

# FLASH VIDEO TECHNOLOGY AND OPTIMIZATIONS

Ing. Fabio Sonnati – **Progetto Sinergia**  
(Revision 1.1 – July 2005)

## Introduction to white paper

This whitepaper analyzes the video technology currently implemented in Flash Player 6 and 7. This is a technical document written to enhance the knowledge of Flash Video capabilities.

Flash Video Technology derives from H.263 standard. To better understand the potentiality of Flash Video, will be usefull to start examining this standard and then the main differences in Flash implementation. The first Chapter of this paper is dedicated to an indeep examination of the basic video compression standards (H.261, H.263, H.263+).

In the second Chapter, we'll analyze the specific Flash codec for real-time video compression. This Topic is very interesting for Flash Communication Server coders.

In the third and last Chapter, we'll describe two form of optimization to improve the real-time encoding performance of standard Flash video codec of, around, 20-30%.

If you are expert of video compression techniques or if you are more interested in how to improve the performance of the real-time video encoding, you may jump to chapter 2 and 3.

This whitepaper is dedicated to all the FlashCom coders.

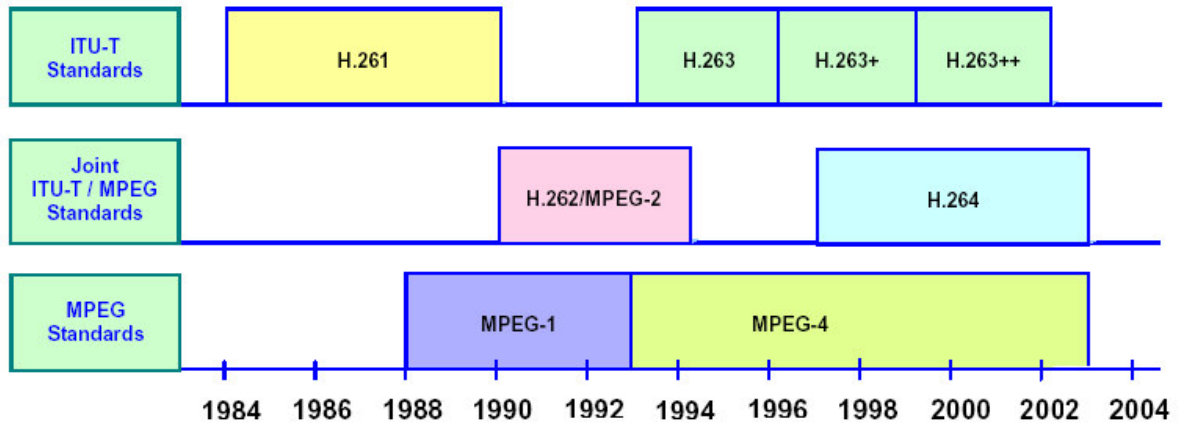
## 1 - Introduction to video standards

Digital video is being adopted in an increasing range of applications covering video telephony, videoconferencing, DVD, digital TV, Internet streaming. The adoption of digital video in so many applications has been accelerated by the development of many video coding standards. These standards provide interoperability between different systems and applications facilitating the growth of the digital video market.

The ITU-T (International Telecommunications Union) and the ISO/IEC (International Standardization Organization/International Electrotechnical Commission) are the only two formal organizations that develop video coding standards. The ITU-T video coding standards, called recommendations and denoted with H.26x (e.g., H.261, H.262, H.263 and H.264) are usually optimized for real-time video communication such as videoconference and video telephony while the ISO/IEC standards, denoted with MPEG-x (e.g., MPEG-1, MPEG-2 and MPEG-4), are mainly designed for storage (DVD) and broadcast (satellite and digital TV).

For the most part, the two standardization committees have worked independently on the different standards. The only exception has been the H.262/MPEG-2 standard, which was developed jointly by the two committees. Recently, the ITU-T and the ISO/IEC JTC1 have agreed to join their efforts in the development of the emerging H.264 standard, which was initiated by the ITU-T committee.

Figure summarizes the evolution of the ITU-T recommendations and the ISO/IEC MPEG standards.



The main objective of the H.26x standards is to provide a mean to achieve substantially high video bandwidth compression for real-time communication applications.

The underlying approach of all H.26x is similar and consists of the following four main stages:

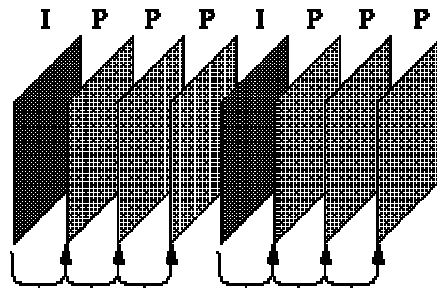
1. *Dividing each video frame into blocks of pixels so that processing of the video frame can be conducted at the block level.*
2. *Exploiting the spatial redundancies that exist within the video frame by coding some of the original blocks through spatial prediction, transform, quantization and entropy coding (or variable-length coding).*
3. *Exploiting the temporal dependencies that exist between blocks in successive frames, so that only changes between successive frames need to be encoded. This is accomplished by using **motion estimation and compensation**. For any given block, a search is performed in the previously coded frame to determine the motion vectors that are then used by the encoder and the decoder to predict the subject block.*
4. *Exploiting any remaining spatial redundancies that exist within the video frame by coding the residual blocks, i.e., the difference between the original blocks and the corresponding predicted blocks, again through transform, quantization and entropy coding.*

## 1.1 - H.261 Compression Standard

H. 261 Compression has been specifically designed for video telecommunication applications. Developed by CCITT (ITU-T) in 1988-1990, H.261 has been widely used in videotelephone applications and videoconference systems over ISDN lines. The Bandwidth target is an integral multiple of baseline ISDN bandwidth ( $p \times 64\text{Kbit/s}$ ).

The basic approach to H. 261 Compression is here summarized:

- Frame types are CCIR 601 CIF (352x288) and QCIF (176x144) images, in color space YCbCr with 4:2:0 chrominance subsampling.
- There are two frame types: Intra-frames (*I-frames*) and Inter-frames (*P-frames*) forming the following frame sequence: I,P,P,...,P,P,I,P,P,...,P,P,I,...



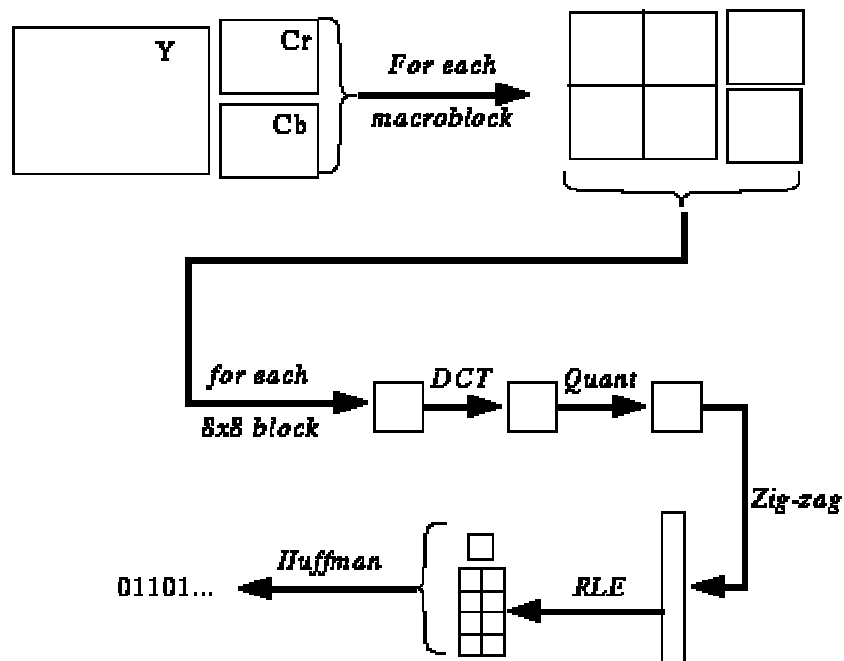
- I-frames use basically JPEG compression (DCT transform, quantization, entropy coding)
- P-frames use **pseudo-differences** from previous frame (P or I), so frames depend on each other and prediction goes forward in time.
- I-frames provide a mean of sincronization between the stream source and the stream destinations. I-frames are also called Key-Frames.

### Intra Frame Coding

In **intra frame coding** the lossy compression techniques are performed relative to information that is contained only within the current frame, and not relative to any other frame in the video sequence.

In other words, no temporal processing is performed outside of the current picture or frame. This mode will be described first because non-intra coding techniques are extensions to these basics.

Figure shows a block diagram of a basic video encoder for intra frames only. It is very similar to that of a JPEG still image video encoder:



The basic processing blocks shown are the video filter, discrete cosine transform, DCT coefficients quantizer, and run-length amplitude/variable length coder.

The video filter processes and prepares the original frame format for the other blocks. The frame is converted in YCbCr (or YUV) color space and then logically subdivided in *Block* (8x8 pixels) and *Macroblocks* (16x16 pixels).

Research into the Human Visual System (HVS) has shown that the eye is most sensitive to changes in luminance, and less sensitive to variations in chrominance. To gain compression advantage from such eye's characteristic, H/261 (and MPEG) uses the YCbCr color space to represent the data values instead of RGB, where Y is the luminance signal and CbCr the chrominance signal (Cb is the blue color difference signal, and Cr is the red color difference signal).

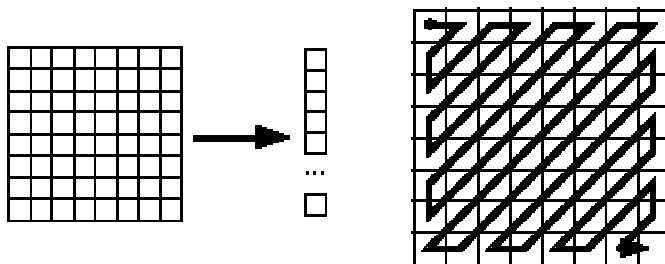
A macroblock can be represented in several different manners when referring to the YCbCr color space. There are 3 mainly known formats: 4:4:4, 4:2:2, and 4:2:0 video. 4:4:4 is full bandwidth YCbCr video, and each macroblock consists of 4 Y blocks, 4 Cb blocks, and 4 Cr blocks. Being full bandwidth, this format contains as much information as the data would if it were in the RGB color space. 4:2:2 contains half as much chrominance information as 4:4:4 (downsampling of 2 along X axis), and 4:2:0 contains one quarter of the chrominance information (downsampling of 2 along X and Y axis). H.261 and the majority of video-codec use this third mode that allows an immediate 1:2 bandwidth compression without visible loss of quality.

After Video filtering, each 8x8 block of the frame is trasformed from space domain to spatial frequency domain using a DCT (Discrete Cosine Transform). The DCT coefficients represent the spatial details of the Picture Block. Quantizing coarsely the higher frequency coefficients retaining more bits of information for the lower frequency coefficients brings to block bit-stream compression with moderate loss of quality.

The Zig-Zag reading of the DCT quantized coefficients is a typical trick to obtain a better organization of values for a more efficient compression in the successive Entropy Coding Block (bit-stream redundancies reduction through the use of Variable Length Coding).

What is the purpose of the Zig-zag Scan:

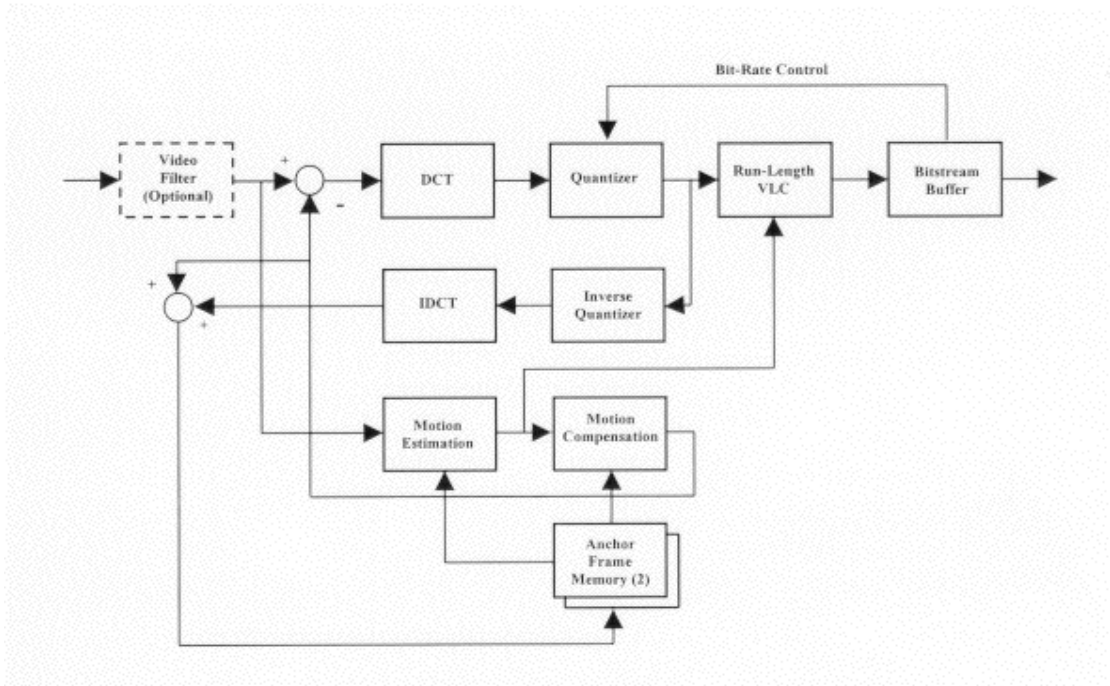
- to group low frequency coefficients in top of vector.
- Maps 8 x 8 to a 1 x 64 vector



### Inter-frame (P-frame) Coding

The previously discussed intra frame coding techniques were limited to processing the video signal on a spatial basis, relative only to information within the current video frame. Considerably more compression efficiency can be obtained however, if the inherent temporal, or time-based redundancies, are exploited as well.

Temporal processings, to exploit this redundancy, use a technique known as **block-based motion compensated prediction, using motion estimation**. A block diagram of the basic encoder with extensions for non-intra frame coding techniques is given in Figure. Of course, this encoder can also support intra frame coding as a subset.

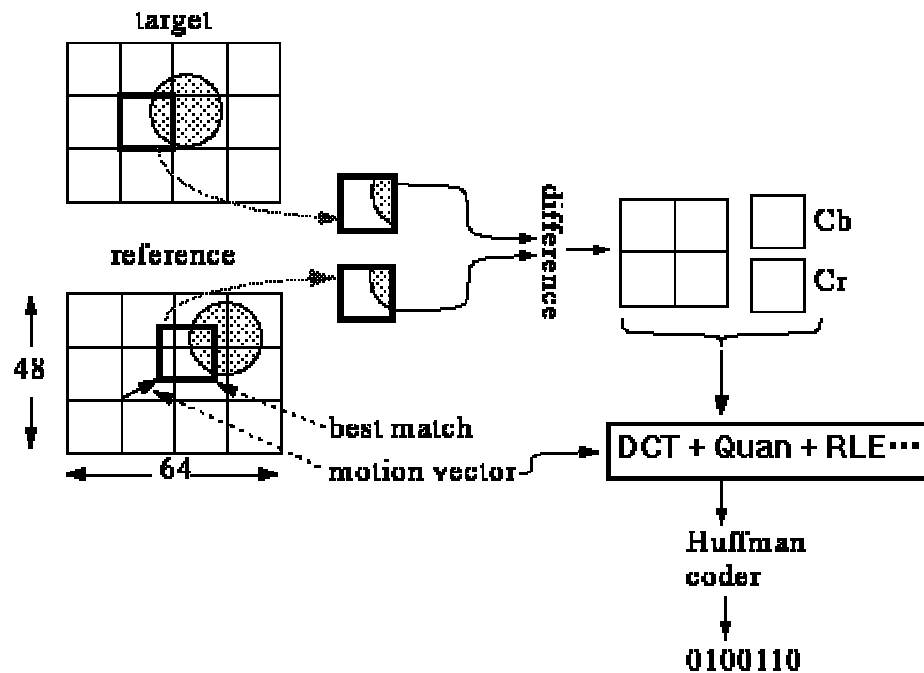


### P-Frame Coding

Starting with an intra, or I frame, the encoder can forward predict a future frame. This is commonly referred to as a P frame, and it may also be predicted from other P frames, although only in a forward time manner. As an example, consider a group of pictures that lasts for N frames. In this case, the frame ordering is given as I,P,P,P,...,P,I,P,P,P,...

Each P frame in this sequence is predicted, on a Macroblock basis, from the frame immediately preceding it, whether it is an I frame or a P frame. As a reminder, I frames are coded spatially with no reference to any other frame in the sequence.

P-coding can be summarised as follows:



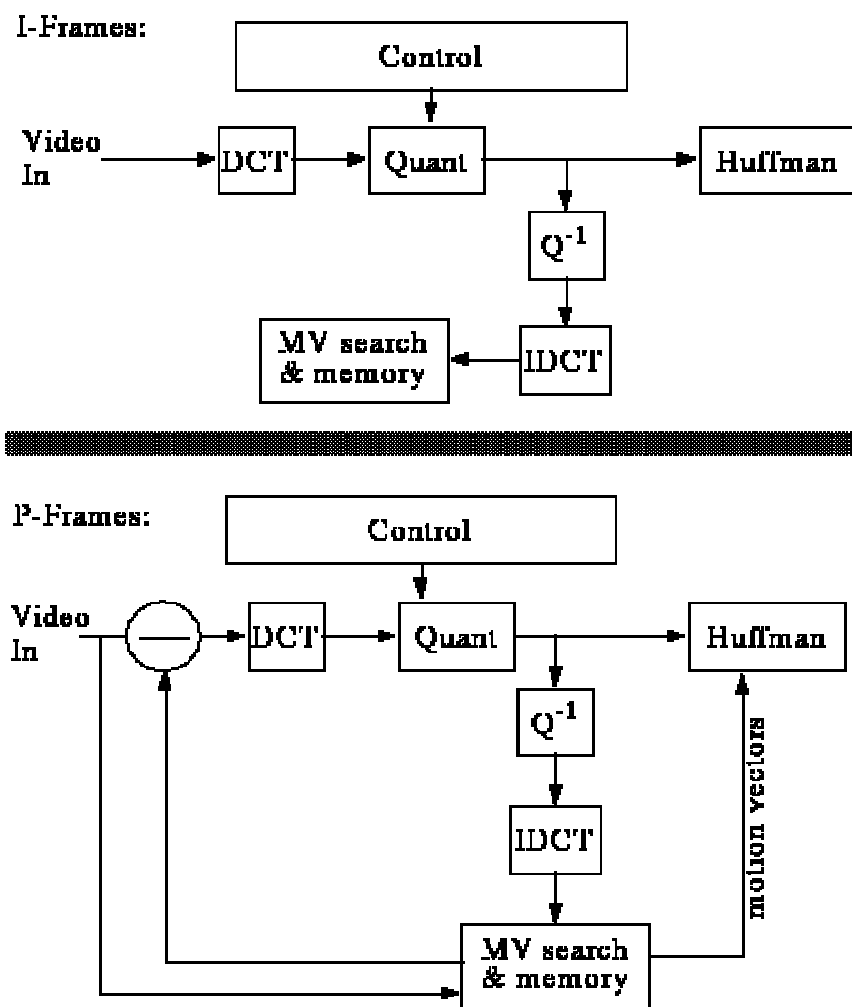
The key principle of Inter-frame compression is to recover, from the previous frame, as much information as possible. This is based on the fact that information between adjacent frame is redundant, especially for slow-varying or still frames. Recovering information using Inter-frame prediction, reduces this temporal-redundancy.

For each Macroblock on the current frame the codec tries to find in the surrounding of the relative macroblock of the previous frame the *reference area* (again of 16x16 pixel) that best matches the current macroblock of the current frame. Indeed the codec tries to minimize the difference signal (Mean Absolute Difference or Mean Squared Error).

Once found the *reference area*, we can process (compress) the difference signal again using DCT and quantization, and transmit this compressed bitstream plus the Motion Vector that point to the reference area. These information are sufficient to the decoder to rebuilt the block with lesser amount of information than that required compressing the stand-alone Macroblock using Intra-compression.

H.261 defines the Motion Vector as an Integer in the range [-16,+15] (1 pixel or pel accuracy).

The processing of Macroblocks in I-frames and P-frames are summarized by the following Figure:



## 1.2 - H.263 Compression Standard

### Introduction

ITU-T developed H.263 in the years 1993-1996. H.263 has the goal to achieve better compression than H.261 with much more flexibility, especially for low bit rate IP channels.

H.263 may be thought as an evolution of H.261 combined with MPEG-like features and others optimizations for very low bit rates. Compared to H.261, H.263 has the following base improvements:

- *More picture formats and different GOB structures.*
- *Half-pel motion compensation.*
- *Improved VLC table.*
- *Four negotiable options.*

The base improvements lead to an average PSNR (Power Signal Noise Ratio) that is 3-4dB better than H.261 at bit-rate lower than 64kbit/s.

### More picture formats

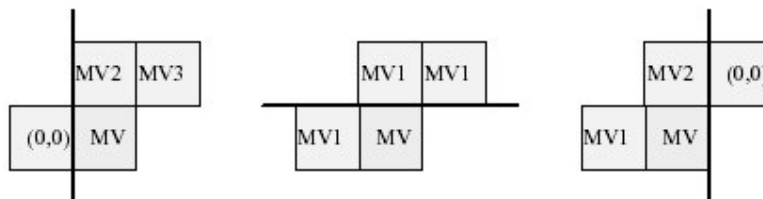
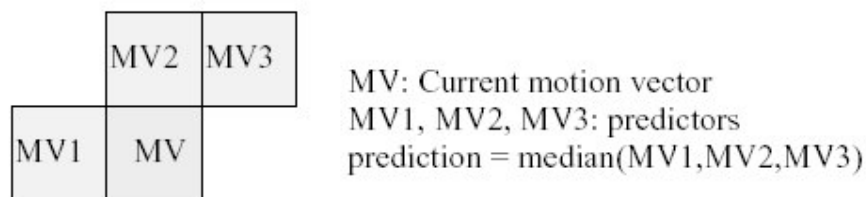
The picture format is the same YCbCr 4:2:0 used in H.261

The allowed resolutions are: Sub-CIF (128x96), QCIF(176x144), 4CIF(704x576) and 16CIF(1408x1152) up to 30-60 Hz (Fps).

### More accurate motion prediction

Resolution of Motion Vectors is now half-pel (half-pixel) in the range [-16,+15.5]

The generic Motion Vector is predicted by the values of the surrounding MV. Infact, in a frame, the motion is a local property and adjacent blocks have similar motion vectors. The predicted value is used to reduce the amount of information trasmitted. Only the difference signal (delta signal) between the real vector and its prediction is transmitted (see Figure).



## Improved VLC Table

Tables containing the Variable Length Code for the Entropy Coding of various parameters (DCT coefficients, quantization levels, motion vectors) have been optimized. This leads to an improvement in bit-stream compression of about 5-7%.

## Negotiable Options in H.263

- *Unrestricted Motion Vector Mode*
- *Advanced Prediction Mode*
- *PB-Frame Mode*
- *Syntax-based Arithmetic Coding Mode*

## Unrestricted Motion Vector (UMV) Mode

- Motion vectors may point outside the picture  
Edge pels are repeated as prediction of the points outside the boundaries of picture. Significant gain for movements along the edge of the pictures, especially for smaller picture formats.
- Extension of the motion vector range  
[-31.5, 31.5] instead of [-16, 15.5]  
Especially useful for 4CIF, 16CIF where motion is usually of wider range.
- Good for camera movement and background motion

## Advanced Prediction Mode (AP)

- Four motion vectors for each MB (one for each 8×8 block)  
The sub-division is useful in areas with compound motion. Use more bits, but give better prediction. Encoder decides which MBs to apply

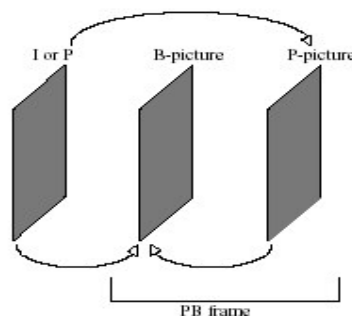
- **Overlapped block motion compensation (OBMC)**

This option reduce blocking artifacts in scenes with strong motion level.

- Motion vectors may point outside as in UMV

## PB-Frame (PB) Mode

- A PB-frame consists of two pictures  
P-picture: Predicted from the last decoded picture as usual  
B-picture: Predicted from both the last valid picture (P or I) and following P-picture (see Figure)  
Bidirection prediction improves compression ratio but introduces latency and the need of a higher processing power. B picture is never referenced by other picture thus it is possible to discard B-pictures in a stream to provide time-scalability and bandwidth scalability.

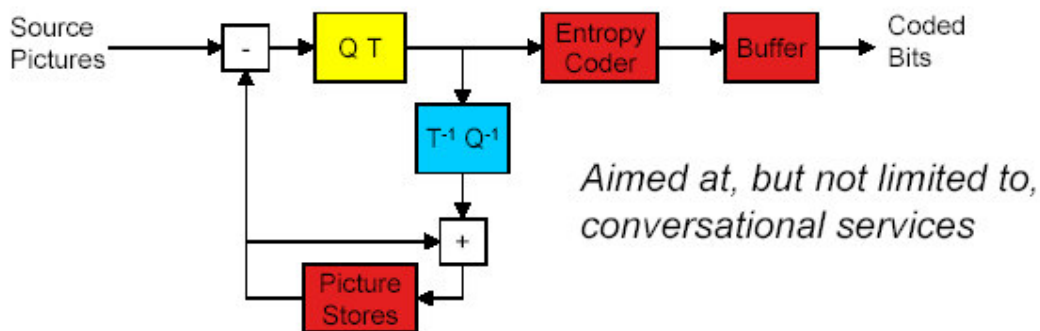


### Syntax-Based Arithmetic Coding (SAC) Mode

- Arithmetic coding is used instead of VLC (Variable Length Coding)  
Instead of using static VLC table, SAC uses Cumulative frequencies calculated both at coder and decoder level.
- SAC gives about 5% less bits at the same SNR
- Inter frames gain: 3~4%
- Intra blocks and frames gain: ~10%

Key features of H.263 and schematic block diagram are here summarized:

## H.263 Version 1 - November 1996



Derived from H.261, MPEG-1 and MPEG-2

- Half Pixel Motion Compensation, 16x16 and 8x8 blocks
- Discrete Cosine Transform
- Motion vectors off picture
- Overlapped block motion compensation
- PB frames

### 1.3 - H.263 Version 2 (H.263v2)

H.263+ is the natural evolution of the base standard. ITU-T developed H.263+ in the years 1996-1998. The basic concepts and techniques of this standard may be found in the later MPEG4 standard.

Enhancements of H.263+ over H.263 are:

- **Extended source formats**
- **16 negotiable optional mode**
- **Supplemental enhancement information**

New Negotiable Options Examples:

### Advanced Intra Coding Mode

The standard introduces a spatial prediction of DCT coefficients in Intra compression. This is similar to Motion Vector prediction but applied to DCT coefficients. There are 3 prediction mode: DC only, vertical DC & AC, horizontal DC & AC.  
10% improvement in Intra compression.

### Alternate Inter VLC Mode

This mode uses separate VLC table for inter DCT and intra DCT. The use of different VLC tables for the various parameters allows better compression at the cost of higher coding complexity.

### Deblocking Filter Mode

Depending on quantization step size the codec applies a deblocking filter to Blocks in order to improve visual quality perception. This is very helpful in the case of very high compression ratio.

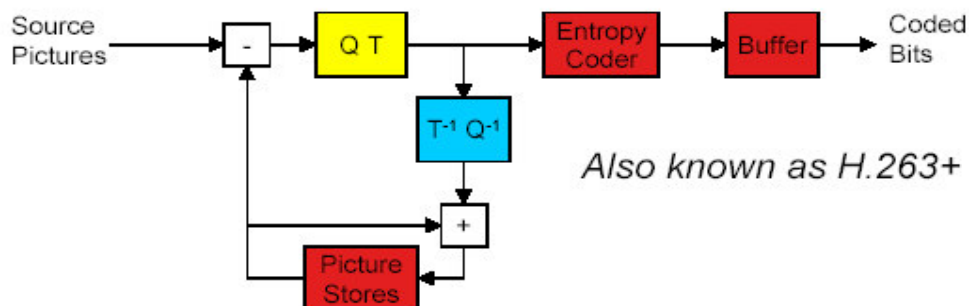
### Modified Quantization Mode

More flexible changes of quantization step sizes and finer quantization for chrominance. The gain in SNR for chrominance levels is considerable. Less chromatic artifacts.

### Improved PB-Frame Mode

For each Macroblock is now possible to choose for Forward, backward, or bi-directional prediction. The motion vectors are predicted from the mean values of the previous and following frame. If prediction is good enough, delta signal is not transmitted at all with a consequent bandwidth gain. Other modes are dedicated to scalability in time, SNR (quality), space, and error resilience. Key features of H.263+ and schematic block diagram are here summarized:

## H.263 Version 2 - February 1998



Further optional modes of operation added:

- Advanced Intra Coding - using spatial prediction
- Deblocking Filter
- Reference Picture Selection
- Scalability and B pictures
- Reference Picture Resampling

## 2 - Technical Analysis of Flash Video Codec

Flash Player 6 and 7 includes Sorenson's Spark, a video codec derived from H.263 standard and licensed by Macromedia. The Codec has a small footprint (approx. 100Kbytes), an important feature for an internet plug-in that must be as light as possible.

Spark supports a subset of H.263 basic standard. In details, it doesn't support arithmetic coding (SAC mode), B frames (PB mode) and Advanced Prediction Mode. These features are probably too much expensive in terms of processing power and codec complexity to be included in a small codec, furthermore B frames are good for pre-recorded video but not for realtime streaming.

B frames introduce latency and double the required processing power. Spark codec replaces B frames with D frames (disposable frames). D Frames are essentially B frames with only forward prediction (a B frame uses both previous and next P or I frame as references, while D frame uses only the previous P or I frame). When the codec uses D frames, it achieves lesser compression but has a method to modulate in realtime the bandwidth required to stream the video because the D frames are "Discartable", like B frames, without the need to resync to the next keyframe. When used as a video compressor to feed a Flash Communication Server, the flash player's spark codec always uses D frames. The video stream appears as a sequence of frames like this:

I-D-P-D-P-D-P-...-D-P-I-D-P....(I=keyframe, D=desposable frame, P=predictive frame)

Spark supports by default the UMV(Unrestricted Motion Vector) mode and admits arbitrary frame dimensions. Furthermore it supports a Deblocking filter mode similar to that of H.263+

With these features, the video codec of Flash player is technologically 20-30% less efficient than a good MPEG4 codec based on H.263+ in terms of quality / bandwidth ratio. (the future version of flash video will fill this gap using a new VP6 based codec licensed from On2).

### 2.1 - Real-time Compression Capabilities

As we have read, the "theoretical" compression capabilities of Flash Video Technology are not the state of the art but are good, expecially if we think of the simplicity, ubiquity and flexibility of using videos embedded in a flash movie.

If you use a dedicated application like Flash Video Importer, Sorenson Squeeze, FlixPro or even FFMPEG to produce a stand alone FLV (the file format of flash video) from a good video source, you are able to obtain a very good result (good quality with a good bandwidth consumption).

A situation completely different is to use a Flash movie to acquire and send a Video stream to a Flash Communication Server for recording or real-time delivery. In this case, the compression is done in real time by the Flash movie itself and the final result, in most cases, may be far from the optimal.

We must consider that video compression is an asymmetric task. Compression is quite complex and time consuming while de-compression is much faster, therefore compression part of the Flash video codec has been simplified to fit in the small plug-in.

For any given codec there may be a great difference in efficiency between different implementation of compression. Compression strategies are various and differ each other in terms of required

processing power, memory and code length. Macromedia programmers have chosen to implement only 3 basic video compression strategies in the Flash Player code to keep the codec footprint and processing power requirement very low.

## 2.2 - Flash Video Compression options

Flash Player takes advantage of essentially 3 video compression strategies. The specific approach is selected by passing different parameters to the `setQuality(Bandwidth,Quality)` method of the Camera Object. We have analysed the behaviour of Flash real-time video encoding using a set of tools. We have used a proprietary **FLV manipulation tool** to know the exact weight and type of each frame in videos recorded by Flash Communication Server, furthermore we have used a proprietary **Video Test Bench** to monitor every parameters during streaming. **Camtasia** is used as screen grab utility and **NetPeeker** as general bandwidth meter and limiter.

### 2.2.1 - Costant Bandwidth

To choose this strategy set `Quality=0` and the desired Bandwidth (`setQuality(Bandwidth,0)`).

The goal is to keep the bandwidth requirement and fps constant varying the block quantization. Starting from the Keyframe, the next frames are compressed as difference with the previous frame. The amount of data dedicated to frame compression is previously calculated and allocated to keep the bandwidth constant. Given a fixed amount of data per frame, the more the frame is different from the previous (high motion) the more the frame will probably be coarse because of insufficient allocation of data. If the frame is similar to the previous, the difference will be more easily coded with that given amount of data.

This approach has an interesting consequence: a progressive sharpening of picture. When frames are very similar (i.e. static scene) the difference signal between previous compressed frame and current frame reduces to zero with a progressive picture quality improvement.

However the codec show a strategy of data allocation between Macroblock of picture that seems to be quite simple and not so efficient. Therefore If scene contain high motion level, the overall quality is low. This is also caused by the forementioned D frames. D Frames can't be used by P frame as reference frame, so the sharpening of picture is lead on only by P frames. In this way, the quality improvement of D frames is, in some way, lost, because it can't propagate to future frames.

Moreover, the sharpening is not "resetted" by a new keyframe. The KeyFrame keeps the same quality as the previous frame. This may be a bad behaviour; Infact this strategy doesn't set a maximum frame quality factor, so, for scene completely static (expecially at low fps as in a screen capture) the frame quality (quantization level of macroblock) reaches the highest levels with the consequence of a very, very heavy keyframe. An heavy Keyframe steals bandwidth to the other frames. If you set a bandwidth of 30KB/s, 10Fps and 1 keyframe every 10 frame, if the keyframe requires 12Kbyte the codec allocates only  $(30-12)/9=2$ Kbytes for the other frames. If motion occurs in the meanwhile, frames appear very coarse.

In some situations the key frame is so big that the algorithm isn't able to keep bandwidth constant and this produce latency, frame rate instability and bandwidth congestion. This effect is more evident hi-res, low fps video as in desktop grabbing.

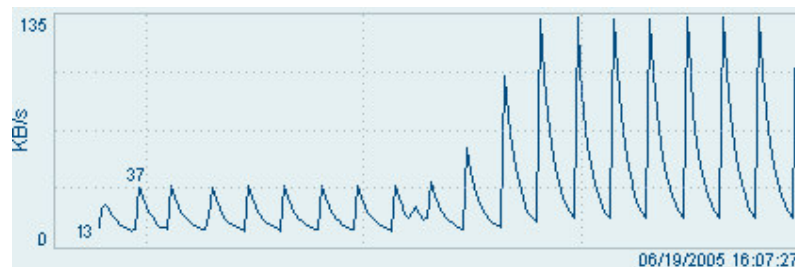
Now we want analyze a peculiar but interesting critical case: a real-time screen grab or document cam video in hi-resolution and low fps. Let's take a look at the bandwidth profiles produced by such situation between the client and the FlashCom server.

**Screen grab or Document cam – static scene – Standard Flash encoding**

*Setting: 800 x 600, 1 fps, 1 Keyframe every 10 frames, 16 KByte/s*

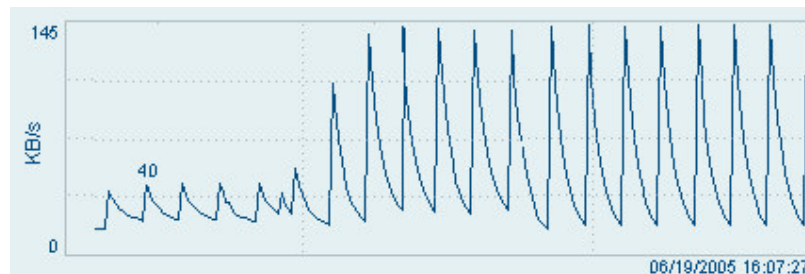
These examples are obtained using Camtasia as screen grab driver. The bandwidth peaks correspond to keyframes (distant each other 10s).

In the first seconds of grabbing, only a low detailed wallpaper is shown. When the bandwidth grow up, I have opened a file explorer window. As you can see, even in the first part the codec is able only to maintain the average bandwidth of 16KB while the keyframe are considerably more 'consistent' (37KB). But the trouble is clear when on screen appears a detailed picture... the keyframe, in a few seconds, explodes to 135KB!



*Setting: 800 x 600, 2 fps, Keyframe every 20 frames, 32 KB/s*

The situation doesn't change. Even with that setting, the keyframe is still enormous and even with more fps (3-5 or even more) the problem still exists.

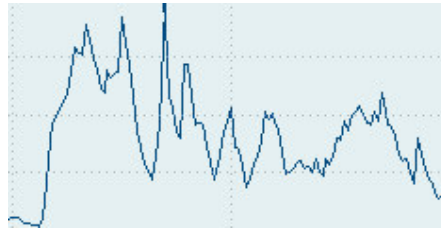


It is clear that the standard constant bandwidth mode isn't usable for screen grabbing and delivery through Flashcom. In part 3 of this whitepaper, we'll show how to improve this compression mode to obtain a good and reliable performance.

**2.2.2 - Constant Quantization**

To choose this strategy set Bandwidth=0 and the desired frame Quality (setQuality(0,Quality)) This method is quite simple: in every frame is applied a static quantization level for every macroblock. This means that picture quality is constant in every frame. The resulting bandwidth is variable with the amount of motion level in scene. Between very similar frame, the difference signal is built only of high frequency coefficients. Depending from the quantization level, that coefficients will be more or less eliminated and not transmitted. For high motion level, the difference signal will contain even low and medium frequency coefficients and the amount of data transmitted to rebuild the frame will be higher.

This is the typical bandwidth profile of this compression method:



As the previous even this method is not perfect. A more complex approach to compression uses a quality range and not a fixed value. To use a range is good to keep the bandwidth low in high motion passages (where is possible to lower frame quality without affecting the video experience) and keep the quality sufficiently high in low motion passages (where is possible to sharpen picture with a low amount of bandwidth).

With this method we cannot know the required bandwidth, that can vary widely. This method keeps fps constant.

### 2.2.3 - Costant Quantization with Bandwidth Limit

This third and last method is exactly the same as the previous with the adding of a Bandwidth limit. The choice of the parameters (bandwidth cap and quality level, together with frame resolution and framerate) are very critic to achieve the best result.

Infact is quite frequent to incur in an unpleasant side effect of bandwidth limit: if a frame is too big to be trasmitted under the specific bandwidth limit, the next frame or frames are dropped. Frame dropping cause motion compensation to be less effective with the result to produce bigger frames and even more bandwidth limit violation. We have analysed the FLV produced by this compression method in a drop frame situation. We have discovered that the codec re-create a Keyframe after each frame drop. This behaviour is another cause of the previously mentioned side effect. So, in this scenario, when the frame dropping begins, it is hard to stop it.

A good video codec uses both quality modulation and frame dropping to keep bandwidth constant. Even with this drawback, this method is the more suitable to a wide range of situation.

In the next Chapter we'll show how to improve the standard behaviour of this compression mode to reach a better exploiting of bandwidth.



## 3 - How to improve Real-time Flash Video compression performance

As a developing team, we have had the problem to stream hi-quality video for Bio-medical and Healthcare customers. For this hi-performance demanding applications, Flashcom gives quite a lot of advantages: first of all flexibility and short time to market, expecially when more sources and destinations of hi-quality stream with interactive interfaces are involved. But the efficiency of Flash real-time video codec become quickly a problem.

In detail, we had to resolve two typical situation: the streaming of low motion, hi resolution video like the one produced by a document cam (for radiography images, laboratory reports, microscopy images, etc...), or by desktop sharing (to show how to use a biomedical software, a pacs, and so on with a lot of picture or even movies in it ).

In this scenario isn't practical to use VNC, or even Breeze, because these applications use gzip and screen area change to compress the desktop, techniques suitable only for normal windows operativities but not for big and hi-quality pictures and totally unusable for videos over the desktop.

Another scenario is the streaming of video from live surgery sessions with better encoding performance than that offered by the standard flash player.

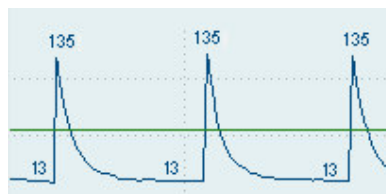
To resolve these problems, my team have developed two coding optimization capable to improve the coding performace for realtime streaming.

At the moment we cannot reveal every details of the code but we want to share with other Flashcom experts the principles of these optimization, and show some results.

### 3.1 - Improving the costant bit rate compression

We have just described the problem of "big keyframe" produced by flash video codec in costant bit rate compression mode. To eliminate this effect and improve overall performance we have created a procedure capable to identify the keyframe and to change on the fly the compression mode from costant to variable bitrate for the keyframe only. We must choose an appropriate value for the quality factor depending by the frame resolution.

Let's consider the standard behavior for a video 800 x 600, 1 fps, 1 Keyframe every 40 frames and 32KByte/s of bandwidth. This is what we obtain in terms of bandwidth between client and Flashcom with a static detailed screen:



With the improved technique and a Keyframe quality of 50 the bandwidth profile became:



The Keyframe is obliged to a standard quality factor, this lower considerably the amount of data dedicated to keyframe, block the uncontrolled raising of quality, and leaves a higher amount of data for the compression of frames between keyframes with a consequent faster progressive sharpening of picture.

The technique is usable only on flash video stream without audio attached to it. In the case of desktop sharing the audio may be presumably attached to another video or left stand alone. In a video stream only, the time property of the net stream, changes only on frame change, so it is possible to count exactly the frame number and find the keyframe. This isn't true for streams with both audio and video attached.

This is the “core” of the code, insert it in a procedure called by an appropriate setInterval

```
K_quality=50; //Keyframe quality, must change in function of frame resolution
bandw=32000; //Stream Bandwidth
Keyframe=40; //Number of frames between keyframes
current_time=out_ns.time;
if (current_time!=old_time){
  frame_count=frame_count+1;
  old_time=current_time;
  resto=math.floor(frame_count-math.floor(frame_count/Keyframe)*Keyframe);
  if (resto==0){
    client_cam.setQuality(bandw, K_quality);}
  else{
    client_cam.setQuality(bandw, 0); }
}
```

We are developing an improved version with automatic Keyframe quality calculation.

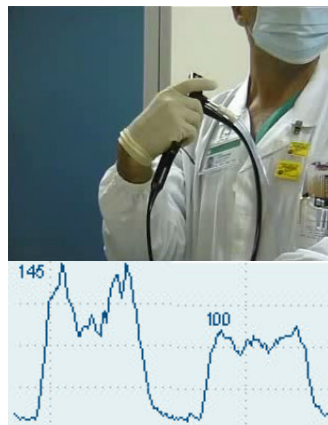
### 3.2 - Improving the variable rate compression

We have seen that variable rate compression without bandwidth cap is too unpredictable to be used and that with bandwidth cap the number of dropped frames may be too high. In an Hi-Quality video streaming scenario isn't acceptable to drop too much frame. Frame dropping must be a rare and extreme measure.

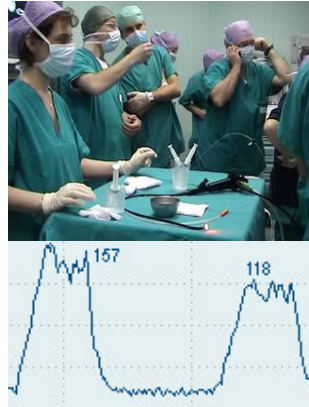
To deliver high quality video with a control of bandwidth we have tried to emulate the behavior of a more complex codec capable of both quality and frame modulation.

The principle of this improved codec is to change the quality setting according to the video motion level. When the scene has high motion levels we lower the frame quality setting, when the scene is quiet we raise the quality setting. The final result is the elimination of bandwidth spikes without frame dropping for a better exploiting of bandwidth.

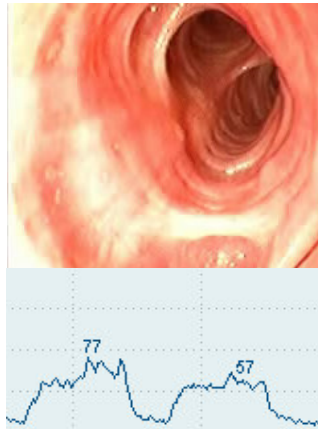
Observe these bandwidth profiles produced by live video streaming from a live surgery session. All video grabs are made in cif resolution (352x288, 25fps).



This camera makes pan & zoom on the surgeon's hands. On the left you find the bandwidth profile of a standard flash video compression with constant frame quality of 90. On the right you find the profile of the optimized codec with variable frame quality in the range 60-90. The gain is -31% of max bandwidth and no dropped frame. (values on figure are in Kbyte/s). This is an high motion scene, the optimization gives good result.



This camera shows the surgeons and assistants (values on figure are in Kbyte/s). This is a mid-low motion scene. The range for variable quality is again 60-90. The gain is -24% of max bandwidth.



This video derives from endoscopic instrumentation. The video is a little blur, so bandwidth requirement is lower than the previous examples. The gain is -25% of max bandwidth. It may be higher with a sharper video source.

The mean gain factor is around 25-30% in this type of videos.

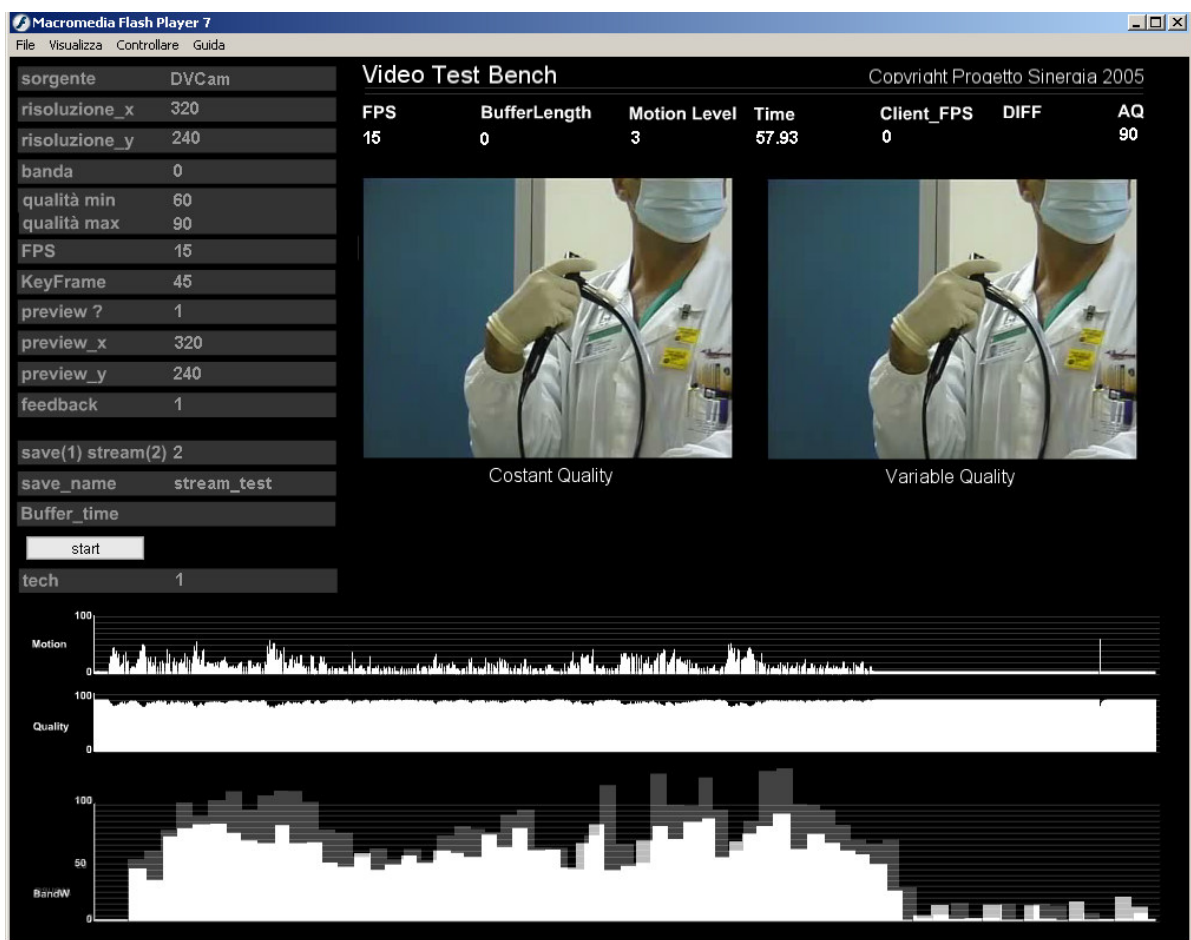
The last example is a classical video conference with limited motion and fixed background. Resolution and frame quality range are the same as the previous. The gain is -40% of max and medium bandwidth.



The easiest way to dynamically change quality may be linear:

```
MaxQ=85; //Max quality – reached on quiet scene  
MinQ=45; //Min quality – reached on high motion scene  
q_actual=((MaxQ)-(MinQ)/100)*(100-client_cam.activityLevel)+MinQ;  
client_cam.setQuality(MaxBandwidth, q_actual);
```

But obviously, it isn't the best. The latest version of our improved codec uses an adaptive approach to achieve a better performance. The algorithm mixes bandwidth and dropped frame control with adaptive quality change to achieve a +30-40% improvement in efficiency over standard Flash coding. Here you can see a picture of our **Video Test Bench**, the tool we use to stream high quality Flash video for our customers.



The graphs represent motion level, applied quality level and resulting bandwidth consumption (the “ghost” graph represent the costant quality result versus the improved variable quality).

To be up to date about other optimization techniques visit our Flash Video optimizations and tools page: <http://flashvideo.progettosinergia.com>