

Encoding Best Practices for H.264 Video Using Flash

Fabio Sonnati

October 5, 2009

Los Angeles, MAX 2009

Twitter TAG: [#adobemax137](#)



Encoding Best Practices for H.264 Video Using Flash

- **Fabio Sonnati** media applications consultant, Flash Community Expert (FMS), FMS developer and beta tester since 2003 with expertise in Video Encoding optimizations. Collaborates with **ValueTeam**, a leading IT consultancy firm, at the development of Video encoding & delivery platforms.

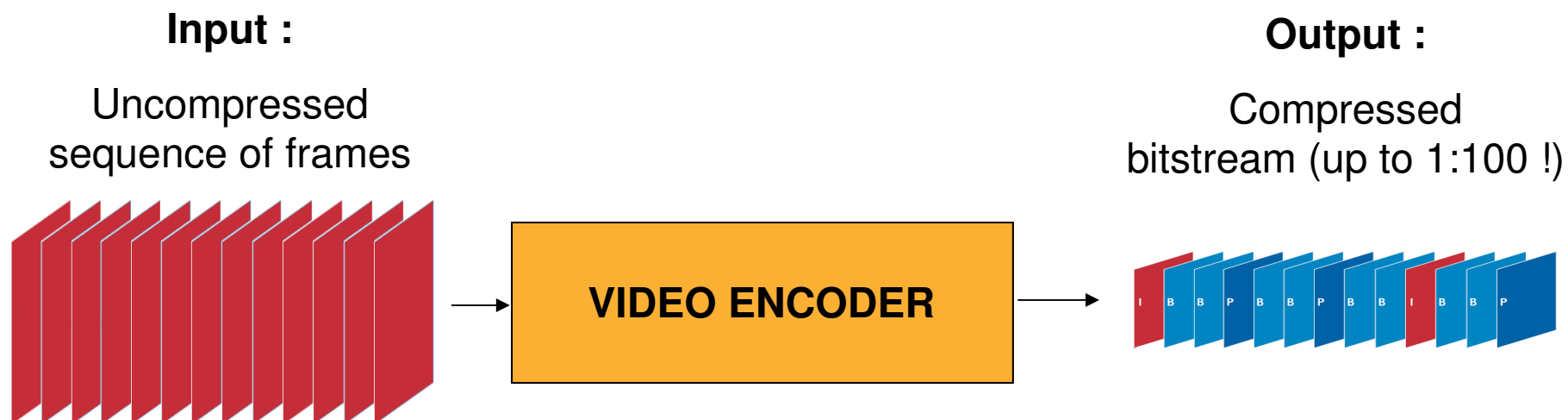
Blog: <http://flashvideo.progettosinergia.com>

Email: sonnati@progettosinergia.com

- **H.264 – Technical Overview**
 - Understanding video compression
 - Video encoding standards overview
 - The H.26x family
- **Encoding Best Practices**
 - Understanding H.264's parameters
 - H.264 encoders
 - Define a content-driven encoding strategy
 - Video prefiltering
- **Delivery Best Practices**
 - Understanding HW acceleration
 - Postfiltering
 - Encoding for Dynamic Streaming
- **Q&A**

Part I – H.264 Technical Overview



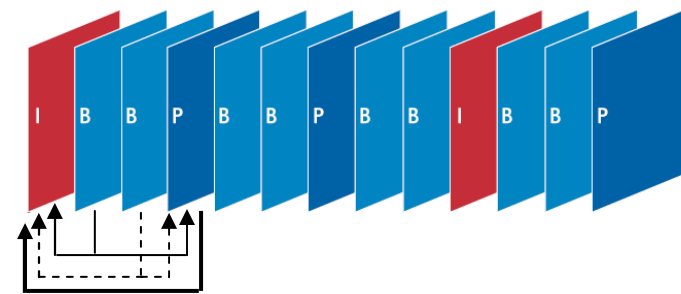


Video encoders compress video data exploiting :

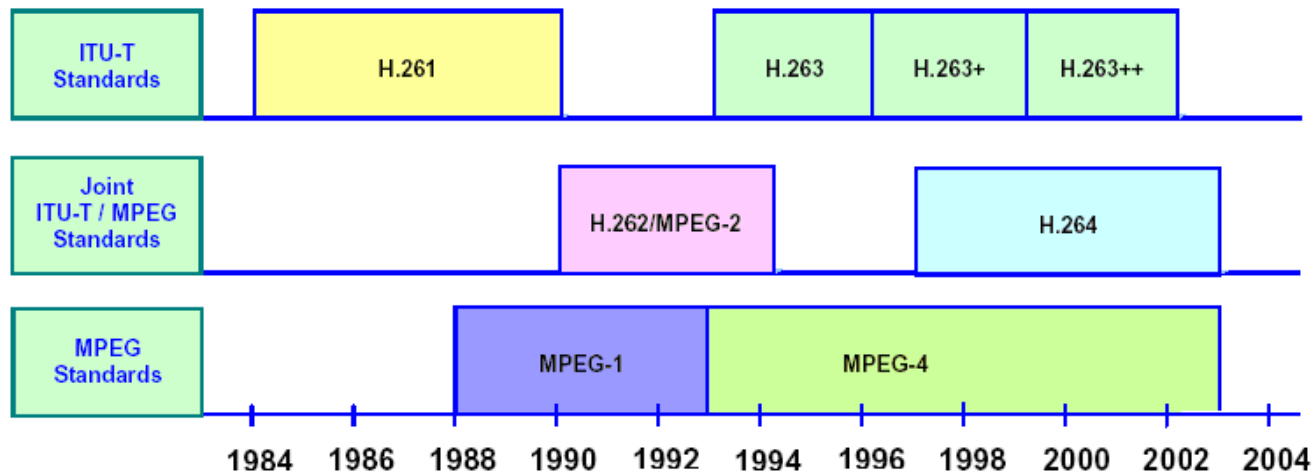
- **Psycho visual models**
- **Spatial & temporal redundancies**
- **Entropy coding (VLC)**

Types of Video Frame :

- I-frames
 - Encoded only spatially (Intra). I-frames (Keyframes) are “self contained” and used for stream accessibility.
- P-frames
 - Predicted from previous reference frames using motion estimation and compensation (Inter). They are not self-contained and form a chain of references.
- B-frames
 - Motion vectors and other data are Bi-directionally Interpolated from previous and next reference frames (Inter). B-frames can achieve the highest compression but must be used properly.



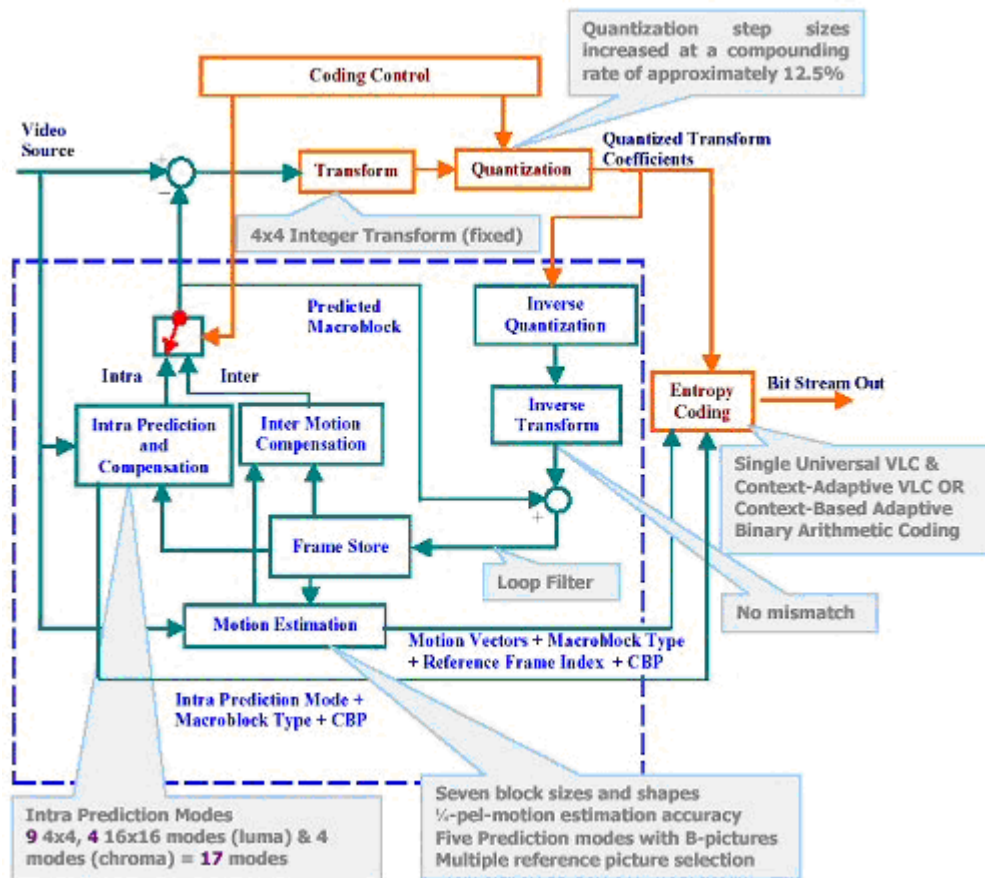
Video encoding standards - overview



- ITU and ISO are the formal organizations for video codec standardization
- ITU develops the “core” logic of the codec (H.26x) for generic applications while ISO defines “industry standards” (MPEG) for storage and broadcasting using H.26x
- In the last two decades ITU and ISO have standardized the most used audio and video encoding standards like MPEG2, MPEG4, H.264

- H.261 (1990) is the grandfather of every video codec.
 - Few compression techniques, very simple motion prediction, base VLC.
- H.263 (1993-1996) is the father of every modern codec.
 - More compression techniques, higher motion prediction and compensation accuracy (half-pixel), longer vectors' range, more efficient VLC.
FP6's Sorenson Spark is derived from H.263.
- H.263v2 and v3 (1998+) represent the technology of MPEG4 ASP
 - Even more complex techniques, motion vector accuracy up to quarter-pixel, Context Adaptive VLC, simplified in-loop deblocking filter, PB-mode.
Divx, VC-1 and VP6 are "MPEG4 ASP generation" codecs.

The H.26x Family – H.264



- **H.264, MPEG-4 Part 10 (or AVC)** was written by the ITU-T together with the ISO/IEC Movie Picture Experts Group (MPEG). The first final drafting work of the standard was completed in May of 2003.
- H.264 contains several new features that allow it to compress video much more effectively than older H.26x standards. Because of the number of options, H.264 is much more complex to exploit than any previous standard.
- H.264 is considered the state of the art in video encoding today and has been widely accepted by the media industry (BD, Satellite and Digital Terrestrial HD, Mobile Video, Camcorder, Internet).

- **New transform design**
- **Several Intra frame prediction modes**
- **Multiple reference frames**
- **Accurate motion compensation**
- **Efficient B-Frames**
- **In-loop deblocking filter**
- **Context-adaptive entropy coding (CABAC)**

Part II – Encoding Best Practices



- The H.264 decoder implemented in Flash Player is very good.
- Supports **baseline**, **main**, **high** profiles and level up to 5.1
- Supports multi-core CPU for decoding (up to 4 cores).
- Supports .mp4 file format as H.264 video container.

- ***Adobe Media Encoder CS4***
- ***Adobe Flash Media Encoder Server***
- ***MainConcept's Reference***
- Sorenson's Squeeze
- On2's Flix
- Apple's QuickTime
- Nero Digital's H.264 encoder
- nVidia Elemental Accelerator for Premiere

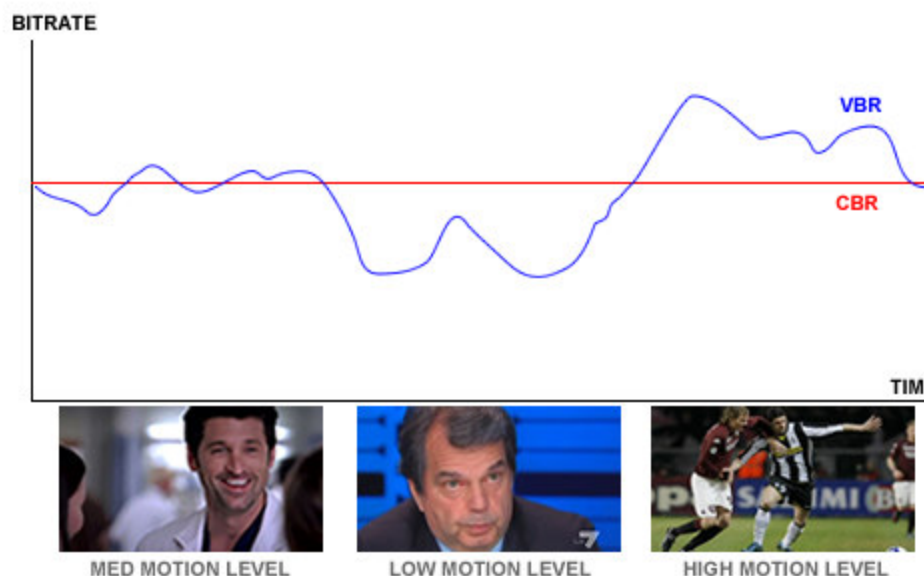
Bitrate related options. The dilemma: CBR or VBR ?

Constant Bitrate (CBR):

An “exact” bitrate is used for the entire video. Useful for FMS streaming, especially for “Dynamic Streaming”. Not optimal for quality because the level of compression varies naturally with the level of motion in the video.

Variable Bitrate (VBR):

An “average” bitrate is used. More bits are allocated in fast moving scenes and less bits in static scenes. Difficult to handle in streaming because of the spikes in bandwidth. Good for progressive downloading.

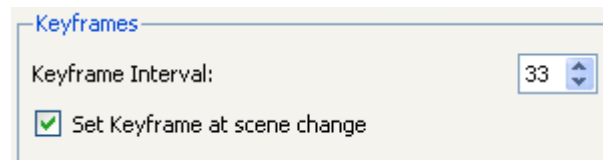
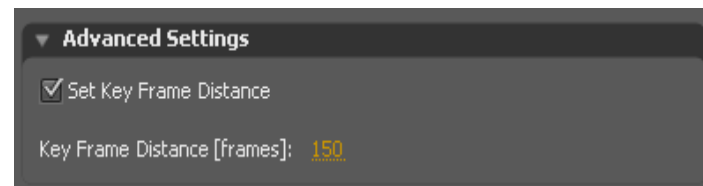


Multi-pass encoding

- When available use 2-pass encoding because of the better distribution of bits.

■ IDR Interval

- It is the distance between keyframes. The value depends by the level of accessibility you want to give to your media. Two consecutive IDR frames isolate a so called Group-of-pictures (GOP).
- Recommended range 60-300 (better if “Dynamic” and guided by a scene change threshold).

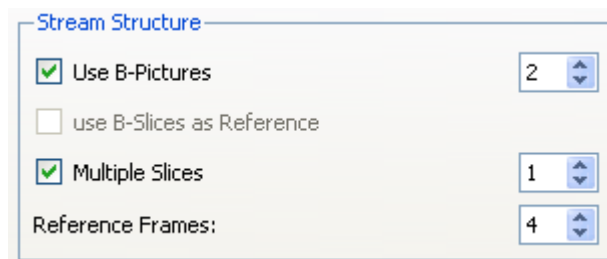


- **Max number of B-Frames**

- Always activate B-frames. An high number of B-frames is more useful in static scenes. The max number of consecutive B-frames can be in the range 1-16. Some encoder does not allow a value higher than 3. Recommended values: 3 to 5.

- **Dynamic B-frames**

- If the encoder supports it, enable an “automatic decision” for the number of B-frames really used.

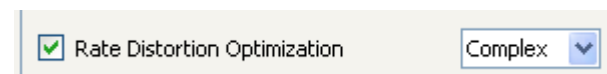


- **B-pyramid**

- This option allows the encoder to use **B-frames as reference** if needed. Enable it if available.

- **Rate Distortion Optimization**

- Optimizes estimation choices to enhance PSNR. Depending by the specific implementation, can be very slow and save only a few % of quality and/or bits.



- **Number of reference frames**

H.264 supports up to 16 reference frames. An high number of reference frames may elevate too much encoding times. Over 4-5 references, there are very little improvements. Recommended 2-5.

- **Motion estimation and compensation**

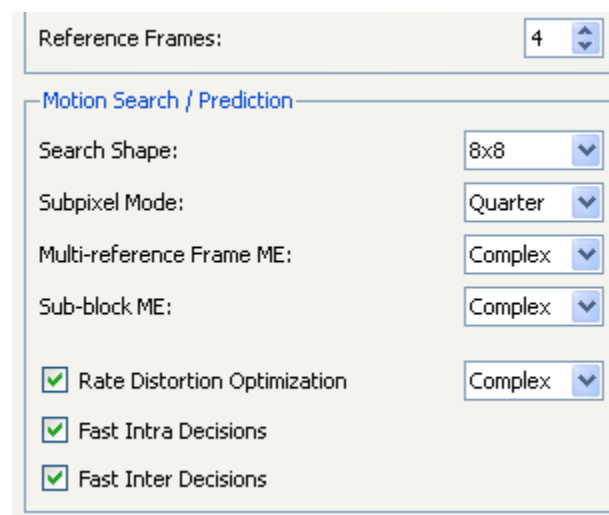
Use *bigger macroblocks size* for fast encoding or for high resolution frame. Use smaller *macroblocks* for accurate encoding or at low resolution.

- **Search modes**

The best search mode in motion estimation is “exhaustive”, but it’s really too slow. Hexagonal search strategies are usually balanced.

- **Search accuracy**

Set quarter-pel accuracy for final encoding and half-pel for fast tests.



- **De-blocking**

The complex in-loop de-blocking filter is one of the major cause of the efficiency of H.264
Never disable de-blocking. I suggest to use the standard values for Alpha and Beta parameters which control the threshold and the strength of the de-blocking filter. If you like a more “crisp” look try -1,-1. If you like a more “smooth” look try 1,1.

- **Entropy Coding**

- **CABAC**

- **Context-Adaptive Binary Arithmetic Coding** is always the best choice for quality sake.

- **CAVLC**

- Requires less processing power but reduce the quality of around 7-10%. Using CAVLC instead of CABAC can save only a limited amount of processing power in decoding (10-15%) because of the optimized H.264 decoder implemented in the Flash Player.

- If encoding time is an issue, which features can be disabled to speed up consistently encoding without affecting too much quality?
 - Reduce **reference frames** number to 2
 - Reduce **search range** (i.e. “simple” instead of “complex”)
 - Increase **macroblock** size (8x8 -> 16x16)
 - Disable **rate distortion optimizations** (RDO)
 - Consider **reducing resolution** and **pre-filtering**

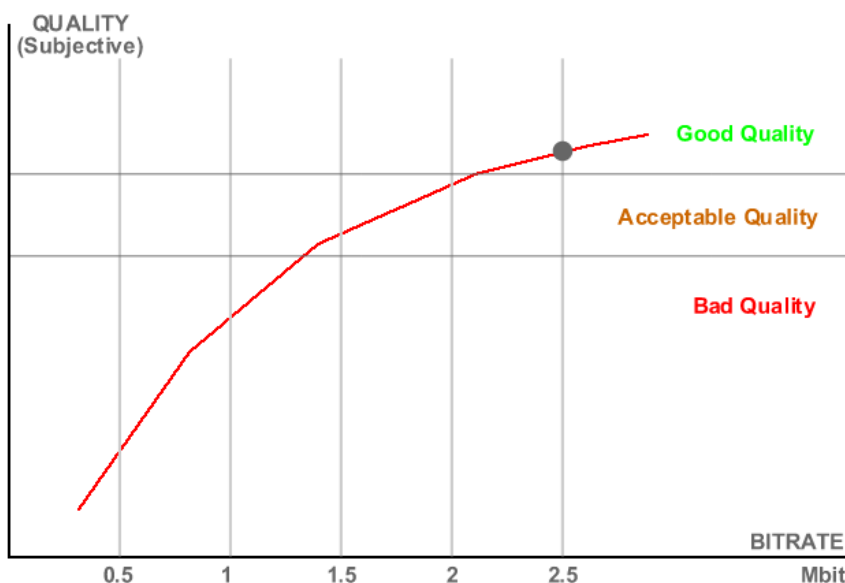
How to maximize the “video-experience” ?

- **Choose the best resolution-bitrate mix for your video**
- **Pre-process source video**
- **Optimize video playback in the player**
- **Use FMS’s dynamic streaming**

Choose the best resolution-bitrate mix for your video

Resolution and bitrate are chosen appropriately to meet the following requirements :

- “Good” video quality
- Bitrate as low as possible to reach the maximum possible number of users.



Example of rate distortion curve for a generic 720p sample

Choose the best resolution-bitrate mix for your video

As a general recommendation choose resolution values multiple of 8/16.
Example:

Aspect ratio 16:9

1920x1080 (1080p)

1280x720 (720p)

1024x576 (576p)

848x480 (480p)

640x360 (360p)

Aspect ratio 4:3

720x576 (576p anamorphic PAL
on a PC needs to be stretched to 768x576)

720x480 (480p anamorphic NTSC
on a PC needs to be stretched to 640x480)

480x360 (360p)

320x240 (240p)

Note: If video has a very wide aspect ratio (like cinema) with black bars above and below, cut off the bar and encode at the original wide resolution to save bits.

Choose the best resolution-bitrate mix for your video

- The level of motion, amount of details and of noise in the original video sequence (video complexity) determine the level of compression that H.264 can achieve.

A very complex clip can require 2-3x the bitrate of a static clip. Therefore, trying to accomodate all the cases with a single “static” resolution-bitrate mix results in an high average bitrate (the bitrate needed to assures high quality in the high complexity case).

- Example of Static Resolutions–Bitrate mix:



HD (720p) @ 3Mbit/s

HQ (480p) @ 1.5Mbit/s



HD (720p) @ 2.5Mbit/s

HQ (480p) @ 1Mbit/s

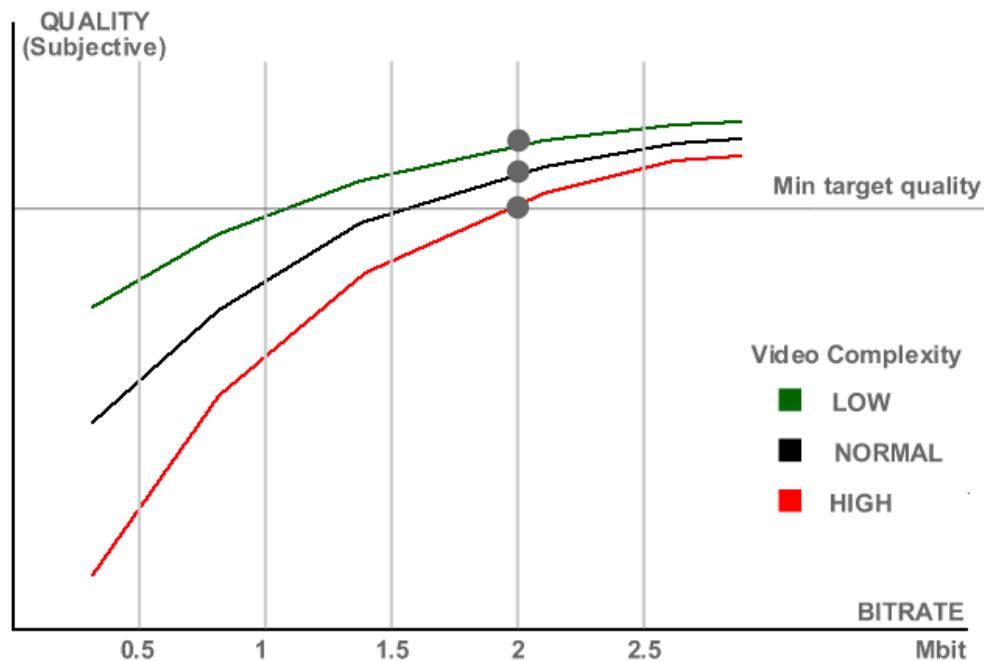


HD (720p) @ 2Mbit/s

HQ (360p) @ 0.9Mbit/s

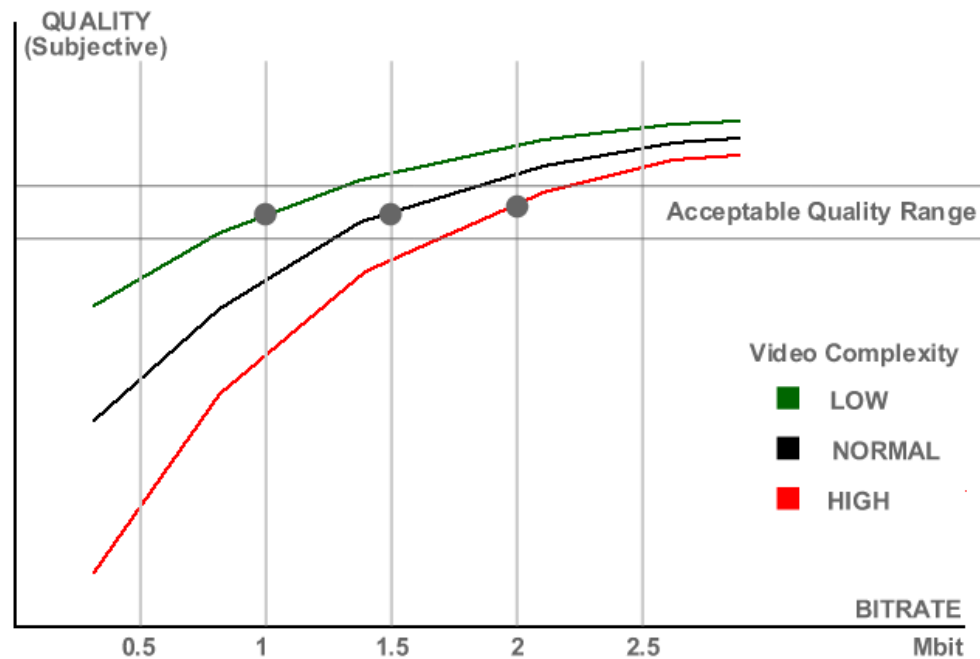
Static resolution-bitrate mix

Example of a static resolution-bitrate mix: 720p @ 2Mbit/s
For every complexity, the same resolution and bitrate is used. Quality varies in a range. The highest complexity video must satisfy a minimum quality requirement.



Context adaptive resolution-bitrate mix (1)

In this example of dynamic resolution-bitrate mix, the bitrate is chosen according to the video complexity and the quality is constant. The total bandwidth usage and the servers' load is optimized.



- How to reduce the bitrate required by complex clips?
 - **Encoding in anamorphic resolutions you can save 20% of bitrate with a negligible loss in quality.**

The video is encoded at a lower width and interpolated back in the player after the decoding.

1920x1080 > 1440x1080 (- 25% pixels)

1280x720 > 1024x720 (- 20% pixels)

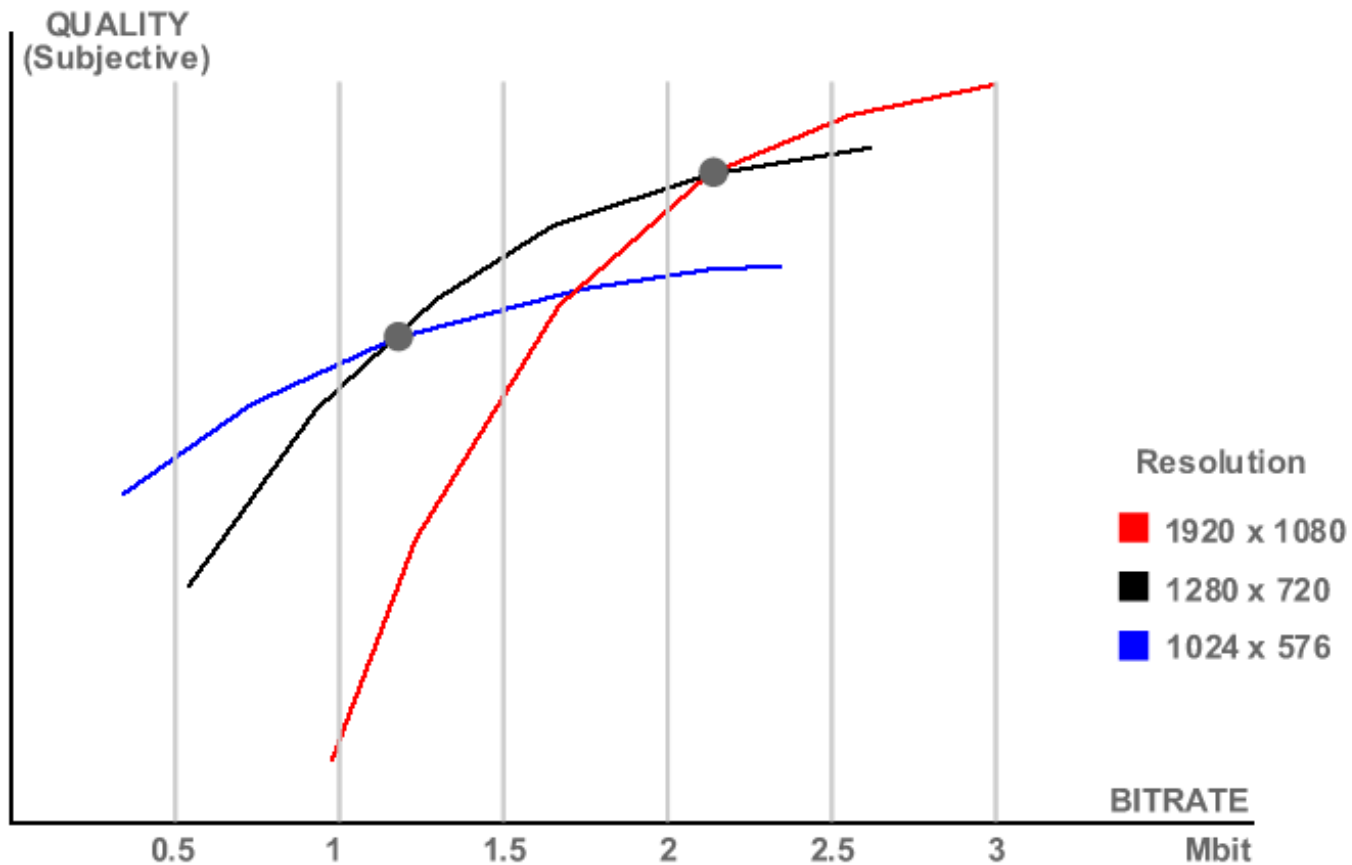
Anamorphic video remains FullHD or HD compliant.

- **Note: The perceived loss in quality caused by the use of a lower resolution is minor than the perceived loss in quality caused by a too much high quantization (level of compression) used to fit the desired bitrate.**

Resolution-Bitrate Mix



Example of rate distortion curves at different resolutions.
Note the “tipping points” that suggest to change resolution.



Context adaptive resolution-bitrate mix (2)



Standard Complexity (tv series, news) :
use a target bitrate and resolution (-> target quality)
(i.e. 720p @ 1.5 Mbit/s)



Low Complexity (talk show, interviews) :
1. save bitrate with the same quality (720p @ 1Mbit/s)
2. same bitrate maximize quality (720p @ 1.5Mbit/s)



High Complexity (sports, action movies) :
1. higher bitrate with the same quality (720p @ 2Mbit/s)
2. same bitrate, anemorphic, acceptable quality (720p @ 1.5Mbit/s)

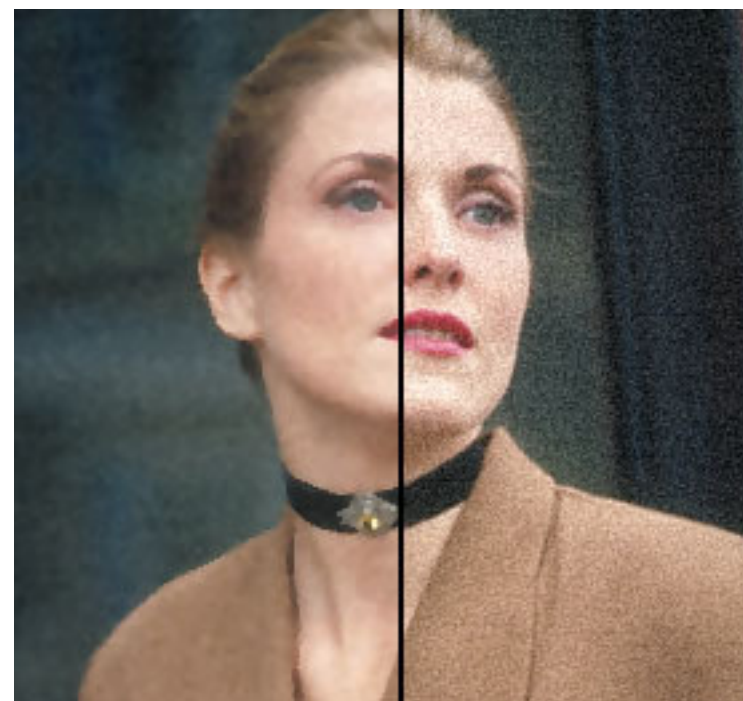
Video source pre-processing

Pre-process accurately the source is very important to maximise the encoding efficiency. The main objective is to reduce video noise, adjust color levels and de-interlace.

- **Video noise** is very frequently present in video footages (due to low lights, film grain, poor cmos sensors). It lowers the efficiency of encoding because the encoder try to recostruct the noise as a very fine detail.

It is very important to reduce video noise with proper filters. The best are the “temporal denoise filters” or “3D denoise filters”.

Note: *resizing (bilinear or bicubic) to a lower output resolution acts as a denoising filter.*



Denoised

Original

Video source pre-processing

- **Interlacing:** SD sources are very often interlaced, and 1080i sources are interlaced. It is very important to use the most professional deinterlacing routine (*motion compensated adaptive deinterlace routines*) to preserve frame resolution and detail because most de-int filters (wave, bob) simply cut a field or blend two fields producing a sensible loss in quality and detail.

AlgoSuite for After Effects or Premiere do the job.



interlaced



deinterlaced

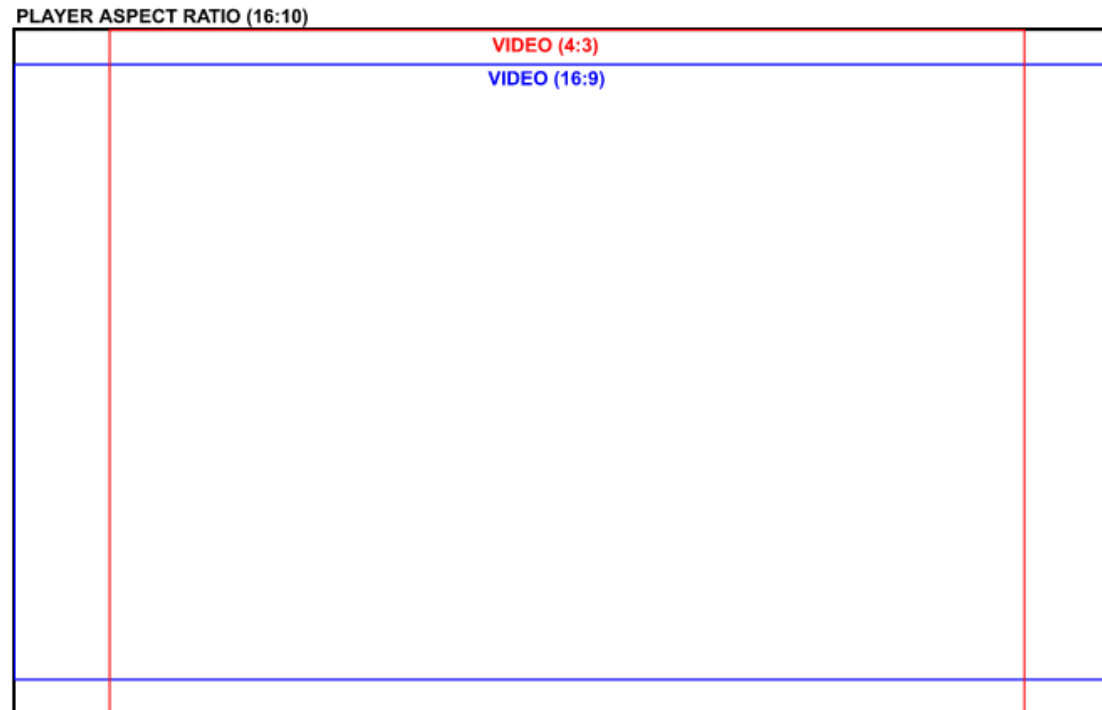
Part III – Delivery Best Practices



How to optimize video playback



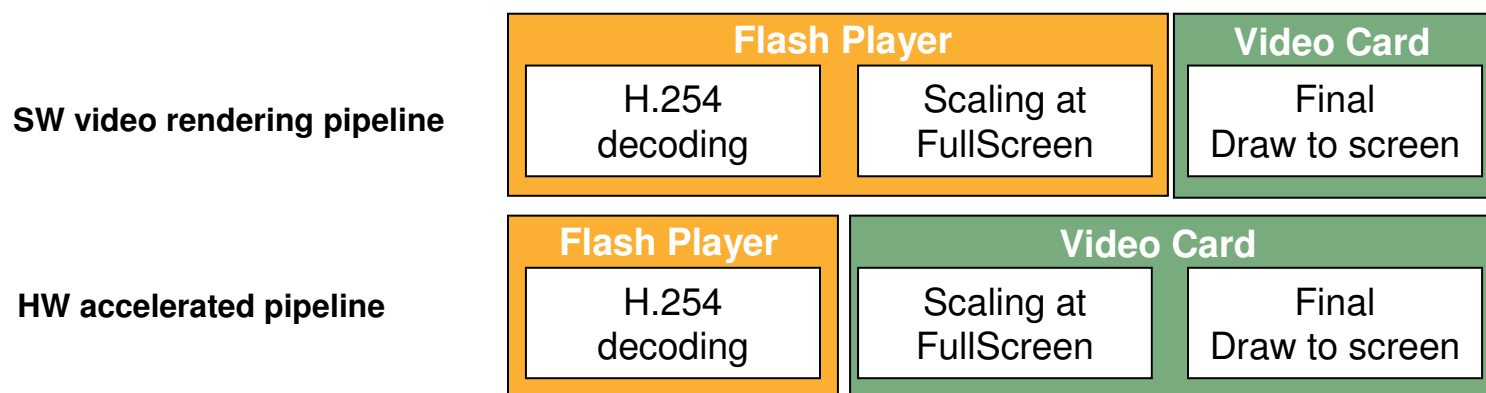
- As a general recommendation, set a 16:10 aspect ratio for the player. This is optimal for displaying both 16:9 and 4:3 video.
- Use the standard overlay method for placing the SWF in the HTML page. Decoding video is very complex and need the more straightforward path to the screen.



Understanding HW acceleration



- Flash Player 9.0.115 introduced the support for hardware acceleration of video scaling at fullscreen.



- To properly enable HW scaling, use the **fullScreenSourceRect** stage's property

```
Stage.fullScreenSourceRect = new Rectangle(0, 0, video.width, video.height);  
Stage.displayState = "fullScreen";
```

- HW scaling can smooth player interface, but I recommend to use it especially for HD video to speed up video playback on older computer.
- Use always **video.smoothing=true** when not at fullscreen, disable it at fullscreen. The only exception is when pixel aspect ratio (PAR) of video is not 1:1 (for example when using anemorphic resolutions).

- The encoding process progressively removes details. This can be done in a clever way, so that the viewer may not recognize the loss of such details. But at very low bitrate, this loss can be strong.

Post-filtering can help to “reconstruct” lost details.

- We can use simple (built-in convolution matrix) or very complex filters (pixel bender based). A common “sharpen filter” can be a good start.

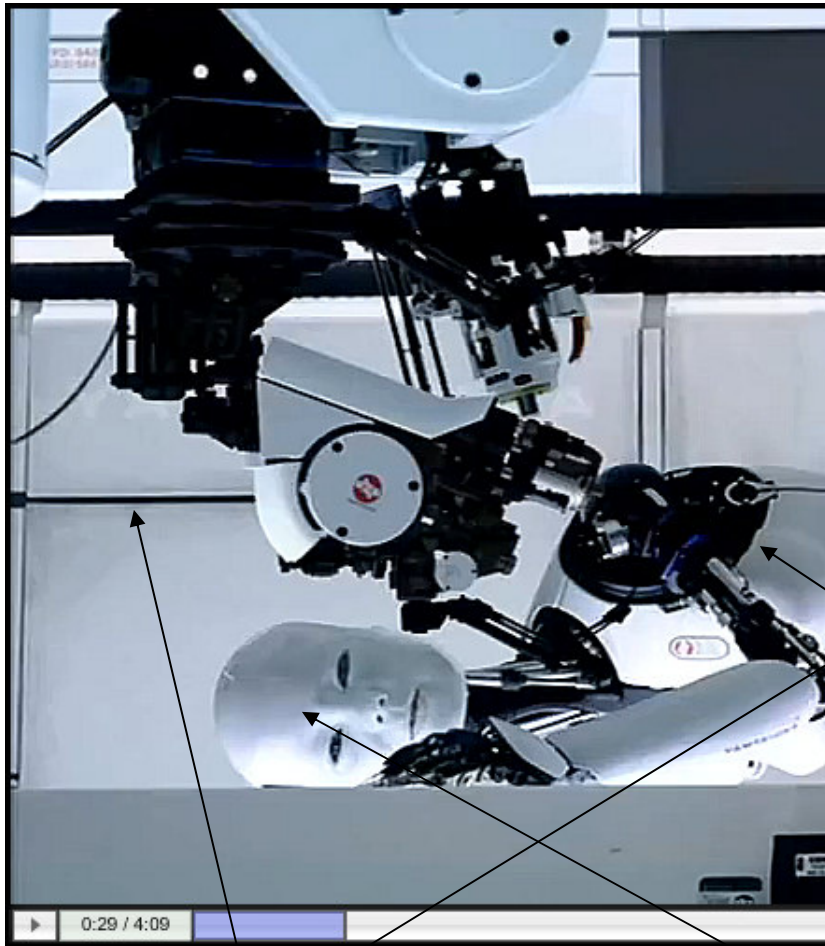
```
var filter;  
filter = new flash.filters.ConvolutionFilter();  
filter.matrixX = 3; filter.matrixY = 3;  
filter.matrix = [-1, -1, -1, -1, 12, -1, -1, -1, -1];  
filter.bias = 0;  
filter.divisor = 4;  
video_mc.filters=[filter];
```

- It’s important to understand that this is only a “perceptual” restore and is very CPU intensive. Use QoS Api (FP10) to adaptively use the enhancement.

Post-filtering

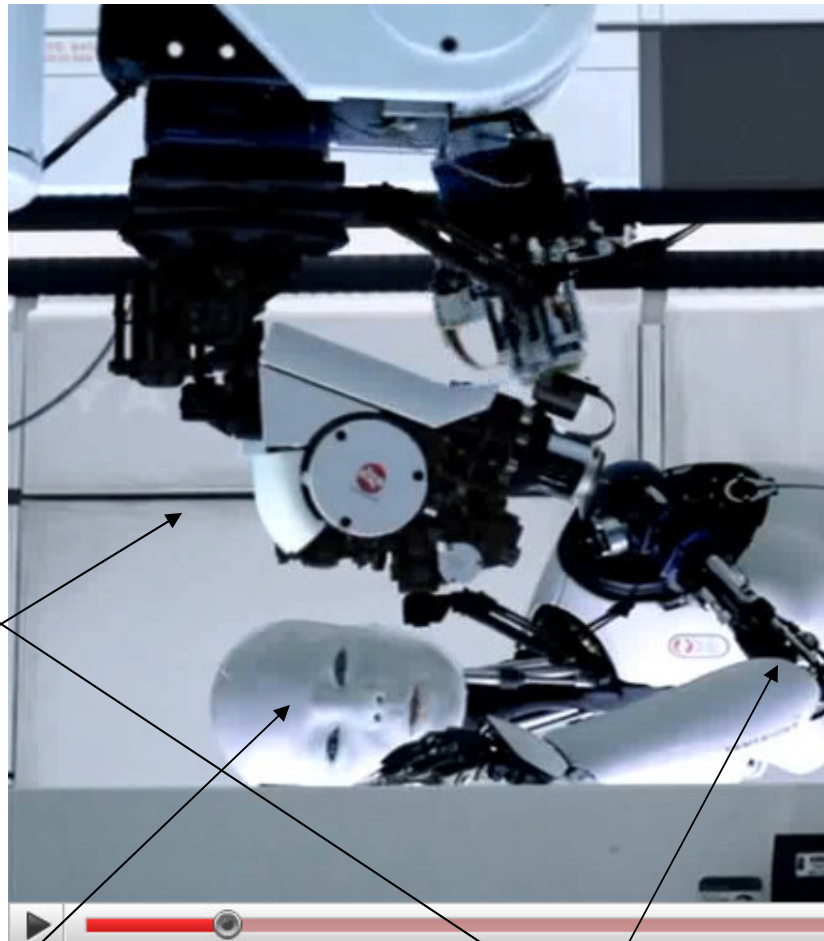


720p @ 500Kbit/s



More contrast

720p @ 2Mbit/s



More texturing

More detail

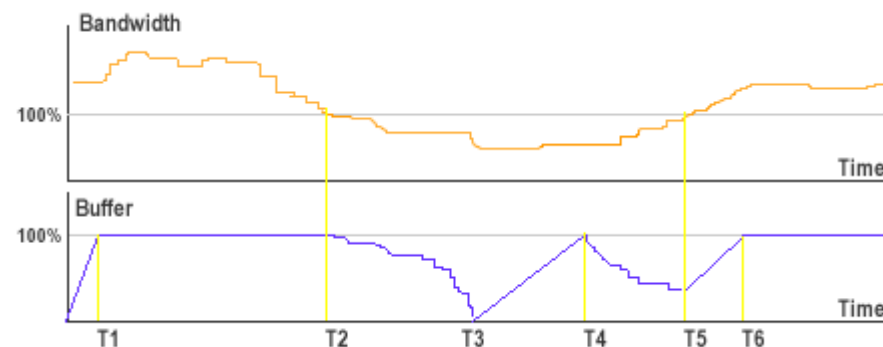
How to improve user experience and QoS ?

- Dynamic Buffering (dual threshold buffering)

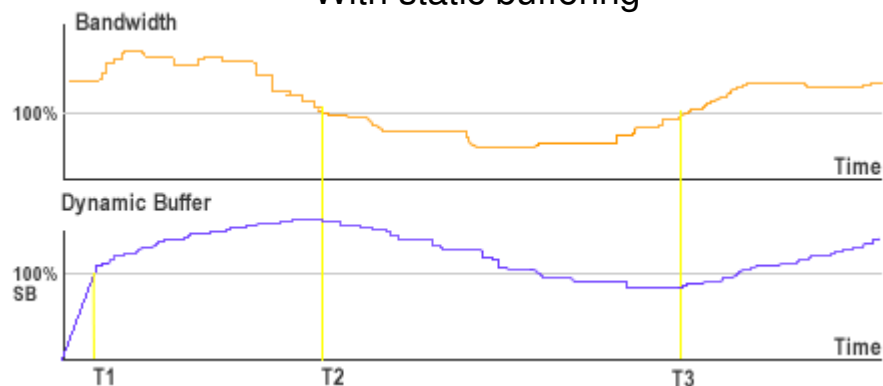
This should be a well-known “trick”. (I explained it on the DevNet in 2006).

Using a dynamic buffer instead of a static one, you can widely enhance resilience to bandwidth fluctuations.

Use a moderate start buffer (3-5 seconds) and when it's full expand to 20-30 seconds.



With static buffering



With dynamic buffering

For more info go to:

http://www.adobe.com/devnet/flashmediaserver/articles/fms_dual_buffering.html

How to improve user experience and QoS ?

- **Dynamic Streaming**

FP10 and FMS 3.5 can provide a superior user experience switching between different bitrates depending by the network condition of each user.

FP10 offers a new QoS Api which gives detailed information about the stream, the bandwidth and the video decoding process. With such info it is possible to choose the best stream to assure the highest QoS in that moment.

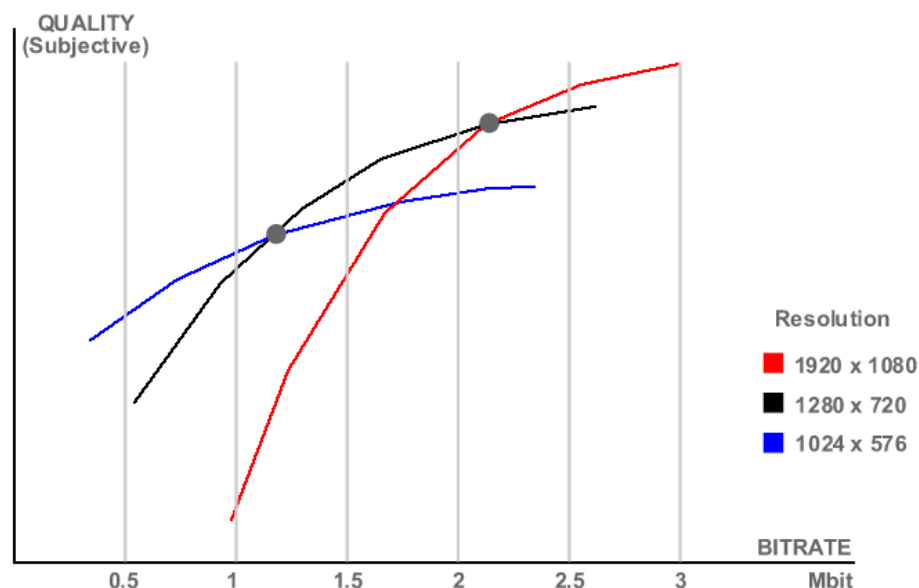
The **DynamicStream class** provided by Adobe implements all the heuristic for the bitrate switching process for vod and live streaming. It implements also the Dynamic Streaming strategy. You can also use **OSMF**.

Optimizing encoding for Multi-bitrate

- Encode all the streams with a fixed **gop** size (suggested 3-4 seconds).
- Encode all the streams in CBR or with a very “light” VBR.
- Encode all the streams with the same audio settings.
- Use few, balanced bitrates :

Es:

720p @ 1.5 Mbit/s
576p @ 1 Mbit/s
480p @ 700 Kbit/s
360p @ 450 Kbit/s



Q & A



CONNECT. DISCOVER. INSPIRE.

Fabio Sonnati

sonnati@progettosingia.com

<http://flashvideo.progettosingia.com>