H.263

Video codec standard

2001-10-30

Anders Lidén, 800424-0035
Magnus Haglund, 780621-6995
Per Sundling, 790910-8917

Luleå Tekniska Universitet

Table of contents

Introduction

The H.263 video codec standard was specified in 1996 by The International Telecommunications Union (ITU). Version 2 of the standard was approved in 1998.

The main objective of the H.263 codec is to provide a video coding standard that is optimized for relatively low motion video and suitable for applications with bit rates below 64kbits/s. It has a strong temporal compression component, and works best on videos in which there is little change between frames. But H.263 is not only for low-bit rate. It has a scalable bit rate encoding that can present video at rates below 20 Kbps, and increasingly improved video quality up to about 600 Kbps.

H.263 can be used in a range of applications:

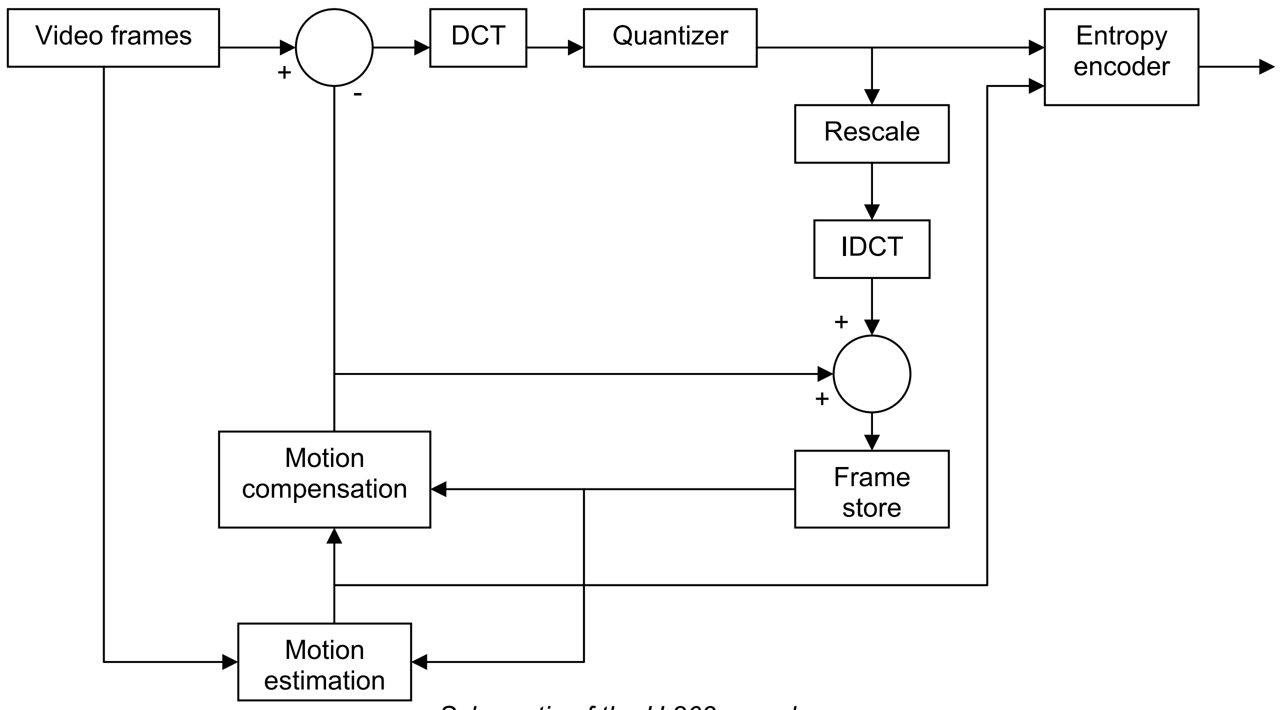
- Desktop videoconferencing
- Internet video
- Surveillance and monitoring
- Transmission over a low bit-rate down to 9600bps radio links
- And much more

H.263 is basically an improved version of H.261. It has improved motion estimation and several new features, for example optional arithmetic coding and the size of motion estimation block can also be adaptively switched between 16x16 and 8x8.

Although the standard only specifies the requirements for a video encoder and decoder, it does not describe the encoder or decoder themselves. Instead, it specifies the format and content of the encoded (compressed) stream.

H.263 Encoder

The aim of the encoder is to take the video frames and make an as short as possible bit stream of them in order to be able to send the video over the Internet or a network. The quality needs to be as good as possible and the bit stream needs to be as short as possible. In order to compile the bit stream in an efficient way there are several methods to go through before we have the bit stream.



Schematic of the H.263 encoder

Motion compensation and estimation

The first step when trying to reduce the bandwidth needed is to subtract the previous transmitted frame from the current frame so that only the difference needs to be encoded and transmitted over the network. This means that areas of the video frame that do not change between the frames are not encoded and transmitted over the network again and thus we save bandwidth. The next thing we can do to limit the bandwidth needed is to estimate where areas of the previous frame have moved in the current frame, and then try to compensate for this movement. The motion estimation is done by comparing each macroblock (normally a 16x16 pixels block) in the current frame with its surrounding area in the previous frame and try to find a match. If we have a match, the matching area is moved into the current macroblock position by the motion compensator module. The motion compensated macroblock is then subtracted from the current macroblock. If the motion compensation and estimation process is efficient the remaining macroblock should contain only as small amount of information that needs to be encoded further.

Discrete cosine transform (DCT)
Next step is to transform a block of pixel values (or the residual values from the motion compensation and estimation module) into a set of spatial frequency coefficients. This is done pretty much the same way as when transforming a time domain signal into a frequency domain signal using Fast Fourier Transform. The difference is that the DCT operates on a 2-dimensional block of pixels instead of a 1-dimensional signal, which is the case of FFT. The DCT is particularly good at compacting the energy in the block of values, into a small number of coefficients. This means that only a small number of coefficients are needed in order to recreate a recognizable copy of the original block of pixels.

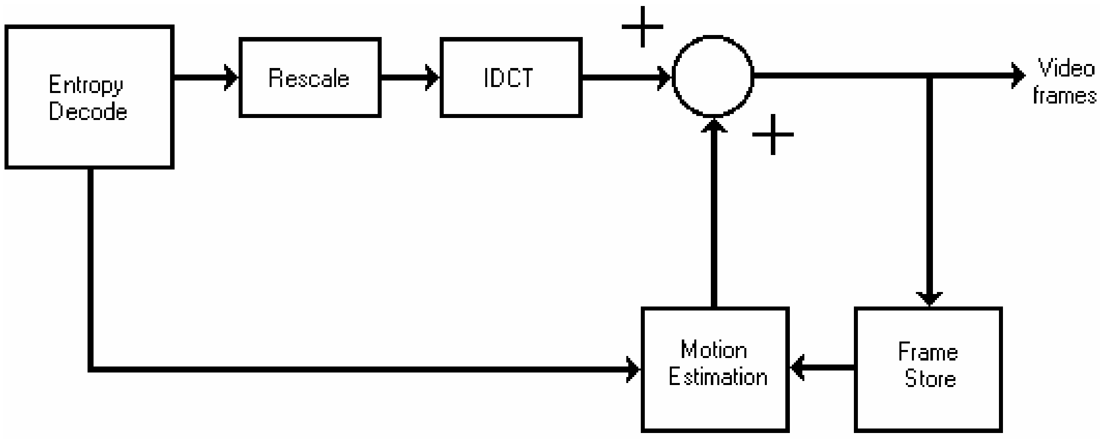
Quantization
The third step is the quantizer. For a typical block of pixels, most coefficients produced by the DCT are close to zero. The quantizer reduces the precision of the coefficients by dividing each coefficient by an integer scale factor and truncating the result. This will mean that coefficients close to zero will be set to zero and only a few significant coefficients are left. It is important to know that the quantizer throws away information that cannot be reproduced later on.

Entropy encoding
The fourth step is the entropy encoder, such as a Huffman encoder. The entropy encoder replaces frequently occurring values of the DCT coefficients with short binary codes and replaces infrequently occurring values with longer binary codes. The result is a sequence of variable-length binary codes. These codes are then combined with synchronization and control information, such as motion compensation vectors required to reconstruct the motion-compensated frame, to form the H.263 bit stream.

Frame store
The last step is the frame store. The current frame has to be stored in order to have a frame to compare with when encoding the next frame of the video. The quantized coefficients are rescaled, inverse transformed using Inverse Discrete Cosine Transform and added to the motion compensated reference block to create a reconstructed frame that is placed in a store called frame store. This will ensure that the content of the reference frame is identical in both the encoder and decoder. When the next frame is encoded the motion estimation module uses the content of this frame store to determine the best matching area for motion compensation.

H.263 Decoder

The aim of the decoder is to reverse the operations of the encoder and estimate the next frame best possible with the help of the previous frame.



Schematic of the H.263 decoder

Entropy decode

The incoming H.263 bit stream, which is made up of variable-length codes, is decoded and the motion vector information and the coefficient values are extracted.

Rescale

Rescale “reverses” the quantization made when encoding. The coefficients are multiplied by the same scaling factor that was used in the quantizer. However, because the quantizer discarded the fractional remainder, the rescaled coefficients are not identical to the original coefficients.

Inverse Discrete Cosine Transform

The IDCT reverses the DCT operation to create a block of samples. These (typically) correspond to the difference values that were produced by the motion compensator in the encoder.

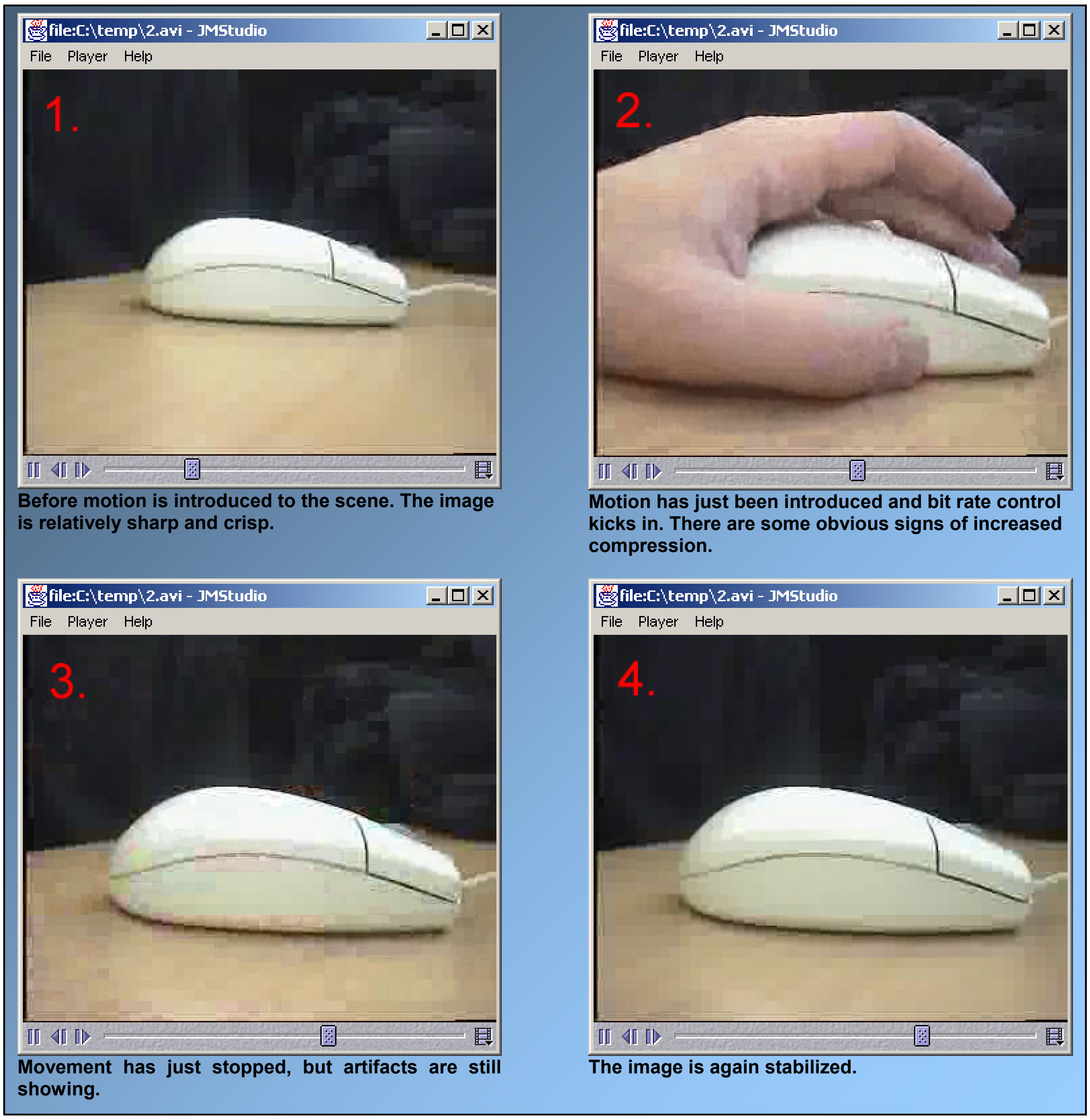
Motion compensation

The difference values are added to a reconstructed area from the previous frame. The motion vector information is used to pick the correct area (the same reference area that was used in the encoder). The result is a reconstruction of the original frame: note that this will not be identical to the original because of the "lossy" quantization stage, i.e. the image quality will be poorer than the original. The reconstructed frame is placed in a frame store and it is used to motion-compensate the next received frame.

Implementation issues (regarding real-time video communications)

Bit rate control

Bit rate control is a mechanism built into H.263 that allows the codec to keep a constant output bit rate no matter what the contents of the video stream looks like. Normally the encoder generates a variable number of bits for each encoded frame, depending on how well the motion compensation process works. Sudden increases in scene content complexity or scene motion makes it much harder for this process to work correctly, and this is when bit rate control is needed to avoid frame dropping. It is carried out by measuring the encoder's output buffer and adjusting the quantization scale factor if it deviates from its predefined size. This way the output is effectively mapped to a certain bandwidth. Illustration 1 shows how bit rate control affects the image.



Before motion is introduced to the scene. The image is relatively sharp and crisp.

Motion has just been introduced and bit rate control kicks in. There are some obvious signs of increased compression.

Movement has just stopped, but artifacts are still showing.

The image is again stabilized.

Illustration 1. A series of frames showing bit rate control in action.

Synchronization

It is important for the encoder and decoder to stay synchronized; otherwise the quality of the decoded image will deteriorate very quickly. Desynchronization or loss of synchronization is common in “noisy” transmission environments, like mobile networks, where bit error rates can be quite high. Under these circumstances it is important to have some kind of countermeasure. H.263 uses synchronization markers. These markers are special codes that indicate to a decoder the position of the current data within a frame and the “time code” of the current frame. If the decoder loses synchronization then it can simply scan forward for the next marker in order to resynchronize and resume decoding. Synchronization markers limit the propagation of errors within frames.