MPEG-4

Santanu Chaudhury
EE Dept., IIT Delhi
MPEG-4: Overview

An architecture and coding methods for representing rich multimedia content
Unified Technology for
MPEG-4 Principles

• **Audio-visual scenes** made of **audio-visual objects** composed together according to a **scene description**:
  – allows interaction with elements within the audio-visual scene,
  – coding scheme can differ for individual objects,
  – allows easy re-use of audio-visual content.
MPEG-4 Principles

• Scene Description provides:
  – the spatial/temporal relationship between the audiovisual objects (2D, 3D, mixed 2D and 3D scene description),
  – the behavior and interactivity of the audio-visual objects and scenes,
  – protocols to modify and animate the scene in time,
  – a binary encoding for the scene.
Audio-visual objects can be of different nature:

- audio (single or multi-channel) or video (arbitrary shape or rectangular),
- natural (natural audio or video) or synthetic (text & graphics, animated faces, synthetic music),
- 2D (Web like pages) or 3D (spatialized sound, 3D virtual world),
- streamed (video movie) or downloaded (audio jingle).
Example of an MPEG-4 Audiovisual Scene (1)

2D Audio-visual scene
Audio and Video + Scrolling Text and Still Images

2D Audio-visual scene
Audio and Video + Still Images
MPEG-4 Scene

An MPEG-4 scene

Graphic objects

Animated 2D mesh

Natural video object
Corresponding Scene Graph

- root
  - background
    - sprite
    - music
  - voice-over
  - baby
    - video
    - bubbles
    - sound
  - bream
  - graphics
    - soil
    - fish
    - seaweed
    - blue
    - red
“Traditional” Features

• Systems Decoder Model

• Management of elementary streams
  – synchronization,
  – buffer management,
  – packetization.

• Intellectual Property Right Management
MPEG-4 Specifics

- Coded Representation of Interactive Audiovisual Scene,
- Identification and Association of Elementary Streams,
- User interaction.
What is MPEG-4 Representation?

“A coded, streamable representation of audio-visual objects and their associated time-variant data along with a description of how they are combined.”
MPEG-4 Standard Structure

Delivery Aware, media unaware:
14496-6 DMIF
14496-1 FlexMux

Media and Delivery unaware:
14496-1 Systems, adaptation of the MPEG-2 traditional activities

Media Aware, delivery unaware:
14496-2/3 Audio and Visual
14496-1 Scene Description and Object Descriptor Protocol
Structure of the Standard

1. Systems
2. Visual
3. Audio
4. Conformance
5. Reference SW
6. DMIF
Transport layer

presentation
decoding
demux & buffer
transport interface
not in standard
MPEG-4 Systems Principle

- Scene Description Stream
- Object Descriptor Stream
- Visual Stream
- Visual Stream
- Visual Stream
- Audio Stream

Interactive Scene Description
Visual Object Coding
Tools and Algorithms

• Natural Textures, Images and Video
  – compression of images, video, textures, 2D and 3D mesh
  – extended manipulation facility
  – scalability
  – error resilience

• Synthetic Objects
  – parametric description of human face and body
  – static and dynamic mesh coding
  – texture coding for view dependent applications
Structure of Tools for Natural Video

- VLBV Core: bit rates between 5..64 kbits/s for low resolution, low frame rate (15 Hz)
  - real time multi-media application
- Extended: Super set at higher bit rates (upto 10 Mb/s) higher spatial and temporal resolution
  - multimedia broadcast
  - interactive retrieval
MPEG4: Visual Object Coding

- Visual Scene consists of Video Objects
- Hierarchical Visual Scene Description
  - Visual Object Sequence
  - Video Object
  - Video Object Layer
  - Group of Video Object Planes
  - Video object Planes
VO and VOL

- Video Object corresponds to a particular 2D object in the scene
- Video Object Layer
  - scalable: multi-layer
    - spatial
    - temporal
  - non-scalable: single layer
VOP

• Time sample of a video object
• GoV groups together video object planes
• VOP contains encoded video data
  – shape information
  – motion parameters
  – texture data
  – encoded using macro blocks
Video Object Extraction

Not part of the standard

VO and VOP
Some Profiles: Natural Video

- Simple: rectangular video frame
- Advanced real time: rectangular frame for real time streaming
- Core: Basic coding of arbitrary shaped video objects
- Simple Scalable: Scalable encoding of rectangular frame
- Core Scalable: Scalable coding of video objects
- Scalable Texture: Scalable Coding of still texture
Simple Profile

- I-VOP
- P-VOP
- Compression Efficiency
  - Four motion vectors per macro block
  - Unrestricted motion vectors
  - Intra prediction
- Transmission Efficiency tools
  - Video packets, data partitioning, etc.
Coding Scheme

• Basic scheme same as MPEG-1 with conditional replenishment

• Compression efficiency
  – Motion Compensation with smaller block size
  – Macro-block, optionally considered as four 8x8 blocks
    • Reduce overhead
  – Useful for reducing residual in case of complex motion or near boundary
Compression Efficiency

• Unrestricted Motion Vectors
  – Motion vector can point outside the boundary of reference frame
    • Match with extrapolated macro block
  – Useful when objects moving in and out of the frame

• Intra Prediction
  – AC coefficient prediction
Transmission Efficiency

• Video Packets: VOP consists of one or more Video packets
  – A Video packet is analogous to a slice consisting of resynchronisation marker, header field and coded macro blocks
  – No error propagates beyond boundary of a slice so no predictive coding (motion vector, AC/DC coefficients) beyond packet boundary
  – Possible recovery of header of VOP due to repeated transmission
Transmission Efficiency

• Data partitioning: reorganisation of data to minimise transmission error
  – first partition of a packet contains coding mode information with DC coefficients and/or motion vector
  – Second partition contains remaining data
H.264

• Enhanced video compression for applications in
  – Broadcast, Storage, Conversational Services, Multimedia Streaming, Multimedia Messaging
• Enhancement on simple profiles of MPEG4 and referred as MPEG4 Part 10
• Coded can represent either an entire frame or field
Visual Coding

• Flexible Slice size
• Flexible macro-block ordering
• Slice Groups
• Each slice coded using different coding types: I,P,B
• Additional types:
  – SP slice and SI slice

Region of Interest

Error Concealment
Coding Structure
Intra-frame Predictive Coding

- Use of 4x4 and 16x16 blocks
- Intra-prediction in spatial domain
Inter-frame

• A range of block sizes (16x16 to 4x4)
  – Partitioning macro-block into motion compensated sub-blocks of varying size is known as tree structured motion compensation
  – Encoder selects the best partition size
• Motion vectors at sub-pixel resolution using interpolation of nearby pixels
• Predictive coding of motion vectors
Multi-frame Prediction

- Frames in multi-picture buffer for both encoder and decoder
  - Multiple past or future frames
  - Different frames for different macro blocks
- P_Skip – predicted directly from indexed frame
  - No residual error or motion vector coded
Multi-frame Prediction

4 Prior Decoded Pictures as Reference

Current Picture
Transform, Scaling, Quantization

- Transformation applied to 4x4 blocks
- A separable integer transform with similar properties as DCT used

\[ H = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \]
Why smaller transform?

• Improved prediction generates residuals with less spatial correlation
• Less computation and smaller processing wavelength
  – Forward transformation involves addition and shift only
  – Inverse with 16-bit integer operations only
• Quantisation parameters absorb scaling factor of DCT
De-blocking filter

• Blocking artifact
  – Block edges are reconstructed with more error

• If a relatively large absolute difference between samples near a block edge is measured, it is likely blocking artifact and should be reduced

1. $|p_0 - q_0| < \alpha(QP)$
2. $|p_1 - p_0| < \beta(QP)$
3. $|q_1 - q_0| < \beta(QP)$
De-blocking filter

• Applied after the inverse transform
  – In the encoder – before reconstructing and storing the macro-block for future predictions
  – In the decoder – before reconstructing and displaying the macro-block

• Averaging filter of different support
Object based Video Coding

- Object of any shape
- Shape, size and position can vary from frame to frame
- Composed of YUV and an alpha plane
- Alpha component defines objects shape from frame to frame basis
Encoding Scheme

• Binary shape encoder for encoding sequence of alpha maps
• A motion compensated discrete cosine transform algorithm for the interior texture coding.
• Possible to represent a textured object with gray-valued alpha map
  – Representing transparency information
Shape Coding Tools

- Coding of the Object support
- Binary Shape Information
  - representation as bit map
  - bounding box of the shape of the VOP
- Gray Scale Shape Information
  - gray value for degree of transparency representation
  - corresponds to the notion of alpha-plane
Binary Shape Coding

• Bounding Rectangle of VOP containing minimum number of blocks of size 16x16 with non-transparent pixels
• Partitioned into binary alpha block (BAB) of size 16x16
• Context based Arithmetic Coding and Motion Compensation
Alpha – plane Coding
• Each binary macro-block is BAB
• If all the pixels in a BAB are transparent/opaque special symbol to indicate
• In inter-frame coding modes, it is possible to use motion compensation where a BAB may be coded with reference to a suitable prediction BAB from the previously coded frame.
• In the simplest case, it may be chosen to code the BAB by motion compensation only, i.e., there is no
BAB Encoding

• BAB macro-block is collocated with YUV macro block
• BAB macro block can have different coding mode
  – Intra, inter, not-coded
• Context Adaptive Encoding is done
Basics of CAE

- Values of pixels in the neighbourhood of \( j \) will dictate the probability of value of the pixel at \( j \)
  - Causal constraint based on local neighbourhoods called templates
  - Used in JBIG also

- Patterns of pixels in a given template represented by a 10 bit number
  - Pixel \( j \) (?) can have 1024 PDF’s
CAE

1. CAE encoding is performed pixel by pixel. At each pixel, a template is formed.

2. From the template, a context number extracted, and this number is used to access a probability table.

3. The accessed probability (the probability that the pixel zero) and the value of the pixel (1 or 0) are used to drive entropy arithmetic encoder.
The current BAB and the motion-compensated BAB with the intra and inter templates overlain on the pixels.
CAE

- Fixed Probability Coder for MPEG-4
  - For each context a histogram generator counts the number of times the associated pixel to be coded is zero or one to a typical selection of moving shape
  - No continuous adaptation of the probability table
- For each BAB only one arithmetic code is generated
- For border pixels, contexts can span over pixels in adjacent blocks to maximize coding efficiency
Shape Encoder

Diagram:
- Previous Shape
- Motion Est.
- Motion Comp.
- Mode Decision
- CAE Coding
- Homo. Block
- Reconstructed Shape
- Entropy Coding
Mode Decision

For Intra VOP

- homogeneous?
  - yes: opaque
  - no: intraCAE

For Inter VOP

- perfect match with MVD=0?
  - yes: noUpdate w/o MV
  - no: homogeneous?
    - yes: opaque
      - yes: perfect match?
        - yes: intra shorter than inter?
          - yes: intra CAE
          - no: inter CAE
        - no: no update w/ MV
      - no: transparent
    - no: motion estimation
      - yes: perfect match?
        - yes: intra shorter than inter?
          - yes: intra CAE
          - no: inter CAE
        - no: no update w/ MV
Arithmetic Coding

- It has been shown that Huffman encoding will generate a code whose rate is within $p_{\text{max}} + 0.086$ of the entropy ($p_{\text{max}}$ is the probability of the most frequent symbol).
- When the size of the alphabet is small or the probabilities are skewed $p_{\text{max}}$ can be quite large.
- Huffman codes become inefficient under these conditions.
- The performance of Huffman codes can be improved by grouping blocks of symbols together.
- This can cause problems in terms of the memory requirements and decoding.
Problems in Huffman Coding

Example

- \( A=\{a1,a2,a3\} \) \( p(a1)=0.95, p(a2)=0.02, p(a3)=0.03 \)
- \( H=0.335 \) bits/symbol

Considering Strings of two alphabets

<table>
<thead>
<tr>
<th>Letter</th>
<th>Probability</th>
<th>Codeword</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1a1</td>
<td>0.9025</td>
<td>0</td>
</tr>
<tr>
<td>a1a2</td>
<td>0.0190</td>
<td>111</td>
</tr>
<tr>
<td>a1a3</td>
<td>0.0285</td>
<td>100</td>
</tr>
<tr>
<td>a2a1</td>
<td>0.0190</td>
<td>1101</td>
</tr>
<tr>
<td>a2a2</td>
<td>0.0004</td>
<td>110011</td>
</tr>
<tr>
<td>a2a3</td>
<td>0.0006</td>
<td>110001</td>
</tr>
<tr>
<td>a3a1</td>
<td>0.0285</td>
<td>101</td>
</tr>
<tr>
<td>a3a2</td>
<td>0.0006</td>
<td>110010</td>
</tr>
<tr>
<td>a3a3</td>
<td>0.0009</td>
<td>110000</td>
</tr>
</tbody>
</table>

\( l=1.05 \) bits/symbol

Better Coding efficiency by considering longer strings with additional memory and delay

\( l=0.611 \) bits/symbol
Arithmetic Coding

- Map a sequence into an interval. The sequence can then be encoded by transmitting a tag that represents the interval.

\[(a_1, a_2, \ldots, a_N)\]

- Key advantage: no need to generate codewords for all sequences of length \(N\).
Arithmetic Coding

- Encoding is constructing and conveying an interval whose length is the product of the probabilities of the input symbols so far encoded.
- Encoding is done recursively. If the sequence \( \{a_1, \ldots, a_{n-1}\} \) has been encoded into the interval \([l(n-1), u(n-1))\), the sequence \( \{a_1, \ldots, a_n\} \) is encoded into an interval \([l(n), u(n))\) obtained by
  - subdividing the previous interval into subintervals of lengths proportional to the probabilities of the symbols.
  - choosing the subinterval corresponding to the symbol \(a_n\).
Example

<table>
<thead>
<tr>
<th>Source Symbol</th>
<th>Probability</th>
<th>Initial Subinterval</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$</td>
<td>0.2</td>
<td>[0.0, 0.2)</td>
</tr>
<tr>
<td>$a_2$</td>
<td>0.2</td>
<td>[0.2, 0.4)</td>
</tr>
<tr>
<td>$a_3$</td>
<td>0.4</td>
<td>[0.4, 0.8)</td>
</tr>
<tr>
<td>$a_4$</td>
<td>0.2</td>
<td>[0.8, 1.0)</td>
</tr>
</tbody>
</table>

Encoding sequence

1. $a_4$ 0.2
2. $a_4$ 0.08
3. $a_4$ 0.072
4. $a_4$ 0.0688
5. $a_4$ 0.0624

0. $a_1$
1. $a_1$ 0.04
2. $a_1$ 0.056
3. $a_1$ 0.0624
4. $a_1$ 0.0624

$0 \rightarrow a_1 \rightarrow a_2 \rightarrow a_3 \rightarrow a_4$
Coding Scheme

- The equations describing encoding of symbol $x_n$:

$$l^{(n)} = l^{(n-1)} + (u^{(n-1)} - l^{(n-1)}) F_X(x_n - 1)$$

$$u^{(n)} = l^{(n-1)} + (u^{(n-1)} - l^{(n-1)}) F_X(x_n)$$

$$F_X(i) = \sum_{k=1}^{i} P(X = k)$$

- What tag to use? Popular choices are:
  - Middle of interval
  - Best: tag requiring smallest number of bits

- If middle of the interval is used as the tag, a binary representation of the tag truncated to

$$l(\underline{x}) = \lceil -\log(p(\underline{x})) \rceil + 1$$

bits is uniquely decodable.
Example

- Encode the sequence “bab”.
  For the first symbol, $p(a)=2/3$ and $p(b)=1/3$ => $F_x(0)=0$, $F_x(1)=2/3$, $F_x(2)=1$.
  For the second symbol, $p(a)=1/2$ and $p(b)=1/2$ => $F_x(0)=0$, $F_x(1)=1/2$, $F_x(2)=1$.
  For the third symbol, $p(a)=2/5$ and $p(b)=3/5$ => $F_x(0)=0$, $F_x(1)=2/5$, $F_x(2)=1$.
- Set $l(0) = 0$ and $u(0) = 1$.
- Input “b”: $l(1) = 0 + (1)(0.66) = 0.66$ and $u(1) = (1)(1) = 1$
- Input “a”: $l(2) = 0.66+(0.33)(0) = 0.66$ and $u(2) = 0.66+(0.33)(0.50) = 0.83$. 
Example (contd.)

- Input “b”: \( l(3) = 0.66 + (0.16)(0.40) = 0.73 \) and \( u(3) = 0.66 + (0.16)(1) = 0.83 \).
- Since \([0.110000..., 0.110100...] \subset [0.73, 0.83]\), send 1100, which is the shortest uniquely decodable code.
- Unique decodability implies that whatever bits the decoder adds to the code, the resulting real number still lies within the current interval.
- For example, if we only send 110 and the decoder adds 1 bits to the right side of the code, the resulting real number will definitely be larger than 0.83. However, sending 1100 will leave no ambiguity. In fact, you can check that \(0.11001111 \in [0.73, 0.83]\).
Example - Decoder

- Decode the sequence 1100. Construct C = 0.11001000… (0.78125), where the right four bits are random bits. Assume we have a copy of the model used at the encoder.
- Set l(0)=0 and u(0)=1.
- Subdivide [0,1) to [0,0.66) and [0.66,1).
- Since C=0.78125 ∈ [0.66,1), select [0.66,1) and emit "b".
- Subdivide [0.66,1) to [0.66, 0.83) and [0.83,1).
- Since C ∈ [0.66, 0.83), emit "a".
- Subdivide [0.66, 0.83) to [0.66, 0.73) and [0.73,0.83).
- Since C ∈ [0.73,0.83), emit "b". End.
Gray Scale Shape Coding

- Encoded using a block based motion compensated DCT
- Allowing lossy coding
Texture Coding

• For a given macroblock, the binary shape component is the first component reconstructed.

• Texture content of the macroblock’s bitstream depends to a great extent on the reconstructed shape information.
  – For example, a video object may contain one or more macroblocks that are totally transparent for which no texture information is included in the bitstream.
Motion Estimation and Compensation

- Block based coding adapted for VOP
- I-VOP, P-VOP, B-VOP
- Motion estimation is performed only for macro-blocks in the bounding box of VOP
- For macro-blocks partially belonging to VOP modified block matching technique is used
Coding Scheme

- I-VOP: texture information resides directly in the luminance and chrominance components
- P-VOP and B-VOP texture information represents the residual error.
- Steps: DCT, Quantisation, Coefficient prediction, Coefficient scan, VLC
Boundary Macro Blocks

- Padding for making blocks rectangular
- When texture data is residual error, blocks are padded with zero values
- For I-VOP Low-pass extrapolation
  - Use mean of the pixels belonging to the VOP
  - Use average of the neighbours belonging to the VOP
MPEG4 Video Coder
Sprite Coding

• Static Sprite
• Sprite consists of those regions of a VO that present in a scene throughout video segment
  – e.g., Background Sprite
• Shape and texture component encoded as in I-VOP, may be followed by a set of S-VOP
• Can provide high Compression Efficiency
Sprite Generation

- Background motion represented by global motion model
- Background motion caused by camera motion such as panning, tilting, zooming
- Modeled in terms of parametric geometric model

\[ x' = \frac{m_0x + m_1y + m_2}{m_6x + m_7y + 1}, \quad y' = \frac{m_3x + m_4y + m_5}{m_6x + m_7y + 1} \]

\((x,y)\) and \((x',y')\) are a pair of coordinates whose positions are in correspondence between two frames
Global Motion

- Camera Motion leads to image warps
- Examples of parametric warps:
  - translation
  - rotation
  - affine
  - perspective
  - aspect
  - cylindrical
2D coordinate transformations

- translation: \( x' = x + t \) \( x = (x, y) \)
- rotation: \( x' = R x + t \)
- similarity: \( x' = s R x + t \)
- affine: \( x' = A x + t \)
- perspective: \( \underline{x}' \cong H \underline{x} \) \( \underline{x} = (x, y, 1) \)
  (\( \underline{x} \) is a homogeneous coordinate)

- These all form a nested group (closed w/ inv.)
Image Warping

• Given a coordinate transform $x' = h(x)$ and a source image $f(x)$, how do we compute a transformed image $g(x') = f(h(x))$?
Warping

• Send each pixel $f(x)$ to its corresponding location $x' = h(x)$ in $g(x')$
• What if pixel lands “between” two pixels?
Forward Warping

- Send each pixel \( f(x) \) to its corresponding location \( x' = h(x) \) in \( g(x') \)
- What if pixel lands “between” two pixels?
- Answer: add “contribution” to several pixels, normalize later (splatting)
Sprite Generation

- Image registration: estimation of motion parameters by optimisation
- Find parameters which minimises sum of squared error over background region
- Warp with respect to reference coordinate frame and blend
- Mosaicing
Image registration

• How do we determine alignment between images?

  – One possible answer: *block matching* (correlation), i.e., find minimum squared error

  \[ E(u,v) = \sum_{(x,y)} \left[ I_1(x+u, y+v) - I_0(x, y) \right]^2 \]
Full-view Panorama
Sprite Generation

- Convert masked images into a background sprite for content-based coding
Sprite Decoding

• First VOP is the entire sprite
  – Decoded and put in sprite buffer

• Next, S-VOP contains warp parameters that are used to move and warp the contents of sprite-buffer in order to produce the desired background display
Example of Sprite Coding
Synthetic Visual Tools in MPEG-4
Motivation

• A new type of data is appearing in multimedia applications: Synthetic
• Both synthetic and natural data can co-exist in today’s applications
• This new data needs to be compressed and streamed in most applications
• New technologies are needed for:
  – Compression and streaming of synthetic data
Some examples of applications

• 3D video
• Augmented reality
• Telepresence
• Scientific visualization
• Virtual reality
• ...
Synthetic Visual Tools in MPEG-4

• Version 1
  – Face animation
  – 2D dynamic mesh
  – Scalable coding of synthetic texture
  – View dependent scalable coding of texture

• Version 2
  – Body animation
  – 3D model compression
Synthetic visual information

MESH DATA
- Vertices coordinates
- Topology
- Normals
- Colors
- Texture coordinates
Synthetic visual information

TEXTURE DATA

- Still or moving pictures
Synthetic visual information

TEXTURE MAPPING & RENDERING

• Viewing conditions
  – View Point
  – Aim Point

Image Based Rendering
Face Animation

MPEG-4 enables integration of face animation with multimedia communications and presentations and allows face animation over low bit rate communication channels, for point to point as well as multi-point connections with low-delay.
Example of applications

The integration of face animation and text to speech synthesizer is of special interest. MPEG-4 defines an application program interface for TTS synthesizer. Using this interface, the synthesizer can be used to provide phonemes and related timing information to the face model. The phonemes are converted into corresponding mouth shapes enabling simple talking head applications. Adding facial expressions to the talking head is achieved using bookmarks in the text. This integration allows for animated talking heads driven just by one text stream at a data rate of less than 200 bits/s.
Face animation

• **Face**: an object ready for rendering and animation
  
  – A realistic representation of a “human” face
  
  – Capable of animation by a reasonable set of parameters driven by speech, facial expressions, or others
Face animation

- Shape, texture and expressions:
  - specified parameters in the incoming bitstream
  - remote as well as local control of these parameters
Neutral face object

- Gaze along the Z axis
- All face muscles relaxed
- Eyelids tangential to the iris
- Pupil one-third of full eye
- Lips: in contact; horizontal
- Mouth: closed; upper and lower teeth touching
- Tongue: flat; tip touching front teeth
Neutral Face
Face animation

• Three sets of parameters used to describe a face and its animation characteristics:
  – *Facial Definition Parameters (FDPs)*
  – *Facial Animation Parameters (FAPs)*
  – *Facial Interpolation Table (FIT)*
Descriptors

• Facial Animation Parameters
  – Allows animation of key feature points in the model at the receiver

• Facial Definition Parameters
  – To configure or adapt a 3D facial model

• FAP Interpolation Table
  – Interpolation rules for the FAP that have to be interpolated at the decoder
Block Diagram
Structure
Facial Definition Parameters - FDPs

- Defines a specific face via
  - 3D feature points
  - 3D mesh/scene graph
  - Face Texture
  - Face Animation Table (FAT)
  - Deformation rules
Face Feature Points

- Normalize animation parameters
- Find feature correspondence in different faces
- Roughly define shape
3D mesh and feature points
## Expression FAP

<table>
<thead>
<tr>
<th>Expression Name</th>
<th>Textual Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joy</td>
<td>The eyebrows are relaxed. The mouth is open and the mouth corners pulled back toward the ears.</td>
</tr>
<tr>
<td>Sadness</td>
<td>The inner eyebrows are bent upward. The eyes are slightly closed.</td>
</tr>
<tr>
<td>Anger</td>
<td>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.</td>
</tr>
<tr>
<td>Fear</td>
<td>The eyebrows are raised and pulled together. The inner eyebrows are bent upward. The eyes are tense and alert.</td>
</tr>
<tr>
<td>Disgust</td>
<td>The eyebrows and eyelids are relaxed. The upper lip is raised and curled, often asymmetrically.</td>
</tr>
<tr>
<td>Surprise</td>
<td>The eyebrows are raised. The upper eyelids are wide open, the lower relaxed. The jaw is opened.</td>
</tr>
</tbody>
</table>
Texture
Talking Head

• Use of facial animation with TTS can reduce bandwidth requirement
• Communication channel – only for text and animation parameters for streaming
• Initial set-up or model required
Body animation
Body animation objectives

• For streaming human-like characters
• Supports various applications
  – from realistic simulation of human motions
  – to network games using simple human-like models.
Example of applications

- Virtual meeting, tele-presence, ...
- Virtual story teller, virtual actor, user interface, ...
- Games, avatars, ...
- Education
- Medicine
Body animation

- **Body**: an object ready for rendering and animation
  - A realistic representation of a “human” body
  - Capable of animation by a reasonable set of parameters
  - Extension of the notion of face object to FBA object
Body animation

• Shape, texture and gesture:
  – specified parameters in the incoming bitstream
  – remote as well as local control of these parameters
Body animation

• Two sets of parameters are used to define and to animate body objects
  
  – Body Definition Parameters (BDPs)
  – Body Animation Parameters (BAPs)

• Body Animation Table (BAT)
  
  – A downloadable function mapping from incoming BAPs to body surface geometry that provides a combination of BAPs for controlling body surface geometry deformation
FBA object

• **FBA object**: a collection of nodes in a scene graph which are animated by the FBA (Face and Body Animation) object bitstream

• FBA object is controlled by two separate bitstreams:
  
  – The first bitstream, contains instances of Body Definition Parameters (BDPs) in addition to Facial Definition Parameters (FDPs)

  – The second bitstream, FBA bitstream, contains Body Animation Parameters (BAPs) together with Facial Animation Parameters (FAPs).
Body animation

- The Body Animation Parameters (BAPs)
  - Produce similar body posture and animation on different body models
  - No need to initialize or calibrate the model
Body animation

- Body Definition Parameters (BDPs)
  - Transform the default body to a customized body
  - Body dimensions
  - Body surface geometry
  - Joint center locations
  - Textures
- Based on VRML 2.0 H-Anim models
  - Close collaboration
Default body

- A generic virtual human or human-like body with the default posture
- Can be rendered immediately
- Can receive BAPs immediately for animation
- If BDPs are received, they are used to transform the decoder’s generic body into a particular body determined by the parameter contents
- No assumption is made and no limitation is imposed on the range of defined mobilities for humanoid animation
Default posture

• 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 as 0
Default posture
BAPs

- BAPs comprise joint angles connecting different body parts.
- They include: toe, ankle, knee, hip, spine (C1-C7, T1-T12, L1-L5), shoulder, clavicle, elbow, wrist, and the hand fingers.
- The unit of rotations (BAPU) is defined as PI/10E-5 radians.
- The unit of translation BAPs is defined in millimeters.
BAPs

- 175 Body Animation Parameters
- 125 BAPs for body, 25 BAPs for each hand
- 19 groups
  - e.g. left_leg1, left_leg2, left_arm1, etc.
  - classified: visual effect on posture
Body animation - lower body
Body animation - hands
Body animation - upper body

- Head top
  - Cervical region:
    - Neck:
      - VC2-tilt, VC2-torsion
      - VC4-tilt, VC4-torsion
  - Thoracic region:
    - VT1-tilt, VT1-torsion
    - VT6-torsion
    - VT10-tilt, VT10-torsion
  - Lumbar region:
    - VL1-tilt, VL1-torsion
    - VL3-tilt, VL3-torsion
    - VL5-tilt

- Skeleton root
Body animation

• Five LODs for spine
  – 9 dof
  – 24 dof
  – 42 dof
  – 60 dof
  – 72 dof
BAP coding

• BAP coding is similar to FAP coding:
  – Quantization, and predictive coding using arithmetic coding
  – DCT coding on 16 frames BAPs
What is not standardized?

- The way to extract the parameters
  - Markers
  - Trackers
  - Image analysis and feature extraction
- The choice on which parameters to code and with which precision
  - Quantization
  - Rate control
2D Dynamic Mesh
Example of applications

• *Video Object Manipulation*
  – Augmented Reality
  – Object Transformation/Editing
  – Spatio-Temporal Interaction
Dynamic 2D mesh

• Specifically refers to *triangular Delaunay* meshes

• Tessellation of a 2D visual object plane into a connection of triangular patches

• No addition and deletion of nodes, i.e. no change in topology
Dynamic 2D Mesh

• Generation of the initial mesh
• Coding of initial node points
• Coding of the node motion vectors
Generation of the initial mesh

- Any technique can be used
- Not imposed by the standard
Coding of node motion vectors
2D dynamic mesh coding
Let’s see an example ...
2D dynamic mesh coding

- Text overlay
3D model compression
3D model compression

How to represent a 3D mesh with a minimum number of bits

- Single resolution
- Progressive resolution
- Progressive piece-wise
- Error resilient
- For compression of static models only
3D model coding

• Connectivity (Topology)
  – How 3D polygons are connected

• Geometry
  – 3D coordinates of nodes or vertices

• Properties (Photometry)
  – Colors
  – Normals
  – Texture
  – ...

3D model encoder

3D mesh model
- Geometry
- Connectivity
- Photometry

Geometry coder

Connectivity coder

Photometry coder

Entropy coding

Compressed 3D mesh model

MPEG System MUX

Bitstream
3D model decoder

Bitstream

MPEG System DEMU

Entropy decoding

Geometry decoder

Connectivity decoder

Photometry decoder

3D mesh model

Geometry

Connectivity

Photometry

Composition & Rendering
Compression of 3D Models

- Texture Coding and Compression
  - Static textures can be compressed by JPEG
  - Moving textures can be compressed by MPEG-1 or H.263
- Geometry Coding and Compression
  - Most often geometry is represented by a triangular 3D mesh.
  - Algorithms that quantize vertex positions and normals, and then apply Huffman or entropy encoding can achieve compression in the range of 15:1 to 65:1, depending on the nature of the model.
  - Mechanical models are easier to compress than anatomical.

Need for Geometry Compression:
- Each vertex represented by three floating pt. numbers.
- If each vertex shared by six polygons, and max number of vertices per model is $2^{20}$, then 76 bits/triangle needed.
- Several hundred Kilobytes for average model (just geometry).

\[
\left( \frac{1}{6} \right) \times \left( \frac{3 \text{vertices}}{\text{triangle}} \right) \times \left( \frac{32 \text{bits}}{\text{vertex}} \right) + \left( \frac{3 \text{vertexIDs}}{\text{triangle}} \right) \times \left( \frac{20 \text{bits}}{\text{vertexID}} \right) = \frac{76 \text{bits}}{\text{triangle}}
\]
Connectivity coding

- Topological surgery
Geometry and properties coding

- Quantization
- Predictive coding
- Entropy coding (arithmetic coding)
Simplification vs multiresolution

- Reduction of number of vertices and faces
- Progressive transmission = Inverse problem
- Forest split
View Dependent Texture Coding

• With respect to upstream viewer chosen viewpoint

• Generation of appropriate texture
  – Scaling down for distance
  – Orientation of the 3D surface
Forest split coding
Piece-wise progressive coding

• Example of progressive decoding
  – part of the model can be rendered before all the bits are received
  – Error resilience