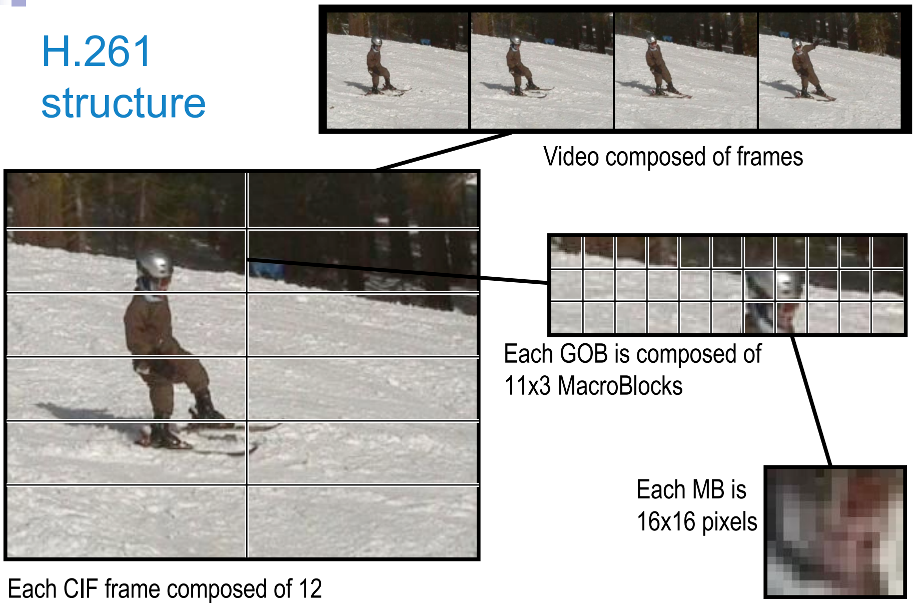
H.261 Video

Mark Handley

H.261 Video

- H. 261 Compression was designed for videotelephony and videoconferencing applications.
  - □ Developed by CCITT (now ITU-T) in 1988-1990
  - □ Intended for use over ISDN telephone lines, as part of the H.320 protocol suite.
  - □ Datarate was specified as multiples of 64Kb/s ("p x 64")
- Goals for ISDN videotelephony:
  - □ Low end-to-end delay.
  - □ Constant bit rate.

H.261 structure



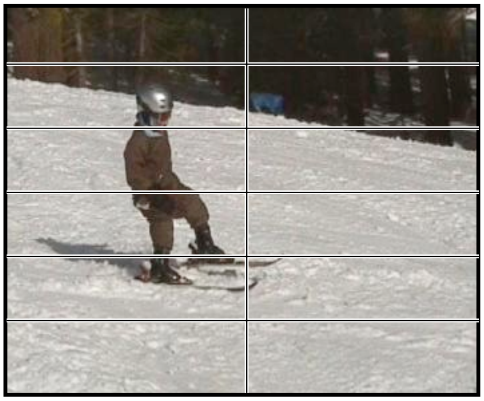
Video composed of frames

Each GOB is composed of 11x3 MacroBlocks

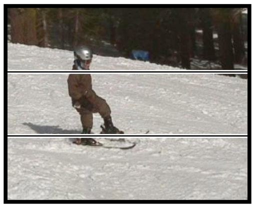
Each MB is 16x16 pixels

Each CIF frame composed of 12 Groups of Blocks (GOBs)

CIF and QCIF Frame Formats



Each CIF frame (352x288 pixels) is composed of 12 Groups of Blocks (GOBs)



Each QCIF frame (176x144 pixels) is composed of 3 Groups of Blocks (GOBs)

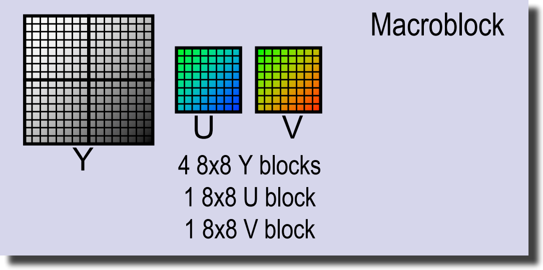
GOB and MacroBlock format is identical in both frame formats.

GOB and Resynchronization

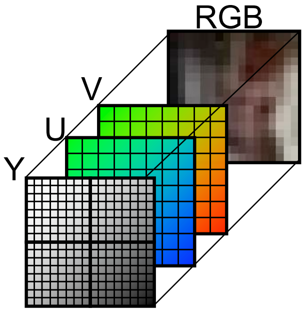
- Purpose of Group of Blocks is resynchronization.
- GOB starts with a sync code (binary: 00000000 00000001)
- Within a GOB, encoded MBs don't even start on byte boundaries.
  - □ If there's a bit error and you lose sync, or you join in the middle, you can't decode the next bits (you don't know where you are in the bitstream).
  - □ Scan for the next GOB sync code, and then you can start decoding.

Macroblocks

- Macroblock is basic unit for compression.
- Each macroblock is 16x16 pixels.
  - □ Represent as YUV 4:2:0 data.
  - □ 16x16 Luminance (Y) and subsampled 8x8 C_r, 8x8 C_b
- Represent this as 6 Blocks of 8x8 pixels:



4 8x8 Y blocks
1 8x8 U block
1 8x8 V block



Macroblock coding

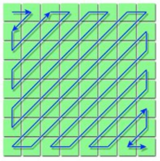
Three ways to code a Macroblock:

1. Don't.
   - If it hasn't changed since last frame, don't send it.
2. Intra-frame compression
   - Do DCT, Quantize, Zig-zag, Run-length encoding, and Huffman coding. Just like JPEG.
3. Inter-frame compression
   - Calculate difference from previous version of same block.
   - Can use motion estimation to indicate block being differenced can from a slightly different place in previous frame.
   - Same DCT/quant/huffman coding as Intra, but data is differences rather than absolute values.

H.261 intra-frame compression

Intra-coding of blocks is very similar to JPEG:

- □ DCT.
- □ Quantize DCT.
  - Unlike JPEG, H.261 uses the same quantizer value for all coefficients.
  - Feedback loop changes quantizer to achieve target bitrate.
- □ Order coefficients in zig-zag order.
- □ Run-length encode.
- □ Huffman code what remains.



H.261 inter-frame compression

- Basic compression process is the same as intra-frame compression, but the data is the differences from the immediately preceding frame rather than the raw samples themselves.

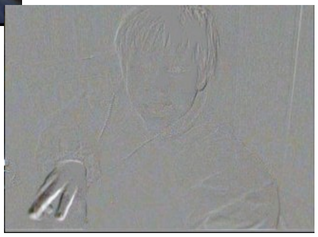
Frame Differencing

Often the amount of information in the difference between two frames is a lot less than in the second frame itself.

Frame 1

Frame 2

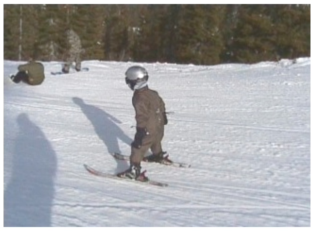
Difference:
Frame 2 - 1



Motion

- Motion in the scene will increase the differences.
- If you can figure out the motion (where each block came from in the previous frame):
  - □ Encode the motion as a motion vector (two small integers indicating motion in x and y directions)
  - □ Encode the differences from the *moved* block using DCT + quantization + RLE + Huffman encoding.

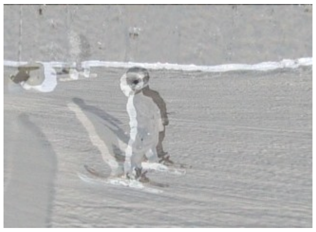
Motion



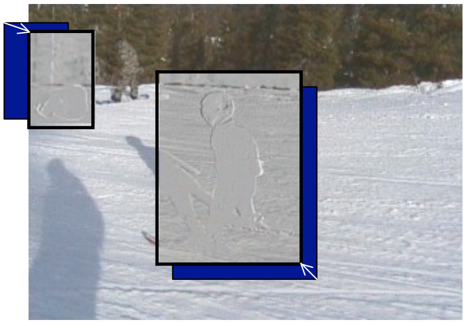
Frame 1

Frame 2

Coding from moved part of previous image can reduce the differences



Frame 2 - 1
(lots of motion)



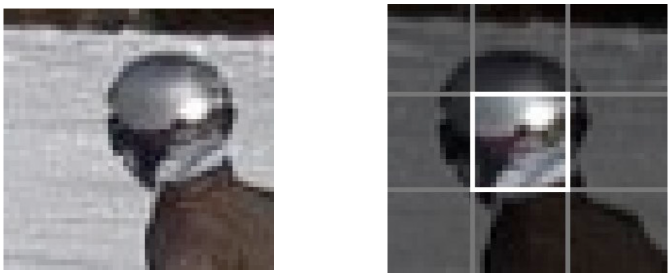
Motion Compensation in H.261

- Each inter-coded 16x16 pixel macroblock has its own motion vector.
  - □ Applies to all six 8x8 blocks in the macroblock.

- Encoder must search the image surrounding the MB to discover where it came from.
  - □ Don’t care whether it’s really motion or not - only that differencing reduces the data to send.
  - □ Motion Vector search can be the most CPU-intensive part of H.261.
  - □ Standard doesn’t say how to do this - only how to decode the results. Plenty of room for innovation.

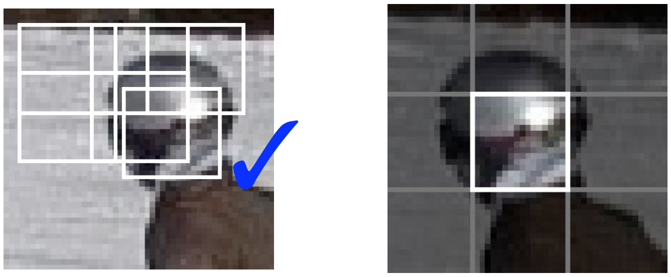
Motion Vector Search

Where did this Macroblock come from in the previous frame?



Motion Vector Search

Where did this Macroblock come from in the previous frame?

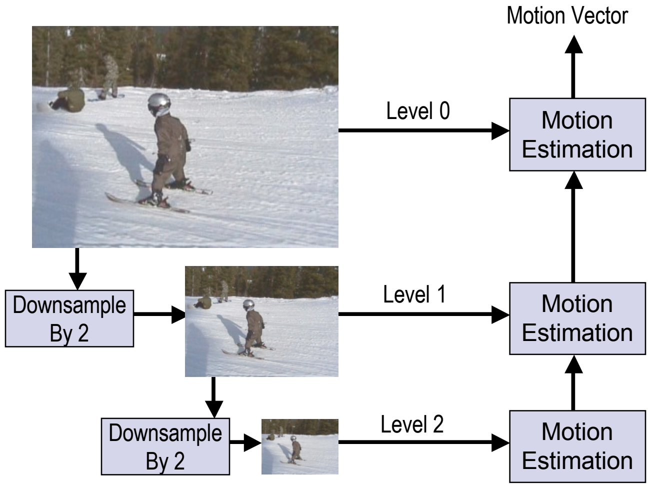


Motion Vector Search: Brute Force

- Each motion vector can encode motions of ±15 pixels in both x and y direction.
- $30^2 = 900$ possible vectors for each Macroblock.

- Calculate mean difference for each possible vector. Choose vector with least mean difference.
  - ⇒256 subtractions and 256 additions per possible vector
  - ⇒ 460K calculations per MB,
  - ⇒ 182M calculations per frame (CIF),
  - ⇒ 5.5 billion calculations per second (30fps NTSC video).
  - ⇒
- Not possible on today’s CPUs.

Hierarchical Search



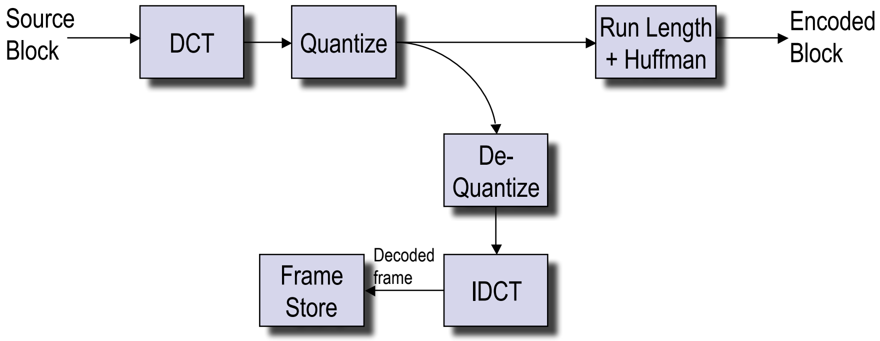
Motion vectors: ±1 around level 2 result
⇒9 positions checked
⇒16^2*2 ops per check
⇒ 4608 ops

Motion vectors: ±1 around level 2 result
⇒9 positions checked
⇒8^2*2 ops per check
⇒ 1152 ops

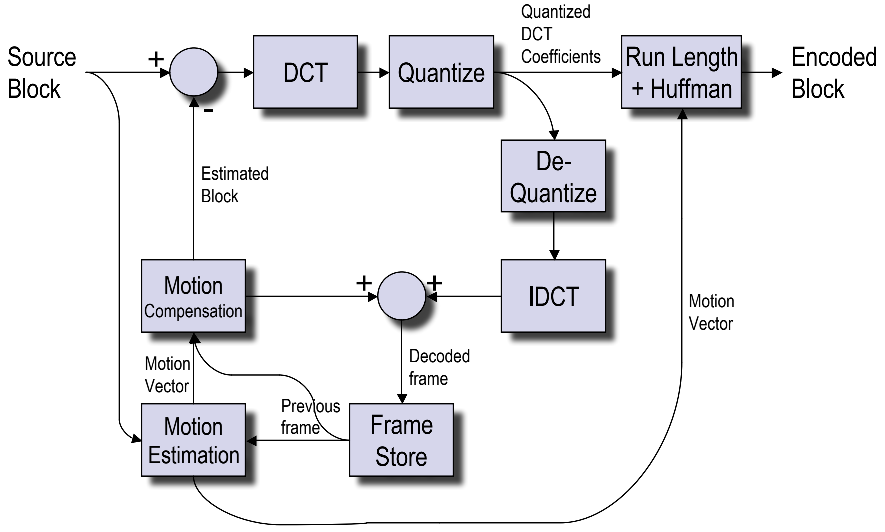
Motion vectors: ±4
⇒64 positions checked
⇒4^2*2 ops per check
⇒2048 ops

Total: 90M ops/sec for 30fps

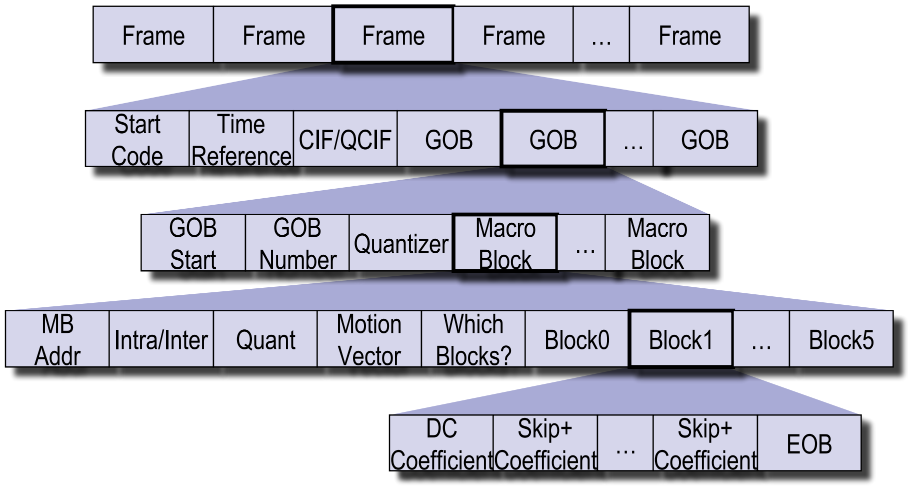
Intra-Block Encoding



Inter-Block Encoding



Bitstream Structure



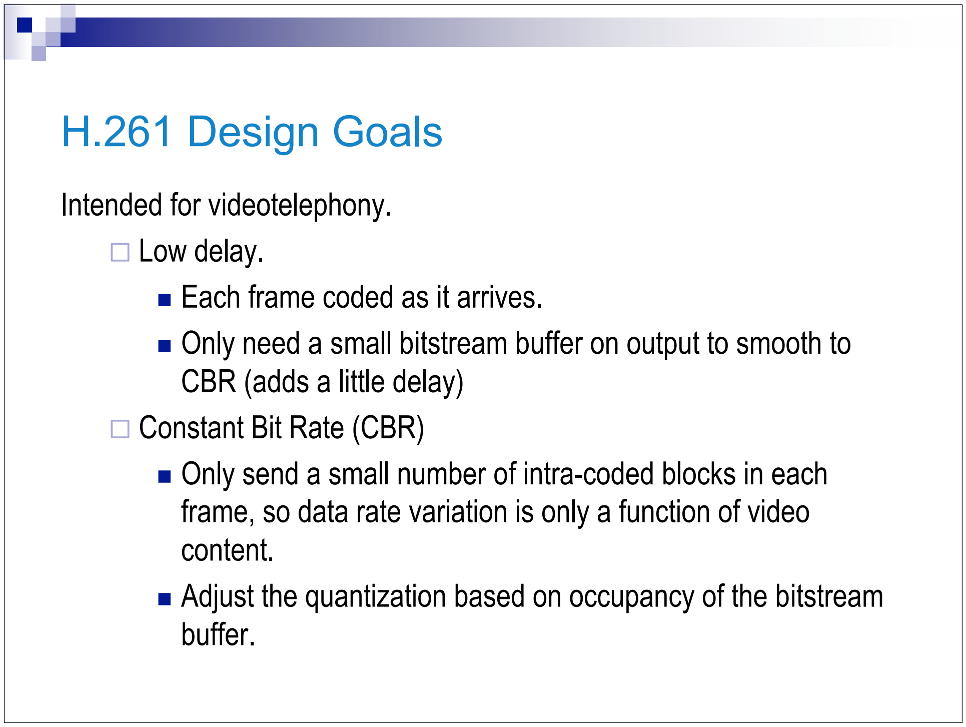
Frame | Frame | Frame | Frame | … | Frame

Start Code | Time Reference | CIF/QCIF | GOB | GOB | … | GOB

GOB Start | GOB Number | Quantizer | Macro Block | … | Macro Block

MB Addr | Intra/Inter | Quant | Motion Vector | Which Blocks? | Block0 | Block1 | … | Block5

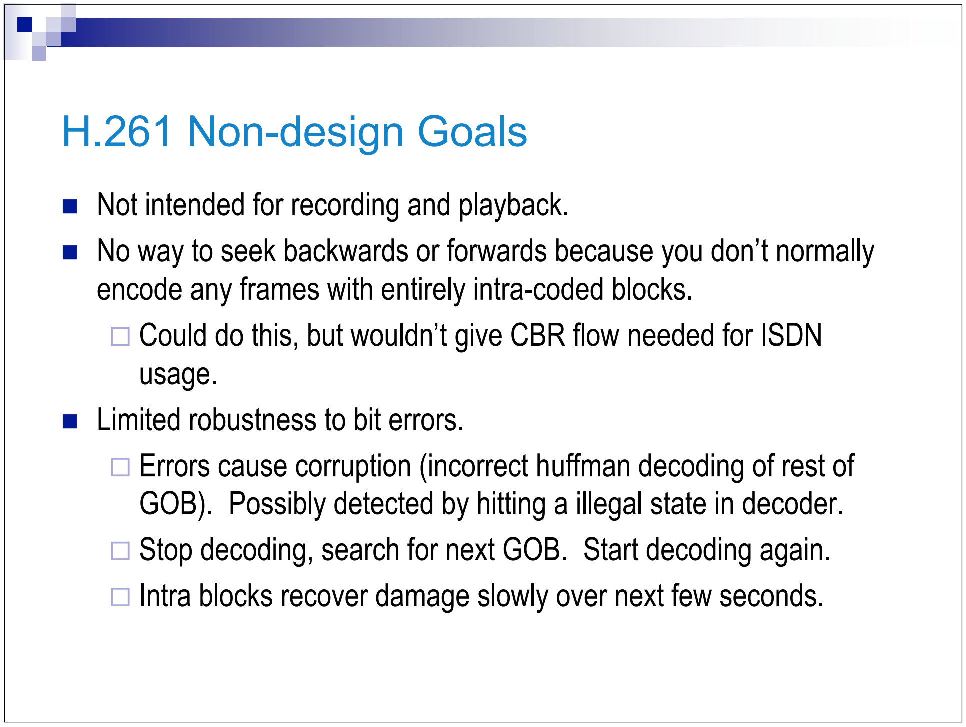
DC Coefficient | Skip+ Coefficient | … | Skip+ Coefficient | EOB



H.261 Design Goals

Intended for videotelephony.

- Low delay.
  - Each frame coded as it arrives.
  - Only need a small bitstream buffer on output to smooth to CBR (adds a little delay)
- Constant Bit Rate (CBR)
  - Only send a small number of intra-coded blocks in each frame, so data rate variation is only a function of video content.
  - Adjust the quantization based on occupancy of the bitstream buffer.



H.261 Non-design Goals

- Not intended for recording and playback.
- No way to seek backwards or forwards because you don’t normally encode any frames with entirely intra-coded blocks.
  - Could do this, but wouldn’t give CBR flow needed for ISDN usage.
- Limited robustness to bit errors.
  - Errors cause corruption (incorrect huffman decoding of rest of GOB). Possibly detected by hitting a illegal state in decoder.
  - Stop decoding, search for next GOB. Start decoding again.
  - Intra blocks recover damage slowly over next few seconds.

H.263

- Son of H.261.
  - Standardized in 1996.
  - Replacing H.261 in many applications.
- Basic design is very similar to H.261 (DCT/Quantization based, using intra or inter frame coding).
  - Numerous optional improvements to improve compression, robustness, and flexibility of use.

H.263 Improvements

- Half-pixel precision in motion vectors (vs full-pixel precision for H.261).
- New options:
  - Unrestricted Motion Vectors,
  - Syntax-based arithmetic coding (replace RLE/Huffman)
  - Advance prediction (uses 4 8*8 blocks instead of 1 16*16: gives better detail.)
  - Forward and backward frame prediction similar to MPEG
- Five resolutions (H.261 only does QCIF and CIF):

SQCIF: 128x96	4CIF: 704x576
QCIF: 176x144	16CIF: 1408x1152
CIF: 352x288	