Department of Computer Science
National Tsing Hua University

CS 5263: Wireless Multimedia Networking Technologies and Applications

Scalable Video Coding

Instructor: Cheng-Hsin Hsu

Acknowledgement: The instructor thanks Prof. Mohamed Hefeeda at Simon Fraser University for sharing his course materials

Motivation

- Receivers of video stream are heterogeneous
  - Connection bandwidth
  - Display resolution
  - Processing power
  - Battery level
- Dynamic conditions
  - Even for the same receiver
  - Internet bandwidth is changing
  - Wireless conditions and mobility
- → Need stream adaptation methods

Stream Adaptation

- **Transcoding**
  - Transform the encoded stream to different format/bitrate/resolution
  - Simple approach: decode then encode again with different parameters
  - There are more sophisticated transcoding schemes, e.g., work in the compressed domain
  - Disadvantages?
    - Computational cost

Stream Adaptation

- **Simulcasting (or Stream Switching)**
  - Encode a video stream multiple times
  - E.g., high, medium, low quality
  - Or high and low resolutions
  - Switch among streams during the session
  - Advantages: simple
  - Disadvantages?
    - Managing multiple versions of same video
    - Larger storage requirements
    - Switching streams is not easy: need to synchronize at I-frames

Stream Adaptation

- **Muti-Descritpion Coding (MDC)**
  - Encode each stream into multiple descriptions
  - Each description improves the received quality
  - Any subset of descriptions can be decoded
  - Advantages:
    - Very flexible
  - Disadvantages?
    - Coding inefficiency: the aggregate bit rate of MDCs is much higher than single-layer (nonscalable stream) at the same quality

Stream Adaptation

- **Scalable Video Coding (SVC)**
  - **Goal:** Each stream is encoded once, but can be decoded/adapted in many different ways
  - **Idea:** each stream has multiple **layers**, and subsets of layers can be decoded (some restrictions on layers)
- **Started early on**
  - H.262/MPEG-2, H.263, MPEG-4 Visual, …
  - But was not widely deployed:
    - Coding/decoding complexity and
    - Coding inefficiency, e.g., 2-5 dB gap is common for MPEG-4 FGS (fine-grained scalability) streams when compared against MPEG-4 streams

Standards Addressing the Open Issues

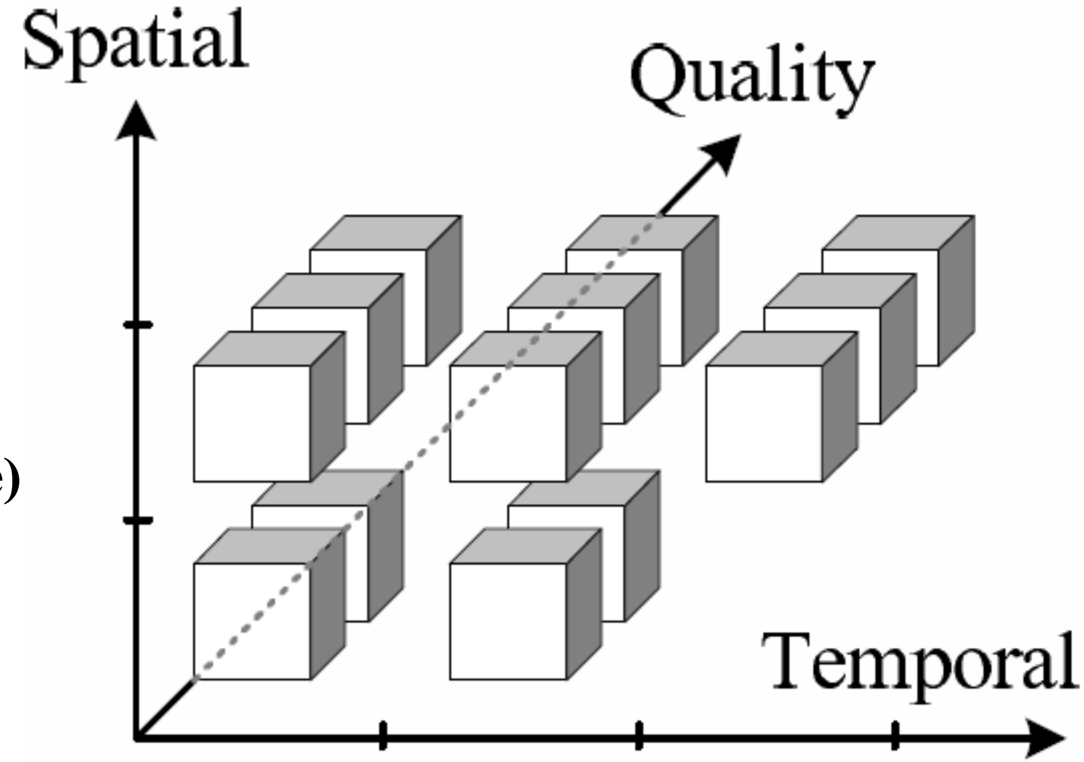
- H.264/SVC
  - Tries to avoid previous problems
  - Gaining momentum (some companies already used it)
- Our discussion today is mostly focused on H.264/SVC
  - May discuss SHVC (H.265 based) features next time
- Read: Schwarz et al., Overview of the Scalable Video Coding Extension of the H.264/AVC Standard, IEEE Trans. on Circuits and Systems for Video Technology, 17(9), 2007

H.264/SVC

- **SVC tries to achieve (scalability wish list)**
  - Similar coding efficiency to single-layer coding (10% bit rate increase at most)
  - Support for temporal, spatial, quality scalability
  - Backward compatibility of the base layer
  - Support for simple bitstream adaptations after encoding
  - Little increase in decoding complexity (arguably failed)
- **SVC has many potential applications**
  - Support heterogeneous receivers (wired and wireless)
  - Unequal error protection
  - Archiving in surveillance applications (store base quality)
  - ….

H.264/SVC: 3-D Scalability

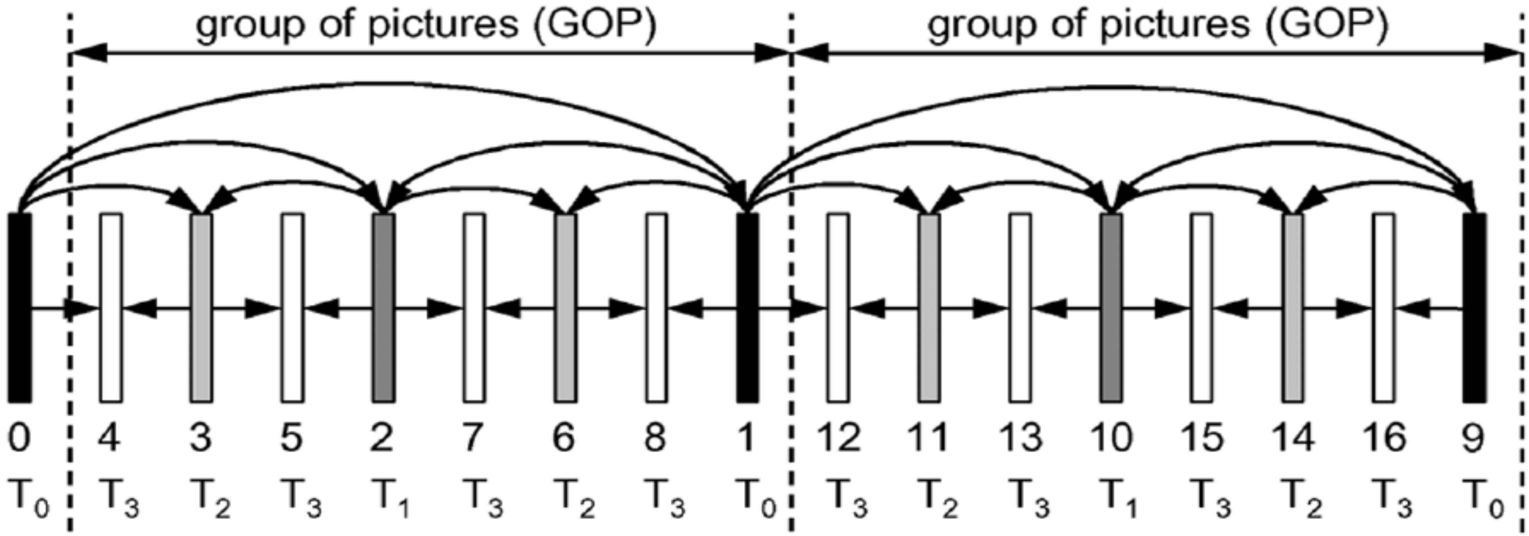
- **Temporal scalability**
  - Frame rate
- **Spatial scalability**
  - Resolution (picture size)
- **Quality scalability**
  - (Fidelity or SNR)
- **SVC ➔ Very flexible adaptation**



Temporal Scalability

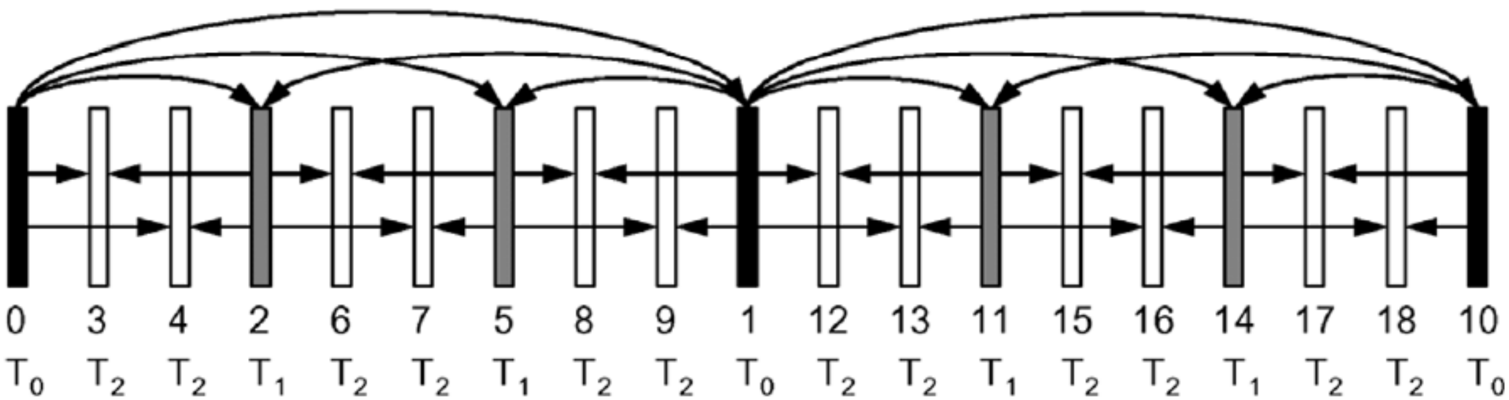
- Divide sequence into temporal layers
  - Restrict motion-compensated prediction
- → Hierarchical prediction structure
  - Already provided by H.264/AVC

Temporal Scalability: Example



- **4 temporal layers: T0, …, T3**
  - **T0 = 1 frames per GoP**
  - **T1 = T0 + 1 = 2 frames per GoP**
  - **T2 = T1 + 2 = 4 frames per GoP**
  - **Numbers below frames indicate decoding order**
  - **Arrows show prediction dependency**
  - **Dyadic (power of 2) temporal enhancement layers**

Temporal Scalability: Example 2



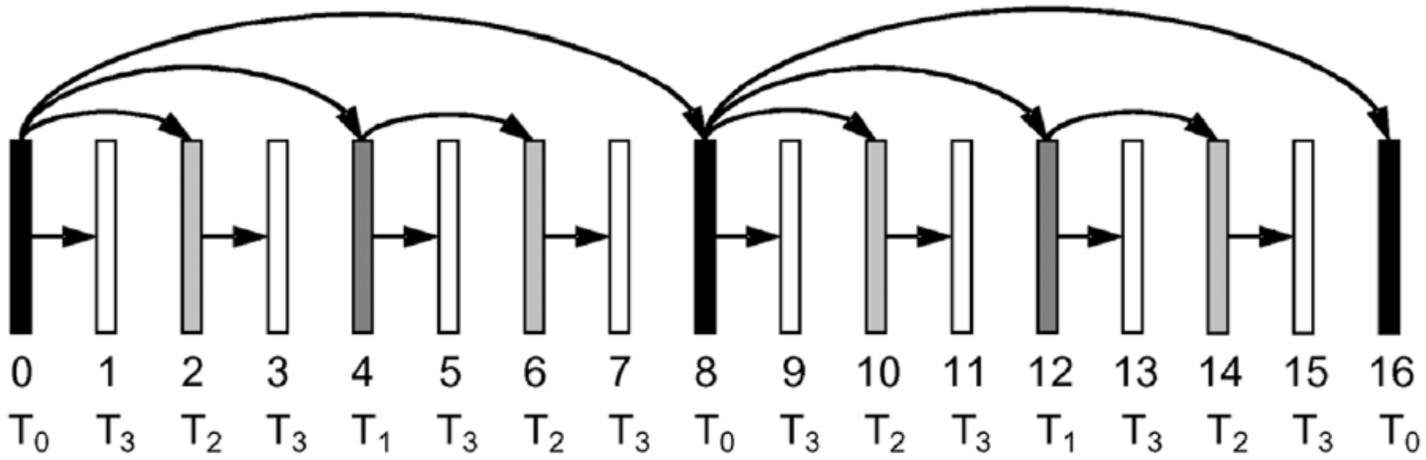
- **Non-dyadic structure is also possible**
  - T0 = 1/9 of full frame rate
  - T1 = 1/3 of full frame rate
  - T2 = full frame rate

Temporal Scalability

- **SVC also supports**
  - Changing the hierarchical prediction structure over time
  - Having the reference frame in the same temporal layer as the target frame
  - Having multiple reference frames (as in H.264/AVC)

Temporal Scalability: Delay

- **Hierarchical structure could increase decoding delay**
  - Some frames cannot be decoded until receiving future frames
  - Not desired in interactive multimedia applications (e.g., video conf)
- **SVC can limit predictions to preceding frames only**
  - Cost?
  - Decreased coding efficiency

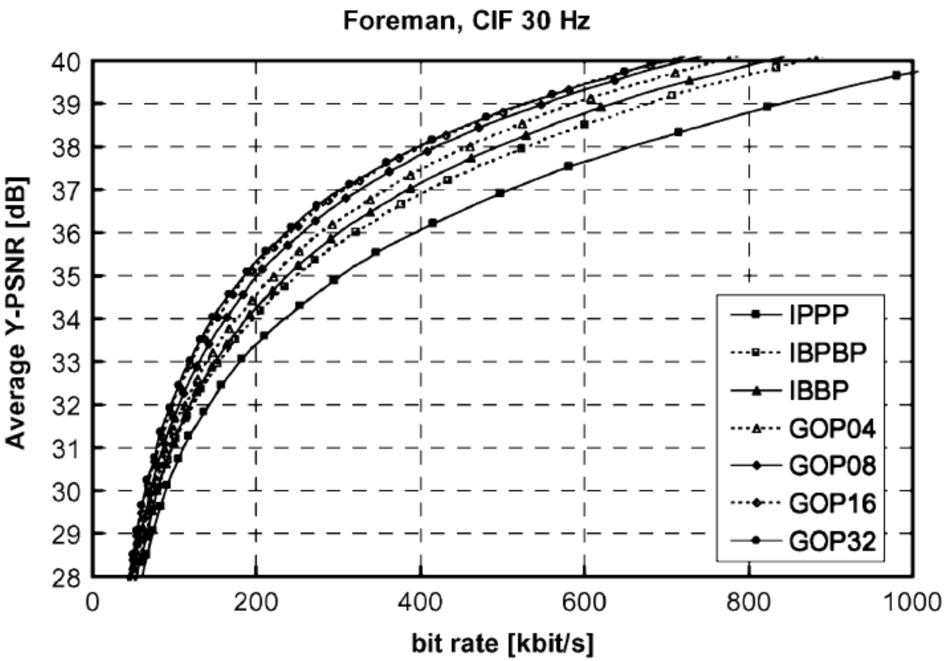


Temporal Scalability: Coding Efficiency

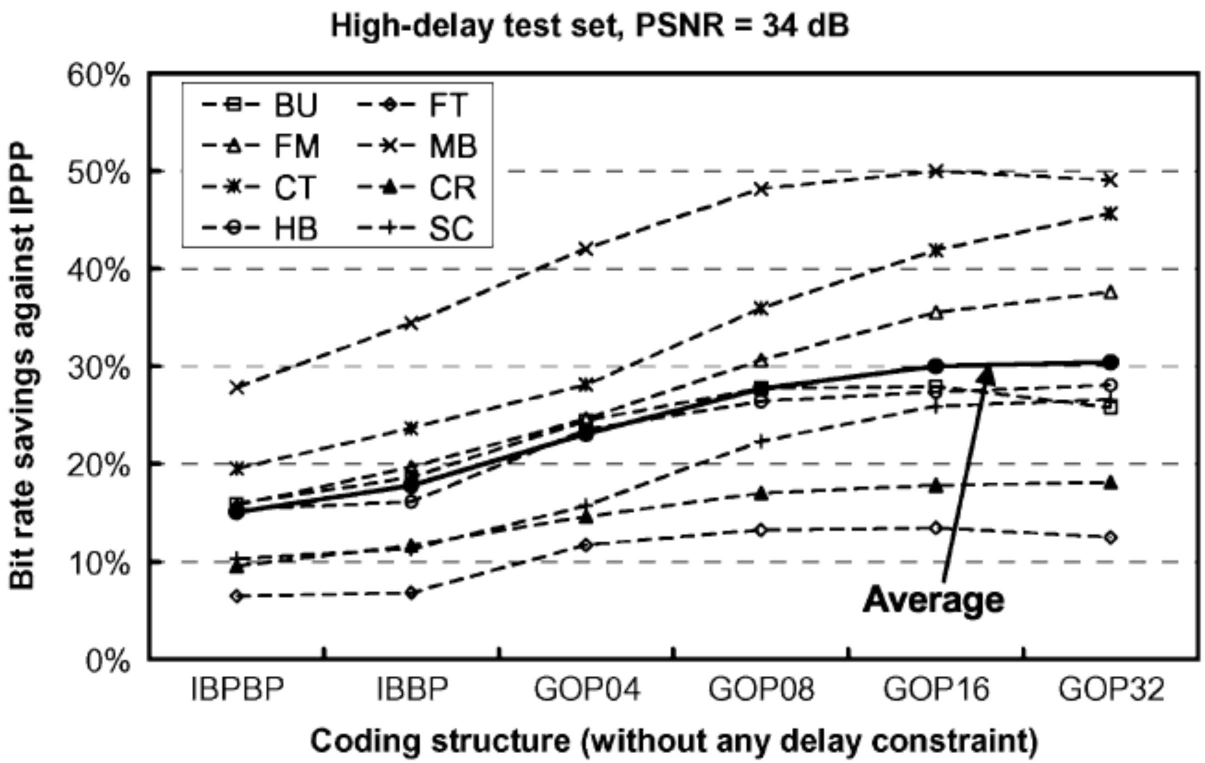
- Comparing dyadic hierarchical B-pictures (no decoding delay constraint)
  vs IPPP, IBPBP, and IBBP

- Hierarchical B-pictures achieve PSNR gain >= 1 dB compared to the widely used IBBP coding structure

- Gain is higher for large GoP sizes

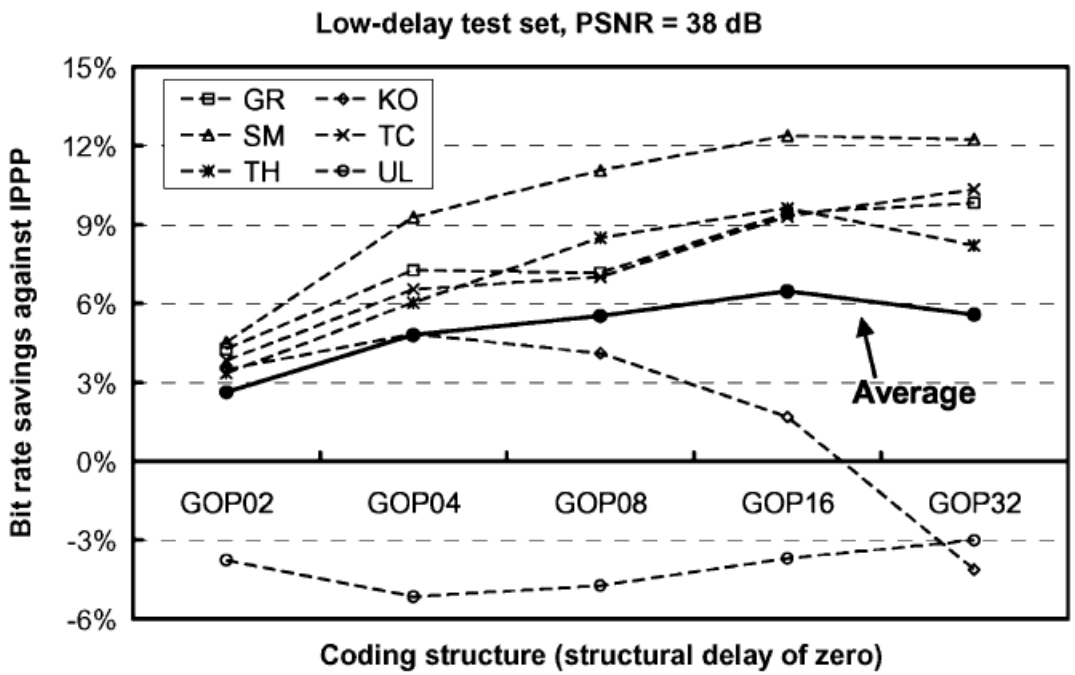


Temporal Scalability: Coding Efficiency



- Using high-delay test set (non-conversational sequences), CIF 30Hz, 34dB, compared to IPPP
- ➔ significant saving in bitrate

Temporal Scalability: Coding Efficiency



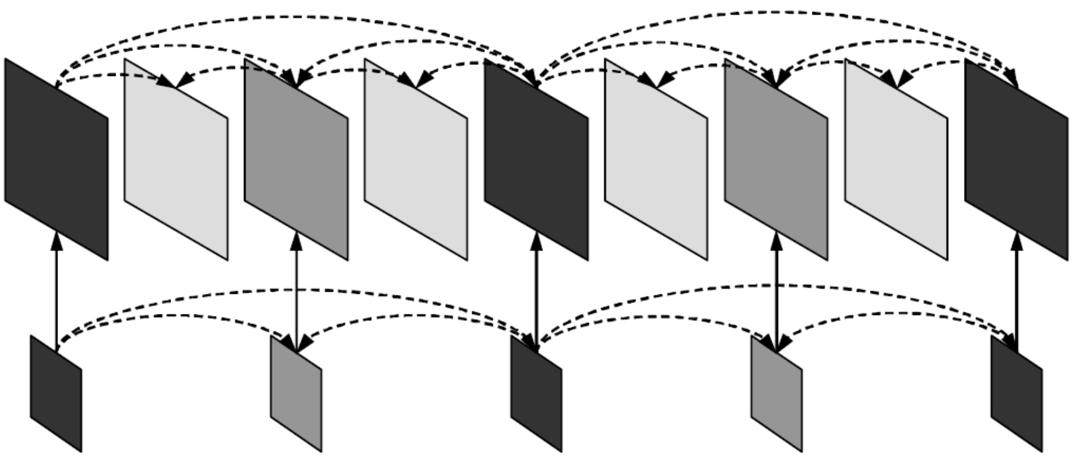
- **Using low-delay (delay = 0) test set (conversational sequences), 365x288, 25-30Hz, 38 dB vs IPPP**
  - **Still some gain but not as high as before**

Temporal Scalability: Summary

- Achieved using hierarchical temporal structures
- Typically no negative impact on coding efficiency
  - Significant improvement, especially when higher delays are tolerable
  - Minor losses in coding efficiency are possible when low delay is required

Spatial Scalability

- **Basic Idea:**
  - Multiple layers with different resolutions
  - Each layer is treated as if it were single-layer coding:
    - i.e., uses motion-compensated prediction and intra-prediction
  - All layers share the same encoding order ← for low complexity
  - Inter-layer prediction is also possible
- **Notice temporal and spatial scalabilities can co-exist**
  - Inter-layer prediction is only performed at access units



Spatial Scalability: Inter-Layer Prediction

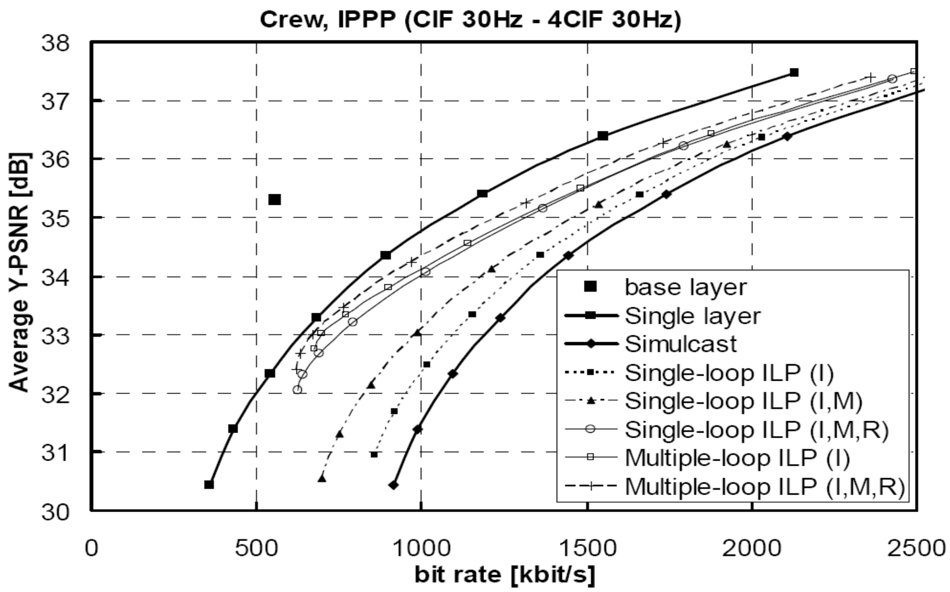
- **Inter-layer prediction**
  - Up-sample lower layer signal (reconstructed samples) and perform prediction ← early standards only support this
  - Perform temporal prediction inside higher-resolution layer (in the enhancement layer)
  - You can either use the first prediction and/or the second
    - Averaging in case of using both
- **Note: same-layer temporal prediction can provide better compression in case of low motion videos with detailed resolution**

Spatial Scalability: Improving Efficiency

- To improve the coding efficiency of inter-layer prediction, two coding tools were added
  - Prediction of macroblock modes and associated motion parameters
  - Prediction of the residual signal
- A new macroblock type is defined
  - Transmits residue signals
  - No intra-prediction mode nor motion parameters
  - If the corresponding macroblock in the reference layer is
    - Intra-coded→intra prediction: upsample reference layer
    - Inter-coded→motion prediction: motion vectors are scaled up

Spatial Scalability: Coding Efficiency

- **Single-loop vs. multiple-loop decoding**
  - Reconstructing inter-coded reference (lower) layers or not…
- **Coding tools: Intra-layer prediction (I), motion prediction (M), residual prediction (R)**
- **Take-Away**
  - I, M, R are beneficial
  - But multiple-loop leads to minor enhancements, while incurring high decoding overhead



Quality Scalability

- **Basic Idea:**
  - Multiple layers created with same resolution but different fidelity (picture quality)
  - Different qualities can be achieved by controlling quantization step
- **H.264/SVC quality scalability models**
  - Coarse-Grained Scalability (CGS)
    - Few layers
  - Medium-Grained Scalability (MGS)
    - More flexible

Quality Scalability: CGS

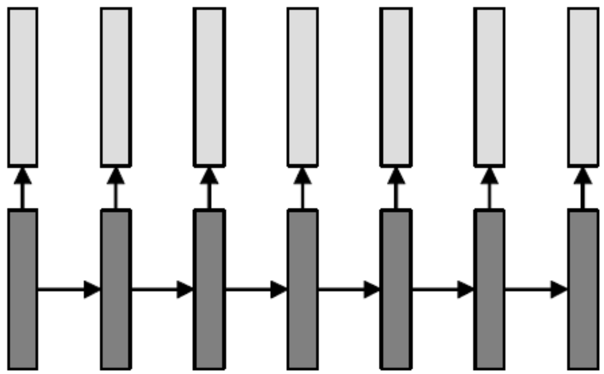
- Similar to spatial scalability, but with same resolution
- Use different quality parameters in different layers
- Supports a few (typically 3 to 6) different bit-rates/layers
- Too many layers → high overhead → low coding efficiency

Quality Scalability: MGS

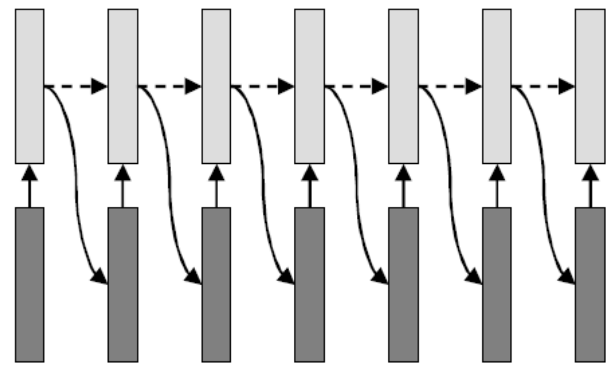
- Medium-Grained Scalability (MGS) improves:
  - Flexibility of the stream
    - Packet-level quality scalability
  - Error robustness
    - Controlling drift propagation
  - Coding efficiency
    - Use of more information for temporal prediction

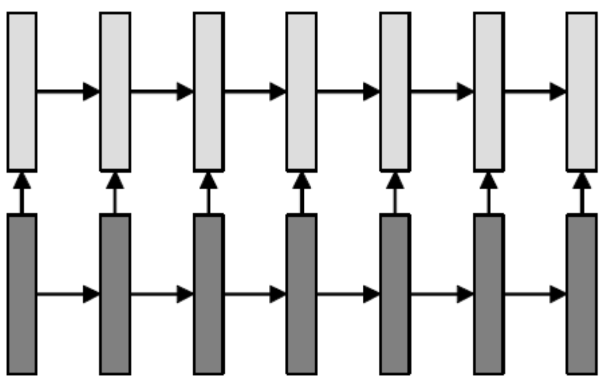
Quality Scalability: MGS Prediction Structure

MPEG-4 FGS

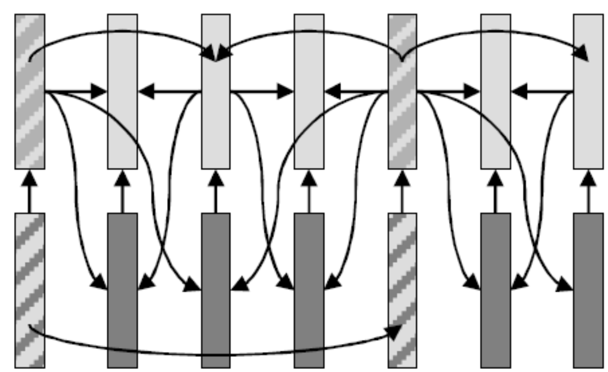


MPEG-2 Quality Scalable





MPEG-2 Spatial Scalable



H.264/SVC MGS

Quality Scalability: MGS Key Frames

- Video frames of coarsest temporal layer are called key frames
- Key frames only use base-layer frames for predictions ← robustness
- Non-key frames can only highest possible layers for prediction ← coding efficiency

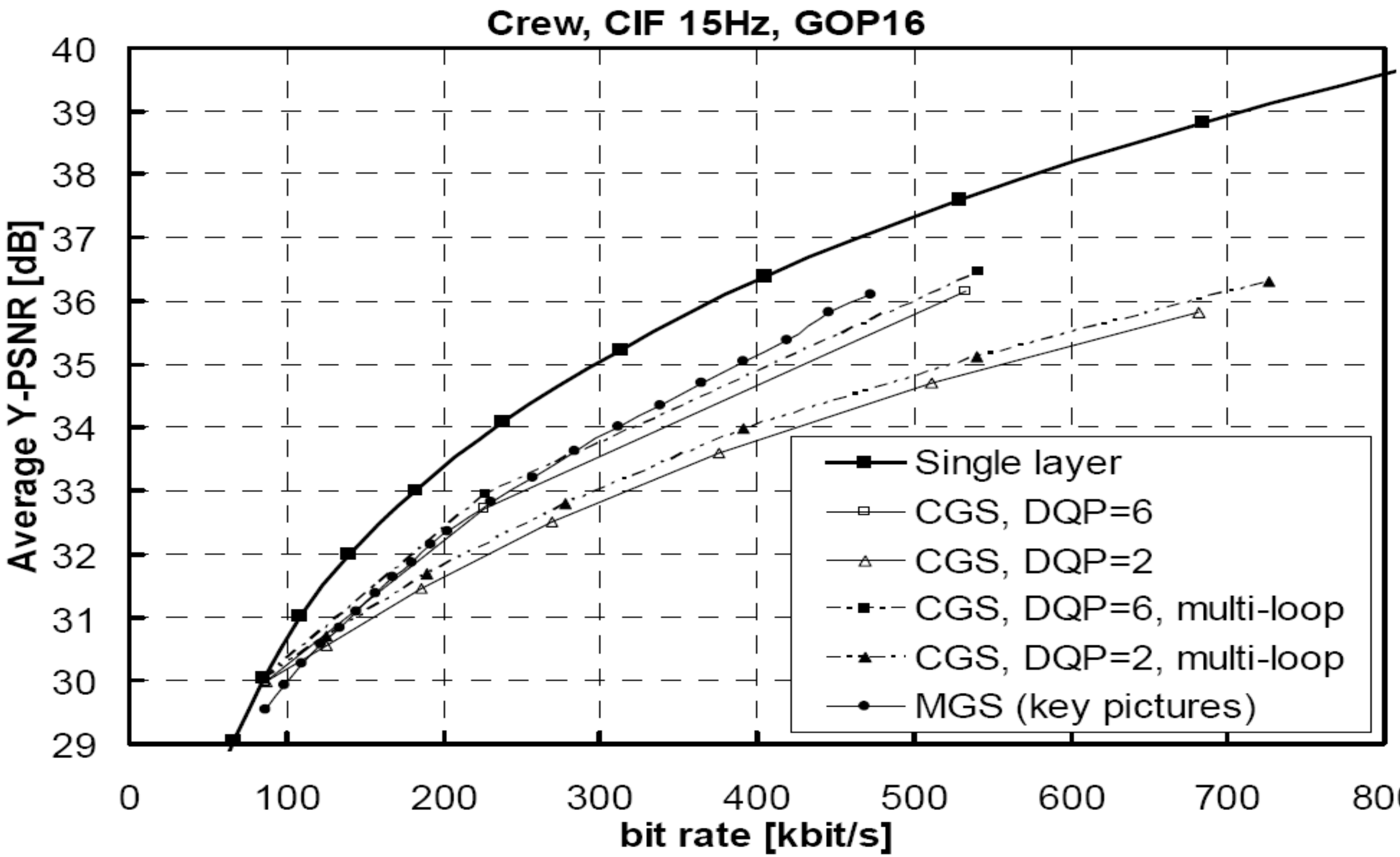
Quality Scalability: MGS

- **MGS: flexibility of the stream**
  - Enhancement layer transform coefficients can be distributed among several slices

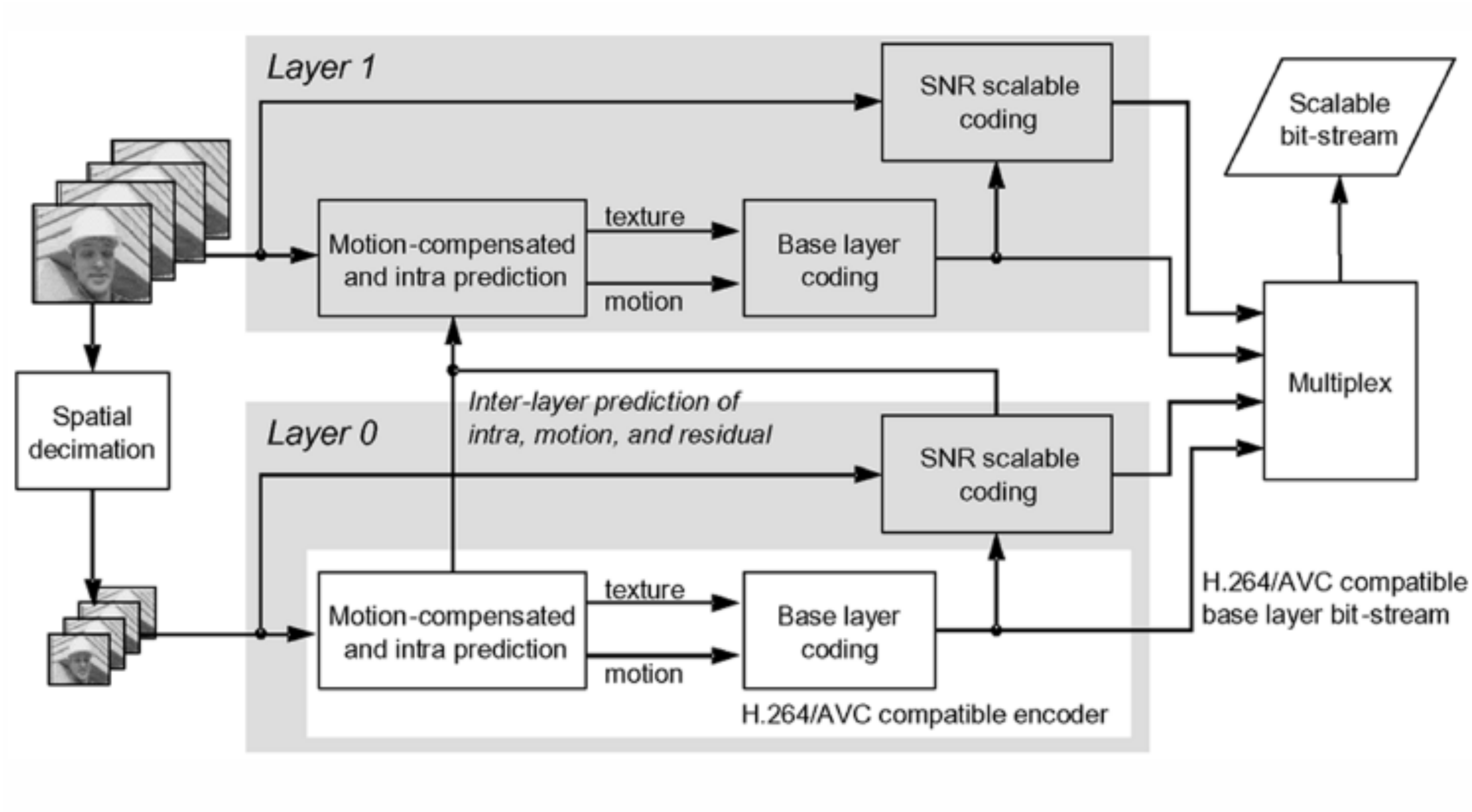
1	1	2	3
2	2	3	4
2	3	3	4
3	3	4	4

  - Packet-level quality scalability

Quality Scalability: MGS vs. CGS



SVC Encoder Structure



- Simple example for 2 spatial layers

Summary

- **Different models of scalability**
  - Simulcast, MDC, SVC
- **SVC**
  - Temporal
  - Spatial
  - Quality
- **H.264/SVC tried to improve coding efficiency while reducing complexity**
  - It achieves the former goal: gap between H.264/SVC and MPEG-4 is reported to be as low as 10%
  - It arguably fails the later goal: very few SVC chip designs are out there, mostly due to memory limitation

Compiling JSVM (and other ref. sw)

- **Create working folder:** mkdir JSVM; cd JSVM
- **CVS login:** cvs -d :pserver:jvtuser:jvt.Amd.2@garcon.ient.rwth-aachen.de:/cvs/jvt login
- **CVS checkout:** cvs -d :pserver:jvtuser@garcon.ient.rwth-aachen.de:/cvs/jvt checkout jsvm
- **Get into the build directory (using Linux as example)**: cd jsvm/JSVM/H264Extension/build/linux
- **Compile**: make
- The resulting binary files are under JSVM/jsvm/bin

JSVM Utilities

- ***H264AVCEncoderLibTestStatic***: reference encoder
- ***H264AVCDecoderLibTestStatic*** : reference decoder
- ***BitStreamExtractorStatic*** : extract a substream from the global scalable stream
- ***PSNRStatic*, *YUVCompareStatic*** : compare two yuv files for PSNR
- ***FixedQPEncoderStatic***: binary search algorithm for rate control (there is no rate control algorithms in JSVM)
- ***H264AVCVideoIoLibStatic***: library for read and write NAL units

Encoding a Two Layer CGS Stream (1/5)

- **Prepare the configure files**
  - **One main configuration file: main.cfg**
  - **One layer configuration file for each layer: layer0.cfg and layer1.cfg**
- **Download the YUV files**
  - wget http://nsl.cs.sfu.ca/video/library/YUV/4CIF/CREW_704x576_30_orig_01.yuv
  - wget http://nsl.cs.sfu.ca/video/library/YUV/CIF/CREW_352x288_30_orig_01.yuv

Encoding a Two Layer CGS Stream (2/5)

- **main.cfg**

```
# JSVM Main Configuration File

OutputFile              CS5263.264   # Bitstream file
FrameRate               30.0        # Maximum frame rate [Hz]
FramesToBeEncoded       150         # Number of frames (at input frame
rate)
GOPSize                 16          # GOP Size (at maximum frame rate)
BaseLayerMode           2           # Base layer mode (0,1: AVC
compatible,
                                    #                    2: AVC w
subseq SEI)
SearchMode              4           # Search mode (0:BlockSearch,
4:FastSearch)
SearchRange             32          # Search range (Full Pel)
NumLayers               2           # Number of layers
LayerCfg                layer0.cfg # Layer configuration file
LayerCfg                layer1.cfg # Layer configuration file
```

Encoding a Two Layer CGS Stream (3/5)

- **layer0.cfg**

```
# JSVM Layer Configuration File

InputFile            CREW_352x288_30_orig_01.yuv # Input  file
SourceWidth          352             # Input  frame width
SourceHeight         288             # Input  frame height
FrameRateIn          30              # Input  frame rate [Hz]
FrameRateOut         30              # Output frame rate [Hz]
```

Encoding a Two Layer CGS Stream (4/5)

- **Encode the video**
  - jsvm/bin/H264AVCEncoderLibTestStatic -pf main.cfg -lqp 0 30 -lqp 1 32
- **Decode at the full quality**
  - jsvm/bin/H264AVCDecoderLibTestStatic CS5263.264 full.yuv
- **Playout the reconstructed yuv file**
  - mplayer -demuxer rawvideo -rawvideo fps=10:w=704:h=576:format=i420 -loop 0 full.yuv

Encoding a Two Layer CGS Stream (5/5)

- **Extract a lower resolution stream**
  - jsvm/bin/BitStreamExtractorStatic CS5263.264 CS5263_LoFi.264 -l 0
- **Decode the video**
  - jsvm/bin/H264AVCDecoderLibTestStatic CS5263_LoFi.264 LoFi.yuv
- **Play the low resolution reconstructed video**
  - mplayer -demuxer rawvideo -rawvideo fps=10:w=352:h=288:format=i420 -loop 0 LoFi.yuv

Compute PSNR

- jsvm/bin/PSNRStatic 704 576 CREW_704x576_30_orig_01.yuv full.yuv

0	38,9117	43,1924	43,8809
1	34,3800	39,0224	39,2633
2	37,0061	42,5263	42,8077
3	36,7575	42,3130	42,4146
....			
147	33,8073	39,1530	39,1838
148	35,4138	39,7400	39,5863
149	34,3667	39,4859	39,0969
total	35,1929	40,6697	40,6936

Compile and Use OpenSVC

- Download the source code from Sourceforge
  - http://sourceforge.net/projects/opensvcdecoder/
- unzip the source code
- cd to Mplayer/ folder
- Configure:
  - `CPPFLAGS="-I/opt/local/include/" LDFLAGS="-L/opt/local/lib" CC=gcc-4.2 ./configure --enable-svc`
- make
- Decode the 4CIF version
  - `./mplayer -fps 30 -loop 0 ../../jsvm/CS5262/CS5262.264`
- Decode the CIF version
  - `./mplayer -fps 30 -setlayer 0 -loop 0 ../../jsvm/CS5262/CS5262.264`
- OpenSVC supports switching among layers, but it doesn’t work on our 264 file, why? How can we fix it? ← homework assignment?