MPEG-1

MPEG-1 VIDEO

How does MPEG-1 VIDEO work ?

First off, it starts with a relatively low resolution video sequence (possibly decimated from the original) of about 352 by 240 frames by 30 frames/s (US--different numbers for Europe), but original high (CD) quality audio. The images are in color, but converted to YUV space, and the two chrominance channels (U and V) are decimated further to 176 by 120 pixels. It turns out that you can get away with a lot less resolution in those channels and not notice it, at least in *natural* (not computer generated) images.

The basic scheme is to predict motion from frame to frame in the temporal direction, and then to use DCTs (Discrete Cosine Transforms) to organize the redundancy in the spatial directions. The DCTs are done on 8x8 blocks, and the motion prediction is done in the luminance (Y) channel on 16x16 blocks. In other words, given the 16x16 block in the current frame that you are trying to code, you look for a close match to that block in a previous or future frame (there are backward prediction modes where later frames are sent first to allow interpolating between frames). The DCT coefficient (of either the actual data, or the difference between this block and the close match) are *quantized*, which means that you divide them by some value to drop bits off the bottom end. Hopefully, many of the coefficients will then end up being zero. The quantization can change for every macroblock (a macroblock is 16x16 of Y and the corresponding 8x8's in both U and V). The results of all of this, which include the DCT coefficients, the motion vectors, and the quantization parameters (and other stuff) is Huffman coded using fixed tables. The DCT coefficients have a special Huffman table that is *two-dimensional* in that one code specifies a run-length of zeros and the non-zero value that ended the run. Also, the motion vectors and the DC DCT components are DPCM, (subtracted from the last one) coded.

There are three types of coded frames. There are **I** or intra frames. They are simply a frame coded as a still image, not using any past history. You have to start somewhere. Then there are **P** or predicted frames. They are predicted from the most recently reconstructed I or P frame. (I'm describing this from the point of view of the decompressor.) Each macroblock in a P frame can either come with a vector and difference DCT coefficients for a close match in the last I or P, or it can just be *intra* coded (like in the I frames) if there was no good match.

Lastly, there are B (bidirectional) frames. They are predicted from the closest two I or P frames, one in the past and one in the future. You search for matching blocks in those frames, and try three different things to see which works best. (Now I have the point of view of the compressor, just to confuse you.) You try using the forward vector, the backward vector, and you try averaging the two blocks from the future and past frames, and subtracting that from the block being coded. If none of those work well, you can intra- code the block.

The sequence of decoded frames usually goes like:

```
IBBPBBPBBPBBIBBPBBPB...
```

Where there are 12 frames from I to I (for US and Japan anyway.) This is based on a random access requirement that you need a starting point at least once every 0.4 seconds or so. The ratio of P's to B's is based on experience. Of course, for the decoder to work, you have to send that first P *before* the first two B's, so the compressed data stream ends up looking like:

```
0xx312645...
```

where those are frame numbers. xx might be nothing (if this is the true starting point), or it might be the B's of frames -2 and -1 if we're in the middle of the stream somewhere.

You have to decode the I, then decode the P, keep both of those in memory, and then decode the two B's. You probably display the I while you're decoding the P, and display the B's as you're decoding them, and then display the P as you're decoding the next P, and so on.

What do B-frames buy you ?

Since bi-directional macroblock predictions are an average of two macroblocks blocks, noise is reduced at low bit rates. At nominal MPEG-1 video (352 x 240 x 30, 1.15 Mbit/sec) rates, it is said that B-frames improves SNR by as much as 2 dB. (0.5 dB gain is usually considered worth-while in MPEG). However, at higher bit rates, B-frames become less useful since they inherently do not contribute to the progressive refinement of an image sequence (i.e. not used as prediction by subsequent coded frames). Regardless, B-frames are still politically controversial.

B pictures are interpolative in two ways:

1. predictions in the bi-directional macroblock are an average from block areas of two pictures
2. B pictures fill in or interpolate the 3-D video signal over a 33 or 25 millisecond picture period without contributing to the overall signal quality beyond that immediate point in time.

In other words, a B picture, regardless of its internal make-up of macroblock types, has a life limited to its immediate self. As mentioned before, its energy does not propagate into other frames. In a sense, bits spent on B pictures are wasted.

Why do some people hate B-frames ?

Computational complexity, bandwidth, delay, and picture buffer size are the four B-frame Pet Peeves. Computational complexity is increased since some macroblock modes require averaging between two macroblocks. Worst case, memory bandwidth is increased an extra 16 MByte/s (601 rate) for this extra prediction. An extra picture buffer is needed to store the future prediction reference (bi-directionality). Finally, extra delay is introduced in encoding since the frame used for backwards prediction needs to be transmitted to the decoder before the intermediate B-pictures can be decoded and displayed.

Cable television (e.g. General Instruments) have been particularly adverse to B-frames since the extra picture buffer pushes the decoder DRAM memory requirements past the magic 8-Mbit (1 Mbyte) threshold into the realm of 16 Mbits (2 MByte) for CCIR-601 frames (704 x 480), yet not for lowly 352 x 480. However, cable does not realize that DRAM does not come in convenient high-volume (low cost) 8-Mbit packages as 16-Mbit does. In a few years, the cost differences between 16 Mbit and 8 Mbit will become insignificant compared to the gain in compression. For the time being, cable boxes will start with 8-Mbit and allow future drop-in upgrades to 16-Mbit.

Can motion vectors be used to measure object velocity ?

Motion vector information cannot be reliably used as a means of determining object velocity unless the encoder model specifically set out to do so. First, encoder models that optimize picture quality form vectors that typically minimize prediction error and, consequentially, the vectors often do not represent

true object translation. Standards converters that re-sample one frame rate to another (as in NTSC to PAL) use different methods (field coding, edge detection, et al) that are not concerned with optimizing SNR vs bitrate. Secondly, motion vectors are not transmitted for all macroblocks anyway.

How do you code interlaced video with MPEG-1 syntax ?

Two methods can be applied to interlaced video that maintain syntactic compatibility with MPEG-1 (which was originally designed for progressive frames only). In the field concatenation method, the encoder model can carefully construct predictions and prediction errors that realize good compression but maintain field integrity (distinction between adjacent fields of opposite parity). Some pre-processing techniques can also be applied to the interlaced source video that would, e.g., lessen sharp vertical frequencies. This technique is not efficient of course. On the other hand, if the original source was progressive (e.g. film), then it is more trivial to convert the interlaced source to a progressive format before encoding. (MPEG-2 would then only offer superior performance through greater DC block precision, non-linear mquant, intra VLC, etc.) Reconstructed frames are re-interlaced in the decoder Display process.

The second syntactically compatible method codes fields separately. Picture types are keyed to motion activity to aid efficiency of prediction.

Where did they get 352x240 ?

That derives from the CCIR-601 digital television standard which is used by professional digital video equipment. It is (in the US) 720 by 243 by 60 fields (not frames) per second, where the fields are interlaced when displayed. (It is important to note though that fields are actually acquired and displayed a 60th of a second apart.) The chrominance channels are 360 by 243 by 60 fields a second, again interlaced. This degree of chrominance decimation (2:1 in the horizontal direction) is called 4:2:2. The source input format for MPEG I, called SIF, is CCIR-601 decimated by 2:1 in the horizontal direction, 2:1 in the time direction, and an additional 2:1 in the chrominance vertical direction. And some lines are cut off to make sure things divide by 8 or 16 where needed. For 50 Hz display standards (PAL, SECAM) change the number of lines in a field from 243 or 240 to 288, and change the display rate to 50 fields/s or 25 frames/s. Similarly, change the 120 lines in the decimated chrominance channels to 144 lines. Since 288*50 is exactly equal to 240*60, the two formats have the same source data rate.

Can MPEG-1 encode higher sample rates than 352x240x30 or 352x288x25?

It seems to be the number one misconception about MPEG-1: that it has fixed or limited frame size, i.e. 352x240x30 fps or 352x288x25 fps.

In fact MPEG-1 can use *any* frame size, including CCIR-601 resolutions (704x480), with frame sizes as high as 4095x4095x60 fps.
MPEG-2 is more limited since frame sizes must be multiples of 16.

The misconception apparently arises from the *standard profile*, a kind of subset known as Constrained Parameters Bitstream (CPB), which is a series of restrictions that the MPEG-1 stream must meet, including bit rate and frame sizes. Most (maybe all) hardware decoders accept only streams that follow the CPB profile.

You can encode with any bit rate or frame size and you still have standard MPEG-1; you just can't say it meets the standard profile or whatever, and you probably can't play it with some existing decoder chips.

What are Constrained Parameters Bitstreams ?

CPB are a limited set of sampling and bitrate parameters designed to normalize computational complexity, buffer size, and memory bandwidth while still addressing the widest possible range of applications. CPB limits video to 396 macroblocks (101,376 pixels) per frame if the frame rate is less than or equal to 25 fps (frames per second), and 330 macroblocks (84,480 pixels) per frame if the frame rate is less or equal to 30 fps. Therefore, MPEG video is typically coded at SIF dimensions (352 x 240 x 30fps or 352 x 288 x 25 fps).

The total maximum sampling rate is 3.8 Ms/s (million samples/sec) including chroma. The coded video rate is limited to 1.862 Mbit/sec. In industrial practice, the bitrate is the most often waived parameter of CPB, with rates as high as 6 Mbit/sec in use.

Why are Constrained Parameters Bitstreams so important?

It is an optimum point that allows (just barely) cost effective VLSI implementations in 1992 technology (0.8 microns). It also implies a nominal guarantee of interoperability for decoders and encoders. MPEG devices which are not capable of meeting SIF rates are not canonically considered to be true MPEG.

Are there ways of getting around Constrained Parameters Bitstreams for SIF class applications and decoder ?

Yes, some. Remember that CPB limits frames to 396 macroblocks (as in 352 x 288 SIF frames). 416 x 240 x 24 Hz sampling rates are still within the constraints, but this only aids NTSC (240 lines/field) displays. Deviating from 352 samples/line could throw off many decoder implementations that have limited horizontal sample rate conversion modes. Due to chip die size constraints (most chips barely pack in the necessary features), many decoders use simple doubling, e.g. 352 to 704 samples/line via binary taps which are simple shift-and-add operations. Future MPEG decoders will have arbitrary sample rate converters on-chip. Also remember that the 1.86 Mbit/sec limit is often ignored in real life.

How much does it compress ?

As mentioned before, audio CD data rates are about 1.5 Mbits/s. You can compress the same stereo program down to 256 Kbits/s with no loss in discernible quality. (So they say. For the most part it's true, but every once in a while a weird thing might happen that you'll notice. However the effect is very small, and it takes a listener trained to notice these particular types of effects.) That's about 6:1 compression. So, a CD MPEG I stream would have about 1.25 MBits/s left for video. The number I usually see though is 1.15 MBits/s (maybe you need the rest for the system data stream). You can then calculate the video compression ratio from the numbers here to be about 26:1. If you step back and think about that, it's little short of a miracle. Of course, it's lossy compression, but it can be pretty hard sometimes to see the loss, if you're comparing the SIF original to the SIF decompressed. There is, however, a very noticeable loss if you're coming from CCIR-601 and have to decimate to SIF, but that's another matter. I'm not counting that in the 26:1.

The standard also provides for other bit rates ranging from 32Kbits/s for a single channel, up to 448 Kbits/s for stereo.