: the selection is based on the peer address information provided by the Application as part of a URL passed to the DAI.

8.2 Demultiplexing, synchronization and description of streaming data

Individual Elementary Streams have to be retrieved on the delivery layer from incoming data from some network connection or a storage device. Each network connection or file is homogeneously considered a TransMux Channel in the MPEG-4 system model. The demultiplexing is partially or completely done by layers outside the scope of MPEG-4, depending on the application. The only multiplexing tool defined by MPEG-4 is the FlexMux tool that may optionally be used for low delay, low overhead multiplexing and for saving network connection resources.

For the purpose of integrating MPEG-4 in system environments, the DMIF Application Interface is the reference point at which elementary streams can be accessed as sync layer-packetized streams. The DMIF Network Interface specifies how either SL(Sync Layer)-packetized streams —no FlexMux used— or FlexMux Streams are to be retrieved from the TransMux Layer. This is the interface to the transport functionalities not defined by MPEG. The data part of the interfaces is considered here, while the control part is dealt with by DMIF.

In the same way that MPEG-1 and MPEG-2 describe the behavior of an idealized decoding device along with the bitstream syntax and semantics, MPEG-4 defines a System Decoder Model. This allows the precise definition of the terminal's operation without making unnecessary assumptions about implementation details. This is essential in order to give implementers the freedom to design real MPEG-4 terminals and decoding devices in a variety of ways. These devices range from television receivers, which have no ability to communicate with the sender, to computers that are fully enabled with bi-directional communication. Some devices will receive MPEG-4 streams over isochronous networks, while others will use non-isochronous means (e.g., the Internet) to exchange MPEG-4 information. The System Decoder Model provides a common model on which all implementations of MPEG-4 terminals can be based.

The specification of a buffer and timing model is essential to encoding devices which may not know ahead of time what the terminal device is or how it will receive the encoded stream. Though the MPEG-4 specification will enable the encoding device to inform the decoding

device of resource requirements, it may not be possible, as indicated earlier, for that device to respond to the sender. It is also possible that an MPEG-4 session is received simultaneously by widely different devices; it will, however, be properly rendered according to the capability of each device.

8.2.1 Demultiplexing

Demultiplexing occurs on the delivery layer that is modeled as consisting of a TransMux layer and a DMIF layer. The retrieval of incoming data streams from network connections or storage media consists of two tasks. First, the channels must be located and opened. This requires a transport control entity that manages, among others, the tables that associate transport channels to specific elementary streams. Stream map tables link each stream to a ChannelAssociationTag that serves as a handle to the channel that carries this stream. Resolving ChannelAssociationTags to the actual transport channel as well as the management of the sessions and channels is addressed by the DMIF part of the MPEG-4 standard.

Second, the incoming streams must be properly demultiplexed to recover SL-packetized streams from downstream channels (incoming at the receiving terminal) to be passed on to the synchronization layer. In interactive applications, a corresponding multiplexing stage will multiplex upstream data in upstream channels (outgoing from the receiving terminal).

The generic term ‘TransMux Layer’ is used to abstract any underlying multiplex functionality – existing or future – that is suitable to transport MPEG-4 data streams. Note that this layer is not defined in the context of MPEG-4. Examples are MPEG-2 Transport Stream, H.223, ATM AAL 2, IP/UDP. The TransMux Layer is assumed to provide protection and multiplexing functionality, indicating that this layer is responsible for offering a specific QoS. Protection functionality includes error protection and error detection tools suitable for the given network or storage medium.

In any concrete application scenario one or more specific TransMux Instances will be used. Each TransMux demultiplexer gives access to TransMux Channels. The requirements on the data interface to access a TransMux Channel are the same for all TransMux Instances. They include the need for reliable error detection, delivery, if possible, of erroneous data with a suitable error indication and framing of the payload, which may consist of either SL-packetized streams or FlexMux streams. These requirements are summarized in an informative way in the TransMux Interface, in the Systems part of the MPEG-4 Standard. An adaptation of SL-packetized streams must be specified to each transport protocol stack of interest according to these requirements and in conjunction with the standardization body that has the proper jurisdiction. This is happening for RTP and mobile channels at the moment.

The FlexMux tool is specified by MPEG to optionally provide a flexible, low overhead, low delay method for interleaving data whenever this is not sufficiently supported by the underlying protocol stack. It is especially useful when the packet size or overhead of the underlying TransMux instance is large, so that a waste of bandwidth or number of network connections would result otherwise. The FlexMux tool is not itself robust to errors and can either be used on TransMux Channels with a high QoS or to bundle Elementary Streams that are equally error tolerant. The FlexMux requires reliable error detection and sufficient framing of FlexMux packets (for random access and error recovery) from the underlying layer. These requirements are also reflected in the data primitives of the DMIF Application Interface, which defines the data access to individual transport channels. The FlexMux demultiplexer retrieves SL-packetized streams from FlexMux Streams.

8.2.2 Synchronization and description of elementary streams

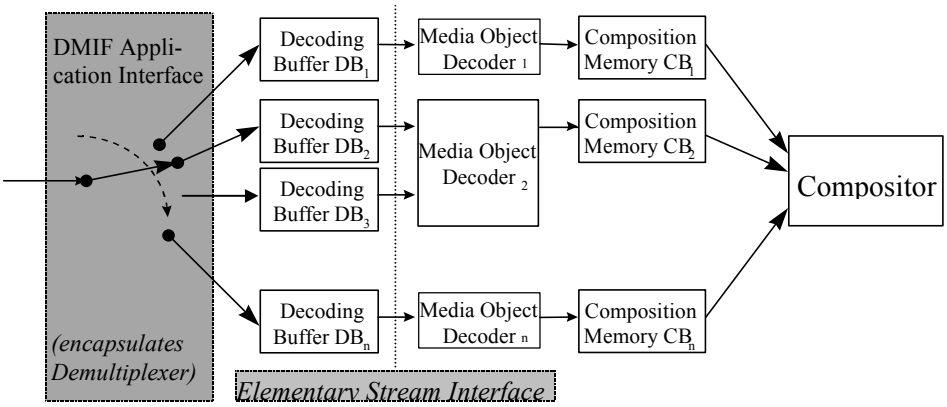


Figure 8 - Buffer architecture of the System Decoder Model

The sync layer has a minimum set of tools for consistency checking, padding, to convey time base information and to carry time stamped access units of an elementary stream. Each packet consists of one access unit or a fragment of an access unit. These time stamped access units form the only semantic structure of elementary streams that is visible on this layer. Time stamps are used to convey the nominal decoding and composition time for an access unit. The sync layer requires reliable error detection and framing of each individual packet from the underlying layer, which can be accomplished, e.g., by using the FlexMux. How data can be accessed by the compression layer is summarized in the informative Elementary Stream Interface, which can be found in the Systems part of the MPEG-4 Standard. The sync layer retrieves elementary streams from SL-packetized streams.

To be able to relate elementary streams to media objects within a scene, object descriptors are used. Object Descriptors convey information about the number and properties of elementary streams that are associated to particular media objects. Object descriptors are themselves conveyed in one or more elementary streams, since it is possible to add and discard streams (and objects) during the course of an MPEG-4 session. Such updates are time stamped in order to guarantee synchronization. The object descriptor streams can be considered as a description of the streaming resources for a presentation. Similarly, the scene description is also conveyed as an elementary stream, allowing to modify the spatio-temporal layout of the presentation over time.

8.2.3 Buffer Management

To predict how the decoder will behave when it decodes the various elementary data streams that form an MPEG-4 session, the Systems Decoder Model enables the encoder to specify and monitor the minimum buffer resources that are needed to decode a session. The required buffer resources are conveyed to the decoder within object descriptors during the setup of the MPEG-4 session, so that the decoder can decide whether it is capable of handling this session.

By managing the finite amount of buffer space the model allows a sender, for example, to transfer non real-time data ahead of time, if sufficient space is available at the receiver side to store it. The pre-stored data can then be accessed when needed, allowing at that time real-time information to use a larger amount of the channel's capacity if so desired.

8.2.4 Time Identification

For real-time operation, a timing model is assumed in which the end-to-end delay from the signal output from an encoder to the signal input to a decoder is constant. Furthermore, the transmitted data streams must contain implicit or explicit timing information. There are two types of timing

information. The first is used to convey the speed of the encoder clock, or time base, to the decoder. The second, consisting of time stamps attached to portions of the encoded AV data, contains the desired decoding time for access units or composition and expiration time for composition units. This information is conveyed in SL-packet headers generated in the sync layer. With this timing information, the inter-picture interval and audio sample rate can be adjusted at the decoder to match the encoder's inter-picture interval and audio sample rate for synchronized operation.

Different media objects may have been encoded by encoders with different time bases, with the accompanying slightly different speed. It is always possible to map these time bases to the time base of the receiving terminal. In this case, however, no real implementation of a receiving terminal can avoid the occasional repetition or drop of AV data, due to temporal aliasing (the relative reduction or extension of their time scale).

Although systems operation without any timing information is allowed, defining a buffering model is not possible for this case.

8.3 Advanced Synchronization (FlexTime) Model

The FlexTime model (Advanced Synchronization Model) augments the traditional MPEG-4 timing model
to permit synchronization of multiple streams and objects, such as video, audio, text, graphics, or even programs, that may originate from multiple sources.

The traditional MPEG-4 timing model has been designed primarily for "push" broadcast applications where temporal synchronization among access units is achieved via "hard" timestamps and reference clocks. While this mechanism provides accurate intra-stream synchronization, it falls short of providing inter-stream synchronization for streams coming from different sources (and possibly with different reference clocks) as is the case in most Internet applications and in emerging more sophisticated broadcast applications.

The FlexTime model allows the content author to specify simple temporal relationships among selected MPEG-4 objects, such as "CoStart," "CoEnd," and "Meet." The content author can also specify flexibility constraints for MPEG-4 objects, as if the objects were on stretchable springs. This allows synchronization among multiple objects according to the specified temporal relationships, while stretching or shrinking them if necessary within specified bounds.

The most immediate and beneficial impact of this new functionality will be on emerging Internet applications and enhanced broadcast applications where multiple sources need to be synchronized at the consumer device.

8.3.1 Flexible duration

In an environment of unreliable delivery it may very well happen that the delivery of a certain Elementary Stream, or portions of the stream, can be delayed beyond its required playback start time.

To be less sensitive to stream delivery delay, the FlexTime model is based upon a so-called "spring" metaphor. A spring comes with a set of 3 constants: the minimum length beyond which it won't shrink, the maximum length beyond which it will break, and the optimal length at which it may rest comfortably.

Following this spring model, Elementary Streams, or stream segments, are viewed temporally as springs, each with the corresponding 3 spring constants. The optimal spring length (the stream playback duration) can be viewed as a hint to aid a receiver to choose a particular duration when

more than one value is possible. Note, that while stretching or shrinking the duration of a continuous media such as video implies respectively slowing down or speeding up playback, when an Elementary Stream consists of a still image, shrinking or stretching is merely holding the display shorter or longer.

8.3.2 Relative start and end time

Two or more Elementary Streams or stream segments can be synchronized with respect to one another, by defining that they either start at the same time ("CoStart"), end at the same time ("CoEnd"), or the end time of one coincides with the start time of another ("Meet").

It is important to note that there are two classes of MPEG-4 objects. The timing and rendering of an MPEG-4 object that uses an Elementary stream, such as video, is not determined by the stream alone, but also by the corresponding BIFS nodes and their timing. Whereas the timing and rending of an MPEG-4 object that does not use a stream, such as text or rectangle, is determined only by the corresponding BIFS nodes and their timing.

The FlexTime model allows the content author to express synchronization among MPEG-4 objects with streams or stream segments, by assigning temporal relationships among them.

The temporal relationships (or relative timestamps) can be seen as "functional" timestamps, and their functional timestamps are resolved at playback time. Thus, a FlexTime player can:

- Compensate for various network delays by supporting a timed wait for the initial arrival of the stream, before the player starts rendering/playing the node associated with it.
- Compensate for various network jitters by supporting a timed wait for the arrival of the stream segment.
- Synchronize multiple media/BIFS nodes with some of the media stream of unknown length or uncontrolled arrival time.
- Synchronize BIFS updates (e.g. events such as field updates) among multiple nodes/streams with some of the streams of unknown length or uncontrolled arrival time.
- Slow down or speed up the rendering/playback speed of portions of streams to re-adjust out-of-sync situations caused by unknown length, uncontrolled arrival time or variation.

8.3.3 The FlexTime support in MPEG-4

The FlexTime model is supported in MPEG-4 by two nodes : TemporalTransform and TemporalGroup nodes, and a descriptor : SegmentDescriptor. The TemporalTransform node specifies temporal properties of an MPEG-4 object that needs to be synchronized (or flexed). The TemporalGroup node specifies temporal relationships among the objects that are represented by the TemporalTransform nodes, and the SegmentDescriptor identifies portions of a stream that can be synchronized.

8.3.3.1 The TemporalTransform node

The **TemporalTransform** supports synchronization of nodes within the scene to a media stream, or segment thereof, and supports flexible transformation to scene time. This grouping node can flexibly support the slowing down, speeding up, freezing or shifting of the scene time

for rendering of nodes contained within. Its **children** field may contain a list of nodes of the type SF3Dnode, and the node can effect the slowing down, speeding up, freezing or shifting the time base of the compositor when it renders the child nodes that are transformed by this node. In addition, this node has a **url** field that may reference an elementary stream or a segment thereof and in this case, the node affects the time base of the referenced stream.

8.3.3.2 The TemporalGroup node

The **TemporalGroup** node specifies the temporal relationship between a given number of **TemporalTransforms** to align in time both nodes, and media streams with nodes, in the scene graph. Temporal adjustment of media to meet the constraint and flexibility being done in the sync layer. **TemporalGroup** can examine the temporal properties of its children and when all the children are ready and the temporal constraint met can allow its children to play.

8.3.3.3 The SegmentDescriptor

An array of **SegmentDescriptor**s are added as an item in the ES_Descriptor. A **SegmentDescriptor** identifies and labels segments of a stream, so that the specific stream segments can be referenced by their url fields in the **TemporalTansform** node.

8.3.4 The Execution Model

TemporaI decoding and clock adjustment of media streams according to timestamps is a function of the sync layer. The FlexTime model requires a small change to the MPEG-4 buffer model in terms of media delivery and decoding. Decoding may be delayed on the client, beyond the standard decoding time, by an amount determined by the flexibility expressed in the relationships.

The buffer model for flextime can thus be specified as follows: “At any time from the instant of time corresponding to its DTS up to a time limit specified by Flextime, and AU is instantaneously decoded and removed from the decoding buffer.” As such the exact time of removal of the AU from the decoding buffer can vary and should not be assumed to be removed ahead of the worst case time (maximum delay for the media stream). By using the worst case time rather than the fixed DTS the decoding buffer can be managed otherwise as prescribed by MPEG-4.

To support media stream flexing the OD/ESD delete command execution shall be delayed by the same amount (or less) that the media stream it refers to has been delayed. If the OD/ESD delete is received after the object has ended then the command can be executed immediately.

8.4 Syntax Description

MPEG-4 defines a *syntactic description language* to describe the exact binary syntax for bitstreams carrying media objects and for bitstreams with scene description information. This is a departure from MPEG’s past approach of utilizing pseudo C. This language is an extension of C++, and is used to describe the syntactic representation of objects and the overall media object class definitions and scene description information in an integrated way. This provides a consistent and uniform way of describing the syntax in a very precise form, while at the same time simplifying bitstream compliance testing. Software tools can be used to process the syntactic description and generate the necessary code for programs that perform validation.

8.5 Binary Format for Scene description: BIFS

In addition to providing support for coding individual objects, MPEG-4 also provides facilities to compose a set of such objects into a scene. The necessary composition information forms the scene description, which is coded and transmitted together with the media objects. Starting from VRML (the Virtual reality Modeling Language), MPEG has developed a binary language for scene description called BIFS. BIFS stands for **BI**nary **F**ormat for **S**cenes.

In order to facilitate the development of authoring, manipulation and interaction tools, scene descriptions are coded independently from streams related to primitive media objects. Special care is devoted to the identification of the parameters belonging to the scene description. This is done by differentiating parameters that are used to improve the coding efficiency of an object (e.g., motion vectors in video coding algorithms), and the ones that are used as modifiers of an object (e.g., the position of the object in the scene). Since MPEG-4 should allow the modification of this latter set of parameters without having to decode the primitive media objects themselves, these parameters are placed in the scene description and not in primitive media objects.
The following list gives some examples of the information described in a scene description.

How objects are grouped together: An MPEG-4 scene follows a hierarchical structure, which can be represented as a directed acyclic graph. Each node of the graph is a media object, as illustrated in Figure 9 (note that this tree refers back to Figure 1). The tree structure is not necessarily static; node attributes (e.g., positioning parameters) can be changed while nodes can be added, replaced, or removed.

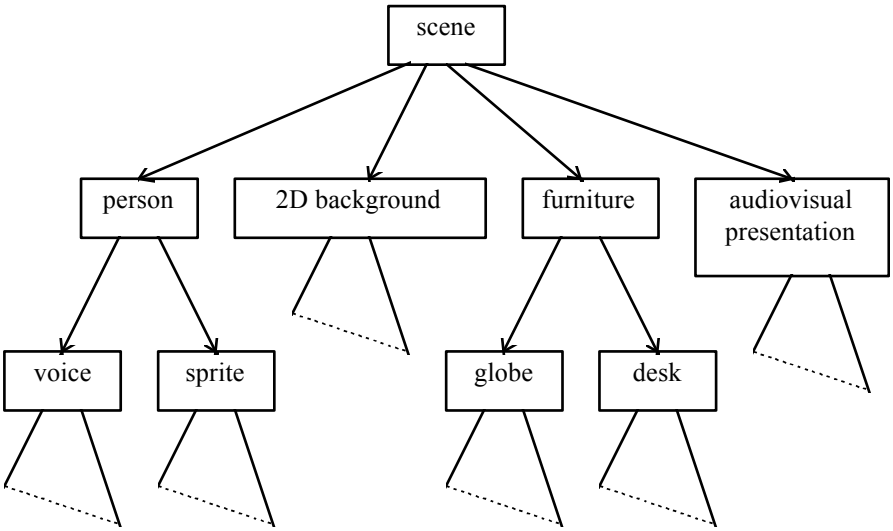


Figure 9- Logical structure of a scene

How objects are positioned in space and time: In the MPEG-4 model, audiovisual objects have both a spatial and a temporal extent. Each media object has a local coordinate system. A local coordinate system for an object is one in which the object has a fixed spatio-temporal location and scale. The local coordinate system serves as a handle for manipulating the media object in space and time. Media objects are positioned in a scene by specifying a coordinate transformation from the object's local coordinate system into a global coordinate system defined by one more parent scene description nodes in the tree.

Attribute Value Selection: Individual media objects and scene description nodes expose a set of parameters to the composition layer through which part of their behavior can be controlled.

Examples include the pitch of a sound, the color for a synthetic object, activation or deactivation of enhancement information for scaleable coding, etc.

Other transforms on media objects: As mentioned above, the scene description structure and node semantics are heavily influenced by VRML, including its event model. This provides MPEG-4 with a very rich set of scene construction operators, including graphics primitives that can be used to construct sophisticated scenes.

8.5.1 Advanced BIFS

The version 2 BIFS (Advanced BIFS) includes the following new functionalities:

- *Advanced sound environment modeling* in interactive virtual scenes, where properties such as room reflections, reverberation, Doppler effect, and sound obstruction caused by objects appearing between the source and the listener are computed for sound sources in a dynamic environment in real time. Also enhanced source directivity modeling is made possible enabling inclusion of realistic sound sources into 3-D scenes.
- Body animation of either a default body model present at the decoder or of a downloadable body model. The animation of the body is performed by sending animation parameters to it in a bitstream. (Also See Section 3.2.2)
- Chroma keying which is used to generate a shape mask and a transparency value for an image or a video sequence.
- Inclusion of hierarchical 3-D meshes to BIFS scenes.
- Associating interactive commands to media nodes. The commands are passed to server over a back channel for specified processing.
- PROTOs and EXTERNPROTOs

8.6 User interaction

MPEG-4 allows for user interaction with the presented content. This interaction can be separated into two major categories: client-side interaction and server-side interaction. Client-side interaction involves content manipulation, which is handled locally at the end-user's terminal, and can take several forms. In particular, the modification of an attribute of a scene description node, e.g., changing the position of an object, making it visible or invisible, changing the font size of a synthetic text node, etc., can be implemented by translating user events. A user event can be a mouse clicks or keyboard command) to scene description updates. The MPEG-4 terminal can process the commands in exactly the same way as if they originated from the original content source. As a result, this type of interaction does not require standardization.

Other forms of client-side interaction require support from the scene description syntax, and are specified by the standard. The use of the VRML event structure provides a rich model on which content developers can create compelling interactive content.

Server-side interaction involves content manipulation that occurs at the transmitting end, initiated by a user action. This, of course, requires that a back-channel is available.

8.7 Content-related IPR identification and protection

MPEG-4 provides mechanisms for protection of intellectual property rights (IPR), as outlined in section 1.5. This is achieved by supplementing the coded media objects with an optional Intellectual Property Identification (IPI) data set, carrying information about the contents, type of content and (pointers to) rights holders. The data set, if present, is part of an elementary stream descriptor that describes the streaming data associated to a media object. The number of data sets

to be associated with each media object is flexible; different media objects can share the same data sets or have separate data sets. The provision of the data sets allows the implementation of mechanisms for audit trail, monitoring, billing, and copy protection.

Next to identifying rights, each of the wide range of MPEG-4 applications has a set of requirements regarding protection of the information it manages. These applications can have different security requirements. For some applications, users exchange information that has no intrinsic value but that must still be protected to preserve various rights of privacy. For other applications, the managed information has great value to its creator and/or distributors requiring high-grade management and protection mechanisms. The implication is that the design of the IPMP framework must consider the complexity of the MPEG-4 standard and the diversity of its applications. This IPMP framework leaves the details of IPMP systems designs in the hands of applications developers. The level and type of management and protection required depends on the content's value, complexity, and the sophistication of the associated business models.

The approach taken allows the design and use of domain-specific IPMP systems (IPMP-S). While MPEG-4 does not standardize IPMP systems themselves, it does standardize the MPEG-4 IPMP interface. This interface consists of IPMP-Descriptors (IPMP-Ds) and IPMP-Elementary Streams (IPMP-ES).

IPMP-Ds and IPMP-ESs provide a communication mechanism between IPMP systems and the MPEG-4 terminal. Certain applications may require multiple IPMP systems. When MPEG-4 objects require management and protection, they have IPMP-Ds associated with them. These IPMP-Ds indicate which IPMP systems are to be used and provide information to these systems about how to manage and protect the content. (See Figure 10)

Besides enabling owners of intellectual property to manage and protect their assets, MPEG-4 provides a mechanism to identify those assets via the Intellectual Property Identification Data Set (IPI Data Set). This information can be used by IPMP systems as input to the management and protection process.

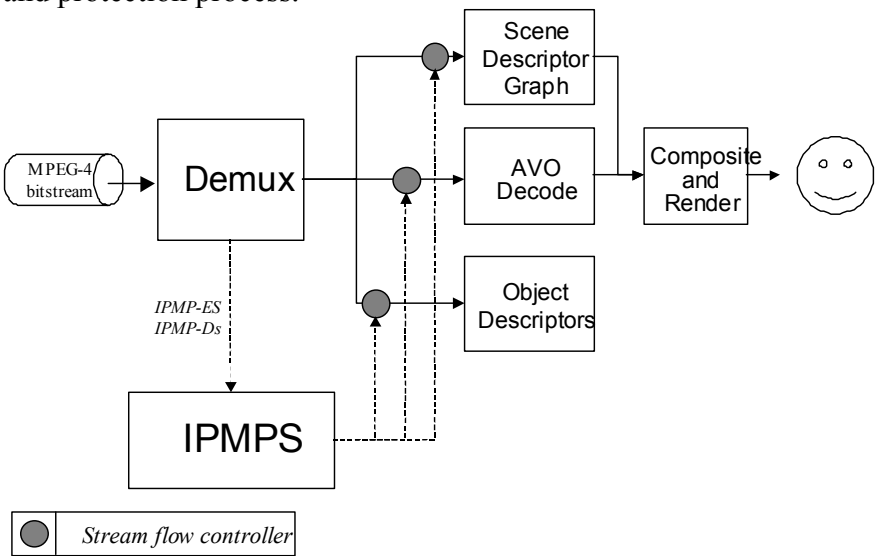


Figure 10 - The IPMP Interfaces within an MPEG-4 System

8.8 Object Content Information

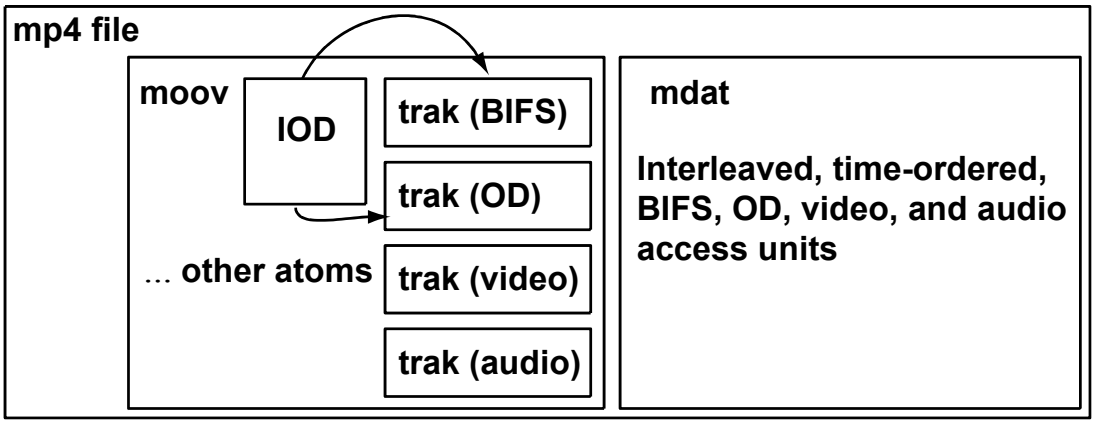
MPEG-4 allows attaching information to objects about their content. Users of the standard can use this 'OCI' (Object Content Information) data stream to send textual information along with MPEG-4 content. It is also possible to classify content according to pre-defined tables, which

will be defined outside of MPEG. Further possibilities are giving unique labels to content, and storing camera parameters.

8.9 MPEG-4 File Format

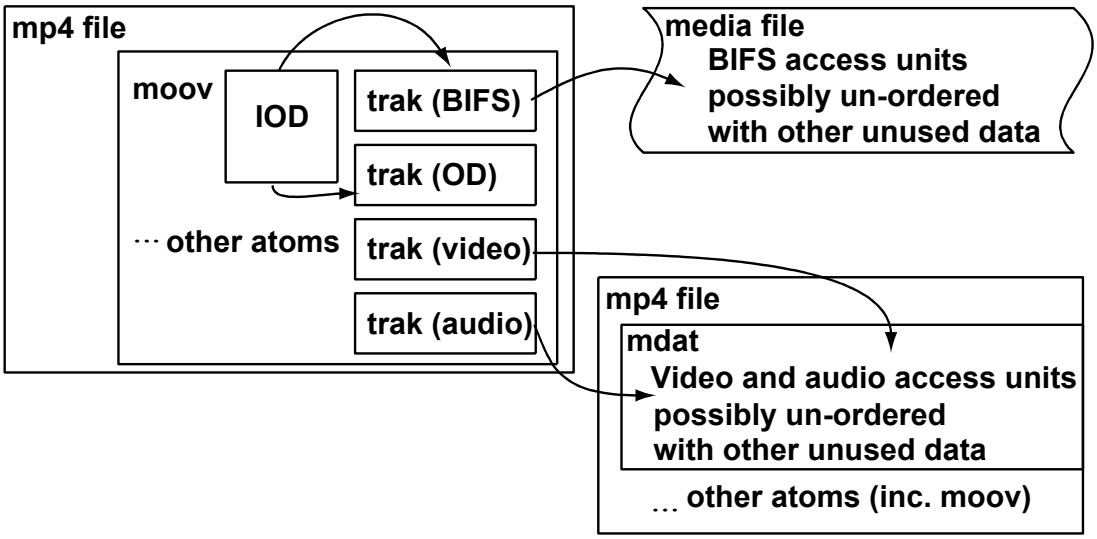
The MP4 file format is designed to contain the media information of an MPEG-4 presentation in a flexible, extensible format which facilitates interchange, management, editing, and presentation of the media. This presentation may be 'local' to the system containing the presentation, or may be via a network or other stream delivery mechanism (a TransMux). The file format is designed to be independent of any particular delivery protocol while enabling efficient support for delivery in general. The design is based on the QuickTime® format from Apple Computer Inc.

The following diagram gives an example of a simple interchange file, containing three streams.

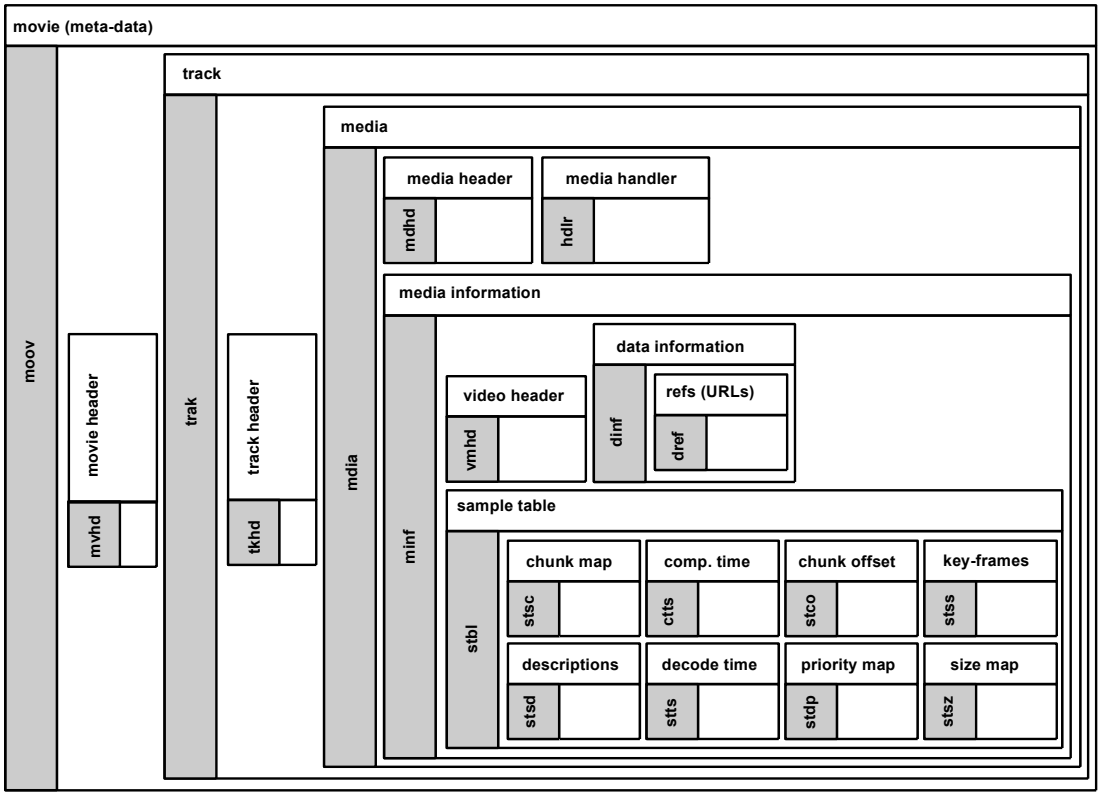


The MP4 file format is composed of object-oriented structures called 'atoms'. A unique tag and a length identify each atom. Most atoms describe a hierarchy of metadata giving information such as index points, durations, and pointers to the media data. This collection of atoms is contained in an atom called the 'movie atom'. The media data itself is located elsewhere; it can be in the MP4 file, contained in one or more 'mdat' or media data atoms, or located outside the MP4 file and referenced via URL's.

The following diagram shows a more complex file with external media data:

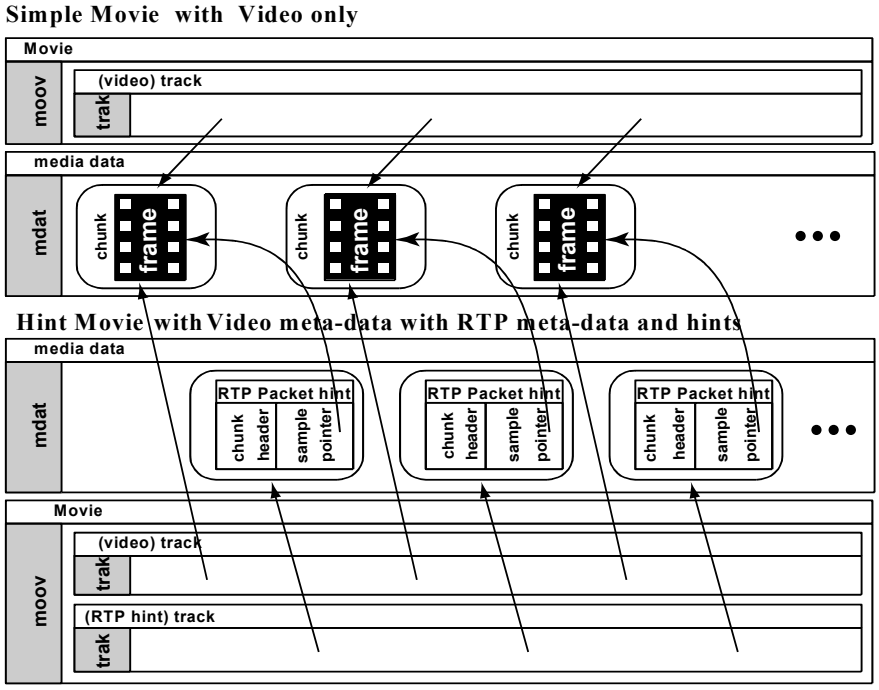


The following diagram shows the container relationship between the different objects or atoms:



The file format is a streamable format, as opposed to a streaming format. That is, the file format does not define an on-the-wire protocol, and is never actually streamed over a transmission medium. Instead, metadata in the file known as ‘hint tracks’ provide instructions, telling a server application how to deliver the media data over a particular delivery protocol. There can be multiple hint tracks for one presentation, describing how to deliver over various delivery protocols. In this way, the file format facilitates streaming without ever being streamed directly

The following diagram shows the container relationship with RTP protocol hint tracks to stream a simple video movie:



The metadata in the file, combined with the flexible storage of media data, allows the MP4 format to support streaming, editing, local playback, and interchange of content, thereby satisfying the requirements for the MPEG4 Intermedia format.

8.10 MPEG-J

The MPEG-J is a programmatic system (as opposed to the parametric system offered by MPEG-4 Version 1) which specifies API for interoperation of MPEG-4 media players with Java code. By combining MPEG-4 media and safe executable code, content creators may embed complex control and data processing mechanisms with their media data to intelligently manage the operation of the audio-visual session. A block diagram of the MPEG-J player in an MPEG-4 system player environment is shown in Figure 11. The lower half of this drawing depicts the parametric MPEG-4 System player also referred to as the Presentation Engine. The MPEG-J subsystem controlling the Presentation Engine, also referred to as the Application Engine, is depicted in the upper half of Figure 11.

The Java application is delivered as a separate elementary stream to the MPEG-4 terminal. There it will be directed to the MPEG-J run time environment, from where the MPEG-J program will have access to the various components and data of the MPEG-4 player, in addition to .the basic packages of the language (java.lang, java.io, java.util). MPEG-J specifically does *not* support downloadable decoders.

For the above-mentioned reason, the group has defined a set of APIs with different scopes. For Scene graph API the objective is to provide access to the scene graph: to inspect the graph, to alter nodes and their fields, and to add and remove nodes within the graph. The Resource Manager API is used for regulation of performance: it provides a centralized facility for managing resources. The Terminal Capability API is used when program execution is contingent upon the terminal configuration and its capabilities, both static (that do not change during

execution) and dynamic. Media Decoders API allow the control of the decoders that are present in the terminal. The Network API provides a way to interact with the network, being compliant to the MPEG-4 DMIF Application Interface. Complex applications and enhanced interactivity are possible with these basic packages.

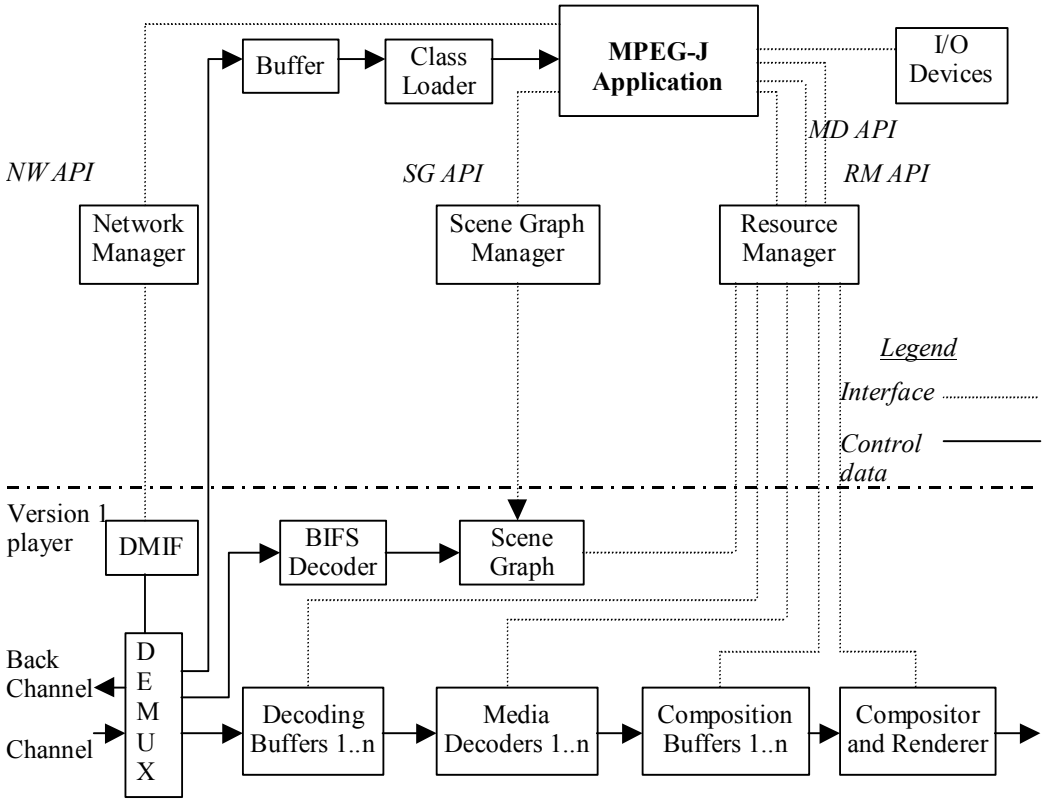


Figure 11 - Location of interfaces in the architecture of an MPEG-J enabled MPEG-4 System

9. Detailed technical description of MPEG-4 Visual

Visual objects can be either of natural or of synthetic origin. First, the objects of natural origin are described. Then follow the synthetic ones.

9.1 Applications of the MPEG-4 Video Standard

MPEG-4 Video offers technology that covers a large range of existing applications as well as new ones. The low-bit rate and error resilient coding allows for robust communication over limited rate wireless channels, useful for e.g. mobile videophones and space communication. There may also be roles in surveillance data compression since it is possible to have a very low or variable frame rate. At high bit-rates, tools are available to allow the transmission and storage of high-quality video suitable for the studio and other very demanding content creation applications. It is likely that the standard will eventually support data-rates well beyond those of MPEG-2.

A major application area is interactive web-based video. Software that provides live MPEG-4 video on a web-page has already been demonstrated. There is much room for applications to make use of MPEG-4's object-based characteristics. The binary and grayscale shape-coding tools allow arbitrary-shaped video objects to be composed together with text and graphics. Doing so, a rich interactive experience for web-based presentations and advertising can be provided; this same scenario also applies to set-top box applications. Additionally, it is possible to make use of scalability tools to allow for a smooth control of user experience with terminal and data link capabilities.

MPEG-4 video has already been used to encode video captures with a hand-held camera. This form of application is likely to grow in popularity with its fast and easy transfer to a web-page, and may also make use of MPEG-4 still-texture mode for still frame capture. The games market is another area where the application of MPEG-4 video, still-texture, interactivity and SNHC shows much promise, with 3-D texture mapping of still images, live video, or extended pre-recorded video sequences enhancing the player experience. Adding live video of users adds to the user experience multi-player 3-D games, as does use of arbitrary-shaped video, where transparency could be combined artistically with 3-D video texture mapping.
The flexibility of MPEG-4 video coding encourages many more applications like this one.

9.2 Natural Textures, Images and Video

The tools for representing natural video in the MPEG-4 visual standard provide standardized core technologies allowing efficient storage, transmission and manipulation of textures, images and video data for multimedia environments. These tools allow the decoding and representation of atomic units of image and video content, called “video objects” (VOs). An example of a VO could be a talking person (without background), which can then be composed with other AVOs (audio-visual objects) to create a scene. Conventional rectangular imagery is handled as a special case of such objects.

In order to achieve this broad goal rather than a solution for a narrow set of applications, functionalities common to several applications are clustered. Therefore, the visual part of the MPEG-4 standard provides solutions in the form of tools and algorithms for:

- efficient compression of images and video
- efficient compression of textures for texture mapping on 2-D and 3-D meshes
- efficient compression of implicit 2-D meshes
- efficient compression of time-varying geometry streams that animate meshes
- efficient random access to all types of visual objects
- extended manipulation functionality for images and video sequences
- content-based coding of images and video
- content-based scalability of textures, images and video
- spatial, temporal and quality scalability
- error robustness and resilience in error prone environments

9.3 Synthetic Objects

Synthetic objects form a subset of the larger class of computer graphics, as an initial focus the following visual synthetic objects will be described:

- Parametric descriptions of
  a) a synthetic the face and body (body animation in Version 2)
  b) Static and Dynamic Mesh Coding with texture mapping
- Texture Coding for View Dependent applications

9.4 Scalable Coding of Video Objects

There are several scalable coding schemes in MPEG-4 Visual: spatial scalability, temporal scalability and object-based spatial scalability. Spatial scalability supports changing the texture quality (SNR and spatial resolution). Object-based spatial scalability extends the 'conventional' types of scalability towards arbitrary shape objects, so that it can be used in conjunction with other object-based capabilities. Thus, a very flexible content-based scaling of video information can be achieved. This makes it possible to enhance SNR, spatial resolution, shape accuracy, etc, only for objects of interest or for a particular region, which can een be done dynamically at play-time.

9.5 Robustness in Error Prone Environments

A newly developed technique in MPEG, called NEWPRED (for 'new prediction'), provides a fast error recovery in real-time coding applications. It uses an upstream channel from the decoder to the encoder. The encoder switches the reference frames adaptively according to the error conditions of the network. NEWPRED does not use intra refresh and it provides the high coding efficiency. This technique has been proven to work under stressful error conditions:

- Burst Error on the wireless networks (averaged bit error rate is 10E-3, 1ms burst length)
- Packet Loss on the internet (packet loss rate is 5%)

9.6 Improved temporal resolution stability with low buffering delay

Another new technique is Dynamic Resolution Conversion (DRC), a way to stabilize the transmission buffering delay by minimizing the jitter of the amount of the coded output bits per VOP. Large frame skips are also prevented and the encoder can control the temporal resolution even in highly active scenes.

9.7 Coding of Textures and Still Images

The following three new tools for coding of textures and still images are provided in V.2:

- *Wavelet tiling* allows an image to be divided into several tiles and each tile to be encoded independently. This means that large images to be encoded/decoded with very low memory requirements, and that random access at the decoder is significantly enhanced.
- *Scalable shape coding* allows encoding of arbitrary shaped textures and still images in a scalable fashion. Using this tool, a decoder can decode an arbitrary shaped image at any desired resolution. This tool enables applications to employ object-based, spatial and quality scalabilities at the same time.
- The *Error resilience tool* adds new error resiliency features. Using packetization and segment marker techniques, it significantly improves the error robustness in applications such as image transmission over mobile and Internet.

The above new tools are used in the two new ‘advanced scalable texture’ and ‘advanced core’ profiles. (See Section 3.4)

9.8 Coding of Multiple Views and Multiple Auxiliary Components

In MPEG-4 Video Version 1 up to one alpha channel per Video Object Layer is supported and in total three shape types are defined. All the three shape types, i.e. binary shape, constant shape and gray scale shape, indicate the **transparency** of the video object. With this definition MPEG-4 cannot support things like **Multiview Video Objects** in an efficient way. Therefore in Version 2 the use of multiple alpha channels to transmit auxiliary components is introduced.
The basic idea is that the gray scale shape is not only used to describe the transparency of the video object, but can be defined in a more general way. A gray scale shape may e.g. represent:

- Transparency shape
- Disparity shape for Multiview Video Objects (horizontal and vertical)
- Depth shape (acquired by laser range finder or by disparity analysis)
- Infrared or other secondary texture

All the alpha channels can be encoded by the shape coding tools, i.e. the binary shape coding tool and the gray scale shape coding tool which employs a motion-compensated DCT, and usually have the same shape and resolution as the texture of the video object.
As an example for employing multiple auxiliary components, the usage of the disparity shape for Multiview Video Objects is described below.
The general concept is to limit the number of pixels that have to be encoded by analysis of the correspondences between the particular views of an object available at the encoder side. All areas of an object that are visible within more than one camera view are encoded only once with

the highest possible resolution. The disparity correspondences can be estimated from the original views to reconstruct all the areas that were excluded from encoding by use of disparity-compensated projection. One or two auxiliary components can be allocated to encode the disparity maps, indicating the correspondences between the pixels within multiple views.
We denominate the areas which are retained for encoding from each of the particular camera views as the "areas of interest" (AOI). These AOIs can now simply be defined as MPEG-4 video objects, and are encoded with their associated disparity values. Due to possible reflection effects in the different views, but as well due to color or exposure deviations between the cameras, borders between the areas which are to be reconstructed from different original views might become visible. To circumvent this problem, it is necessary to preprocess pixels near the borders of an AOI, such that a smooth transition is achieved by interpolating pixels from different adjacent views within a transition area.
To reconstruct different viewpoints from the texture surface disparity-compensated projection is performed from the texture data within the particular AOIs, with disparities from the disparity map available within the auxiliary component decoded from the MPEG-4 video stream. Each of the AOIs is processed separately and then the projected images from all AOIs are assembled to obtain the final view towards the video object from the selected viewpoint. This procedure can be employed for a system with two cameras with parallel setup, but is also extendable to convergent and multiple-camera cases as well.

9.8.1 facial animation

The 'facial animation object' can be used to render an animated face. The shape, texture and expressions of the face are controlled by Facial Definition Parameters (FDPs) and/or Facial Animation Parameters (FAPs). Upon construction, the face object contains a generic face with a neutral expression. This face can already be rendered. It can also immediately receive the animation parameters from the bitstream, which will produce animation of the face: expressions, speech etc. Meanwhile, definition parameters can be sent to change the appearance of the face from something generic to a particular face with its own shape and (optionally) texture. If so desired, a complete face model can be downloaded via the FDP set.

Face Animation in MPEG-4 Version 1 provides for highly efficient coding of animation parameters that can drive an unlimited range of face models. The models themselves are not normative, although (see above) there are normative tools to describe the appearance of the model. Frame-based and temporal-DCT coding of a large collection of FAPs can be used for accurate speech articulation. Viseme and expression parameters are used to code specific speech configurations of the lips and the mood of the speaker.

The Systems Binary Format for Scenes (BIFS, see section 2.6) provides features to support Face Animation when custom models and specialized interpretation of FAPs are needed:

1 the Face Definition Parameters (FDP) in BIFS (model data downloadable to configure a baseline face model pre-stored in the terminal into a particular face before FAP decoding, or to install a specific face model at the beginning of a session along with the information about how to animate it);
2 the Face Animation Table (FAT) within FDPs (downloadable functional mapping from incoming FAPs to feature control points in the face mesh. This provides piecewise linear mappings of incoming FAPs for controlling facial movements. Example: the FAP could say 'open_jaw (500) and the table then defines what this means in terms of moving the feature points;
3 the Face Interpolation Technique (FIT) in BIFS (downloadable definition of mapping of incoming FAPs into a total set of FAPs before their application to feature points, through

weighted rational polynomial functions invoked by conditional evaluation of a Face Interpolation Graph). This can be used for complex cross-coupling of FAPs to link their effects, or to interpolate FAPs missing in the stream using the FAPs that are available in the terminal).

These specialized node types in BIFS effectively provide for tailored face models including calibration of an established face model in a terminal or downloading of a fully custom model including its shape, texture, and color.

9.8.2 Body animation

The Body is an object capable of producing virtual body models and animations in form of a set of 3-D polygonal meshes ready for rendering. Two sets of parameters are defined for the body: Body Definition Parameter (BDP) set, and Body Animation Parameter (BAP) set. The BDP set defines the set of parameters to transform the default body to a customized body with its body surface, body dimensions, and (optionally) texture. The Body Animation Parameters (BAPs), if correctly interpreted, will produce reasonably similar high level results in terms of body posture and animation on different body models, without the need to initialize or calibrate the model.

Upon construction, the Body object contains a generic virtual human body with the default posture. This body can already be rendered. It is also immediately capable of receiving the BAPs from the bitstream, which will produce animation of the body. If BDPs are received, they are used to transform the generic body into a particular body determined by the parameters contents. Any component can be null. A null component is replaced by the corresponding default component when the body is rendered. The default posture is defined by standing posture. This posture is defined as follows: the feet should point to the front direction, the two arms should be placed on the side of the body with the palm of the hands facing inward. This posture also implies that all BAPs have default values.

No assumption is made and no limitation is imposed on the range of motion of joints. In other words the human body model should be capable of supporting various applications, from realistic simulation of human motions to network games using simple human-like models. The work on Body Animation includes the assessment of the emerging standard as applied to hand signing for the listening-impaired.

The Body Animation standard has been developed by MPEG in concert with the Humanoid Animation Working Group within the VRML Consortium, with the objective of achieving consistent conventions and control of body models which are being established by H-Anim.

9.8.3 2-D animated meshes

A 2-D *mesh* is a tessellation (or partition) of a 2-D planar region into polygonal patches. The vertices of the polygonal patches are referred to as the *node points* of the mesh. MPEG4 considers only triangular meshes where the patches are triangles. A 2-D dynamic mesh refers to 2-D mesh geometry and motion information of all mesh node points within a temporal segment of interest. Triangular meshes have long been used for efficient 3-D object shape (geometry) modeling and rendering in computer graphics. 2-D mesh modeling may be considered as projection of such 3-D triangular meshes onto the image plane. An example of a 2-D mesh is depicted in Figure 12.

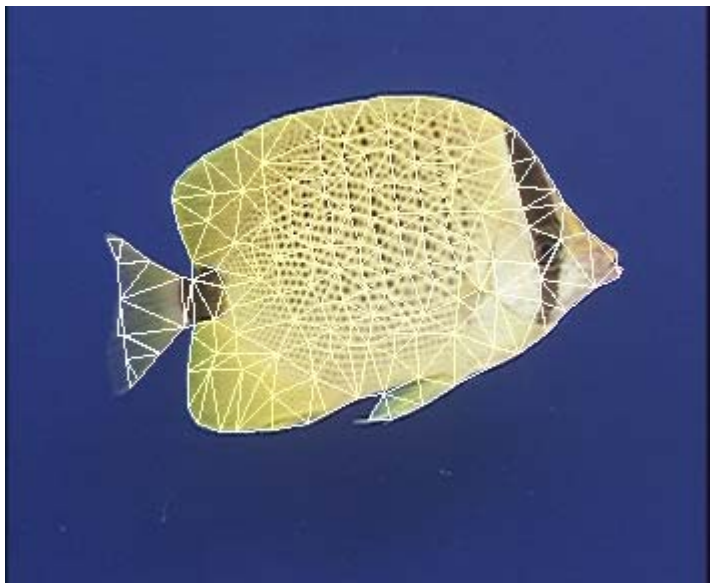


Figure 12- 2-D mesh modeling of the "Bream" video object.
By deforming the mesh, the fish can be animated very efficiently, and be made to 'swim'. Also, a logo could be projected onto the fish, and made to move in accordance with the fish
A dynamic mesh is a forward tracking mesh, where the node points of the initial mesh track image features forward in time by their respective motion vectors. The initial mesh may be regular, or can be adapted to the image content, which is called a *content-based mesh*. 2-D content-based mesh modeling then corresponds to non-uniform sampling of the motion field at a number of salient feature points (node points) along the contour and interior of a video object. Methods for selection and tracking of these node points are not subject to standardization.

In 2-D mesh based texture mapping, triangular patches in the current frame are deformed by the movements of the node points into triangular patches in the reference frame. The texture inside each patch in the reference frame is *warped* onto the current frame using a parametric mapping, defined as a function of the node point motion vectors. For triangular meshes, the affine mapping is a common choice. Its linear form implies texture mapping with low computational complexity. Affine mappings can model translation, rotation, scaling, reflection and shear, and preserve straight lines. The degrees of freedom given by the three motion vectors of the vertices of a triangle match with the six parameters of the affine mapping. This implies that the original 2-D motion field can be compactly represented by the motion of the node points, from which a continuous, piece-wise affine motion field can be reconstructed. At the same time, the mesh structure constrains movements of adjacent image patches. Therefore, meshes are well-suited to represent mildly deformable but spatially continuous motion fields.

2-D mesh modeling is attractive because 2-D meshes can be designed from a single view of an object without requiring range data, while maintaining several of the functionalities offered by 3-D mesh modeling. In summary, the 2-D object-based mesh representation is able to model the shape (polygonal approximation of the object contour) and motion of a VOP in a unified framework, which is also extensible to the 3-D object modeling when data to construct such models is available. In particular, the 2-D mesh representation of video objects enables the following functionalities:

A. Video Object Manipulation

- Augmented reality: Merging virtual (computer generated) images with real moving images (video) to create enhanced display information. The computer-generated images must remain in perfect registration with the moving real images (hence the need for tracking).

- Synthetic-object-transfiguration/animation: Replacing a natural video object in a video clip by another video object. The replacement video object may be extracted from another natural video clip or may be transfigured from a still image object using the motion information of the object to be replaced (hence the need for a temporally continuous motion representation).
- Spatio-temporal interpolation: Mesh motion modeling provides more robust motion-compensated temporal interpolation (frame rate up-conversion).

B. Video Object Compression

- 2-D mesh modeling may be used for compression if one chooses to transmit texture maps only at selected key frames and animate these texture maps (without sending any prediction error image) for the intermediate frames. This is also known as self-transfiguration of selected key frames using 2-D mesh information.

C. Content-Based Video Indexing

- Mesh representation enables animated key snapshots for a moving visual synopsis of objects.
- Mesh representation provides accurate object trajectory information that can be used to retrieve visual objects with specific motion.
- Mesh representation provides vertex-based object shape representation which is more efficient than the bitmap representation for shape-based object retrieval.

9.8.4 3D Meshes

Capabilities for 3-D mesh coding include:

- Coding of generic 3-D polygonal meshes enables the efficient encoding of 3-D polygonal meshes. The coded representation is generic enough to support both manifold and non-manifold meshes.
- Incremental representation enables a decoder to reconstruct a number faces in a mesh proportional to the number of bits in the bit stream that have been processed. This furthermore enables incremental rendering.
- Error resilience enables a decoder to partially recover a mesh when subsets of the bit stream are missing and/or corrupted.
- LOD (Level Of Detail) scalability enables a decoder to reconstruct a simplified version of the original mesh containing a reduced number of vertices from a subset of the bit stream. Such simplified representations are useful to reduce the rendering time of objects which are distant from the viewer (LOD management), but also enable less powerful rendering engines to render the object at a reduced quality.

9.8.5 View-dependent scalability

The view-dependent scalability enables to stream texture maps, which are used in realistic virtual environments. It consists in taking into account the viewing position in the 3-D virtual world in order to transmit only the most visible information. Only a fraction of the information is then sent, depending on object geometry and viewpoint displacement. This fraction is computed both at the encoder and at the decoder side. This approach allows to reduce greatly the amount of transmitted information between a remote database and a user, given that a back-channel is available. This scalability can be applied both with Wavelet and DCT based encoders.

9.9 Structure of the tools for representing natural video

The MPEG-4 image and video coding algorithms give an efficient representation of visual objects of arbitrary shape, also supporting so-called content-based functionalities. They support most functionalities already provided by MPEG-1 and MPEG-2, including efficient compression

of standard rectangular sized image sequences at varying levels of input formats, frame rates, pixel depth, bit-rates, and various levels of spatial, temporal and quality scalability.

A basic classification of the bit rates and functionalities currently provided by the MPEG-4 Visual standard for natural images and video is depicted in Figure 13 below, which clusters bit-rate levels versus sets of functionalities.

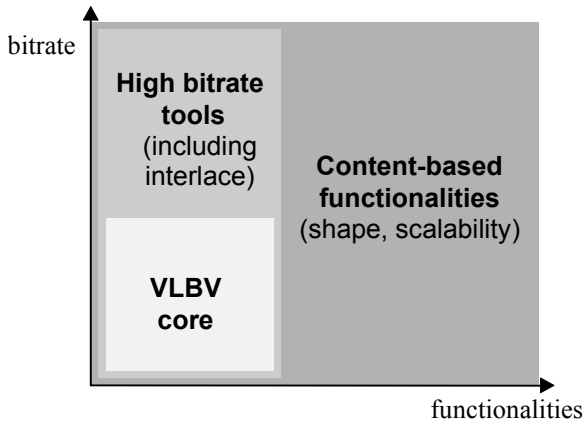


Figure 13 - Classification of the MPEG-4 Image and Video Coding Algorithms and Tools

At the bottom end a "VLBV Core" (VLBV: Very Low Bit-rate Video) provides algorithms and tools for applications operating at bit-rates typically between 5...64 kbits/s, supporting image sequences with low spatial resolution (typically up to CIF resolution) and low frame rates (typically up to 15 Hz). The basic applications specific functionalities supported by the VLBV Core include:

a) coding of conventional rectangular size image sequences with high coding efficiency and high error robustness/resilience, low latency and low complexity for real-time multimedia communications applications, and

b) "random access" and "fast forward" and "fast reverse" operations for VLB multimedia data-base storage and access applications.

The same basic functionalities outlined above are also supported at higher bit-rates with a higher range of spatial and temporal input parameters up to ITU-R Rec. 601 resolutions and larger - employing identical or similar algorithms and tools as the VLBV Core. The bit-rates envisioned range typically from 64 kbits/s up to 10 Mb/s and applications envisioned include multimedia broadcast or interactive retrieval of signals with a quality comparable to digital TV. For these applications at higher bit-rates, also interlaced can be represented by MPEG-4 coding tools.

Content-based functionalities support the separate encoding and decoding of content (i.e. physical objects in a scene, VOs). This MPEG-4 feature provides the most elementary mechanism for interactivity; flexible representation and manipulation with/of VO content of images or video in the compressed domain, without the need for further segmentation or transcoding at the receiver.

For the hybrid coding of natural as well as synthetic visual data (e.g. for virtual presence or virtual environments) the content-based coding functionality allows mixing a number of VO's from different sources with synthetic objects, such as a virtual backgrounds.

The extended MPEG-4 algorithms and tools for content-based functionalities can be seen as a superset of the VLBV core and high bit-rate tools - meaning that the tools provided by the VLBV and higher bitrate cores are complemented by additional elements.

9.10 Support for Conventional and Content-Based Functionalities

As mentioned before, MPEG-4 Video supports conventional rectangular images and video as well as images and video of arbitrary shape. This concept is illustrated in Figure 14 below.

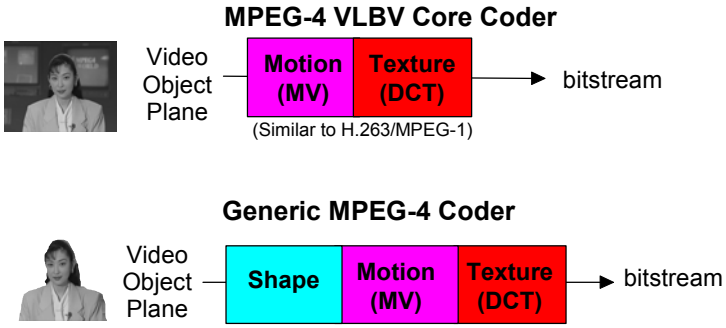


Figure 14 - the VLBV Core and the Generic MPEG-4 Coder

The coding of conventional images and video is similar to conventional MPEG-1/2 coding. It involves motion prediction/compensation followed by texture coding. For the content-based functionalities, where the image sequence input may be of arbitrary shape and location, this approach is extended by also coding shape and transparency information. Shape may be either represented by an 8 bit transparency component - which allows the description of transparency if one VO is composed with other objects - or by a binary mask.

The extended MPEG-4 content-based approach can be seen as a logical extension of the conventional MPEG-4 VLBV Core or high bit-rate tools towards input of arbitrary shape.

9.11 The MPEG-4 Video Image and Coding Scheme

Figure 15 below outlines the basic approach of the MPEG-4 video algorithms to encode rectangular as well as arbitrarily shaped input image sequences.

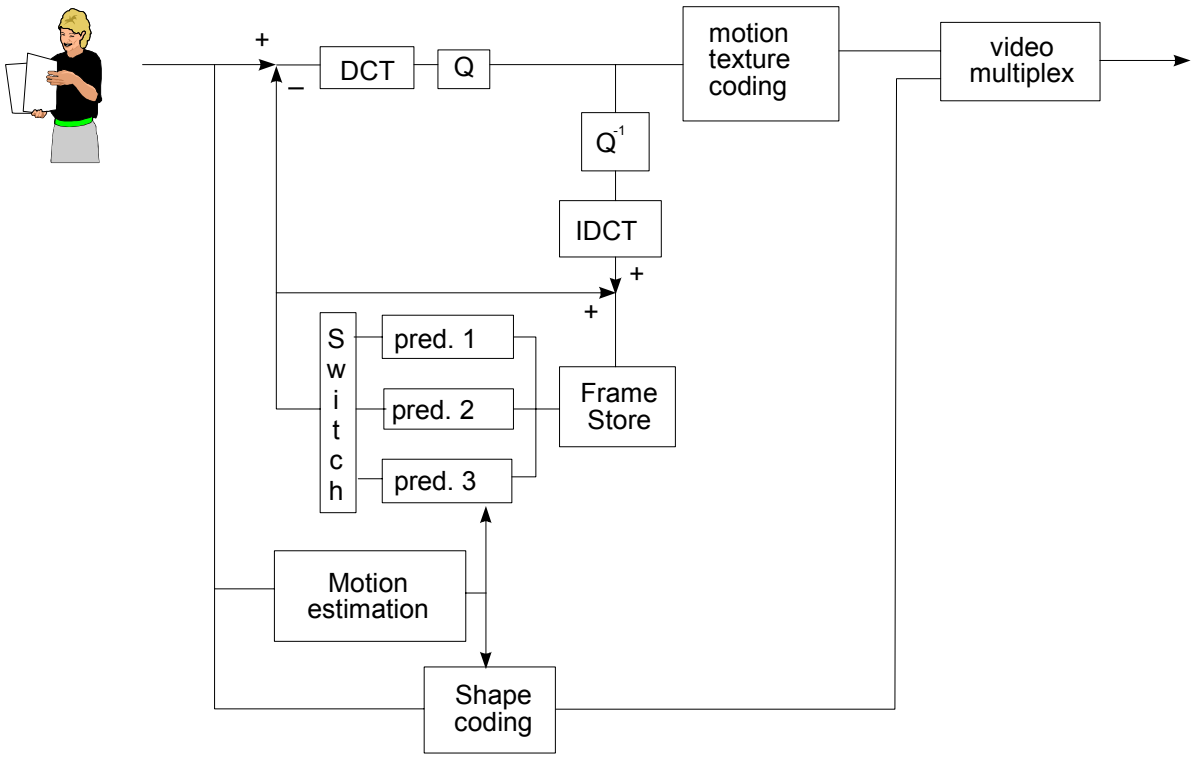


Figure 15 - Basic block diagram of MPEG-4 Video Coder

The basic coding structure involves shape coding (for arbitrarily shaped VOs) and motion compensation as well as DCT-based texture coding (using standard 8x8 DCT or shape adaptive DCT).

The basic coding structure involves shape coding (for arbitrarily shaped VOs) and motion compensation as well as DCT-based texture coding (using standard 8x8 DCT or shape adaptive DCT).

An important advantage of the content-based coding approach MPEG-4 is that the compression efficiency can be significantly improved for some video sequences by using appropriate and dedicated object-based motion prediction "tools" for each object in a scene. A number of motion prediction techniques can be used to allow efficient coding and flexible presentation of the objects:

- Standard 8x8 or 16x16 pixel block-based motion estimation and compensation.
- Global motion compensation based on the transmission of a static "sprite". A static sprite is a possibly large still image, describing panoramic background. For each consecutive image in a sequence, only 8 global motion parameters describing camera motion are coded to reconstruct the object. These parameters represent the appropriate affine transform of the sprite transmitted in the first frame.

Figure 16 depicts the basic concept for coding an MPEG-4 video sequence using a sprite panorama image. It is assumed that the foreground object (tennis player, image top right) can be segmented from the background and that the sprite panorama image can be extracted from the sequence prior to coding. (A sprite panorama is a still image that describes as a static image the content of the background over all frames in the sequence). The large panorama sprite image is transmitted to the receiver only once as first frame of the sequence to describe the background – the sprite remains is stored in a sprite buffer. In each consecutive frame only the camera

parameters relevant for the background are transmitted to the receiver. This allows the receiver to reconstruct the background image for each frame in the sequence based on the sprite. The moving foreground object is transmitted separately as an arbitrary-shape video object. The receiver composes both the foreground and background images to reconstruct each frame (bottom picture in figure below). For low delay applications it is possible to transmit the sprite in multiple smaller pieces over consecutive frames or to build up the sprite at the decoder progressively.

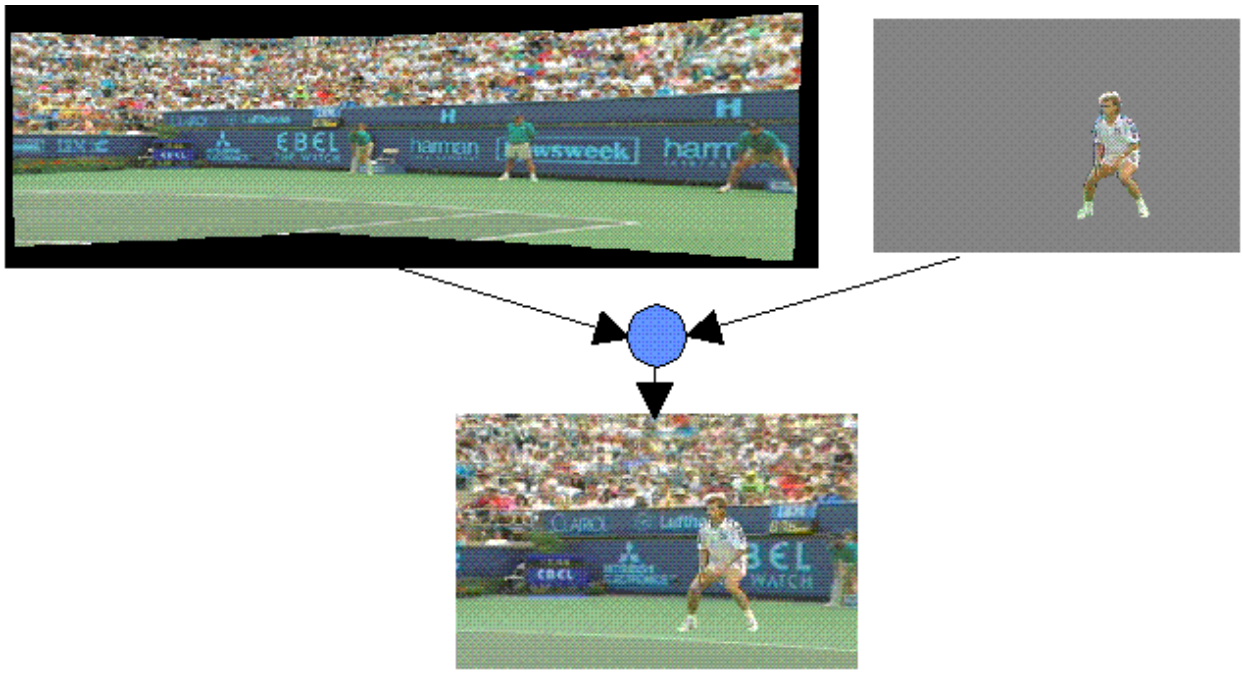


Figure 16 - Example of Sprite Coding of Video Sequence

9.11.1 Coding Efficiency in V.2

MPEG-4 V.2 improves the motion estimation and compensation of rectangular and arbitrary shaped objects and the texture coding of arbitrary shaped objects significantly. In the area of motion estimation and compensation two new techniques are introduced:

- Global Motion Compensation (GMC): Encoding of the global motion for a object using a small number of parameters. GMC is based on global motion estimation, image warping, motion trajectory coding, and texture coding for prediction errors.
- Quarter Pel Motion Compensation enhances the precision of the motion compensation scheme, at the cost of only small syntactical and computational overhead. A accurate motion description leads to a smaller prediction error and, hence, to better visual quality.

In the area of texture coding the shape-adaptive DCT (SA-DCT) improves the coding efficiency of arbitrary shaped objects. The SA-DCT algorithm is based on predefined orthonormal sets of one-dimensional DCT basis functions.
Subjective evaluation tests within MPEG have shown that the combination of these techniques can result in a bitstream saving of up to 50% compared with the version 1, depending on content type and datarate.

9.12 Coding of Textures and Still Images

Efficient Coding of visual textures and still images (e.g. to be mapped on animated meshes) is supported by the visual texture mode of the MPEG-4. This mode is based on a zerotree wavelet algorithm that provides very high coding efficiency over a very wide range of bitrates. Together with high compression efficiency, it also provides spatial and quality scalabilities (up to 11

levels of spatial scalability and continuous quality scalability) and also arbitrary-shaped object coding. The wavelet formulation provides for scalable bitstream coding in the form of an image resolution pyramid for progressive transmission and temporal enhancement of still images. The coded bitstream is also intended for downloading of the image resolution hierarchy into the terminal to be formatted as 'MIPmap texture' as used in 3-D rendering systems. This technology provides the resolution scalability to deal with a wide range of viewing conditions more typical of interactive applications and the mapping of imagery into 2-D and 3-D virtual worlds.

9.13 Scalable Coding of Video Objects

MPEG-4 supports the coding of images and video objects with spatial and temporal scalability, both with conventional rectangular as well as with arbitrary shape. Scalability refers to the ability to only decode a part of a bitstream and reconstruct images or image sequences with:

- reduced decoder complexity and thus reduced quality
- reduced spatial resolution
- reduced temporal resolution
- with equal temporal and spatial resolution but with reduced quality.

This functionality is desired for progressive coding of images and video sent over heterogeneous networks, as well as for applications where the receiver is not capable of displaying the full resolution or full quality images or video sequences. This could for instance happen when processing power or display resolution is limited.

For decoding of still images, the MPEG-4 standard will provide spatial scalability with up to 11 levels of granularity and also quality scalability up to the bit level. For video sequences a initial maximum of 3 levels of granularity will be supported, but work is ongoing to raise this level to 9.

9.14 Robustness in Error Prone Environments

MPEG-4 provides error robustness and resilience to allow accessing image or video information over a wide range of storage and transmission media. In particular, due to the rapid growth of mobile communications, it is extremely important that access is available to audio and video information via wireless networks. This implies a need for useful operation of audio and video compression algorithms in error-prone environments at low bit-rates (i.e., less than 64 kbit/s).

The error resilience tools developed for MPEG-4 can be divided into three major areas: resynchronization, data recovery, and error concealment. It should be noted that these categories are not unique to MPEG-4, but instead have been used by many researchers working in the area error resilience for video. It is, however, the tools contained in these categories that are of interest, and where MPEG-4 makes its contribution to the problem of error resilience.

9.14.1 Resynchronization

Resynchronization tools attempt to enable resynchronization between the decoder and the bitstream after a residual error or errors have been detected. Generally, the data between the synchronization point prior to the error and the first point where synchronization is reestablished, is discarded. If the resynchronization approach is effective at localizing the amount of data discarded by the decoder, then the ability of other types of tools that recover data and/or conceal the effects of errors is greatly enhanced.

The resynchronization approach adopted by MPEG-4, referred to as a packet approach, is similar to the Group of Blocks (GOBs) structure utilized by the ITU-T standards of H.261 and H.263. In these standards a GOB is defined as one or more rows of macroblocks (MBs). At the start of a

new GOB, information called a GOB header is placed within the bitstream. This header information contains a GOB start code, which is different from a picture start code, and allows the decoder to locate this GOB. Furthermore, the GOB header contains information which allows the decoding process to be restarted (i.e., resynchronize the decoder to the bitstream and reset all predictively coded data).
The GOB approach to resynchronization is based on spatial resynchronization. That is, once a particular macroblock location is reached in the encoding process, a resynchronization marker is inserted into the bitstream. A potential problem with this approach is that since the encoding process is variable rate, these resynchronization markers will most likely be unevenly spaced throughout the bitstream. Therefore, certain portions of the scene, such as high motion areas, will be more susceptible to errors, which will also be more difficult to conceal.

The video packet approach adopted by MPEG-4 is based on providing periodic resynchronization markers throughout the bitstream. In other words, the length of the video packets are not based on the number of macroblocks, but instead on the number of bits contained in that packet. If the number of bits contained in the current video packet exceeds a predetermined threshold, then a new video packet is created at the start of the next macroblock.

A resynchronization marker is used to distinguish the start of a new video packet. This marker is distinguishable from all possible VLC codewords as well as the VOP start code. Header information is also provided at the start of a video packet. Contained in this header is the information necessary to restart the decoding process and includes: the macroblock number of the first macroblock contained in this packet and the quantization parameter necessary to decode that first macroblock. The macroblock number provides the necessary spatial resynchronization while the quantization parameter allows the differential decoding process to be resynchronized.

Also included in the video packet header is the header extension code. The HEC is a single bit that, when enabled, indicates the presence of additional resynchronization information; including modular time base, VOP temporal increment, VOP prediction type, and VOP F code. This additional information is made available in case the VOP header has been corrupted.

It should be noted that when utilizing the error resilience tools within MPEG-4, some of the compression efficiency tools are modified. For example, all predictively encoded information must be confined within a video packet so as to prevent the propagation of errors.

In conjunction with the video packet approach to resynchronization, a second method called fixed interval synchronization has also been adopted by MPEG-4. This method requires that VOP start codes and resynchronization markers (i.e., the start of a video packet) appear only at legal fixed interval locations in the bitstream. This helps avoiding the problems associated with start codes emulations. That is, when errors are present in a bitstream it is possible for these errors to emulate a VOP start code. In this case, when fixed interval synchronization is utilized the decoder is only required to search for a VOP start code at the beginning of each fixed interval. The fixed interval synchronization method extends this approach to be any predetermined interval.

9.14.2 Data Recovery

After synchronization has been reestablished, data recovery tools attempt to recover data that in general would be lost. These tools are not simply error correcting codes, but instead techniques that encode the data in an error resilient manner. For instance, one particular tool that has been endorsed by the Video Group is Reversible Variable Length Codes (RVLC). In this approach,

the variable length codewords are designed such that they can be read both in the forward as well as the reverse direction.

An example illustrating the use of a RVLC is given in Figure 17. Generally, in a situation such as this, where a burst of errors has corrupted a portion of the data, all data between the two synchronization points would be lost. However, as shown in the Figure, an RVLC enables some of that data to be recovered. It should be noted that the parameters, QP and HEC shown in the Figure, represent the fields reserved in the video packet header for the quantization parameter and the header extension code, respectively.

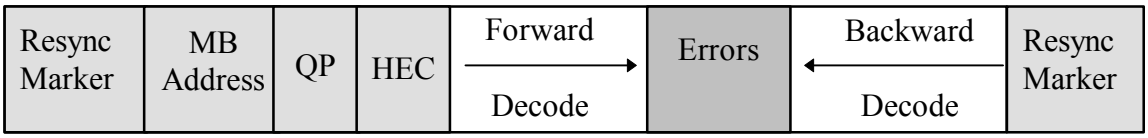


Figure 17 - example of Reversible Variable Length Code

9.14.3 Error Concealment

Error concealment is an extremely important component of any error robust video codec. Similar to the error resilience tools discussed above, the effectiveness of an error concealment strategy is highly dependent on the performance of the resynchronization scheme. Basically, if the resynchronization method can effectively localize the error then the error concealment problem becomes much more tractable. For low bitrate applications, low delay applications the current resynchronization scheme provides very acceptable results with a simple concealment strategy, such as copying blocks from the previous frame.

In recognizing the need to provide enhanced concealment capabilities, the Video Group has developed an additional error resilient mode that further improves the ability of the decoder to localize an error.
Specifically, this approach utilizes data partitioning by separating the motion and the texture. This approach requires that a second resynchronization marker be inserted between motion and texture information. If the texture information is lost, this approach utilizes the motion information to conceal these errors. That is, due to the errors the texture information is discarded, while the motion is used to motion compensate the previous decoded VOP.

10. Detailed technical description of MPEG-4 Audio

MPEG-4 coding of audio objects provides tools for both representing natural sounds (such as speech and music) and for synthesizing sounds based on structured descriptions. The representation for synthesized sound can be derived from text data or so-called instrument descriptions and by coding parameters to provide effects, such as reverberation and spatialization. The representations provide compression and other functionalities, such as scalability and effects processing.

The MPEG-4 Audio coding tools covering 6kbit/s to 24kbit/s have undergone verification testing for an AM digital audio broadcasting application in collaboration with the NADIB (Narrow Band Digital Broadcasting) consortium. With the intent of identifying a suitable digital audio broadcast format to provide improvements over the existing AM modulation services, several codec configurations involving the MPEG-4 CELP, TwinVQ, and AAC tools have been compared to a reference AM system. (see below for an explanation about these algorithms.) It was found that higher quality can be achieved in the same bandwidth with digital techniques and that scalable coder configurations offered performance superior to a simulcast alternative.

Additional verification tests were carried out by MPEG, in which the tools for speech and general audio coding were compared to existing standards.

10.1 Natural Sound

MPEG-4 standardizes natural audio coding at bitrates ranging from 2 kbit/s up to and above 64 kbit/s. When variable rate coding is allowed, coding at less than 2 kbit/s, such as an average bitrate of 1.2 kbit/s, is also supported. The presence of the MPEG-2 AAC standard within the MPEG-4 tool set provides for general compression of audio in the upper bitrate range. For these, the MPEG-4 standard defines the bitstream syntax and the decoding processes in terms of a set of tools. In order to achieve the highest audio quality within the full range of bitrates and at the same time provide the extra functionalities, speech coding techniques and general audio coding techniques are integrated in a common framework:

- Speech coding at bitrates between 2 and 24 kbit/s is supported by using Harmonic Vector eXcitation Coding (HVXC) for a recommended operating bitrate of 2 - 4 kbit/s, and Code Excited Linear Predictive (CELP) coding for an operating bitrate of 4 - 24 kbit/s. In addition, HVXC can operate down to an average of around 1.2 kbit/s in its variable bitrate mode. In CELP coding, two sampling rates, 8 and 16 kHz, are used to support narrowband and wideband speech, respectively. The following operating modes have been subject to verification testing: HVXC at 2 and 4 kbit/s, narrowband CELP at 6, 8.3, and 12 kbit/s, and wideband CELP at 18 kbit/s. In addition various of the scalable configurations have been verified.
- For general audio coding at bitrates at and above 6 kbit/s, transform coding techniques, namely TwinVQ and AAC, are applied. The audio signals in this region typically have sampling frequencies starting at 8 kHz.

To allow optimum coverage of the bitrates and to allow for bitrate and bandwidth scalability, a general framework has been defined. This is illustrated in Figure 18.

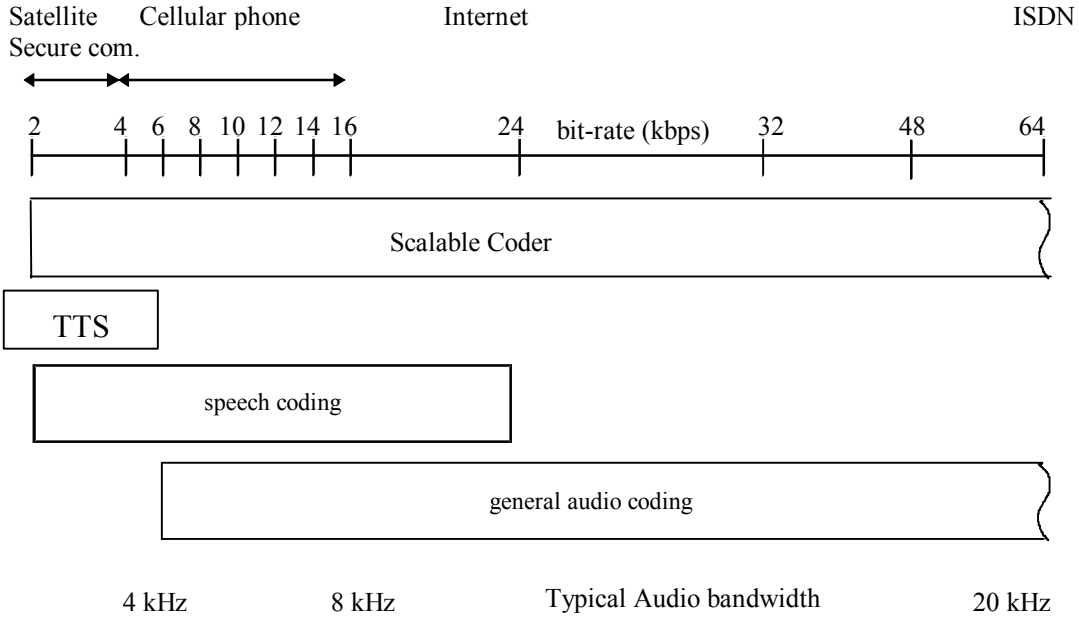


Figure 18 - General block diagram of MPEG-4 Audio

Starting with a coder operating at a low bitrate, by adding enhancements to a general audio coder, both the coding quality as well as the audio bandwidth can be improved.

Bitrate scalability, often also referred to as embedded coding, allows a bitstream to be parsed into a bitstream of lower bitrate that can still be decoded into a meaningful signal. The bitstream parsing can occur either during transmission or in the decoder. Bandwidth scalability is a particular case of bitrate scalability whereby part of a bitstream representing a part of the frequency spectrum can be discarded during transmission or decoding.
Encoder complexity scalability allows encoders of different complexity to generate valid and meaningful bitstreams. The decoder complexity scalability allows a given bitstream to be decoded by decoders of different levels of complexity. The audio quality, in general, is related to the complexity of the encoder and decoder used. Scalability works within some MPEG-4 tools, but can also be applied to a combination of techniques, e.g. with CELP as a base layer and AAC for the enhancement layer(s).

The MPEG-4 systems layer allows codecs according to existing (MPEG) standards, e.g. MPEG-2 AAC, to be used. Each of the MPEG-4 coders is designed to operate in a stand-alone mode with its own bitstream syntax. Additional functionalities are realized both within individual coders, and by means of additional tools around the coders. An example of such a functionality within an individual coder is speed or pitch change within HVXC.

10.2 Improvements in MPEG-4 Audio Version 2

10.2.1 Error Robustness

The error robustness tools provide improved performance on error-prone transmission channels. They can be distinguished into codec specific error resilience tools and a common error protection tool.
Improved error robustness for AAC is provided by a set of error resilience tools. These tools reduce the perceived deterioration of the decoded audio signal that is caused by corrupted bits in the bit stream. The following tools are provided to improve the error robustness for several parts of an AAC frame:

- Virtual CodeBook tool (VCB11)
- Reversible Variable Length Coding tool (RVLC)
- Huffman Codeword Reordering tool (HCR)

Improved error robustness capabilities for all coding tools are provided through the error resilient bit stream payload syntax. It allows advanced channel coding techniques, which can be adapted to the special needs of the different coding tools. This error resilient bit stream payload syntax is mandatory for all Version 2 object types.
The error protection tool (EP tool) provides error protection for all MPEG-4 Audio version 2 audio objects with flexible configuration applicable for wide range of channel error conditions. The main features of the EP tool are as follows:

- providing a set of error correcting/detecting codes with wide and small-step scalability, in performance and in redundancy
- providing a generic and bandwidth-efficient error protection framework which covers both fixed-length frame bit streams and variable-length frame bit streams
- providing an Unequal Error Protection (UEP) configuration control with low overhead

MPEG-4 Audio version 2 coding algorithms provide a classification of each bit stream field according to its error sensitivity. Based on this, the bit stream is divided into several classes, which can be separately protected by the EP tool, such that more error sensitive parts are protected more strongly.

10.2.2 Low-Delay Audio Coding

While the MPEG-4 General Audio Coder provides very efficient coding of general audio signals at low bit rates, it has an algorithmic encoding/decoding delay of up to several 100ms and is thus

not well suited for applications requiring low coding delay, such as real-time bi-directional communication. As an example, for the General Audio Coder operating at 24 kHz sampling rate and 24 kbit/s, this results in an algorithmic coding delay of about 110 ms plus up to additional 210 ms for the use of the bit reservoir. To enable coding of general audio signals with an algorithmic delay not exceeding 20 ms, MPEG-4 Version 2 specifies a Low-Delay Audio Coder which is derived from MPEG-2/4 Advanced Audio Coding (AAC). . Compared to speech coding schemes, this coder allows compression of general audio signal types, including music, at a low delay. It operates at up to 48 kHz sampling rate and uses a frame length of 512 or 480 samples, compared to 1024 or 960 samples used in standard MPEG-2/4 AAC. Also the size of the window used in the analysis and synthesis filter bank is reduced by a factor of 2. No block switching is used to avoid the ”look-ahead" delay due to the block switching decision. To reduce pre-echo artifacts in case of transient signals, window shape switching is provided instead. For non-transient parts of the signal a sine window is used, while a so-called low overlap window is used in case of transient signals. Use of the bit reservoir is minimized in the encoder in order to reach the desired target delay. As one extreme case, no bit reservoir is used at all. Verification tests have shown that the reduction in coding delay comes at a very moderate cost in compression performance.

10.2.3 Fine grain scalability

Bit rate scalability, also known as embedded coding, is a very desirable functionality. The General Audio Coder of Version 1 supports large step scalability where a base layer bit stream can be combined with one or more enhancement layer bit streams to utilize a higher bit rate and thus obtain a better audio quality. In a typical configuration, a 24 kbit/s base layer and two 16 kbit/s enhancement layers could be used, permitting decoding at a total bit rate of 24 kbit/s (mono), 40 kbit/s (stereo), and 56 kbit/s (stereo). Due to the side information carried in each layer, small bit rate enhancement layers are not efficiently supported in Version 1. To address this problem and to provide efficient small step scalability for the General Audio Coder, the Bit-Sliced Arithmetic Coding (BSAC) tool is available in Version 2. This tool is used in combination with the AAC coding tools and replaces the noiseless coding of the quantized spectral data and the scale factors. BSAC provides scalability in steps of 1 kbit/s per audio channel, i.e. 2 kbit/s steps for a stereo signal. One base layer bit stream and many small enhancement layer bit streams are used. The base layer contains the general side information, specific side information for the first layer and the audio data of the first layer. The enhancement streams contain only the specific side information and audio data for the corresponding layer. To obtain fine step scalability, a bit-slicing scheme is applied to the quantized spectral data. First the quantized spectral values are grouped into frequency bands. Each of these groups contains the quantized spectral values in their binary representation. Then the bits of a group are processed in slices according to their significance. Thus first all most significant bits (MSB) of the quantized values in a group are processed, etc. These bit-slices are then encoded using an arithmetic coding scheme to obtain entropy coding with minimal redundancy. Various arithmetic coding models are provided to cover the different statistics of the bit-slices. The scheme used to assign the bit-slices of the different frequency bands to the enhancement layer is constructed in a special way. This ensures that, with an increasing number of enhancement layers utilized by the decoder, providing more of the less significant bits refines quantized spectral data. But also providing bit-slices of the spectral data in higher frequency bands increases the bandwidth.

Verification tests have shown that the scalability aspect of this tool performs well over a wide range of rates. At the highest rate it is as good as AAC main profile operating at the same rate, while at the lowest rate the scalability function requires a moderate overhead relative to AAC main profile operating at the same rate.

10.2.4 Parametric Audio Coding

The Parametric Audio Coding tools combine very low bit rate coding of general audio signals with the possibility of modifying the playback speed or pitch during decoding without the need for an effects processing unit. In combination with the speech and audio coding tools of Version 1, improved overall coding efficiency is expected for applications of object based coding allowing selection and/or switching between different coding techniques.
Parametric Audio Coding uses the Harmonic and Individual Lines plus Noise (HILN) technique to code general audio signals at bit rates of 4 kbit/s and above using a parametric representation of the audio signal. The basic idea of this technique is to decompose the input signal into audio objects, which are described by appropriate source models and represented by model parameters. Object models for sinusoids, harmonic tones, and noise are utilized in the HILN coder.
This approach allows to introduce a more advanced source model than just assuming a stationary signal for the duration of a frame, which motivates the spectral decomposition used e.g. in the MPEG-4 General Audio Coder. As known from speech coding, where specialized source models based on the speech generation process in the human vocal tract are applied, advanced source models can be advantageous in particular for very low bit rate coding schemes.
Due to the very low target bit rates, only the parameters for a small number of objects can be transmitted. Therefore a perception model is employed to select those objects that are most important for the perceptual quality of the signal.
In HILN, the frequency and amplitude parameters are quantized according to the "just noticeable differences" known from psycho-acoustics. The spectral envelope of the noise and the harmonic tone is described using LPC modeling as known from speech coding. Correlation between the parameters of one frame and between consecutive frames is exploited by parameter prediction. The quantized parameters are finally entropy coded and multiplexed to form a bit stream.
A very interesting property of this parametric coding scheme arises from the fact that the signal is described in terms of frequency and amplitude parameters. This signal representation permits speed and pitch change functionality by simple parameter modification in the decoder. The HILN parametric audio coder can be combined with MPEG-4 parametric speech coder (HVXC) to form an integrated parametric coder covering a wider range of signals and bit rates. This integrated coder supports speed and pitch change. Using a speech/music classification tool in the encoder, it is possible to automatically select the HVXC for speech signals and the HILN for music signals. Such automatic HVXC/HILN switching was successfully demonstrated and the classification tool is described in the informative Annex of the Version 2 standard.
Verification test have shown that HILN coding has performance comparable to other MPEG-4 coding technology operating at similar bit rates while providing the additional capability of independent audio signal speed or pitch change when decoding. The test has also shown that the scalable HILN coder provides quality comparable to that of a fixed-rate HILN coder at the same bit rate.

10.2.5 CELP Silence Compression

The silence compression tool reduces the average bit rate thanks to a lower bit rate compression for silence. In the encoder, a voice activity detector is used to distinguish between regions with normal speech activity and those with silence or background noise. During normal speech activity, the CELP coding as in Version 1 is used. Otherwise a Silence Insertion Descriptor (SID) is transmitted at a lower bit rate. This SID enables a Comfort Noise Generator (CNG) in the decoder. The amplitude and spectral shape of this comfort noise is specified by energy and LPC parameters similar as in a normal CELP frame. These parameters are an optional part of the SID and thus can be updated as required.

10.2.6 Error Resilient HVXC

The Error Resilient (ER) HVXC object is supported by the Parametric speech coding (ER HVXC) tools, which provides fixed bit-rate modes(2.0-4.0kbps) and variable bit-rate mode(<2.0kbps, <4.0kbps) both in a scalable and non-scalable scheme. In the Version 1 HVXC, variable bit rate mode of 2.0 kbit/s maximum is already supported and the variable bit rate mode of 4.0 kbit/s maximum is additionally supported in Version 2 ER HVXC. In the variable bit rate modes, non-speech parts are detected in unvoiced signals, and a smaller number of bits is used for these non-speech parts to reduce the average bit rate. ER HVXC provides communications-quality to near-toll-quality speech in the 100-3800 Hz band at 8kHz sampling rate. When the variable bit-rate mode is allowed, operation at lower average bit-rate is possible. Coded speech with variable bit-rate mode at typical bit-rate of 1.5kbps average, and at typical bit-rate of 3.0kbps average has essentially the same quality as 2.0 kbps fixed rate and 4.0 kbps fixed rate respectively. The functionality of pitch and speed change during decoding is supported for all modes. ER HVXC has the syntax with the error sensitivity classes to be used with the EP-Tool, and the error concealment functionality are supported for the use for error-prone channel like mobile communication channels. The ER HVXC speech coder targets applications from mobile and satellite communications, to Internet telephony, to packaged media and speech databases.

10.2.7 Environmental Spatialization

The Environmental Spatialization tools enable composition of an audio scene with more natural sound source and sound environment modeling than is possible in Version 1. Both, a physical and a perceptual approach to spatialization are supported. The *physical approach* is based on a description of the acoustical properties of the environment (e.g. room geometry, material properties, position of sound source) and can be used in applications like 3-D virtual reality. The *perceptual approach* on the other hand permits a high level perceptual description of the audio scene based on parameters similar to those of a reverberation effects unit. Thus, the audio and the visual scene can be composed independently as usually required by applications like movies. Although the Environmental Spatialization tools are related to audio, they are part of the BInary Format for Scene description (BIFS) in MPEG-4 Systems and are referred to as Advanced AudioBIFS.

10.2.8 Back channel

The back channel allows a request of client and/or client terminal to server. With this capability, interactivity can be achieved. In MPEG-4 System, the need for an up-stream channel (back channel) is signaled to the client terminal by supplying an appropriate elementary stream descriptor declaring the parameters for that stream. The client terminal opens this upstream channel in a similar manner as it opens the down-stream channels. The entities (e.g. media encoders & decoders) that are connected through an upstream channel are known from the parameters in its elementary stream descriptor and from the association of the elementary stream descriptor to a specific object descriptor. In MPEG-4 Audio, the back channel allows feedback for bit rate adjustment, the scalability and error protection adaptation.

10.2.9 Audio transport stream

The MPEG-4 Audio transport stream defines a mechanism to transport MPEG-4 Audio streams without using MPEG-4 Systems and is dedicated for audio-only applications. The transport mechanism uses a two-layer approach, namely a multiplex layer and a synchronization layer. The multiplex layer (Low-overhead MPEG-4 Audio Transport Multiplex: LATM) manages multiplexing of several MPEG-4 Audio payloads and audio specific configuration information. The synchronization layer specifies a self-synchronized syntax of the MPEG-4 Audio transport stream which is called Low Overhead Audio Stream (LOAS). The Interface format to a transmission layer depends on the conditions of the underlying transmission layer.

10.3 Synthesized Sound

MPEG-4 defines decoders for generating sound based on several kinds of 'structured' inputs. Text input is converted to speech in the Text-To-Speech (TTS) decoder, while more general sounds including music may be normatively synthesized. Synthetic music may be delivered at extremely low bitrates while still describing an exact sound signal.

Text To Speech. TTS coders bitrates range from 200 bit/s to 1.2 Kbit/s, which allows a text or a text with prosodic parameters (pitch contour, phoneme duration, and so on) as its inputs to generate intelligible synthetic speech. It supports the generation of parameters that can be used to allow synchronization to associated face animation, international languages for text and international symbols for phonemes. Additional markups are used to convey control information within texts, which is forwarded to other components in synchronization with the synthesized text. Note that MPEG-4 provides a standardized interface for the operation of a Text To Speech coder (TTSI = Text To Speech Interface), but *not* a normative TTS synthesizer itself.

Score Driven Synthesis.
The Structured Audio tools decode input data and produce output sounds. This decoding is driven by a special synthesis language called SAOL (Structured Audio Orchestra Language) standardized as a part of MPEG-4. This language is used to define an "orchestra" made up of "instruments" (downloaded in the bitstream, not fixed in the terminal) which create and process control data. An instrument is a small network of signal processing primitives that might emulate some specific sounds such as those of a natural acoustic instrument. The signal-processing network may be implemented in hardware or software and include both generation and processing of sounds and manipulation of pre-stored sounds.

MPEG-4 does not standardize "a single method" of synthesis, but rather a way to describe methods of synthesis. Any current or future sound-synthesis method can be described in SAOL, including wavetable, FM, additive, physical-modeling, and granular synthesis, as well as non-parametric hybrids of these methods.

Control of the synthesis is accomplished by downloading "scores" or "scripts" in the bitstream. A score is a time-sequenced set of commands that invokes various instruments at specific times to contribute their output to an overall music performance or generation of sound effects. The score description, downloaded in a language called SASL (Structured Audio Score Language), can be used to create new sounds, and also include additional control information for modifying existing sound. This allows the composer finer control over the final synthesized sound. For synthesis processes that do not require such fine control, the established MIDI protocol may also be used to control the orchestra.

Careful control in conjunction with customized instrument definition, allows the generation of sounds ranging from simple audio effects, such as footsteps or door closures, to the simulation of natural sounds such as rainfall or music played on conventional instruments to fully synthetic sounds for complex audio effects or futuristic music.

For terminals with less functionality, and for applications which do not require such sophisticated synthesis, a "wavetable bank format" is also standardized. Using this format, sound samples for use in wavetable synthesis may be downloaded, as well as simple processing, such as filters, reverbs, and chorus effects. In this case, the computational complexity of the required decoding process may be exactly determined from inspection of the bitstream, which is not possible when using SAOL.

11. Annexes

Annex A - The MPEG-4 development process

The Moving Picture Coding Experts Group (MPEG) is a working group of ISO/IEC in charge of the development of international standards for compression, decompression, processing, and coded representation of moving pictures, audio and their combination.
The purpose of MPEG is to produce standards. The first two standards produced by MPEG were:

- **MPEG-1**, a standard for storage and retrieval of moving pictures and audio on storage media (officially designated as ISO/IEC 11172, in 5 parts)
- **MPEG-2**, a standard for digital television (officially designated as ISO/IEC 13818, in 9 parts).

MPEG is has recently finalized MPEG-4 Version 1, a standard for multimedia applications, that will officially reach the status of International Standard in February 1999, with the ISO number 14496.
MPEG has also started work on a new standard known as MPEG-7: a content representation standard for information search, scheduled for completion in Fall 2001. The Call for Proposals was issued in October 1998.
MPEG-1 has been a very successful standard. It is the de-facto form of storing moving pictures and audio on the World Wide Web and is used in millions of Video CDs. Digital Audio Broadcasting (DAB) is a new consumer market that makes use of MPEG-1 audio coding.
MPEG-2 has been the timely response for the satellite broadcasting and cable television industries in their transition from analogue to digital. Millions of set-top boxes incorporating MPEG-2 decoders have been sold in the last 3 years.
Since July 1993 MPEG is working on its third standard, called MPEG-4. MPEG considers of vital importance to define and maintain, without slippage, a work plan. This is the MPEG-4 work plan:

Part	Title	WD	CD	FCD	FDIS	IS
1	Systems		97/11	98/03	98/10	99/04
2	Visual		97/11	98/03	98/10	99/04
3	Audio		97/11	98/03	98/10	99/04
4	Conformance Testing	97/10	98/12	99/07	99/12	00/02
5	Reference Software		97/11	98/03	99/03	99/05
6	Delivery Multimedia Integration Framework (DMIF)	97/07	97/11	98/03	98/10	99/04
7	Optimized Software for MPEG-7 tools (video encoders)				01/03	01/05
1/Amd 1	Systems Extensions (v.2)	97/10	99/03	99/07	99/12	00/02
2/Amd 1	Visual Extensions (v.2)	97/10	99/03	99/07	99/12	00/02
3/Amd 1	Audio Extensions (v.2)	97/10	99/03	99/07	99/12	00/02
4/Amd1	Conformance Testing Extensions (v.2)	98/12	99/12	00/07	01/01	01/03
5/Amd 1	Reference Software Extensions	97/10	99/07	99/12	01/01	01/03
6/Amd 1	DMIF Extensions (v.2)	97/10	99/03	99/07	99/12	00/08
1/Amd 2	Extended BIFS			00/03	01/01	01/03

1/Amd 3	Textual Format			01/03	01/07	01/09
2/Amd 3	Studio Profile		00/03	00/07	01/01	01/03
2/Amd 4	Streaming Video Profiles		00/03	00/07	01/01	01/03
8	4 on IP framework		01/03	01/07	01/12	02/02

Table 1 - MPEG-4 work plan (The abbreviations are explained below)

Because of the complexity of the work item, it took 2 years before a satisfactory definition of the scope could be achieved and half a year more before a first call for proposals could be issued. This call, like all MPEG calls, was open to all interested parties, no matter whether they were within or outside of MPEG. It requested technology that proponents felt could be considered by MPEG for the purpose of the developing the MPEG-4 standard. After that first call, other calls were issued for other technology areas.
The proposals of technology received were assessed and, if found promising, incorporated in the so-called Verification Models (VMs). A VM describes, in text and some sort of programming language, the operation of encoder and decoder. VMs are used to carry out simulations with the aim to optimize the performance of the coding schemes.
Because (next to the envisaged hardware environments for MPEG-4) software platforms are gaining in importance for multimedia standards, MPEG decided to maintain software implementations of the different standard parts. These can be used for the purpose of development of the standard and for commercial implementations of the standard. At the Maceió meeting in November ’96 MPEG reached sufficient confidence in the stability of the standard under development, and produced the Working Drafts (WDs). Much work has been done since, resulting in the production of Committee Drafts (CD), Final Committee Drafts (FCD) and finally Final Drafts of International Standard (FDIS), in Atlantic City, October 1998.
The WDs already had the structure and form of a standard but they were kept internal to MPEG for revision. Starting from the Seville meeting in February ’97, MPEG decided to publish the WDs to seek first comments from industry. The CD underwent a formal ballot by national Bodies and the same happened to the FCD. This applies to all parts of the standard, although ‘Conformance’ is scheduled for later completion than the other parts.

Ballots by NBs are usually accompanied by technical comments. These ballots were considered at the March ’98 meeting in Tokyo for CD and in Atlantic City (October 1998) for the FCD. This process entailed making changes. Following the Atlantic City meeting, the standard was sent out for a final ballot, where NBs could only cast a yes/no ballot, without comments, within two months. After that, the FDIS became International Standard (IS) and was sent to the ISO Central Secretariat for publication. A similar process was followed for the amendments that form MPEG-4 Version 2. In December '99, at the Maui meeting, MPEG-4 Version 2 has acquired the status of FDIS. After that, several extensions were added to the standard.

Annex B - Organization of work in MPEG

Established in 1988, MPEG has grown to form an unusually large committee. Some 300 to 400 experts take part in MPEG meetings, and the number of people working on MPEG-related matters without attending meetings is even larger.
The wide scope of technologies considered by MPEG and the large body of available expertise, require an appropriate organization. Currently MPEG has the following subgroups:

1. Requirements	Develops requirements for the standards under development (currently, MPEG-4 and MPEG-7).

2. Systems	Develops standards for the coding of the combination of individually coded audio, moving images and related information so that the combination can be used by any application.
3. Video	Develops standards for coded representation of moving pictures of natural origin.
4. Audio	Develops standards for coded representation of audio of natural origin.
5. SNHC (Synthetic- Natural Hybrid Coding)	Develops standards for the integrated coded representation of audio and moving pictures of natural and synthetic origin. SNHC concentrates on the coding of synthetic data.
6. Multimedia Description	Develops Structures for multimedia descriptions. This group only works for MPEG-7,
7. Test	Develops methods for and the execution of subjective evaluation tests of the quality of coded audio and moving pictures, both individually and combined, to test the quality of moving pictures and audio produced by MPEG standards
8. Implementation	Evaluates coding techniques so as to provide guidelines to other groups upon realistic boundaries of implementation parameters.
9. Liaison	Handles relations with bodies external to MPEG.
10. HoD Heads of Delegation	The group, consisting of the heads of all national delegations, acts in advisory capacity on matters of general nature.

Work for MPEG takes place in two different instances. A large part of the technical work is done at MPEG meetings, usually lasting one full week. Members electronically submit contributions to the MPEG FTP site (several hundreds of them at every meeting). Delegates are then able to come to meetings well prepared without having to spend precious meeting time to study other delegates' contributions.
The meeting is structured in 3 plenary meetings (on Monday morning, on Wednesday morning and on Friday afternoon) and in parallel subgroup meetings.
About 100 output documents are produced at every meeting; these capture the agreements reached. Documents of particular importance are:

- Drafts of the different parts of the standard under development;
- New versions of the different Verification Models, that are used to develop the respective parts of the standard.
- "Resolutions", which document the outline of each agreement and make reference to the documents produced;
- "Ad-hoc groups", groups of delegates agreeing to work on specified issues, usually until the following meeting;

Output documents are also stored on the MPEG FTP site. Access to input and output documents is restricted to MPEG members. At each meeting, however, some output documents are released for public use. These can be accessed from the home page: www.cselt.it/mpeg
Equally important is the work that is done by the ad-hoc groups in between two MPEG meetings. They work by e-mail under the guidance of a Chairman appointed at the Friday (closing) plenary meeting. In some exceptional cases, they may hold physical meetings. Ad-hoc groups produce recommendations that are reported at the first plenary of the MPEG week and function as valuable inputs for further deliberation during the meeting.

Annex C - Glossary and Acronyms

AAC	Advanced Audio Coding
AAL	ATM Adaptation Layer

Access Unit	A logical sub-structure of an Elementary Stream to facilitate random access or bitstream manipulation
ACE	Advanced Coding Efficiency (Profile)
Alpha plane	Image component providing transparency information ??? (Video)
Amd	Amendment
AOI	Area Of Interest
API	Application Programming Interface
ARTS	Advanced Real-time Simple (Profile)
ATM	Asynchronous Transfer Mode
BAP	Body Animation Parameters
BAP	Body Animation Parameter
BDP	Body Definition Parameters
BDP	Body Definition Parameter
BIFS	Binary Format for Scenes
BSAC	Bit-Sliced Arithmetic Coding
CD	Committee Draft
CE	Core Experiment
CELP	Code Excited Linear Prediction
CIF	Common Intermediate Format
CNG	Comfort Noise Generator
DAI	DMIF-Application Interface
DCT	Discrete Cosine Transform
DMIF	Delivery Multimedia Integration Framework
DNI	DMIF Network Interface
DRC	Dynamic Resolution Conversion
DS	DMIF signaling
EP	Error Protection
ER	Error Resilient
ES	Elementary Stream: A sequence of data that originates from a single producer in the transmitting MPEG-4 Terminal and terminates at a single recipient, e.g. an media object or a Control Entity in the receiving MPEG-4 Terminal. It flows through one FlexMux Channel.
FAP	Facial Animation Parameters
FBA	Facial and Body Animation
FCD	Final Committee Draft
FDIS	Final Draft International Standard
FDP	Facial Definition Parameters
FlexMux stream	A sequence of FlexMux packets associated to one or more FlexMux Channels flowing through one TransMux Channel
FlexMux tool	A Flexible (Content) Multiplex tool
FTTC	Fiber To The Curb
GMC	Global Motion Compensation
GSTN	General Switched Telephone Network
HCR	Huffman Codeword Reordering
HFC	Hybrid Fiber Coax
HILN	Harmonic Individual Line and Noise
HTTP	HyperText Transfer Protocol
HVXC	Harmonic Vector Excitation Coding
IP	Internet Protocol
IPI	Intellectual Property Identification

IPMP	Intellectual Property Management and Protection
IPR	Intellectual Property Rights
IS	International Standard
ISDN	Integrated Service Digital Network
LAR	Logarithmic Area Ratio
LATM	Low-overhead MPEG-4 Audio Transport Multiplex:
LC	Low Complexity
LOAS	Low Overhead Audio Stream
LOD	Level Of Detail
LPC	Linear Predictive Coding
LSP	Line Spectral Pairs
LTP	Long Term Prediction
M4IF	MPEG-4 Industry Forum
MCU	Multipoint Control Unit
Mdat	media data atoms
Mesh	A graphical construct consisting of connected surface elements to describe the geometry/shape of a visual object.
MIDI	Musical Instrument Digital Interface
MPEG	Moving Pictures Experts Group
MPEG-J	Framework for MPEG Java API's
MSB	Most Significant Bits
NB	National Body
OCI	Object Content Information
OD	Object descriptor
OD	Object Descriptor
PDA	Personal Digital Assistant
PDU	Protocol Data Unit
PSNR	Peak Signal to Noise Ratio
QCIF	Quarter Common Intermediate Format
QoS	Quality of Service
Rendering	The process of generating pixels for display
RTP	Real Time Transport Protocol
RTSP	Real Time Streaming Protocol
RVLC	Reversible Variable Length Coding
SA-DCT	shape-adaptive DCT
SID	Silence Insertion Descriptor
SL	Sync(hronization) layer
SMIL	Synchronized Multimedia Integration Language
SNHC	Synthetic- Natural Hybrid Coding
SNR	Signal to Noise Ratio
Sprite	A static sprite is a - possibly large - still image, describing panoramic background.
SRM	Session Resource Manager
SVG	Scalable Vector Graphics
T/F coder	Time/Frequency Coder
TCP	Transmission Control Protocol
TransMux	generic abstraction for any transport multiplex scheme
TTS	Text-to-speech
UDP	User Datagram Protocol
UEP	Unequal Error Protection
UMTS	Universal Mobile Telecommunication System
VCB	Virtual CodeBook

Viseme	Facial expression associated to a specific phoneme
VLBV	Very Low Bitrate Video
VM	Verification Model
VOP	Video Object Plane
VRML	Virtual Reality Modeling Language
W3C	World Wide Web Consortium
WD	Working Draft
WWW	World Wide Web
XMT	Extensible MPEG-4 textual format

Light Field Mapping:
Efficient Representation and Hardware Rendering of Surface Light Fields

Wei-Chao Chen [*]
University of North Carolina at Chapel Hill

Jean-Yves Bouguet [†]
Intel Corporation

Michael H. Chu [‡]
Intel Corporation

Radek Grzeszczuk [§]
Intel Corporation

Abstract: A light field parameterized on the surface offers a natural and intuitive description of the view-dependent appearance of scenes with complex reflectance properties. To enable the use of surface light fields in real-time rendering we develop a compact representation suitable for an accelerated graphics pipeline. We propose to approximate the light field data by partitioning it over elementary surface primitives and factorizing each part into a small set of lower-dimensional functions. We show that our representation can be further compressed using standard image compression techniques leading to extremely compact data sets that are up to four orders of magnitude smaller than the input data. Finally, we develop an image-based rendering method, light field mapping, that can visualize surface light fields directly from this compact representation at interactive frame rates on a personal computer. We also implement a new method of approximating the light field data that produces positive only factors allowing for faster rendering using simpler graphics hardware than earlier methods. We demonstrate the results for a variety of non-trivial synthetic scenes and physical objects scanned through 3D photography.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—color, shading, shadowing, texture I.4.8 [Image Processing And Computer Vision]: Scene Analysis—color, shading I.2.10 [Artificial Intelligence]: Scene Analysis—3D/stereo scene analysis, texture

Keywords: Image-based Rendering, Texture Mapping, Compression Algorithms, Rendering Hardware.

1 Introduction

The recent proliferation of inexpensive but powerful graphics hardware and new advances in digital imaging technology are enabling novel methods for realistic modeling of the appearance of physical objects. On the one hand, we see a tendency to represent complex analytic reflectance models with their sample-based approximations that can be evaluated efficiently using new graphics hardware features [15; 16; 17]. On the other hand, we are witnessing a proliferation of image-based rendering and modeling techniques [26; 22; 14; 4] that attempt to represent the discrete radiance data directly in the sample-based format without resorting to the analytic models at all. These techniques are popular because they promise a simple acquisition and an accurate portrayal of the physical world. The approach presented here combines these two trends. Similar to other image-based methods, our approach produces a sample-based representation of the surface light field data. Additionally, the proposed representation can be evaluated efficiently with the support of existing graphics hardware.

Figure 1: A combination of synthetic and scanned objects rendered using light field mapping. Complex, physically realistic reflectance properties of this scene are preserved.

1.1 Overview of Light Field Mapping Approach

A surface light field is a 4-dimensional function $f(r,s,\theta,\phi)$ that completely defines the outgoing radiance of every point on the surface of an object in every viewing direction. The first pair of parameters of this function (r,s) describes the surface location and the second pair of parameters (θ,ϕ) describes the viewing direction. A direct representation and manipulation of surface light field data is impractical because of the large size. Instead, we propose to approximate the light field function as a sum of a small number of products of lower-dimensional functions

$$f(r,s,\theta,\phi) \approx \sum_{k=1}^{K} g_k(r,s)\, h_k(\theta,\phi). \qquad (1)$$

We show that it is possible to construct the approximations of this form that are both compact and accurate by taking advantage of the spatial coherence of the surface light fields. We accomplish this by partitioning the light field data across small surface primitives and building the approximations for each part independently. The partitioning is done in such a way as to ensure continuous approximations across the neighboring surface elements. Because the

[*]e-mail: ciao@cs.unc.edu
[†]e-mail: jean-yves.bouguet@intel.com
[‡]e-mail: michael.h.chu@intel.com
[§]e-mail: radek.grzeszczuk@intel.com

functions $g_k(r,s)$ and $h_k(\theta,\phi)$ encode the light field data and are stored in a sampled form as texture maps, we call them the *light field maps*. Similarly, we refer to the process of rendering from this approximation as *light field mapping.* In the remainder of the paper we will refer to functions $g_k(r,s)$ as the *surface maps* and functions $h_k(\theta,\phi)$ as the *view maps*.

As shown in earlier work, similar types of factorizations can be used to produce high quality renderings of objects with complex surface reflectance properties at interactive frame rates [15; 17; 16]. Most previous work, however, assumes that the surface of the object has uniform reflectance properties, which greatly simplifies the problem. Our method shows for the first time how to compute such factorization for surfaces where each point has a unique reflectance property. This generalization makes light field mapping ideally suited for modeling and rendering of physical objects and scenes scanned through 3D photography.

1.2 Contributions

We make the following contributions to analysis and representation of image-based data and to hardware-accelerated rendering of image-based models.

Data Partitioning: We introduce a novel type of partitioning of the light field data that ensures a continuous approximation across individual triangles. It represents the light field data for each triangle as a linear blend of 3 light fields corresponding to the vertices of the triangle. See Section 2.1.

Approximation Through Factorization: We introduce a new type of factorization of light field data. Dense and uniform sampling of the view-dependent information, enabled through the use of view maps $h_k(\theta,\phi)$, ensures correct image synthesis from any camera location. See Section 2.2.

Positive Factorization: We introduce a new method of approximating the light field data that uses non-negative matrix factorization [20] to produce positive only factors. This method allows for faster rendering using generic graphics hardware at high approximation quality. It is significantly easier to implement than homomorphic factorization [25]. See Section 2.2.

Hardware-Accelerated Rendering: We develop a fully hardware-accelerated rendering routine that can visualize the light field data from the proposed, compact representation at highly interactive frame rates. The rendering algorithm is extremely simple to implement, since it reduces to a multi-pass texture mapping operation. Rendering quality can be improved progressively by increasing the number of rendering passes. See Section 3.

Compression: We report the highest compression of surface light fields to date. Since the light field maps are in essence collections of images that themselves exhibit redundancy, they can be further compressed using standard image compression techniques. This compression combined with the redundancy reduction achieved through the approximation results in extremely high compression ratios. Our method routinely approaches 4 orders of magnitude compression ratio and allows for hardware-accelerated rendering directly from the compressed representation. Additional compression is possible for transmission and storage purposes. See Section 4.

1.3 Related Work

Traditionally, computer graphics researchers have favored analytic reflectance models for their intuitive simplicity, compactness and flexibility [37; 7; 38; 29]. Although simple to represent, these models are often costly to compute and difficult to develop. Several approaches have been developed to approximate reflectance functions with a set of predefined basis function [34; 18; 19] but without much consideration for rendering efficiency. Inverse rendering methods [33; 40; 9; 2] approximate scene attributes, such as lighting and surface reflectance, by fitting analytic models to sample-based radiance data. Ramamoorthi and Hanrahan [31] propose a signal processing framework for inverse rendering. Light field mapping is a related method that uses a factor analysis approach rather than a signal processing approach.

Image-based methods [26; 22; 14; 4; 24] represent the radiance data directly in the sample-based format. Compression of light field data is an important area of research. Levoy and Hanrahan [22] use VQ [13], Magnor and Girod [23] have developed a series of disparity-compensated hybrid light field codecs. Chai *et al.* [5] use spectral analysis to compute the "minimum sampling rate" for light field rendering and show that scene depth information significantly decreases this rate. Miller *et al.* [27] propose a method of rendering surface light fields from input images compressed using JPEG-like compression. Malzbender *et al.* [24] approximate a sequence of images captured under varying lighting conditions using biquadratic polynomials. The representation is very compact and can be implemented in hardware.

The work on sample-based approximations for some of the analytic models listed above includes [15; 17; 21]. Kautz and McCool [16] propose a method for hardware assisted rendering of arbitrary BRDFs through their decomposition into a sum of 2D separable functions that is based on an idea proposed by Fournier [12]. Our application is fundamentally different from this work, since it allows each surface point to have different reflectance properties and is therefore ideally suited for modeling and rendering of objects and scenes scanned through 3D photography. Homomorphic factorization of McCool *et al.* [25] generates a BRDF factorization with positive factors only, which are easier and faster to render on the current graphics hardware, and deals with scattered data without a separate resampling and interpolation algorithm. We present a novel method for factorization of light field data that also produces positive only factors using non-negative matrix factorization [20]. Lensch *et al.* [21] reconstruct a spatially varying BRDF from a sparse set of images. First, the object is split into cluster with different BRDF properties, then a set of basis BRDFs is generated for each cluster, finally the original samples are reprojected into the space spanned by the basis BRDFs. Data factorization is this case is extremely time consuming and rendering is not real-time.

The work on surface light fields includes view-dependent texture mapping [10; 11; 30]. Additionally, Wood *et al.* [39] use a generalization of VQ and PCA (principal component analysis) [1] to compress surface light fields and propose a new rendering algorithm that displays compressed light fields at interactive frame rates. Their 2-pass rendering algorithm interpolates the surface normals using Gouraud shading producing incorrect results for large triangles. To alleviate the problem they introduce a view-dependent geometry refinement at the cost of increasing the complexity of the renderer.

Nishino *et al.* [28] propose the eigen-texture method that also performs factorization on image data. Their basic theoretical approach is similar to ours however it applies only to a 3D subset of the 4D light field and cannot be directly extended to the full 4D data. Additionally, our approach uses a vertex-based partitioning which ensures a continuous reconstruction, we explore a positive factorization technique, and we develop a fully hardware-accelerated rendering of surface light fields directly from our representation.

2 Approximation through Factorization

We start the description of the approximation method with a novel partitioning scheme that ensures continuous approximations throughout the surface of the model.

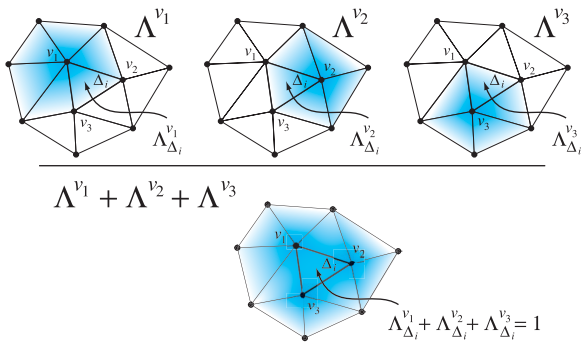


Figure 2: The finite support of the hat functions Λ^{v_j} around vertex v_j, $j = 1,2,3$. $\Lambda^{v_j}_{\triangle_i}$ denotes the portion of Λ^{v_j} that corresponds to triangle $\triangle_i$. Functions Λ^{v_1}, Λ^{v_2} and Λ^{v_3} add up to one inside Δ_i.

2.1 Surface Light Field Partitioning

We assume that the geometry of the models is represented as a triangular mesh. An obvious partitioning of the light field function is to split it between individual triangles. Unfortunately, an approximation of surface light field partitioned this way results in visible discontinuities at the edges of the triangles. We demonstrate this effect in the video accompanying the paper. To eliminate the discontinuities across triangle boundaries we propose to partition surface light field data around every vertex. We refer to the surface light field unit corresponding to each vertex as the *vertex light field* and for vertex v_j denote it as $f^{v_j}(r,s,\theta,\phi)$. Partitioning is computed by weighting the surface light field function

$$f^{v_j}(r,s,\theta,\phi) = \Lambda^{v_j}(r,s)\, f(r,s,\theta,\phi) \tag{2}$$

where Λ^{v_j} is the barycentric weight of each point in the ring of triangles relative to vertex v_j. Because of their shape, the weighting functions are often referred to as the *hat functions*. The top row of Figure 2 shows hat functions Λ^{v_1}, Λ^{v_2}, Λ^{v_3} for three vertices v_1, v_2, v_3 of triangle $\triangle_i$. As shown at the bottom of Figure 2, these 3 hat functions add up to unity inside triangle $\triangle_i$. Therefore, Equation (2) defines a valid partitioning of the surface light field because the sum of all vertex light fields equals to the original light field data.

In the final step of vertex-centered partitioning we reparameterize each vertex light field to the local coordinate system of the vertex. To this end, we define the viewing direction angles θ and ϕ as the azimuth and elevation angles of the viewing direction vector in the local reference frame of the vertex. We define the vertex reference frame in such a way that its z-axis is parallel to the surface normal at the vertex. We use the same letters to denote the local parameters of the vertex light field in order to simplify the notation.

2.2 Vertex Light Field Approximation

Vertex-centered partitioning of light field data allows us to approximate each vertex light field independently while maintaining continuity across the whole model. Analogous to Equation (1), we approximate each vertex light field as

$$f^{v_j}(r,s,\theta,\phi) \approx \sum_{k=1}^{K} g_k^{v_j}(r,s)\, h_k^{v_j}(\theta,\phi). \tag{3}$$

We present two methods of approximating vertex light fields: PCA (principal component analysis) [1] and NMF (non-negative matrix factorization) [20]. Both of these methods approximate the light field data as a linear combination of a small number of basis images but each one produces an approximation with a different set of features. For example, each PCA-based basis image gives a global approximation of the data. This means that we can use a subset of them to produce a consistent approximation, since adding successive basis images simply improves the accuracy preceding basis images. NMF, on the other hand, produces basis images that form a parts-based representation and therefore all of them need to be used to produce a consistent approximation of the input data. PCA-based approximation is therefore more *progressive* than NMF-based approximation. However, PCA allows the approximation factors to be of arbitrary sign. This makes rendering more difficult and requires more specialized graphics hardware. See Section 3.1 for details. NMF does not allow negative values in the factors resulting in an approximation that is much easier and faster to render on generic graphics hardware.

We will use the matrix factorization framework to describe how PCA and NMF construct the light field data approximations. We first discretize the vertex light field function $f^{v_j}(r,s,\theta,\phi)$ into a 4-dimensional grid $f^{v_j}[r_p,s_p,\theta_q,\phi_q]$, where index $p = 1,\ldots,M$ refers to the discrete values $[r_p,s_p]$ describing the surface location within the triangle ring of vertex v_j, and index $q = 1,\ldots,N$ refers to the discrete values $[\theta_q,\phi_q]$ of the two viewing angles. We then rearrange the discretized vertex light field into the matrix

$$\mathbf{F}^{v_j} = \begin{bmatrix} f^{v_j}[r_1,s_1,\theta_1,\phi_1] & \cdots & f^{v_j}[r_1,s_1,\theta_N,\phi_N] \\ \vdots & \ddots & \vdots \\ f^{v_j}[r_M,s_M,\theta_1,\phi_1] & \cdots & f^{v_j}[r_M,s_M,\theta_N,\phi_N] \end{bmatrix}, \tag{4}$$

where M is the total number of surface samples inside the triangle ring and N is the total number of views for each surface sample. We refer to matrix $\mathbf{F}^{v_j}$ as the *vertex light field matrix*. Each column of this matrix represents the appearance of the vertex ring under a different viewing direction. A detailed discussion of resampling the irregular input light field data can be found in Section 5.

Both PCA and NMF construct approximate factorizations of the form

$$\widetilde{\mathbf{F}}^{v_j} = \sum_{k=1}^{K} \mathbf{u}_k \mathbf{v}_k^T \tag{5}$$

where $\mathbf{u}_k$ is a vectorized representation of discrete surface map $g_k^{v_j}[r_p,s_p]$ and $\mathbf{v}_k$ is a vectorized representation of discrete view map $h_k^{v_j}[\theta_q,\phi_q]$. The differences between the two methods arise from the constraints imposed on the matrix factors $\mathbf{u}_k$ and $\mathbf{v}_k$. PCA constrains $\mathbf{u}_k$'s to be orthonormal and $\mathbf{v}_k$'s to be orthogonal to each other. NMF, on the other hand, does not allow negative entries in the matrix factors $\mathbf{u}_k$ and $\mathbf{v}_k$. Singular value decomposition can be used to compute PCA factorization. Since only the first few summation terms of the factorization are needed, one can use an efficient algorithm that computes partial factorization. See Appendix for details on how to compute the factorizations.

Partitioning of light field data into vertex light fields ensures that in the resulting approximation each triangle shares its view maps with the neighboring triangles. Therefore, even though we decompose each vertex light field independently, we obtain an approximation that is continuous across triangles regardless of the number of approximation terms K. Let $g^{v_j}[r_p,s_p]$ be the surface map and $h^{v_j}[\theta_q,\phi_q]$ be the view map corresponding to one approximation term of vertex light field $f^{v_j}[r_p,s_p,\theta_q,\phi_q]$. Let $\widetilde{f}^{\triangle_i}[r_p,s_p,\theta_q,\phi_q]$ denote the corresponding approximation term of the light field data for triangle $\triangle_i$. The following equality holds

$$\widetilde{f}^{\triangle_i}[r_p,s_p,\theta_q,\phi_q] = \sum_{j=1}^{3} g^{v_j}_{\triangle_i}[r_p,s_p]\, h^{v_j}[\theta_q,\phi_q] \tag{6}$$

where index j runs over the three vertices of triangle $\triangle_i$ and $g^{v_j}_{\triangle_i}[r,s]$ denotes the portion the surface map corresponding to the triangle

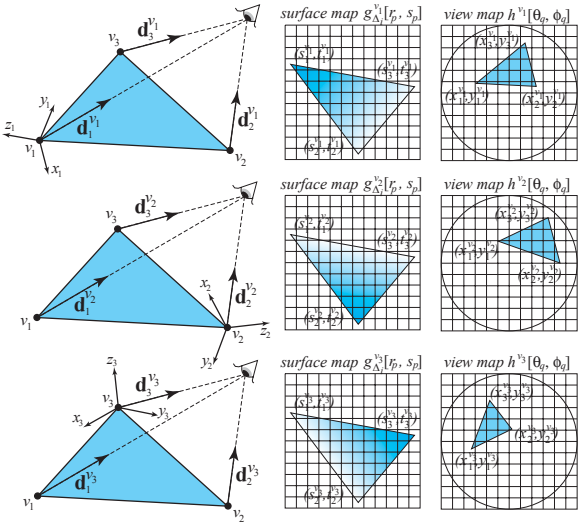


Figure 3: Light field maps for one approximation term of one triangle. Vertex reference frames are shown in the left column.

$\triangle_i$. Note that this equality holds for all approximation terms. Next, we show how this property can be used to develop a very simple and efficient rendering algorithm.

3 Rendering Algorithm

Our rendering algorithm takes advantage of the property of vertex-centered partitioning, described by Equation (6), which says that the light field for each triangle can be expressed independently as a sum of its 3 vertex light fields. This allows us to write a very efficient rendering routine that repeats the same sequence of operations for each mesh triangle. Additionally, since each light field approximation term is also evaluated in exactly the same way, we only need to explain how to evaluate one approximation term of one triangle.

Figure 3 shows six light field maps used in Equation (6) to compute one approximation term of light field for triangle $\triangle_i$. The middle column shows surface maps $g^{v_j}_{\triangle_i}[r_p, s_p]$. The pixels covered by the shaded triangle correspond to the points inside triangle $\triangle_i$ where we sampled the light field function. We will describe the texture coordinates of these points as (s,t). Note that due to partitioning the pixels of the surface maps are weighted, as indicated in the figure by gradient shading, but this does not alter the rendering algorithm in any way.

The right column shows view maps $h^{v_j}[\theta_q, \phi_q]$. In each image, the pixels inside the circle correspond to the orthographic projection of the hemisphere of viewing directions, expressed in the local coordinate system xyz of vertex v_j, onto the plane xy shifted and scaled to the range $(0,1)$. We will describe the texture coordinates of these points as (x,y). This projection allows a simple texture coordinate computation

$$x = (\mathbf{d}\cdot\mathbf{x}+1)/2, \qquad y = (\mathbf{d}\cdot\mathbf{y}+1)/2 \tag{7}$$

where $\mathbf{d}$ represents the normalized local viewing direction and vectors $\mathbf{x}$ and $\mathbf{y}$ correspond to the axes of the local reference frame. Other transformations from 3D directions to 2D maps are possible [15] but we found the one described here efficient and accurate.

Rendering of light field approximations is done using texture mapping. Based on where the camera is located, the rendering algorithm needs to access a different 2D subset of the 4D light field function. This is done by recomputing the view map coordinates $(x_i^{v_j}, y_i^{v_j})$ every time the camera moves. To this end, we apply Equations (7) to vectors $\mathbf{d}_i^{v_j}$, which represent the viewing directions to vertex v_i expressed in the reference frame of vertex v_j. This results in 3 texture fragments shown in the right column of Figure 3. Note that the texture coordinates are different for each fragment because we are using a different reference frame to compute them. Since the surface map texture coordinates $(s_i^{v_j}, t_i^{v_j})$ do not depend on the viewing angle, they do not need to be recomputed when the camera moves.

Evaluating one complete approximation term is equivalent to multiplying pixel-by-pixel the image projections of the surface map and view map texture fragment pairs for the 3 vertex light fields of the triangle, each shown in a separate row of Figure 3, and adding the results together. The multiple term approximation of each triangle light field is computed by simply adding the results of each approximation term.

3.1 Hardware Accelerated Implementation

In this section we discuss efficient implementations of light field mapping using specific hardware features such as multitexturing and extended color range. Recall that one of the fundamental operations of the proposed rendering algorithm was the pixel-by-pixel multiplication, or modulation, of the surface map fragment by the corresponding view map fragment. Multitexturing hardware support enables us to compute the modulation of multiple texture fragments very efficiently in one rendering pass. Consequently, for the NMF-based approximation of light field data, which produces strictly positive light field maps, we need at most 3 rendering passes to render each light field approximation term using multitexturing graphics hardware that supports 2 texture sources. Without multitexturing hardware support we can implement the rendering algorithms described above using an accumulation buffer, though significantly less efficiently.

For the PCA-based approximation, which in general will produce light field maps that contain negative values, rendering can benefit from graphics hardware that permits a change to the limits of the color range from the traditional $[0,1]$ range to $[min, max]$, e.g., $[-1,1]$. NVIDIA cards support extended color range through the register combiners extension [35] but they still clamp the negative output to zero. This means that we can render PCA-based approximations using register combiners at the expense of doubling the rendering passes as follows. Let M be the result of modulation of two texture fragments A and B using register combiners. Let M_+ be the result of clamped modulation of fragments A and B and let M_- be the result of modulating $-A$ and B the same way. We can compute M by subtracting the outputs of the two modulations $M = M_+ - M_-$. Even when the graphics hardware does not support any extended pixel range we can still render PCA-based approximations, though the number of rendering passes required to evaluate one approximation term increases to 4.

PCA-based matrix factorization requires that we subtract the *mean view* from the light field matrix before performing the decomposition. It changes the rendering routine only slightly—we first texture map the triangle using its mean view and then add the approximation terms of the modified light field matrix exactly as it was done before. One of the advantages of extracting the mean view and rendering it separately is that in many cases this view represents an approximation to the diffuse component of the surface material and, as such, is interesting to visualize independently from the view-dependent component of the light field data.

4 Compression of Light Field Maps

Approximation through matrix factorization described in Section 2.2 can be thought of as a compression method that removes local redundancy present in the light field data. The compression ratio

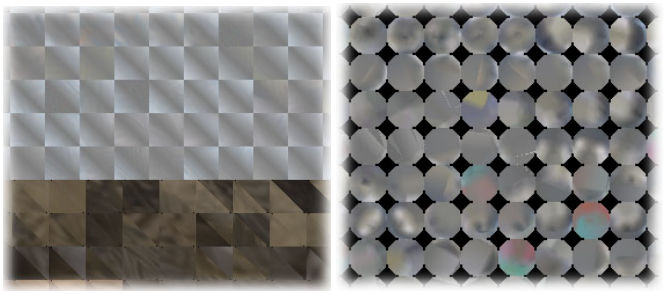


Figure 4: Surface maps (left) and view maps (right) computed using PCA-based approximation for the bust model. The lower portion of the left image represents the extracted mean textures. All other light field maps are stored in $[-1,1]$ range, where -1 is mapped to black.

of this method is closely related to the size of the surface primitives used for partitioning. On the one hand, a dense mesh with many vertices will have unnecessarily large number of view maps. On the other hand, a coarse mesh requires more approximation terms for the same approximation quality. See Section 6.1 for the analysis of the relationship of the compression ratio and the approximation quality to the triangle size. Currently, we choose the size of triangles empirically so that we obtain about two orders of magnitude compression ratio through approximation while maintaining high quality without using many approximation terms. Section 6.2 gives a detailed analysis of compression ratios obtained through the approximation.

Figure 4 shows, for the bust model, surface maps (left) and view maps (right) computed using PCA-based approximation. It is easy to see that the light field maps are still redundant. First, the individual maps are similar to each other, suggesting global redundancy of the data. Second, some of the light field maps have very little information content and can be compressed further using a variety of existing image compression techniques.

Figure 5 gives an overview of the different levels of compression we apply to the light field data. For optimal run-time performance, compressed light field maps need to fit entirely in the texture memory cache and be decompressed on-the-fly during rendering. This process should only introduce minimal run-time memory overhead. In the following paragraphs we discuss several techniques that satisfy these criteria. Other image compression techniques can be used to further reduce the offline storage size, as shown in Figure 5, but are not discussed in the paper.

Data redundancy *across* individual light field maps can be reduced effectively using VQ [13]. Our implementation represents each triangle surface map $g^{v_j}_{\triangle_i}[r_p, s_p]$ and each view map $h^{v_j}[\theta_q, \phi_q]$ as a vector. The algorithm groups these vectors based on their size and generates a separate codebook for every group. We initialize the codebooks using either pairwise nearest neighbor or split algorithm. The codebooks are improved by the generalized Lloyd algorithm utilizing square Euclidean distance as the cost function. We then store the resulting codebooks as images. The rendering algorithm from VQ-compressed images does not change in any way–it simply indexes a different set of images.

We use either a user-specified compression ratio or the average distortion to drive the VQ compression. With the distortion-driven algorithm, the light field maps corresponding to the higher approximation terms exhibit more redundancy and thus are often compressed into a smaller codebook. In practice, light field maps can be compressed using VQ by an order of magnitude without significant loss of quality.

Data redundancy *within* individual light field maps can be reduced efficiently using block-based algorithms. One such method, called S3TC^{TM}, is often supported on commodity graphics cards

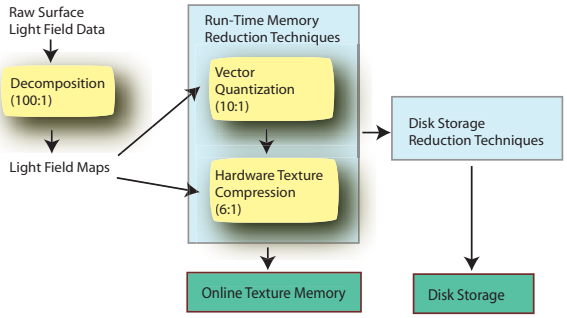


Figure 5: Compression Overview. The number under each technique describes its approximate compression ratio.

today. It offers compression ratios between 6:1 and 8:1 and can be cascaded with VQ for further size reduction. Limited by hardware implementation cost, these algorithms are not very sophisticated in nature. For example, the S3TC algorithm divides the textures into 4-by-4 texel blocks and performs color interpolation within each block. Since the algorithm uses blocks that are smaller than most light field maps, when compared to VQ, it generates noisier images but it preserves the specularities better.

For PCA-based approximation, we observe that the light field maps associated with higher approximation terms contain lower spatial frequency. To test if we can reduce the resolution of these light field maps without significant impact on the rendering quality, we implemented a simple method that subsamples the image resolution uniformly within each term. The results, although not reported in the paper, proved effective. A more sophisticated approach would apply selective resolution reduction to each light field map, or simply reduce light field matrix resolution during resampling.

5 Implementation Details

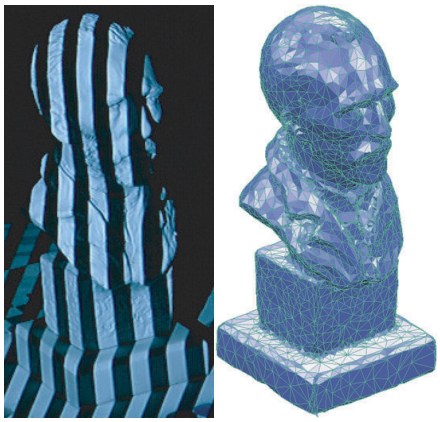
This section describes the following three implementation related issues: the acquisition of the geometry and the radiance data of physical objects, resampling of the radiance data, and tiling of the light field maps.

5.1 Data Acquisition

Figure 6 gives an illustration of the overall data acquisition process. A total of N_I ($200 < N_I < 400$) images are captured with a handheld digital camera (Fig. 6a). Figure 6b shows one sample image. Observe that the object is placed on a platform designed for the purpose of automatic registration. The color circles are first automatically detected on the images using a simple color segmentation scheme. This provides an initial guess for the position of the grid corners that are then accurately localized using a corner finder. The precise corner locations are then used to compute the 3D position of the camera relative to the object. This may be done, given that the camera has been pre-calibrated. The outcome of this process is a set of N_I images captured from known vantage points in 3D space.

The object geometry is computed using a structured lighting system consisting of a projector and a camera. The two devices are visible in Figure 6a. Figure 6c shows an example camera image acquired during scanning. The projector is used to project a translating stripped pattern onto the object. A similar temporal analysis employed by Curless *et al.* [8] and Bouguet *et al.* [3] is used for accurate range sensing. In order to facilitate scanning, the object is painted with white removable paint, a technique especially useful when dealing with dark, highly specular or semi-transparent objects. We take between 10 and 20 scans to completely cover the

(a) (b)



(c) (d) (e)

Figure 6: Data acquisition. (a) The user is capturing approximately 300 images of the object under a fixed lighting condition using a hand-held digital camera. (b) One sample image. The color circles guide the automatic detection of the grid corners that are used for computing the 3D location of the camera. (c) The painted object being scanned using the structured lighting system. (d) The complete 3D triangular mesh consisting of 7228 triangles constructed from 20 scans. (e) The projection of the triangular mesh onto image (b).

surface geometry of the object. Between two consecutive scans, the object is rotated in front of the camera and projector by about 20 degrees. For that purpose, the calibration platform has been designed to rotate about its central axis. The individual scans are automatically registered together in the object reference frame using the same grid corners previously used for image registration. The resulting cloud of points (approx. 500,000 points) is then fed into mesh editing software to build the final triangular surface mesh shown in Figure 6d. Since we use the same calibration platform for both image and geometry acquisition, the resulting triangular mesh is naturally registered to the camera images. Figure 6e shows the projection of the mesh onto the camera image displayed in Figure 6b illustrating the precise alignment of the geometry and the images. The image reprojection error is less than one pixel.

5.2 Data Resampling

Before we start resampling, we need to determine the visible cameras for each mesh triangle. A triangle is considered visible only if it is completely unoccluded. Repeating this process for all N_I images results in a list of visible views for each triangle. The visible triangle views correspond to a set of texture patches of irregular size captured from various viewing directions. We now explain the two-step process of resampling the irregularly sized views into a 4-dimensional grid.

Recall that the approximation algorithms described in Section 2.2 assume light field data in the form of matrices. This requires that we normalize each texture patch to have the same shape and size as the others. The size of the normalized patch is chosen to be equal to the size of the largest view. Resampling is done using bilinear interpolation of the pixels in the original views.

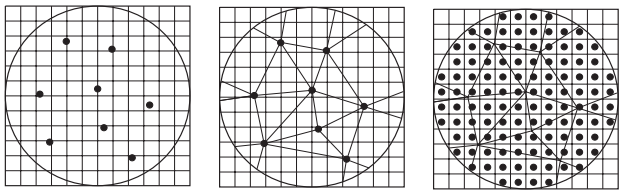


Figure 7: Resampling of views. Projection of original views (left), Delaunay triangulation of projected views (center), uniform grid of views computed by blending the original set of views (right).

At this stage, we have a uniform number of samples for each triangle view but the sampling of views is still irregular. Having a regular grid of viewing directions in the light field matrix is very important for two reasons. As was shown in [6], factorization of the fully resampled matrix results in a more precise approximation and encodes the view-dependent information allowing us to synthesize a correct image for any camera location using the rendering algorithm that was described in Section 3. We will use Figure 7 to explain the view resampling step. Let vectors $\mathbf{d}^{v_j}_{\triangle_i}$ be the viewing directions for the visible views of a given triangle expressed in the reference frame of vertex v_j. We start by projecting these directions onto the xy plane of the reference frame of vertex v_j using Equations (7). The result of this operation is a set of texture coordinates (Figure 7, left). Next we perform the Delaunay triangulation of these coordinates (Figure 7, middle) and compute the regular grid of views by blending the original triangle views using the weighting factors obtained from the triangulation (Figure 7, right).

We assume that the viewing direction is constant across a triangle, since the distance of the camera from the object during light field capture is generally quite large compared to the size of the triangles. The typical resolution of the view grid is $32 \times 32 = 1024$, but it can vary from triangle to triangle and between the approximation terms as explained in Section 4.

5.3 Tiling of Light Field Maps

To avoid excessive texture swapping and improve rendering efficiency, we tile individual light field maps together to create collection of light field maps. To simplify the problem, we allow a predefined set of sizes for the light field maps during the resampling process, and then tile same-size light field maps together, as shown in Figure 4. Since one triangle requires three surface maps per approximation term, we tile all these maps in the same collection. We split the geometry of the model into p segments so that the tiled view maps for each segment do not exceed maximum texture size allowed. We then produce one view map collection per segment: $[V_1, V_2 \ldots, V_p]$. Let $[S^i_1, S^i_2, \ldots, S^i_{q_i}]$ be the list of surface map collections for vertex map collection V_i. For each approximation term, the rendering algorithm is as follows

for $i = 1, \ldots, p$ **do**
  load view map collection V_i into texture unit 1
  for $j = 1, \ldots, q_i$ **do**
    load surface map collection S^i_j into texture unit 2
    render all triangles with surface maps in collection S^i_j
  end for
end for

6 Results

We have acquired the surface light fields and computed the light field map representations for four physical objects with diverse and

Models	Triangles	Number of Images	Raw Image Size	Resampled Data Size
Bust	6531	339	289 MB	5.1 GB
Dancer	6093	370	213 MB	2.9 GB
Star	4960	282	268 MB	5.2 GB
Turtle	3830	277	233 MB	3.8 GB

Table 1: The sizes of the experimental data sets. The second to last column shows the total size of renormalized triangles. The last column shows the size of the resampled surface light field data used for the decomposition.

complex reflection properties, including the metallically-painted clay *bust* in Figure 13, *dancer* in Figure 12, semi-transparent and anisotropically reflective glass *star* in Figure 14, and the furry toy *turtle* in Figure 11. We also constructed a synthetic museum room as shown in Figure 1. The scene has approximately 60K polygons. Using 1-term approximation, we can render it at 18 frames per second using the GeForce^{TM} 3 graphics card on a 2GHz Pentium^{TM} 4 PC.

6.1 Acquisition and Resampling

Table 1 provides information about the data sizes for the scanned models used in our experiments. We use 24-bit RGB images in all the experiments. The raw image size in this table represents the total amount of radiance data after the visibility computation and triangle normalization process as described in Section 5. When resampling the viewing directions we assume a resolution of 32×32. Note that, traditionally, research on light field compression reports results based on the size of the resampled light field data [22; 23]. The compression ratios reported in the next section are computed based on the size of resampled data.

Figure 8 illustrates the relationship of the compression ratio and the approximation quality to the triangle size for the bust object. As seen from the figure, smaller triangles provide better approximation quality but lower compression ratio. It is also interesting that for the same target error larger triangles always give a better compression ratio. Note also that smaller triangles are slower to render for each approximation term. Depending on the target hardware platform, user may choose to reduce the number of triangles while increasing the number of required passes, or vice versa, to achieve desired frame rate and compression ratio. For the following experiments, based on the goal of 100:1 compression obtained for the 3 term approximation with no compression of light field maps, we choose mesh resolution that gives us approximately 314 samples per triangle for all our models.

6.2 Approximation and Compression

We measure the quality of the surface light field approximation using RMS error computed between the resampled light field matrix and the matrix reconstructed from the approximation. Figure 9 shows PSNR (Peak Signal-to-Noise Ratio) for PCA-based and NMF-based approximations. The results show that both techniques produce high quality approximations using few approximation terms. Note that the scale in Figure 9 is logarithmic and that all approximations that use more than 3 terms have the RMS error of less than 2.0. This result provides a quantitative proof of the effectiveness of light field approximation through matrix factorization.

Between the two algorithms, PCA produces better quality than NMF, however, the difference is visually almost indistinguishable. Note that the frame buffer quantization that occurs during each texture blending operation is not considered in Figure 9. This quantization error implies that an approximation beyond $48dB$ would not improve the quality of hardware-accelerated renderer.

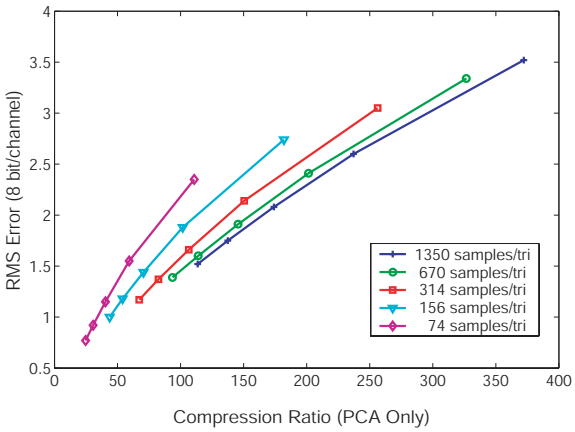


Figure 8: Relationship of the compression ratio and the approximation quality to the triangle size for the bust object. The points on each curve correspond to the different number of approximation terms for a given mesh resolution. Coarser mesh has more samples per triangle.

Table 2 lists the size and compression ratio of the light field data obtained through the light field data approximation and the additional compression of the light field maps. The size of geometric data falls below 10KB for all objects listed in the table when compressed using topological surgery [36] and therefore is negligible compared to the size of light field maps. By combining VQ with S3TC hardware texture compression, we achieve a *run-time* compression ratio of over 5000:1 for a 3-term approximation. For interactive purposes, 1-term approximation is often sufficient and thus the resulting compression ratio approaches 4 orders of magnitude.

6.3 Rendering

Figure 10 compares the rendering performance of PCA-based and NMF-based approximations. NMF-based rendering is 50% faster than PCA-based for the same number of approximation terms. The performance disparity will be larger if the target platform only supports positive texture values. The rendering performance of our algorithm is not very sensitive to the size of light field map data—doubling the image size reduces the frame rate by less than 20%. Rendering from compressed and uncompressed light fields is equally fast if image sets in both cases fit into the texture memory.

Figures 11-14 compare the rendering quality of our routines against the input images and report the corresponding errors. The errors reported in the figures are computed based on the difference between the input image and the rendered image using both APE (average pixel error) and PSNR for the *foreground* pixels only. Currently we discard partially visible triangles in the resampling process, which also contributes to the error. In the future, we plan to address the problem of partially occluded triangles by looking at factor analysis algorithms that use data statistics to fill in missing information [32].

7 Conclusion and Future Work

We have developed a new representation of surface light fields and demonstrated its effectiveness on both synthetic and real data. Using our approach, surface light fields can be compressed several thousand times and efficiently rendered at interactive speed on modern graphics hardware directly from their compressed representation. Simplicity and compactness of the resulting representation leads to a straightforward and fully hardware-accelerated rendering algorithm. Additionally, we present a new type of light field

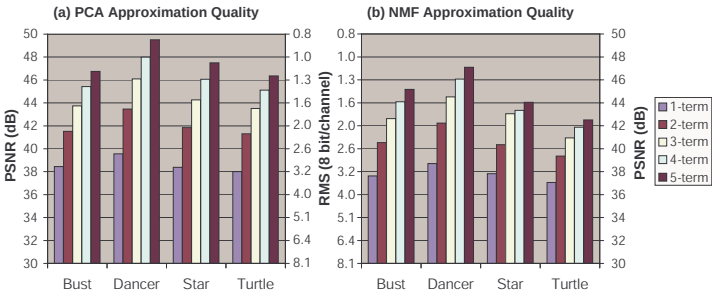


Figure 9: Approximation quality for different models and different number of decomposition terms. PSNR and RMS are based on the weighted average of the approximation errors for all light field matrices.

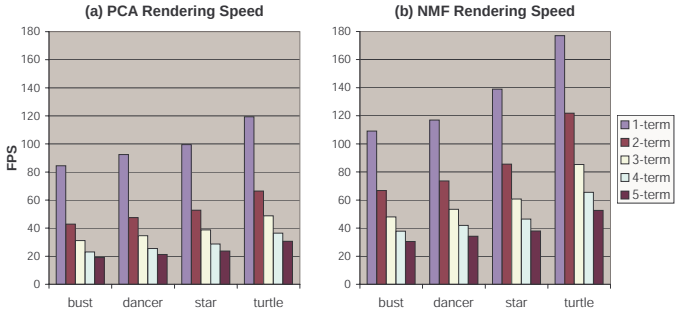


Figure 10: Rendering performance using NVIDIA GeForce 3 graphics card on a 2GHz Pentium 4 PC displayed at 1024×768 window with objects occupying approximately 1/3 of the window.

data factorization that produces positive only factors. This method allows faster rendering using commodity graphics hardware. Furthermore, the paper contains a detailed explanation of the data acquisition and preprocessing steps, providing a description of the complete modeling and rendering pipeline. Finally, our PCA-based approximation technique is particularly useful for network transport and interactive visualization of 3D photography data because it naturally implies progressive transmission of radiance data.

One of the limitations of a surface light field is that it models only the outgoing radiance of a scene. Consequently it is not possible to use surface light fields to render dynamic scenes where lighting changes and objects move. In the future, we are planning to work on extending the approach presented in this paper to relighting and animation of image-based models. If successful, these results would prove that image-based modeling and rendering is a practical and an appealing paradigm for enhancing the photorealism of interactive 3D graphics.

Acknowledgments

We thank Srihari Sukumaran for helping with the early version of the resampling and tiling algorithm. We are indebted to Omid Moghadam for his support in making light field mapping part of the MPEG4 standard. We thank Gary Bishop, Bernd Girod and Pat Hanrahan for critical reads of this paper. Timely proofreading by Gordon Stoll is greatly appreciated. We thank Henry Fuchs, Anselmo Lastra, Ara Nefian, Lars Nyland and Herman Towles for helpful suggestions and discussions and Gary Bradski, Bob Liang and Justin Rattner for encouraging this work. This research is in part supported by NSF Cooperative Agreement "Science and Technology Center for Computer Graphics and Scientific Visualization", "National Teleimmersion Initiative" funded by Advanced Networks and Services, and Link Foundation Fellowships.

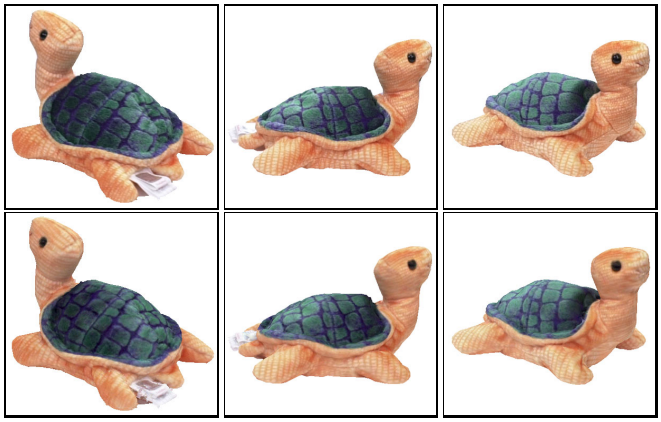


Figure 11: The figure shows a comparison for the turtle model between the input images shown at the top row and the images synthesized from the 1-term PCA approximation compressed using both VQ and S3TC shown at the bottom row. APE = 9.5, PSNR = 25.5 dB, the compression ration is 8202:1, and the size of compressed light field maps is 468 KB.

References

[1] C. M. Bishop. *Neural Networks for Pattern Recognition*. Clarendon Press, 1995.

[2] S. Boivin and A. Gagalowicz. Image-Based Rendering of Diffuse, Specular and Glossy Surfaces From a Single Image. *Proceedings of SIGGRAPH 2001*, pages 107–116, August 2001.

[3] J-Y. Bouguet and P. Perona. 3D Photography Using Shadows in Dual-Space Geometry. *International Journal of Computer Vision*, 35(2):129–149, December 1999.

[4] C. Buehler, M. Bosse, L. McMillan, S. J. Gortler, and M. F. Cohen. Unstructured Lumigraph Rendering. *Proceedings of SIGGRAPH 2001*, pages 425–432, August 2001.

[5] J-X. Chai, X. Tong, S-C. Chan, and H-Y. Shum. Plenoptic Sampling. *Proceedings of SIGGRAPH 2000*, pages 307–318, July 2000.

[6] W-C. Chen, R. Grzeszczuk, and J-Y. Bouguet. Light Field Mapping: Hardware-accelerated Visualization of Surface Light Fields. Published as part of "Acquisition and Visualization of Surface Light Fields," SIGGRAPH 2001 Course Notes for Course 46, pages 410-416, August 2001.

[7] R. L. Cook and K. E. Torrance. A Reflectance Model for Computer Graphics. *ACM Transactions on Graphics*, 1(1):7–24, January 1982.

[8] B. Curless and M. Levoy. Better Optical Triangulation through Spacetime Analysis. *Proc. 5th Int. Conf. Computer Vision, Boston, USA*, pages 987–993, 1995.

[9] P. Debevec, T. Hawkins, C. Tchou, H-P. Duiker, W. Sarokin, and M. Sagar. Acquiring the Reflectance Field of a Human Face. *Proceedings of SIGGRAPH 2000*, pages 145–156, July 2000.

[10] P. E. Debevec, C. J. Taylor, and J. Malik. Modeling and Rendering Architecture from Photographs: A Hybrid Geometry- and Image-Based Approach. *Proceedings of SIGGRAPH 96*, pages 11–20, August 1996.

[11] P. E. Debevec, Y. Yu, and G. D. Borshukov. Efficient View-Dependent Image-Based Rendering with Projective Texture-Mapping. *Eurographics Rendering Workshop 1998*, pages 105–116, June 1998.

[12] A. Fournier. Separating Reflection Functions for Linear Radiosity. *Eurographics Rendering Workshop 1995*, pages 296–305, June 1995.

[13] A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, 1992.

[14] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. The Lumigraph. *Proceedings of SIGGRAPH 96*, pages 43–54, August 1996.

[15] W. Heidrich and H-P. Seidel. Realistic, Hardware-Accelerated Shading and Lighting. *Proceedings of SIGGRAPH 99*, pages 171–178, August 1999.

[16] J. Kautz and M. D. McCool. Interactive Rendering with Arbitrary BRDFs using Separable Approximations. *Eurographics Rendering Workshop 1999*, June 1999.

[17] J. Kautz and H-P. Seidel. Towards Interactive Bump Mapping with Anisotropic Shift-Variant BRDFs. *2000 SIGGRAPH / Eurographics Workshop on Graphics Hardware*, pages 51–58, August 2000.

[18] J.J. Koenderink and A.J. van Doorn. Bidirectional Reflection Distribution Function Expressed in Terms of Surface Scattering Modes. In *European Conference on Computer Vision*, pages II:28–39, 1996.

[19] E. P. F. Lafortune, S-C. Foo, K. E. Torrance, and D. P. Greenberg. Non-Linear

Approximation of Reflectance Functions. *Proceedings of SIGGRAPH 97*, pages 117–126, August 1997.

[20] D. D. Lee and H. S. Seung. Learning the Parts of Objects by Non-Negative Matrix Factorization. *Nature*, 401:788–791, 1999.

[21] Hendrik P. A. Lensch, Jan Kautz, Michael Goesele, Wolfgang Heidrich, and Hans-Peter Seidel. Image-Based Reconstruction of Spatially Varying Materials. In *Twelveth Eurographics Rendering Workshop 2001*, pages 104–115. Eurographics, June 2001.

[22] M. Levoy and P. Hanrahan. Light Field Rendering. *Proceedings of SIGGRAPH 96*, pages 31–42, August 1996.

[23] M. Magnor and B. Girod. Data Compression for Light Field Rendering. *IEEE Trans. Circuits and Systems for Video Technology*, 10(3):338–343, April 2000.

[24] T. Malzbender, D. Gelb, and H. Wolters. Polynomial Texture Maps. *Proceedings of SIGGRAPH 2001*, pages 519–528, August 2001.

[25] Michael D. McCool, Jason Ang, and Anis Ahmad. Homomorphic Factorization of BRDFs for High-Performance Rendering. *Proceedings of SIGGRAPH 2001*, pages 171–178, August 2001.

[26] L. McMillan and G. Bishop. Plenoptic Modeling: An Image-Based Rendering System. *Proceedings of SIGGRAPH 95*, pages 39–46, August 1995.

[27] G. S. P. Miller, S. Rubin, and D. Ponceleon. Lazy Decompression of Surface Light Fields for Precomputed Global Illumination. *Eurographics Rendering Workshop 1998*, pages 281–292, June 1998.

[28] K. Nishino, Y. Sato, and K. Ikeuchi. Eigen-Texture Method: Appearance Compression Based on 3D Model. In *Proceedings of the IEEE Computer Science Conference on Computer Vision and Pattern Recognition (CVPR-99)*, pages 618–624, 1999.

[29] P. Poulin and A. Fournier. A Model for Anisotropic Reflection. *Proceedings of SIGGRAPH 90*, 24(4):273–282, August 1990.

[30] K. Pulli, M. Cohen, T. Duchamp, H. Hoppe, L. Shapiro, and W. Stuetzle. View-based Rendering: Visualizing Real Objects from Scanned Range and Color Data. *Eurographics Rendering Workshop 1997*, pages 23–34, June 1997.

[31] R. Ramamoorthi and P. Hanrahan. A Signal-Processing Framework for Inverse Rendering. *Proceedings of SIGGRAPH 2001*, pages 117–128, August 2001.

[32] Sam Roweis. EM algorithms for PCA and SPCA. In *Advances in Neural Information Processing Systems*, volume 10. The MIT Press, 1998.

[33] Y. Sato, M. D. Wheeler, and K. Ikeuchi. Object Shape and Reflectance Modeling from Observation. *Proceedings of SIGGRAPH 97*, pages 379–388, August 1997.

[34] P. Schröder and W. Sweldens. Spherical Wavelets: Efficiently Representing Functions on the Sphere. *Proceedings of SIGGRAPH 95*, pages 161–172, August 1995.

[35] J. Spitzer. Texture Compositing With Register Combiners. Game Developers Conference, April 2000.

[36] G. Taubin and J. Rossignac. Geometric Compression Through Topological Surgery. *ACM Transactions on Graphics*, 17(2):84–115, April 1998.

[37] K. E. Torrance and E. M. Sparrow. Polarization, Directional Distribution, and Off-Specular Peak Phenomena in Light Reflected from Roughened Surfaces. *Journal of Optical Society of America*, 56(7), 1966.

[38] G. J. Ward. Measuring and Modeling Anisotropic Reflection. *Proceedings of SIGGRAPH 92*, 26(2):265–272, July 1992.

[39] D. N. Wood, D. I. Azuma, K. Aldinger, B. Curless, T. Duchamp, D. H. Salesin, and W. Stuetzle. Surface Light Fields for 3D Photography. *Proceedings of SIGGRAPH 2000*, pages 287–296, July 2000.

[40] Y. Yu, P. E. Debevec, J. Malik, and T. Hawkins. Inverse Global Illumination: Recovering Reflectance Models of Real Scenes From Photographs. *Proceedings of SIGGRAPH 99*, pages 215–224, August 1999.

Appendix: Matrix Factorization

Let $\mathbf{F}$ be a matrix of size $M \times N$. The goal of matrix factorization leading to Equation (5) is to compute an $M \times K$ matrix $\mathbf{U} = [\mathbf{u}_1 \cdots \mathbf{u}_K]$ and an $N \times K$ matrix $\mathbf{V} = [\mathbf{v}_1 \cdots \mathbf{v}_K]$ such that the matrix

$$\widetilde{\mathbf{F}} = \mathbf{U}\mathbf{V}^T \tag{8}$$

best approximates $\mathbf{F}$ in the least squares sense. The PCA and NMF factorizations are two algorithms achieving this goal while enforcing different constraints on the vectors $\mathbf{u}_k$ and $\mathbf{v}_k$. PCA enforces the vectors $\mathbf{u}_k$ and $\mathbf{v}_k$ to be orthogonal and keeps the factorization progressive—that is, once the order K factorization is computed, the K-1 first pairs of vectors provide the best approximation at the order K-1. NMF, on the other hand, enforces vectors $\mathbf{u}_k$ and $\mathbf{v}_k$ to have all positive entries. Unlike PCA, NMF produces a non-progressive factorization. In other words, if a different approximation order K is chosen, the overall matrices $\mathbf{U}$ and $\mathbf{V}$ have to be recomputed. The next two sections describe the implementation.

PCA Algorithm

The PCA factorization is based on computing the partial singular value decomposition of matrix $\mathbf{F}$. Since only the first $K \ll N$ eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_K$ of matrix $\mathbf{A} = \mathbf{F}^T\mathbf{F}$ need to be computed, the power iteration algorithm is well suited to achieve this goal. The pseudo-code for this algorithm looks as follows:

for $p = 1, \ldots, K$ **do**
  $\mathbf{F}_p \leftarrow \mathbf{F} - \sum_{k=1}^{p-1} \mathbf{u}_k \mathbf{v}_k^T$
  $\mathbf{A}_p \leftarrow \mathbf{F}_p^T \mathbf{F}_p$
  Initialize $\mathbf{v}_p \leftarrow$ random $N \times 1$ non-zero vector
  repeat
    $\mathbf{v}_p \leftarrow A_p \mathbf{v}_p$
    $\mathbf{v}_p \leftarrow \mathbf{v}_p / \|\mathbf{v}_p\|$
  until $\mathbf{v}_p$ converges
  $\lambda_p \leftarrow \sqrt{A_p \mathbf{v}_p}$
  $\mathbf{u}_p \leftarrow \mathbf{F}_p \mathbf{v}_p / \lambda_p$
  $\max_u \leftarrow$ max of the absolute values of all the entries of $\mathbf{u}_p$
  $\max_v \leftarrow$ max of the absolute values of all the entries of $\mathbf{v}_p$
  $\alpha \leftarrow \sqrt{\max_u \lambda_p / \max_v}$
  $\mathbf{u}_p \leftarrow \lambda_p \mathbf{u}_p / \alpha$
  $\mathbf{v}_p \leftarrow \alpha \mathbf{v}_p$
  Quantize the entries of $\mathbf{u}_p$ and $\mathbf{v}_p$ for texture storage
end for

The coefficient α introduced in the code helps splitting the multiplicative factor λ_p among the vectors $\mathbf{u}_p$ and $\mathbf{v}_p$ so as to minimize the maximum quantization error.

NMF algorithm

The NMF factorization achieves the same matrix decomposition while enforcing all entries of the two matrices $\mathbf{U}$ and $\mathbf{V}$ to be positive. We propose to apply the algorithm presented by Lee *et al.* [20] to compute $\mathbf{U}$ and $\mathbf{V}$ iteratively:

Initialize $\mathbf{U} \leftarrow$ random $M \times K$ matrix of all strictly positive entries
Initialize $\mathbf{V} \leftarrow$ random $N \times K$ matrix of all strictly positive entries
repeat
  $\mathbf{U}_{n_1} = [u_{n_1}(i,j)] \leftarrow \mathbf{F}\mathbf{V}$
  $\mathbf{U}_{n_2} = [u_{n_2}(i,j)] \leftarrow \mathbf{U}\mathbf{V}^T\mathbf{V}$
  $\mathbf{U}_n = [u_n(i,j)] \leftarrow [u_{n_1}(i,j)/u_{n_2}(i,j)]$
  $\mathbf{V}_{n_1} = [v_{n_1}(i,j)] \leftarrow \mathbf{F}^T\mathbf{U}$
  $\mathbf{V}_{n_2} = [v_{n_2}(i,j)] \leftarrow \mathbf{V}\mathbf{U}^T\mathbf{U}$
  $\mathbf{V}_n = [v_n(i,j)] \leftarrow [v_{n_1}(i,j)/v_{n_2}(i,j)]$
  $\mathbf{U} = [u(i,j)] \leftarrow [u(i,j) * u_n(i,j)]$
  $\mathbf{V} = [v(i,j)] \leftarrow [v(i,j) * v_n(i,j)]$
  $\mathbf{U} = [u(i,j)] \leftarrow [u(i,j) / \sum_{r=1}^{M} u(r,j)]$
until $\mathbf{U}$ and $\mathbf{V}$ converge

Models	Light Field Maps (3-term)	Compression of Light Field Maps		
		VQ	S3TC	VQ+S3TC
Bust	47.7 MB (106:1)	5.7 MB (885:1)	8.0 MB (636:1)	951 KB (5475:1)
Dancer	35.7 MB (82:1)	5.4 MB (542:1)	5.9 MB (490:0)	904 KB (3303:1)
Star	42.3 MB (122:1)	7.2 MB (716:1)	7.0 MB (737:1)	1.2 MB (4301:1)
Turtle	31.7 MB (121:1)	4.5 MB (847:1)	5.3 MB (726:1)	755 KB (5084:1)

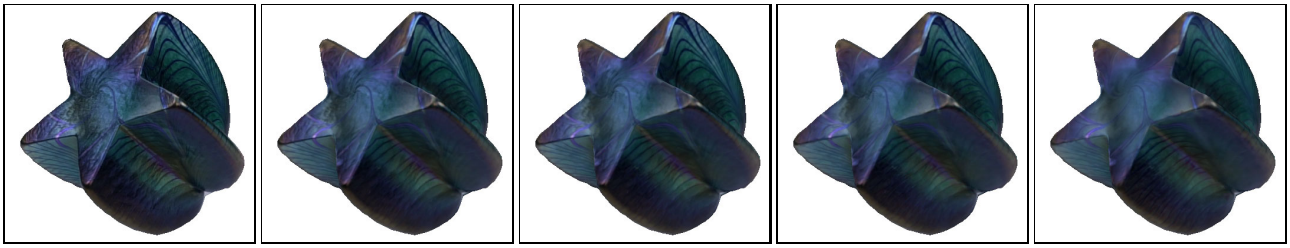
Table 2: The size and compression ratio of the radiance data obtained through the light field map approximation and additional compression of the surface light field maps.

Photograph | 4-term PCA APE=5.42, PSNR=27.63 dB 46.4 MB(63:1) | 4-term NMF APE=5.82, PSNR=27.35 dB 46.4 MB(63:1) | 2-term PCA APE=6.43, PSNR=26.77 dB 24.9 MB(117:1) | 2-term NMF APE=6.76, PSNR=26.54dB 24.9 MB(117:1)

Figure 12: Comparison between PCA and NMF approximation methods. Using the same number of terms, PCA light field maps produce less error, but are slower to render than NMF.

Photograph | 3-term PCA APE=4.57, PSNR=31.04 dB 47.7 MB (106:1) | 3-term PCA+VQ APE=6.98, PSNR=27.90 dB 5.7MB (885:1) | 3-term PCA+VQ+S3TC APE=7.49, PSNR=27.51 dB 951KB (5475:1) | Triangle Mesh

Figure 13: Comparison between different light field map compression algorithms using the bust model. VQ tends to diminish the highlight while S3TC preserves highlights better at the expense of color quality.



Photograph | 4-term PCA APE=4.79, PSNR=30.48 dB 54.4MB (95:1) | 3-term PCA APE=5.28, PSNR=29.85 dB 42.3MB (122:1) | 2-term PCA APE=5.99, PSNR=28.95 dB 30.2MB (171:1) | 1-term PCA APE=7.34, PSNR=27.51 dB 18.1MB (285:1)

Figure 14: The figure demonstrates the progressive nature of PCA approximation. The same star model is rendered using different number of approximation terms.

MPEG-4 Animation Framework eXtension (AFX) VM—

Node Description for Light Field Mapping

1 Geometry tools

1.1 Introduction

The geometry tools enhance the existing geometry tools defined in BIFS. In this section we propose new texture mapping techniques as well as new shapes.

1.2 Textures

1.2.1 Light Field Mapping

1.2.1.1 Overview

Light Field Mapping (LFM) is a method for progressive encoding, efficient compression and interactive rendering of surface light fields. A surface light field is a 4-dimensional function $f(r,s,\theta,\phi)$ that completely defines the radiance of every point on the surface of an object in every viewing direction. The first pair of parameters of this function (r,s) describes the surface location and the second pair of parameters (θ,ϕ) describes the viewing direction.

Intuitively, the LFM representation can be thought of as a special type of texture map that changes its appearance with the viewing angle. Because it is compact and allows for hardware-accelerated rendering, this representation is ideal for accurate modeling of the appearance of many physical and synthetic objects with complex surface reflectance properties. The system consists of three main components: approximation, compression and rendering that are described briefly.

Approximation Step

The first component approximates the 4-dimensional surface light field function f as a sum of a small number of products of lower-dimensional functions

$$f(r,s,\theta,\varphi) \approx \sum_{k=1}^{K} g_k(r,s)\, h_k(\theta,\phi)$$

Equation 1: Describes the approximation of light field data as a sum of a small number of produces of lower-dimensional functions.

The discrete functions g_k and h_k encode the light field data and are stored in a sampled form as texture maps, called *light field maps*. The functions g_k are called the *surface maps* and functions h_k as the *view maps* based on their parameterization. By taking advantage of the existing hardware support for texture mapping and composition surface light fields are visualized directly from the proposed representation at highly interactive frame rates. The process of rendering from this representation is called *light field mapping*.

Compression Step

Although it is possible to render surface light fields directly from the approximation described in the last section, further compression of the surface light field data is possible. Since the light field maps are in essence collections of images that themselves exhibit redundancy, they can be further compressed using standard image compression techniques.

Rendering Step

The sequence of rendering operations for each surface primitive is always the same. It starts with computing the view-dependent texture coordinates for the view maps. Subsequently, it proceeds to evaluate each approximation term and adds them together. Each approximation term is evaluated the same way:

1. The algorithm texture maps the surface primitive using the surface map.
2. It texture maps the surface primitive using the fragment of the view map determined by the view-dependent texture coordinates.
3. It performs pixel-by-pixel multiplication of the results of the two texture mapping operations. The following sections describe the rendering algorithm in more details.

Partitioning of Data

Since the geometry of the models is represented as a triangular mesh, an obvious partitioning of the light field function is to split it between individual triangles. Unfortunately, an approximation of surface light field partitioned this way results in visible discontinuities at the edges of the triangles. To eliminate the discontinuities across triangle boundaries the light field data is partitioned around every vertex. The surface light field unit corresponding to each vertex is called the *vertex light field*. For vertex v_j, it is denoted as $f^{v_j}[r_p, q_p, \theta_q, \phi_q]$. Partitioning is computed by weighting the surface light field function

$$f^{v_j}[r_p, q_p, \theta_q, \phi_q] = \Lambda^{v_j}[r_p, q_p] f[r_p, q_p, \theta_q, \phi_q]$$

where $\Lambda^{v_j}[r_p, q_p]$ is the barycentric weight of each point in the ring of triangles around vertex v_j.

In the final step of vertex-centered partitioning each vertex light field is reparameterized to the local coordinate system of the vertex. To this end, the viewing direction angles are defined as the azimuth and elevation angles of the viewing direction vector in the local reference frame of the vertex. The vertex reference frame is defined in such a way that its z-axis is parallel to the surface normal at the vertex. The same letters are used to denote the local parameters of the vertex light field in order to simplify the notation.

Partitioning of light field data into vertex light fields ensures that in the resulting approximation each triangle shares its view maps with the neighboring triangles. Therefore, even though we approximate each vertex light field independently, we obtain an approximation that is continuous across triangles regardless of the number of approximation terms K in Equation 1. Let $g^{v_j}[r_p, q_p]$ be the surface map and $h^{v_j}[\theta_q, \phi_q]$ be the view map corresponding to one approximation term of vertex light field $f^{v_j}[r_p, q_p, \theta_q, \phi_q]$, i.e.,

$$f^{v_j}[r_p, q_p, \theta_q, \phi_q] = g^{v_j}[r_p, q_p]\, h^{v_j}[\theta_q, \phi_q]$$

Equation 2: One term approximation of light field for vertex v_j.

Let $f^{\Delta_i}[r_p, q_p, \theta_q, \phi_q]$ denote the corresponding approximation term of the light field data for triangle Δ_i. The following equality holds

$$f^{\Delta_i}[r_p, q_p, \theta_q, \phi_q] = \sum_{j=1}^{3} g_{\Delta_i}^{v_j}[r_p, q_p]\, h^{v_j}[\theta_q, \phi_q]$$

Equation 3: One term approximation of light field for triangle Δ_i expressed as a sum of the triangle's 3 vertex light fields.

In above equation index *j* runs over the three vertices of triangle Δ_i and $g_{\Delta_i}^{v_j}[r_p, q_p]$ denotes the portion of the surface map corresponding to the triangle Δ_i. This equality holds for all approximation terms.

Rendering Algorithm

The rendering algorithm takes advantage of the property of vertex-centered partitioning, described by Equation 3, which says that the light field for each triangle can be expressed independently as a sum of its 3 vertex light fields. It enables a very efficient rendering routine that repeats the same sequence of operations for each mesh triangle. As each light field approximation term is evaluated the same way, the description of how to evaluate one approximation term of one triangle is sufficient to completely describe the rendering algorithm.

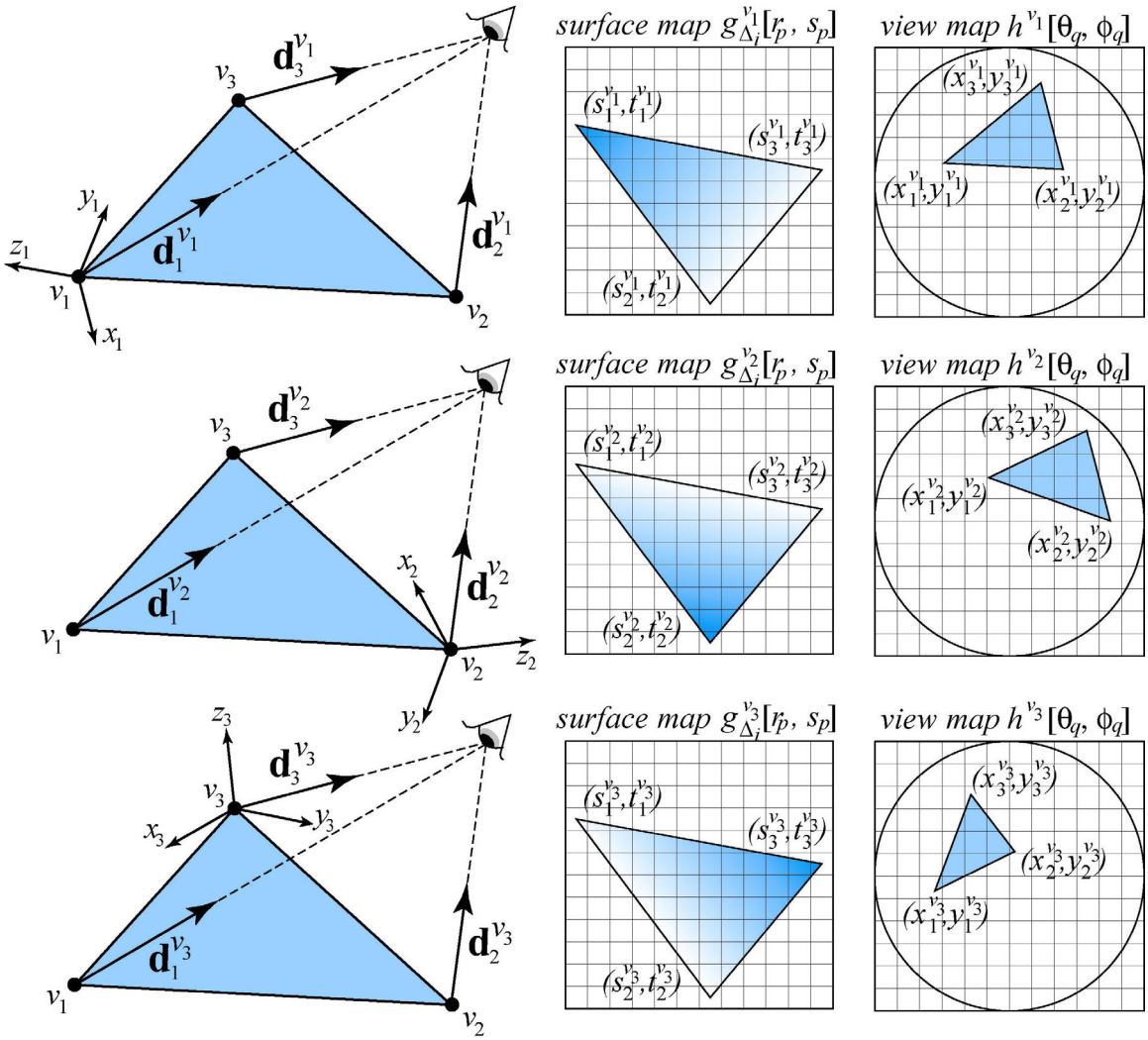


Figure 1: Light field maps for one approximation term of one triangle. Vertex reference frames are shown in the left column.

Figure 1 shows 6 light field maps used in Equation 3 to compute one approximation term of light field for triangle Δ_i. The middle column shows surface maps $g_{\Delta_i}^{v_j}[r_p, q_p]$. In each image, the pixels covered by the shaded triangle correspond to the points inside triangle Δ_i where we sampled the light field function. We will describe the texture coordinates of these points as (*s*, *t*). As a result of the weighting applied during the construction of function $f^{v_j}[r_p, q_p, \theta_q, \phi_q]$ the pixels of the surface maps are weighted, as indicated in the figure by gradient shading, but this does not alter the rendering algorithm in any way.

The right column shows view maps $h^{v_j}[\theta_q, \phi_q]$. In each image, the pixels inside the circle correspond to the orthographic projection of the hemisphere of viewing directions, expressed in the local coordinate system *xyz* of vertex v_j, onto the plane *xy* shifted and scaled to the range (0,1). We will describe the texture coordinates of these points as (*x*,*y*). This projection allows a simple texture coordinate computation

$$x = (\mathbf{d} \cdot \mathbf{x} + 1)/2 \quad y = (\mathbf{d} \cdot \mathbf{y} + 1)/2$$

Equation 4: Equations describing parameterization of viewing directions.

In the equation above **d** represents the normalized local viewing direction and vectors **x** and **y** correspond to the axes of the local reference frame. Other transformations from 3D directions to 2D maps are possible but the one described here is efficient and accurate.

Based on where the camera is located, the rendering algorithm needs to access a different 2D subset of the 4D light field function. This is done by recomputing the view map coordinates $(x_i^{v_j}, y_i^{v_j})$ every time the camera moves. To this end, we apply Equation 4 to vectors $\mathbf{d}_i^{v_j}$ that represent the viewing directions to vertex v_i expressed in the reference frame of vertex v_j. This results in 3 texture fragments shown in the right column of Figure 1. Note that the texture coordinates are different for each fragment because we are using a different reference frame to compute them. Note also that the surface map texture coordinates $(s_i^{v_j}, t_i^{v_j})$ do not depend on the viewing angle and they do not need to be recomputed when the camera moves.

Evaluating one complete approximation term is equivalent to multiplying pixel-by-pixel the image projections of the surface map and view map texture fragment pairs for the 3 vertex light fields of the triangle, each shown in a separate row of Figure 1 and adding the results together. The multiple term approximation of each triangle light field is computed by simply adding the results of each approximation term.

Tiling of Light Field Maps

To avoid excessive texture swapping and improve rendering efficiency, we tile individual light field maps together to create tiled texture maps. Through out this document we are going to use term **tile** to refer to the image containing tiled light field maps. To simplify the problem, we allow a predefined set of triangle sizes during the resampling process, and then tile same-size light field maps together, as shown in Figure 2. We will use the term **surface map tile** to refer to the image containing tiled surface maps and we will use the term **view map tile** to refer to the image containing tiled view maps.

Since one triangle requires three surface maps per approximation term, all these maps are arranged in the same texture. The geometry of the model is split into *p* segments so that the tiled view maps for each

segment do not exceed maximum texture size allowed. One view map tile is produced per segment: [V_1, V_2, ..., V_p]. Let $[S_1^i, S_2^i, \ldots, S_{q_i}^i]$ be the list of surface map tiles for vertex tile V_i. Note that, in general, each triangle represented by a given tile will have more than one surface map in this tile. For each approximation term, the rendering algorithm is as follows

```
for i=1,...,p do
  load view map tile  into texture unit 1
  for j=1,...,q_i do
    load surface map tile S_j^i into texture unit 2
    render all triangles with surface maps in tile S_j^i
  end for
end for
```

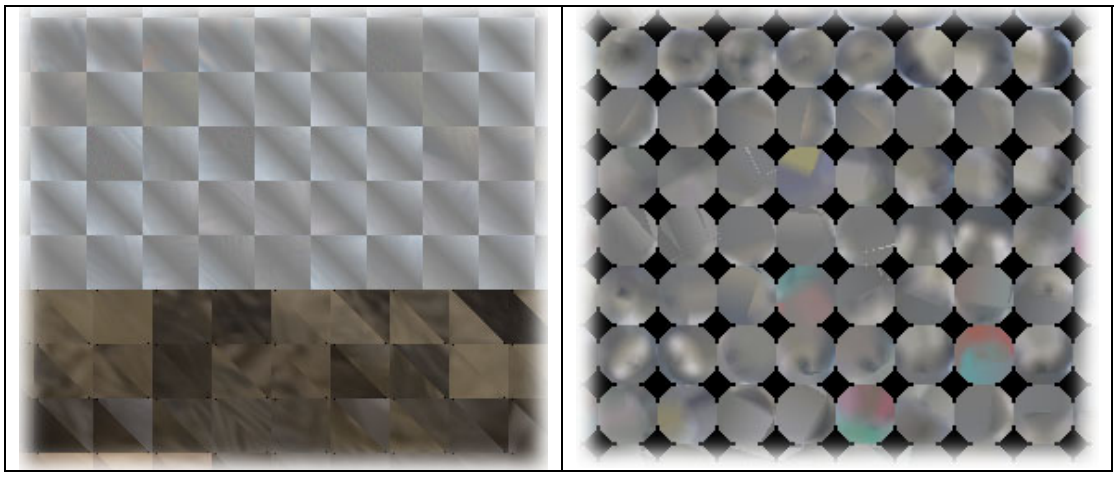


Figure 2: The tiled surface maps (left) and view maps (right)

1.2.1.2 Node specification

The root node that describes the Light Field is LFM_Appearance. This node should be present in the appearance field of a Shape node, while its geometry field should contain an IndexedFaceSet.

LFM_Appearance {			
exposedField	**MFNode**	**tileList**	**[]**
exposedField	**MFNode**	**lightMapList**	**[]**
exposedField	**SFNode**	**blendList**	**NULL**
exposedField	**LFM_FrameList**	**vertexFrameList**	**NULL**
}			

The field **tileList** describes the list of images containing the tiled light maps. Elements of this list should be nodes of type **ImageTexture**. In general, surface maps will be tiled separately from view maps and that not all of them are necessarily stored in a single image.

The LFM algorithm requires that each vertex of the mesh has a reference frame associated with it. The reference frames can be either computed automatically, as described in Appendix A, or they can be specified explicitly through **LFM_FrameList** node. For vertices that do not have the reference frame specified through the **LFM_FrameList** node, the reference frames are computed using the automatic rules. A node describing the reference frames for the vertices is defined as

LFM_FrameList {

exposedField	**MFInt32**	**index**	**[-1]**
exposedField	**MFVec3f**	**frame**	**[1 0 0, 0 1 0, 0 0 1]**

}

The field **lightMapList,** used in the definition of the **LFM_Appearance** node, describes how to access individual light maps from within the image tiles. The number of the elements in the light map list corresponds to the number of the decomposition terms used in the surface light field approximation. The list **lightMapList** should contain elements of type **LFM_LightMap**. The format for accessing surface maps is slightly different from the format for accessing the view maps; therefore, each element in the light map list consists of a field describing how to access the surface maps and a field describing how to access the view maps. Note that each one of those two fields is going to be a list as well.

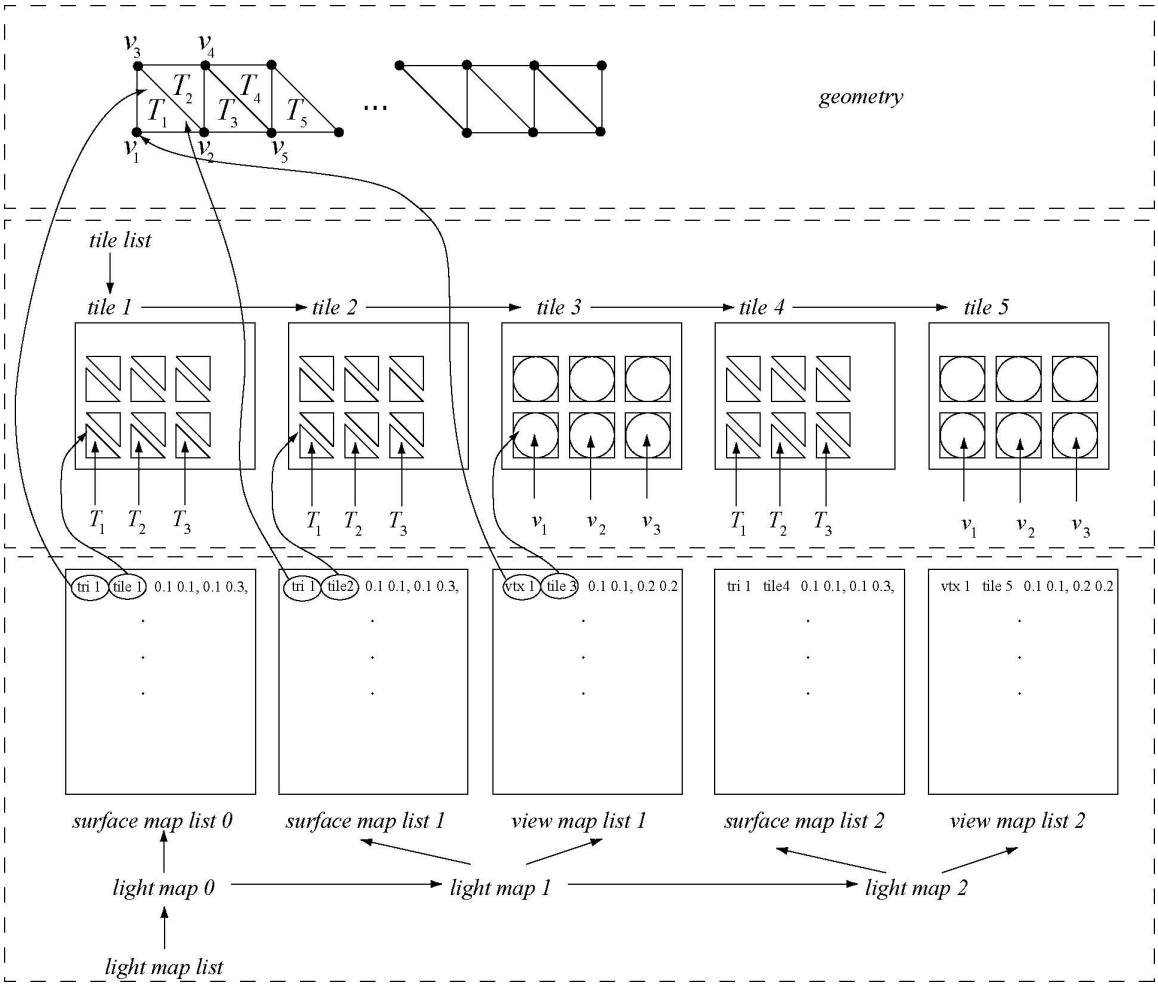


Figure 3 - This figure shows a diagram of the nodes used to define the shape that uses light field mapping appearance. The top of the figure shows the geometry of the object defined as a triangular. The middle of the figure shows the list of images containing the light map tiles. Finally, the bottom of the figure shows the light map list that specifies how to access individual light maps from within the tiles. Note that the first element of the light map list contains only a surface map list and no view map list. This element of the light map list would be used just like a regular texture map. The remaining two elements of the light map list each contain the par of surface map and view map list. In this case we would perform a multiplication of the corresponding surface maps and view maps. We made certain simplification in this diagram. For example, the diagram shows a one-to-one correspondence between the tiles and the surface/view map lists. In practice, this is not always the case, since we can have multiple lists for each tile.

LFM_LightMap {

exposedField	**SFNode**	**surfaceMapList**	**NULL**
exposedField	**SFNode**	**viewMapList**	**NULL**
exposedField	**SFVec3f**	**scaleRGB**	**1 1 1**
exposedField	**SFVec3f**	**biasRGB**	**0 0 0**
exposedField	**SFInt32**	**priorityLevel**	**0**

}

Scale and **bias** map the default color range [0, 1] of the image into the color range required for proper image synthesis. Let **s** be the RGB scale vector, let **b** be the RGB bias vector and let **c** be the RGB color value. The new color range $\mathbf{c}_{new}$ for color vector **c** can be computed using vectors **s** and **b** as follows

$$\mathbf{c}_{new} = \mathbf{s}\,\mathbf{c} + \mathbf{b}$$

Equation 5 – Scale and Bias map default color range to image synthesis color range..

The field **priorityLevel** is a non-negative integer value specifying the level of importance of a given **LightMapList** node. The lower the value associated with the node, the more important it is. If the value is **0**, the node must be rendered. When the renderer is set to a given priority value, then all nodes with the **priorityLevel** below that value must be rendered. For example, if the priority value of the render is set to 2, then all nodes with priority level 0, 1, and 2 must be rendered.

The lists **surfaceMapList** and **viewMapList** containing information about the surface maps and the view maps are defined as follows

LFM_SurfaceMapList {

exposedField	**MFInt32**	**triangleIndex**	**[]**
exposedField	**MFInt32**	**tileIndex**	**[]**
exposedField	**MFInt32**	**viewMapIndex**	**[]**
exposedField	**SFNode**	**triangleCoordinate**	**NULL**

}

where **triangleIndex** refers to the index of a given triangle as defined by the geometry node, **tileIndex** refers to the index of the tile as defined by the list of tiles, **viewMapIndex** indicates the number of view maps that should be used to render a given triangle (it equals –1 if there is no **viewMapList** in the current lightMap) and **triangleCoord** is a list of triples of texture coordinates of every triangle that specifies which part of the tile contains the surface map for the current triangle.

LFM_ViewMapList {

exposedField	**MFInt32**	**vertexIndex**	**[]**
exposedField	**MFInt32**	**tileIndex**	**[]**
exposedField	**SFNode**	**textureOrigin**	**NULL**
exposedField	**SFNode**	**textureSize**	**NULL**

}

where **vertexIndex** refers to the list of indices of vertices as defined by the geometry node, **tileIndex** refers to the list of indices of the tiles as defined by the list of tiles. The fields **textureOrigin** and **textureSize** are two lists of texture coordinates that specify the origin and the size of the view map, respectively, they should contain **TextureCoordinate nodes**.

The field **blendList,** used in the definition of the **LFM_Appearance** node, describes how to combine the individual light maps together during rendering. It may contain a **LFM_blendList** node

LFM_blendList {

	exposedField	**MFInt32**	**lightMapIndex**
	exposedField	**MFInt32**	**blendMode**

}

where **lightMapIndex** refers to the index of the lightMap in **lightMapList** and **tileIndex** refers to the index of the tile as defined by the list of tiles and the field **blendMode** specifies what type of blending operation to use when combining the given **LightMapList** node with the data in the framebuffer. This field can take on the following values: 0 or LFM_ADD, 1 or LFM_SUBTRACT.

Note the following implementation facts

1. For each triangle, each approximation term consists of 3 surface map/view map pairs.
2. In general, each element in the **LFM_LightMap** will have a pair of valid pointers, one pointing to a valid **LFM_SurfaceMapList** node and one pointing to a valid **LFM_ViewMapList** node. If this is the case, during rendering the corresponding surface maps and view maps get multiplied together and blended with the earlier rendering results.
3. In case, when an element of the **LFM_LightMap** list contains a valid pointer to the **LFM_SurfaceMapList** node, but the pointer to the **LFM_ViewMapList** node is **NULL**, we simply use the surface maps as ordinary texture maps.
4. Let **sm1**, **vm1**, ..., **smK**, **vmK** be the first K pairs of surface map/view map for a given triangle. Multitexturing can be supported in the form **sm0 + sm1*vm1 + sm2*vm2 + sm3*vm3 + ...**, where addition refers to adding (or subtracting, depending on the value of field **blendMode**) the results of rendering through blending, and multiplication refers to pixel-by-pixel modulation of rendering results. This set of operations is repeated for each mesh triangle of the object.

1.2.1.3 Examples

The following example shows the definition of the **Shape** node that uses **LFM_Appearance** field. In the example, the light map list has two elements but its first element has the view map list pointer set to **NULL**, which means that the surface map list in this element will be used as a regular texture map. In practice, we need at least 3 pairs of surface/view map lists for correct light field mapping.

```
Shape {
        geometry IndexedFaceSet {
                coord Coordinate { point [ v1, v2, ..., vN ] }
                coordIndex [ t1, t2, ..., tM ]
        }

        appearance LFM_Appearance {
                tileList [ ImageTexture { url "surfaceMapTile0.jpg" }
                           ImageTexture { url "surfaceMapTile1.jpg" }
                           ImageTexture { url "viewMapTile1.jpg" }
                ]
                lightMapList [
                        LFM_LightMap {
                                surfaceMapList LFM_SurfaceMapList {
                                        triangleIndex [
                                                0 1 2 3 .. ]
                                        tileIndex [
                                                0 0 0 1 .. ]
                                        viewMapIndex [
                                                -1 -1 -1 -1 .. ]
                                        triangleCoord TextureCoordinate { point [
                                0.00390625  0.00390625,  0.00390625  0.12890625,  0.12890625
                                0.12890625,  0.29296875  0.00390625,  0.29296875  0.12890625,
                                0.41796875  0.12890625,  0.58203125  0.00390625,  0.58203125
                                0.12890625,  0.70703125  0.12890625,  0.00390625  0.13671875,
                                0.00390625 0.26171875, 0.12890625 0.26171875,
                                        .
```

```
                                .
                                .
                                ] }
                        }
                        viewMapList NULL
                        scaleRGB 1 1 1
                        biasRGB 0 0 0

                }

                LFM_LightMap {
                        surfaceMapList LFM_SurfaceMapList {
                                triangleIndex [
                                        0 1 2 3 .. ]
                                tileIndex [
                                        0 0 0 1 .. ]
                                viewMapIndex [
                                        0 3 6 9 .. ]
                                triangleCoord TextureCoordinate { point [
                        0.01562500  0.00390625,  0.14062500  0.00390625,  0.14062500
                        0.12890625,  0.30468750  0.00390625,  0.42968750  0.00390625,
                        0.42968750  0.12890625,  0.59375000  0.00390625,  0.71875000
                        0.00390625,  0.71875000  0.12890625,  0.01562500  0.13671875,
                        0.14062500 0.13671875, 0.14062500 0.26171875,
                                        .
                                        .
                                        .
                                ] }
                        }
                        viewMapList LFM_ViewMapList {
                                vertexIndex [
                                        0 1 2 3 .. ]
                                tileIndex [
                                        2 2 2 2 .. ]
                                textureOrigin TextureCoordinate { point [
                                        0.00830078 0.00830078,
                                        0.02490234 0.00830078,
                                        0.04150391 0.00830078,
                                        0.05810547 0.00830078,
                                        .
                                        .
                                        .
                                ] }
                                textureSize TextureCoordinate { point [
                                        0.00781250 0.00781250,
                                        0.00781250 0.00781250,
                                        0.00781250 0.00781250,
                                        0.00781250 0.00781250,
                                        .
                                        .
                                        .
                                ] }
                        }

                        scaleRGB 2 2 2
                        biasRGB -1 -1 -1
                }
        ]
        blendList LFM_blendList {
                        lightMapIndex [ 1 2 . . ]
                        blendMode [ 1 1 . . ]
        }
        vertexFrame NULL

    }
}
```

1.2.1.4 Appendix A

The rule for computing the reference frame of a given triangle consists of two steps:

1. Compute the following vectors

$$\mathbf{e}_1^{\Delta_i} = \mathbf{v}_1 - \mathbf{v}_2,\ \mathbf{e}_2^{\Delta_i} = \mathbf{v}_2 - \mathbf{v}_3,\ \mathbf{e}_3^{\Delta_i} = \mathbf{e}_1^{\Delta_i} \times \mathbf{e}_2^{\Delta_i}$$

   where $\mathbf{v}_1$, $\mathbf{v}_2$, $\mathbf{v}_3$ are the vectors describing the positions of the 3 vertices of the given triangle.

2. Construct the change of basis matrix $\mathbf{R} = [\ \mathbf{x}\ \mathbf{y}\ \mathbf{z}\ ]$, where

$$\mathbf{x} = \frac{\mathbf{e}_1^{\Delta_i}}{\left\|\mathbf{e}_1^{\Delta_i}\right\|},\ \mathbf{z} = \frac{\mathbf{e}_3^{\Delta_i}}{\left\|\mathbf{e}_3^{\Delta_i}\right\|},\ \mathbf{y} = \mathbf{x} \times \mathbf{z}.$$

The rule for computing the reference frame of a vertex consists of the following steps:

1. Compute the direction of the normal of the vertex to be the average of the directions of the ring of triangles for that vertex
   where R_j is the set of triangles contained in the ring around vertex v_j.

$$\mathbf{e}_3^{v_j} = \sum_{i \in R_j} \mathbf{e}_1^{\Delta_i} \times \mathbf{e}_2^{\Delta_i}$$

2. Pick the first edge of the first triangle in the ring as the direction of the x-axis of the vertex reference frame

$$\mathbf{e}_1^{v_j} = \mathbf{e}_1^{\Delta_1}$$

3. Construct the change of basis matrix $\mathbf{R} = [\ \mathbf{x}\ \mathbf{y}\ \mathbf{z}\ ]$, where

$$\mathbf{x}' = \frac{\mathbf{e}_1^{v_j}}{\left\|\mathbf{e}_1^{v_j}\right\|},\ \mathbf{z} = \frac{\mathbf{e}_3^{v_j}}{\left\|\mathbf{e}_3^{v_j}\right\|},\ \mathbf{y} = \mathbf{x}' \times \mathbf{z},\ \mathbf{x} = \mathbf{z} \times \mathbf{y}.$$

2 References

W-C. Chen, R. Grzeszczuk, J-Y. Bouguet. Light Field Mapping: Hardware-Accelerated Visualization of Surface Light Fields. Part of "Acquisition and Visualization of Surface Light Fields," *SIGGRAPH 2001 Course Notes for Course #46*. Available from http://www.intel.com/research/mrl/research/lfm