SwissQual
A Rohde & Schwarz Company

Testing and Benchmarking of Video Services

Dr. Jens Berger

August 2012

SwissQual AG
Allmendweg 8   CH-4528 Zuchwil   Switzerland
t +41 32 686 65 65   **f** +41 32 686 65 66   **e** info@swissqual.com
www.swissqual.com

Part Number: 12-070-200912-4

Contents

Figures

1 Video Services in Cellular and Cable Networks

With the continued evolution of global data networks, consumers have taken full advantage of the exponentially increasing bandwidth available to them. As bandwidth has increased, the ability to deliver high quality video is significantly better. Combined with the rise in popularity of video delivery services such as NetFlix, Hulu, iTunes, etc, the transition of video delivery from current cable and satellite TV formats to Internet Protocol (IP) is well under way. In fact, Video is rapidly becoming the most prevalent IP traffic generator through the internet. As expected, initial consumer use of this video delivery mechanism has been limited to cable modems and high quality DSL connections. With the continued evolution of wireless air interface technologies, digital video transmission is no longer limited to cable and satellite networks. Efficient compression techniques and ever increasing bandwidth makes transmission of video the fastest growing data content in cellular networks. There is no end in sight to the insatiable uptake of available bandwidth; Cisco Systems predicts that 66% of all mobile data will be used for delivery of video by 2014.

The latest data technologies offer higher and higher bandwidths in mobile or quasi-stationary connections using cellular networks. This allows an extended set of high quality multi-media applications in mobile environments such as teleconferencing, video-on-demand and live TV services. As a result, typical TV contents and resolutions are used in mobile applications as well. TV becomes part of a multi-media application set in mobile or PC-based setups.

The difference to common terrestrial, satellite or cable TV isn’t the content or the resolution or bandwidth, it is mainly the transport stream and the used protocols and container formats only.

Compared to telephony, the classical application in cellular networks, where in principle a real-time peer-to-peer connection is required, video services are uni-cast or multicast connections. Usually, a server provides a (compressed) video stream that can be requested by several clients.

This paper discusses challenges for operators and networks in delivering a high quality and consistently available video service and ways to measure, to prove and to optimize it.

2 Streaming vs. Progressive Download

Progressive Download

Video services over IP come in different flavors. One application is Video on Demand using services such as Netflix or YouTube. Here a video clip is completely pre-stored on a server and is accessed and played back by the users client player. Clients can access that video at any time and get the video stream 'delivered'. With YouTube and other services the user can even select the image resolution he or she would like to receive. Some provider servers even take into account information about the available capacity of the transfer channel or quality problems at the user's side and adjust the bit stream bandwidth dynamically.

Often, particularly with PC clients and commercial Video on Demand (VOD) services that are included in most internet-ready TVs and DVD players, a considerable part or even the entire video is downloaded into a buffer or temporary file before being played back. In these cases the transfer rate from the server to the player is higher than the bitrate of the video stream. That way, the player can start playing the video while the entire video is downloaded. This is called progressive download. This functionality has to be enabled by the player and the server. The player must manage the buffer or temporary file to allow for smooth playback of video if there are temporary interruptions in available bandwidth and the server has to allow a higher transfer rate than the playback bitrate of the video. Often the video is downloaded well before the stream is completely played back. If the transmission is interrupted, there is data in the buffer to bridge the gap and there is a high chance that the buffer will receive new data again before it runs empty.

Progressive download is usually applied via HTTP over TCP. It is a 'safe protocol', i.e. no erroneous packets will arrive at the player. The only problem could be an interruption of the IP transfer, leading to a re-buffering and potentially a freezing of the video in case the buffer can't be re-filled in time.

Streaming Video

The second application is real-time streaming. It applies to all videos that are not completely pre-stored, i.e. live video or live TV. Here a progressive download isn't possible without a considerable delay to the real-time content. This is usually not acceptable for sports transmissions or TV news. However, to bridge short interruptions in the transmission, a short buffer of a few seconds is typically used. This is just a so-called jitter buffer; it compensates for jitter in the receiving time of packets. Some players even use a dynamic buffer size that is slightly adjusted using the history of packet jitter analyzed by the player. For this type of video transmission RTP (Real-time Transport Protocol) over UDP is the preferred transmission method used. RTP is specially designed for real time streaming transmissions but is not a safe protocol as HTTP over TCP. UDP can be imagined as 'fire and forget'; there is no confirmation of reception of a packet or even a re-transmission procedure in place. Here the player may receive erroneous packets and needs to apply individual concealment strategies.

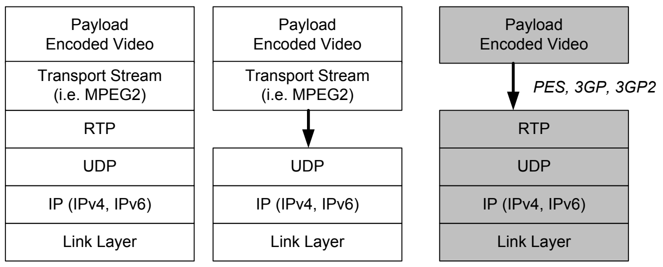


Figure 1: Packet layer structure for typical streaming protocols

We should consider that the above scheme shows a simplified structure. There is for example a parallel session description and session control structure.

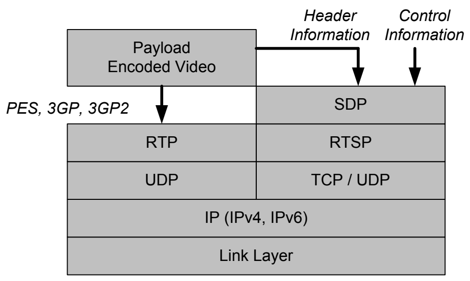


Figure 2: Packet layer structure demonstrating session description and control

Usually, video players running on mobile phones are not able to build up huge buffers. Therefore today's mobile players access video, real time video as well as video on demand via RTP and do not utilize progressive download.

An example that serves both applications is YouTube. YouTube is a typical provider of Video on Demand. In case of a common use of YouTube via PC, Flash Player is used to play back the video using its own container format over HTTP. Here a typical progressive download can be observed. The video download starts at first and playback begins after a few seconds of video has been buffered. The further download and the actual play back positions can be easily observed in the YouTube service right below the video clip by a moving bar.

Mobile phones accessing YouTube do not use Flash Player and they don't buffer a considerable part of video as required for progressive download. Many mobile phones have Real Player integrated, the iPhone uses Apple's QuickTime, and some devices use PacketVideo. For those clients YouTube provides the video content in 3GP format and enables streaming via RTP.

How does it work? YouTube re-processes each uploaded video into different bitrates, resolutions and container formats, such as Flash format and 3GP. The individual re-processed videos are assigned individual URLs that are accessed depending on the client. YouTube is reached via an HTTP entry page. At this stage the client provides information about its operating system and the used video player. Based on this information, YouTube provides the link that fits to the requesting player i.e. 3GP streams for mobile players.

YouTube also considers requests by the client regarding desired image size or simple quality demands, e.g. low, medium or high. Based on this the served bitrate is selected. If the user chooses a lower resolution or a lower quality, he or she will be served lower rate. The quality will be reduced but this enables faster (or ‘safer’) transfer and requires lower amounts of data to be transferred. The experience today shows that even by choosing the best quality in a mobile environment, the bitrate served by YouTube is limited to approximately 300 kbps.

3 Basic introduction in video codecs and compression

Since the explosion of video delivered via digital networks, most people have been exposed to a confusing array of program names, standards, and terms. For the purposes of this section of the document, we will focus on the introduction of the term "codec" as it applies to video. Prior to diving into a discussion of codecs and their purpose, a brief discussion about digital video is necessary.

Video is essentially a series of two dimensional images played in rapid succession. Each image is represented by a matrix of individual small colored dots. If we look at a typical high definition movie, we know that playback is typically 24 frames per second at 1080p. A 1080p image, by definition, is 1920 pixels wide by 1080 pixels tall. One other element for video display is color depth. A Blue Ray DVD player will play in 24 bit color (which allows for up to 256 shades for each Red, Green, and Blue colors per pixel). What does all this mean? The math is pretty simple: 1920x1080x24 = 49.77 Mbit in total for one frame of a movie. Multiplying this out for a 2 hour HD movie and suddenly you need almost 1200 Mbps which equates to about 1 Terabyte of information displayed during the entire movie! Keep in mind there is continued pressure to display more frames per second and more color depth, forcing the need for higher bandwidth. This is a much bigger dilemma than exists with audio. With speech/audio we have a continuous sound event that is really nothing more than air pressure changes over time. This audio signal can easily be transferred into an analogue electrical signal. A typical CD plays 2 channel audio at a rate of 1.4 Mbps, which equates to approximately 1.2 GB for that same 2 hour movie. Fortunately, the human eyes and ears are not so perfect and through the use of various techniques, digital audio and video can be compressed for better bandwidth utilization during playback. With compression comes loss of data, and fortunately the human eyes are much less sensitive to this loss of data than the ears. The challenge then becomes developing encoders and complimenting decoders (hence the term codec) to stuff as much information into as few bits and bytes possible.

Video Content: Color Space

One area to target for reduction of video file size is color. As mentioned above, the representation of an image as matrix of colored dots requires a huge amount of data. Usually, the color information of a pixel is described in Red, Green and Blue (RGB) in a so-called absolute color space. The actual color space itself depends of the definition of the individual color range. There are only 8 color bits per channel used in RGB24, a very widely used approach for color video. Each dot or pixel requires 24 bits (8 bits each for Red, Green, and Blue) however, there are professional image formats supporting even 14 bits per channel that would lead to more than 40 bits per pixel. Over time, the pressure for higher quality video will continue, with richer colors leading to higher bandwidth needs in order to transport the video digitally. When watching a video, the human eye is much more sensitive to brightness than it is to color. Therefore, bandwidth can be optimized by devoting more bits to the 'luma' component than to color components. To accomplish this, the first step is to convert the absolute color space RGB information into a format that can separate the luma (brightness) and chroma (color). The resulting video becomes encoded into the YC_bC_r format, which is no longer an absolute color space. For YC_bC_r, the Luminance (Y) value and two differential Chrominance (C_bC_r) values are derived from the RGB absolute color space (see *Figure 1*).

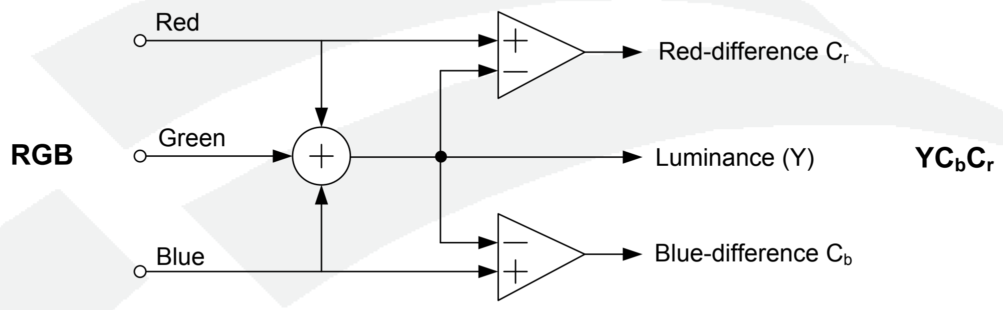


Figure 3: Conversion between RGB and YC_bC_r

The Y-channel transports most of the information. It can be imagined as a grey scale conversion of the RGB image. The two chrominance channels carry less information compared to Y. However, the complete RGB matrix can be re-produced by the full YC_bC_r information.

The full YC_bC_r format is called YC_bC_r 4:4:4. This is the full counterpart to RGB. In this case each pixel is described by the luminance and the chrominance information and a lossless conversion between the planes is possible. The 4:4:4 represents for every 4 pixels of luminance (Y) 4 pixels of blue difference chrominance (C_b) and 4 pixels of red-difference chrominance (C_r) are also included in the data transmission.

In consumer and even in higher quality video coding or storing methods, the chrominance information can be reduced since the human eye is less sensitive to color information than to brightness. In fact, less chrominance information can be included without visible impairments. This means that each pixel can get a luminance value, but you can actually drop some pixels of chrominance and a typical viewer would not see a difference in video quality. The chrominance value is 'bundled' for a small area of pixels. This so-called 'chroma sub-sampling' simply omits the C_bC_r information for a percentage of pixels while the Y (luminance) information is always included. There are two forms of chroma sub-sampling used in digital videos today. The first technique used is called YC_bC_r 4:2:2 format. The second is called YC_bC_r 4:2:0. For the 4:2:2 version, for every 4 pixels, the Y value is included in all, whilst the C_bC_r values are included in half. For the 4:2:0 version, the Y value is included in all pixels, and the C_bC_r values are included in every 2nd pixel of the first line of the video but they are excluded from every pixel in the 2nd line of the video. Essentially every other line of the video is completely missing the C_bC_r values (see Figure 4).

Even studio standards use this technique called YC_bC_r in 4:2:2 formats, and indication that from a human perspective dropping half the chrominance information is generally not perceivable by the human eye. Digital video and other consumer devices even use YC_bC_r in 4:2:0 formats, which reduce the color information by a substantial 75%. See Figure 2 for a graphical representation of the 4:2:2 and 4:2:0 YC_bC_r formats.

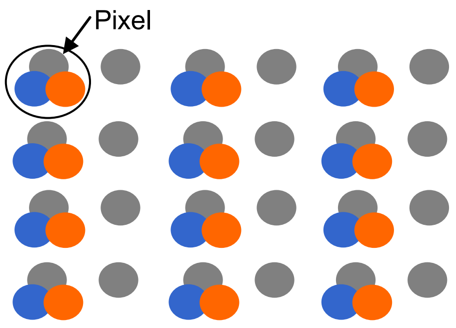
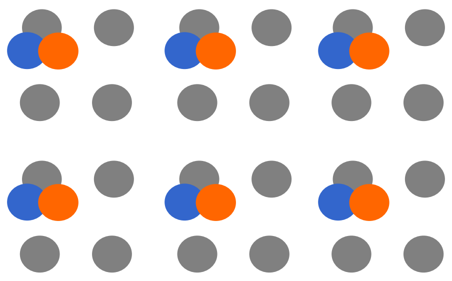


Figure 4: Scheme for YC_bC_r in 4:2:2



Scheme for YC_bC_r in 4:2:0

Compared to RGB 24bit these formats reduce the total amount of data to 66% or 50% of its original size respectively. Component formatted signals usually form the input to video codecs. Going back to our 2 hour HD Blue Ray movie, utilizing YC_bC_r 4:2:0 can reduce the video size from 1 TByte to 500 GByte, which is still far too large a file for efficient transmission.

Video Content: Transformation of Images

In addition to the reduction of data from managing color bits as described above, there are simple ideas of loss-less compression of the individual images as run-length encoding or similar. Those compressions may reduce by a factor – depending on complexity – between 1 and maybe 3. The frame capture below can be used to illustrate the concept behind transformation of images. In this image, the small black box has been expanded 800%. In this expanded image, you can clearly see that the palm has been altered. In this alteration it can be seen that a relatively large block of information is being described with only a few bits of data. The color of the sky, however, looks unaltered. In truth, the entire image has been altered. The sky, since its color and luminance is consistent over a large area is described with very few bits of data and it is not perceivable. A well designed algorithm can determine the minimal change over a large area and subsequently describe this area with fewer bits of information. As image complexity goes up, the ability to dramatically reduce the bits to describe that image goes down. For reference, the image below if 640 x 480 pixels. Mathematically, this image could be as large as 900KB. This image is actually 256KB, the result of a simple compression.

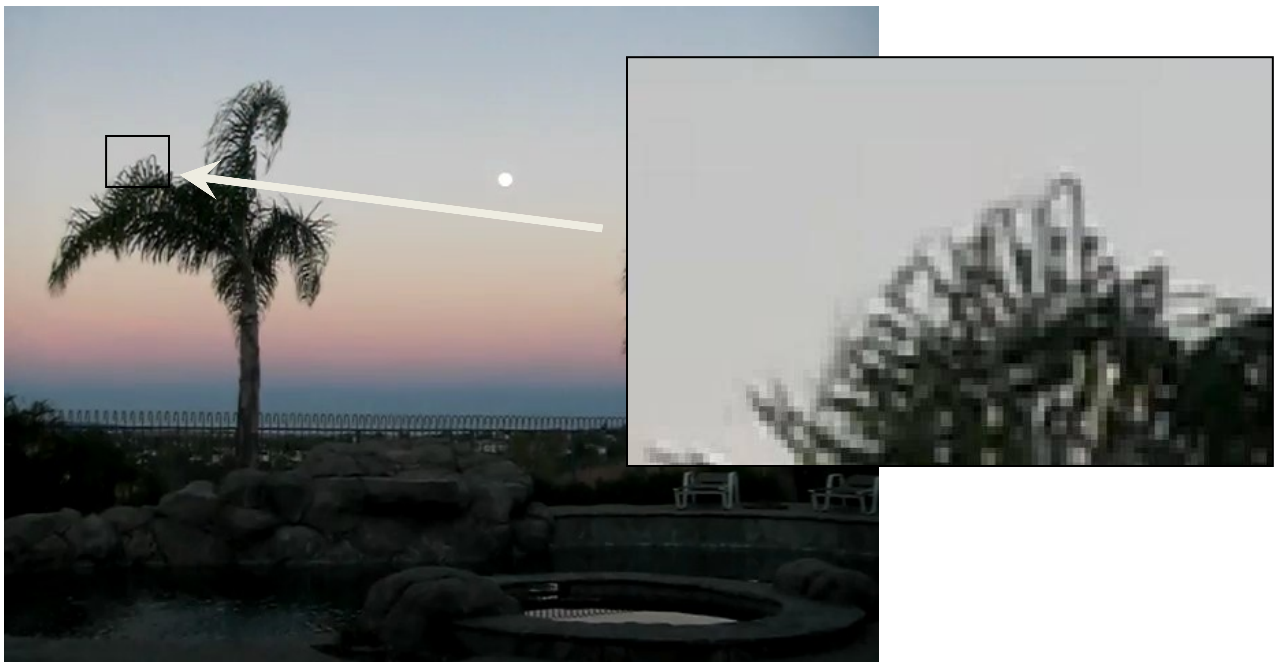
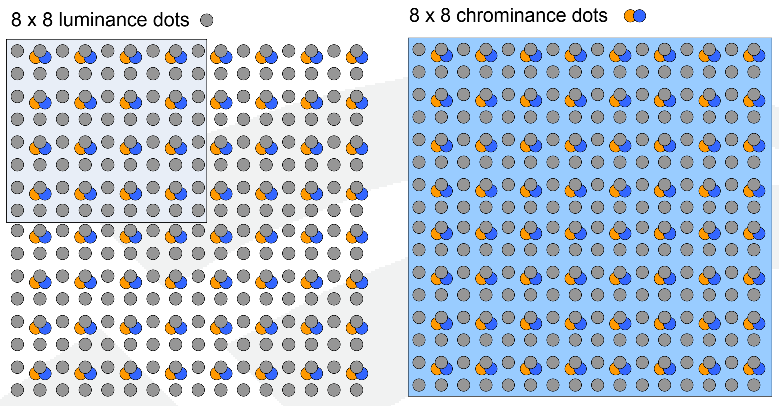


Figure 5: Example of areas with different spatial complexity

The principle idea of all established video compression algorithms, for example H.261, H.263, MPEG4-part 2 and even H.264 is the sub-division of the bitmap matrix in squares (often referred to as “blocks”). The following principles are basics already used in H.261 and H.263. Newer technologies such as H.264 or advanced profiles in MPEG4-part2 follow the same principles but enabling much more scalability and flexibility regarding block sizes, block orders and redundancy detection. All example images in the following section were encoded by MPEG4 in simple profile.

If we imagine the YC_bC_r 4:2:0 input signal as usual for simple profile video compression, an 8 x 8 square of the Y (luminance) information represents actually a square of 8 x 8 pixels. The luminance information is available for each pixel. This is a so-called micro-block.[1]

In YC_bC_r 4:2:0 the two chrominance values (C_b and C_r) are only available for a small area of 2 x 2 pixels; an 8 x 8 information block represents therefore 16 x 16 pixels in the image. This is a so-called macro-block.[2]



[1] Note: The term micro-block is not used with H.264 / AVC anymore. There are only macro-blocks defined.

[2] Especially H.264 / AVC codecs allows a flexible border size and can process input formats with higher chrominance density up to 4:4:4. There are different profiles for different coding complexity. H.264 is also known as MPEG4 part 10.

Figure 6: Common micro- and macro block structures in YC_bC_r 4:2:0

All data reduction happens in these blocks. Therefore, coding distortions are visible as a loss of detail inside the block and visible block borders. The following picture illustrates the micro- and macro-block structure on a heavily compressed image.

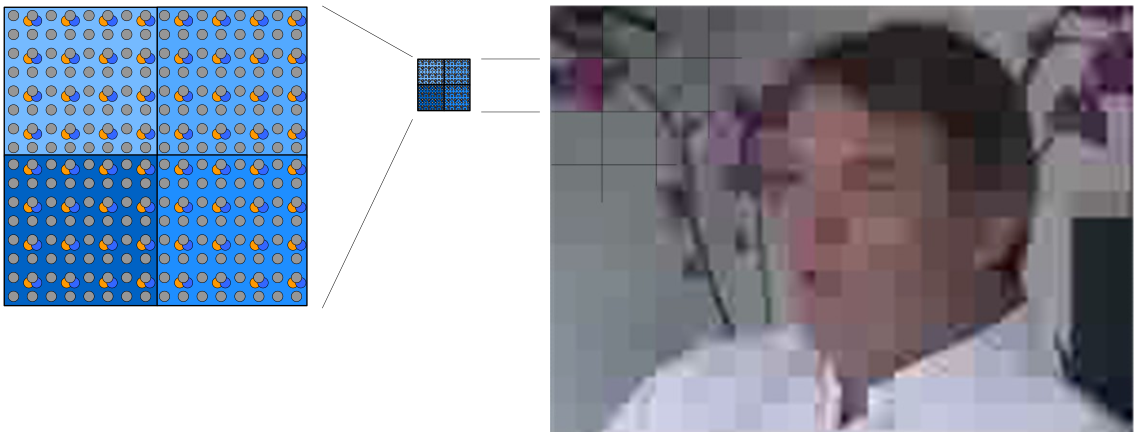


Figure 7: Micro- and macro block structures in an example image

The image conversion makes use of a quasi two-dimensional Discrete Cosine Transformation (DCT) of the 8 x 8 matrices. The bins of the DCT describe the luminance or chrominance changes from pixel to pixel in a kind of frequency domain. The used transformation reduces complexity by being restricted to 2^n number of sample points, similar to a Fast Fourier Transformation. The outcome of the DCT can be imagined as an 8 x 8 matrix as well, where the component describing the lowest frequency component is in the left upper edge while the highest component is in the lower right corner.

Let's explain the idea on the conversion of the luminance information: The cosine transformation converts the sequence of the pixel luminance in the block into the frequency domain, as a spectral analysis of the luminance changes. The f_0 component for the luminance for example describes the average luminance for the entire micro-block. A component that can be imagined as f_1 represents one transition in the block and so on. If all 64 bins or DCT components are available, an equivalent inverse transformation can reproduce the luminance matrix of the micro-block exactly.

The idea is now to focus on the most relevant information in the block. Based on the desired compression ratio and the spatial complexity of the block, only the dominant 'frequency' bins are selected and used for describing the entire block. The squared DCT coefficient matrix becomes sampled in a kind of a diagonal zig-zag sequence from the left upper corner (lowest frequency component) down to the right side (highest frequency component), as very often the high frequency components are close to zero. In case starting from a certain point only zero components or components with a low value (threshold depends on the targeted bitrate) are found, the coding of the block ends at this pint with a unique EOB (end of block) marker. The block is considered as described sufficiently at this point. In case there are sequences of zeros (or low values) in the matrix these 'zero sequences' just become run-length encoded by a value ('zero-run-length'). This allows the efficient coding of i.e. the base component f_0 along with a few or even single high frequency components without getting all values in between in the coded block.

In the example image there are many micro-blocks, where only the overall luminance (quasi the f_0 component) is used (see left part of the example figure). This happens mostly in parts without spatial complexity or demanding texture. There are other examples too, where at least one or two gradients are used to describe the texture in a simplified manner (examples on right side).

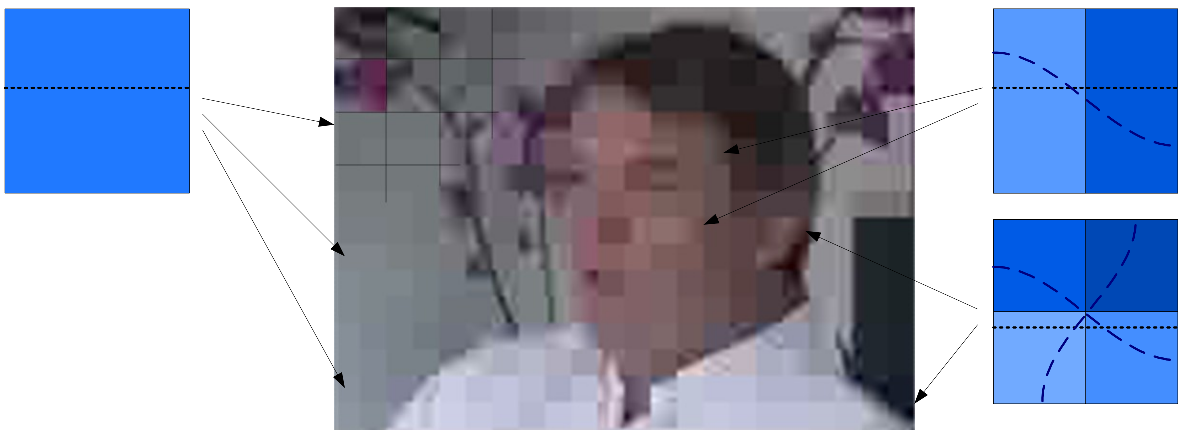


Figure 8: Examples of micro-block coding

In case of more complex patterns, it could be even more efficient just to transmit one or two high frequency bins without the lower spectral components. Examples are at the collar of the shirt in the image.

The same principle is applied to the chrominance information. Gradients in chrominance are also represented by DCT bins in the C_r and C_b domains. By observing the example image, it is well visible that the luminance changes in between the smaller micro-blocks while the color information remains the same over the larger macro-block. In case of less transmitted information, the entire macro-block has the same ‘base color’, only the luminance of the micro-blocks change.

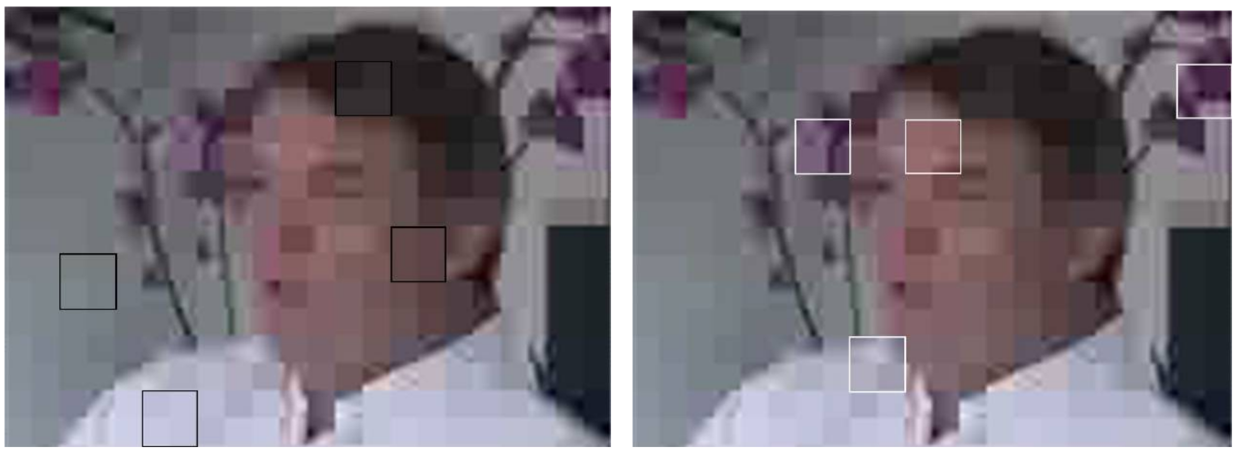


Figure 9: Examples of macro-block coding

The black surrounded squares in the left image of Figure 9 illustrate very simple and highly compressed information. The entire macro-block has the same ‘color’, i.e. grey or skin tone and only the luminance of the micro-blocks change. That means instead of 16 x 16 x 8 bits luminance plus two times 8 x 8 x 8 bits for the two chrominance values (yields 3072 bit net) only the f_0 components of Y, C_r and C_b were transmitted (using a quantization of 8 bits each it would just yield 24 bits!).

The white surrounded squares in the right image of Figure 9 illustrate a more detailed image part. Although the base color doesn’t change in the macro-block, more detailed luminance textures are visible. Here just a few higher frequency bins of the luminance DCT are transmitted and used.

Neither the number of bins per block nor the order of the bins is fixed. During the compression, the coding algorithm decides how many and which dominant bins per block are transmitted to get the perceptually best impression of the image considering the desired bitrate.

The most important entity in this kind of compression is the macro-block. All information in the image is grouped by macro-blocks. A macro-block can be decoded ‘in itself’. There are further grouping information such as 'group of blocks', where a sequence of macro-blocks is bundled and getting an extra header and sync information. This grouping costs some bits but it allows a re-synchronization at the next group in case of destroyed and un-decodable information in the image.

Video as a series of images

Now that we've addressed opportunities for data compression in the bits transmitted (recall that YC_bC_r 4:2:0 gave us a 50% reduction in bits transmitted) and with image transformation, it is time to look at the opportunities associated with motion. Indeed, video, as described at the start of this document, is a series of images played back in rapid succession. Significant reduction in data is possible, for instance, if a video has a sequence of a few seconds where there is hardly any motion. It is possible to just repeat data that was already transmitted in this event. This section of the document will focus on the details behind compression associated with frame to frame changes in video.

If all information from a frame of video is available, the entire image can be drawn – of course with coding artifacts. Images or frames drawn from a complete set of data are called I-frames or key-frames. Although the compression is efficient; the required amount of data for a high quality image remains quite high. As implied above, the next image in sequence can be significantly reduced in size by only transmitting the data that has changed from the prior image. Therefore, with video compression algorithms used today, only differential information is transmitted that refers to a complete image, an I-frame. Such differential information to the previous image is referred to as a P-frame. In principle, P-frames only update information for each macro-block using two methods:, first, differences in luminance and chrominance, and second, the movement of a macro-block if necessary. That way, each macro-block can be adapted in luminance and color to the changes in the next image and can be moved (or better: re-drawn at another place) as well. There are cases, where a difference indication becomes insufficient for describing the changes; in those cases a complete macro-block information as in a I-frame can be transmitted within a P-frame as well. That applies even to entire groups of macro-blocks. That is often called 'partial update'.

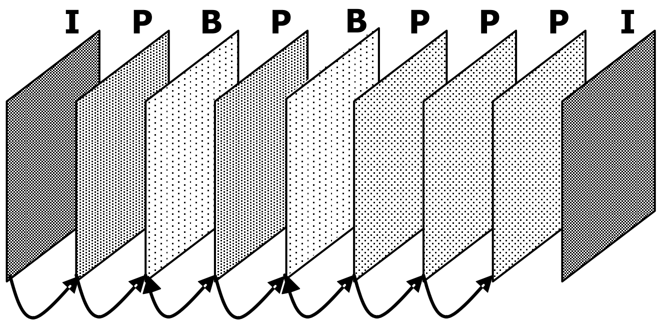


Figure 10: Scheme of frames of a compressed video

There are other frame types, especially for more recent coding schemes such as B-frames, which are using backward and forward differential information.

As known, differential coding becomes critical when errors occur and information is falsely interpreted. If an error occurs during transmission, that error will be propagated since the error does not become corrected by the subsequent image or frame. For that reason, in regular distances key-frames are inserted for synchronizing the information by providing a complete image. Usually, one key-frame per second is transmitted. There are other strategies as well, for example forced partial updates of Groups of Blocks (GOB) in P-frames in between key-frames. Information from one key-frame to the next one are often organized and called a group of pictures (GOP).

However, the different frame types require a different amount of data for storing the image information. Basically, an I- or key-frame requires many more bits than a (differential) P-frame or B-frame. In case the compression is locally constant, a full image description requires of course more bits. Secondly, spatial complexity has a significant effect on bitrate. A spatially less complex image requires less data, and even less after compression. Conversely, a spatially complex image requires more bits to describe. As a result, the bitrate will vary over time depending on the complexity of the images within a segment of the video. Increased spatial complexity has visible peaks where I-frames are transmitted. However, due to the constant quantization coefficients in the algorithm, the quality can be considered as constant over time.

Another strategy for compression is the desire to have a constant bitrate, even in a short term perspective. For this reason, the compression is dynamically adjusted to the frame type and the spatial complexity. However, this approach can be applied only to some extent. The result is a fairly constant bit rate without extreme peaks but with varying quality over time due to the varying compression ratios.

Especially for low bitrates and a constant key-frame rate (e.g. one key-frame per second), the key-frames are perceived as regular drops in quality characterized by increased blockiness or more blurry images. The following differential information will give the difference to that frame and also adds more details again, thus the blocks disappear and the video becomes more detailed after a few frames until the next key-frame is received.

The following series of frames illustrates this pattern. The received I-frame has a high amount of blocks due to the high compression. There is almost no information in the so-called micro-blocks of 8 x 8 pixels. The entire block is described by only a few DCT coefficients (sometimes even one or two only) giving the luminance and chrominance information.

The following differential information does not only describe changes in the movie like movement, but rather adds more and more details. This can be observed quite well at the eyes and mouth region. When the next I-frame is received, a full image is transferred again and the loss of details due to the high compression is visible again.

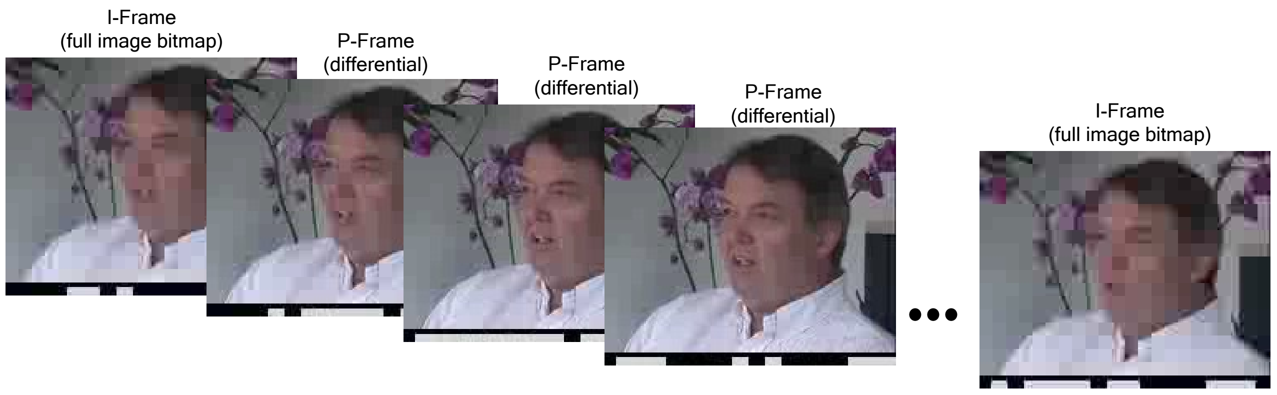


Figure 11: Illustration of an I- and P-frame sequence of a constant bitrate video

For video on demand or video storage purposes, a constant short-term bitrate is less important. There, mostly ‘constant quality’ compression is applied. For real-time streaming or live video, huge peaks or variances should be avoided or at least minimized, leading to a varying quality over time or a different perceived quality (e.g. by regularly appearing key-frames). Practical streaming cases are tending towards the latter approach, though the variance in quality generally decreases when the bitrate increases. This means that the described key-frame effect disappears for higher bandwidths, but is well perceptible for lower ones.

Artifacts caused by video coding and transmission

Video coding and compression as described above is of course not lossless. How far the lost information is causing lower perceived quality depends on the coding scheme and its settings as well as on the bitrate. However, the artifacts introduced by those coding schemes as explained are similar and can be grouped into a few main categories. They are caused mainly by the sub-division of the image into small squares.

Blockiness and Tiling

Due to the reduction of transmitted information in the micro- and macro-blocks they often look uniform and there is a clear border to the neighboring blocks. The picture looks like a chess board pattern. This effect is known as Blockiness. Often it is not visible equally over the entire picture; it may by more present in certain regions of the frame. This effect is just caused by compression in video codec. Good examples for Blockiness are shown in Figure 9.

If transmission errors occur and blocks become falsely decoded or not updated correctly, either wrong information or outdated information are shown in the affected blocks. Here 'misplaced' macro-blocks are easily visible, since they don't match to their actual neighborhood anymore. There might be even complete strange colored blocks anywhere in the image. Those effects – more based on the macro-block sizes are usually called Tiling. Examples for falsely decoded macro-blocks are shown in Figure 12.

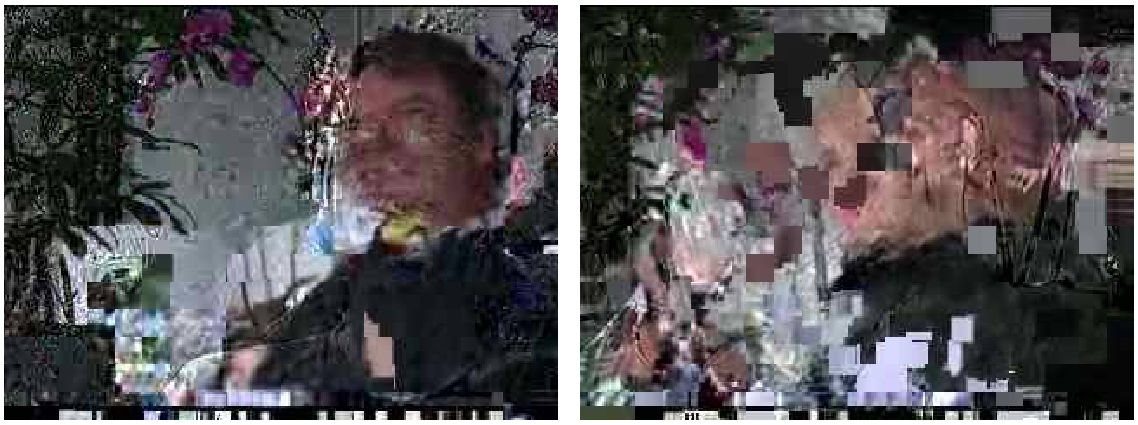


Figure 12: Examples of Tiling artifacts

The left image shows wrongly decoded or falsely updated macro-blocks in the lower left corner. They still contain detail information about luminance and chrominance but are misplaced or wrongly decoded and don't match to the neighborhood. This makes them visible as degraded video quality.

The right image shows macro-blocks without details. However, there is chrominance information that may occur in image but at other places (as for example the skin tones).

Blurring

Blurring is in principle the opposite of sharpness. The picture or parts of it appears un-sharp, blurry. Blurriness is usually measured by analyzing sharp edges in the picture (excluding block borders) and in well reproduced spatial textures. However, blurriness can also just be a characteristic of an image. There are contents (i.e. a cloudy sky over the sea) where are no sharp edges by nature. That makes blurriness difficult to measure without a reference picture for comparison.

In case we have a picture with sharp textures as a source video, it may become less sharp due to compression (details are taken away) and post-processing steps applied in the decoder.[3] (See: Figure 13)

[3] There are strategies to smear block borders by low-pass filters. They can smear spatial details as well and the entire image appears less blocky but more blurry.

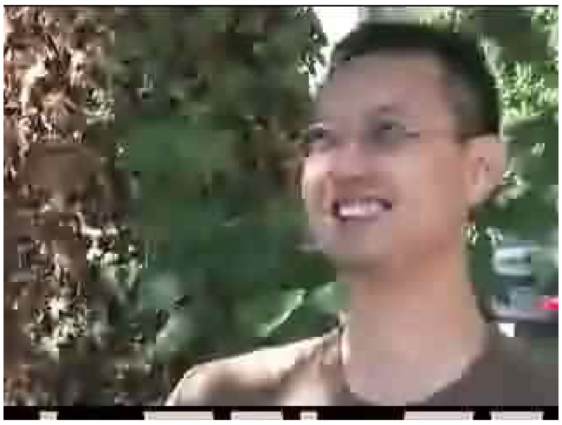


Figure 13: Examples of Blurring by post-processing

Blurriness is a perceptual dimension. There is an overlap with technical measures such as Blockiness. With larger format video and wider viewing distance, blockiness caused by compression is perceived as 'un-sharpness' by a viewer, the picture appears blurry. If one was to expand the image, it would be obvious that blockiness is the primary cause for the blur.

Mosquito Noise and Color Bleeding

Bleeding and Mosquito Noise are side effects of the macro-block encoding. Mosquito Noise is caused by regular patterns produced by the decoding of blocks. It means that a regular pattern only describes a part of block; the same pattern appears outside that region until the macro-block borders. It is perceived as noise, since the texture changes from image to image and appears not static.

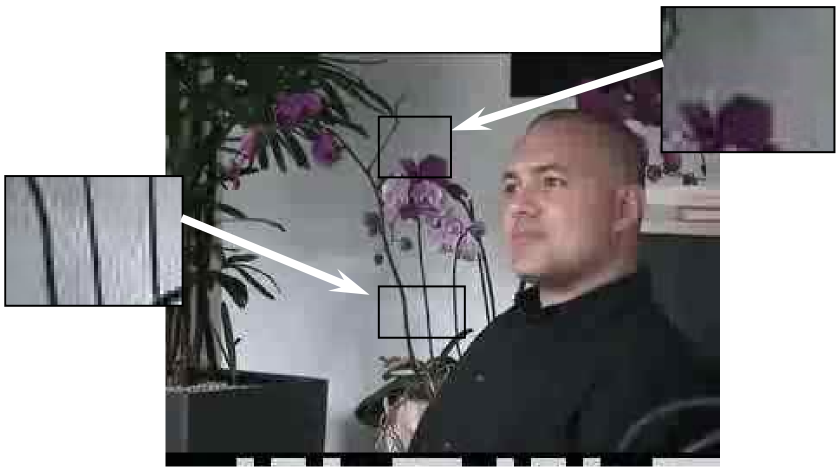


Figure 14: Examples of Mosquito Noise and Color Bleeding

In Figure 14 the Mosquito Noise can be observed in the surrounding of the leaves at the left side and around the stalks of the flowers in the middle. The 'borders' of the Mosquito noise are the macro-block borders too.

Bleeding is a similar but is more related to the Chrominance information. In case there color changes in the macro-block and this gradient is not reproduced due to too fewer coefficients for chrominance, a color exceeds its area and is dominating the entire macro-block. It is to some extend visible above the flowers in the middle, where the pink color is slightly visible even at the wall. Color Bleeding is also visible in Figure 13 where green from the leaves are 'bleeding' in the person's neck.

Slicing and Shadowing

Slicing is an artifact where only a part of the picture is updated. Slicing usually appears in videos using 'slices' or Group of Blocks to sub-divide the image in parts with own synchronization information. It can happen, in case of erroneous transmission, that only a part ('slice') of the image becomes updated and other slices are distorted and therefore not decoded. In this case the old information remains in these parts.

The effect of Slicing is well illustrated in the left image of Figure 16. The lower part of the picture is still from the previous image, where a person with a black shirt was shown and where flowers are in middle background. The following image switches to a person in white shirt in slightly different viewing angle. The image becomes only partially updated (the upper part only).

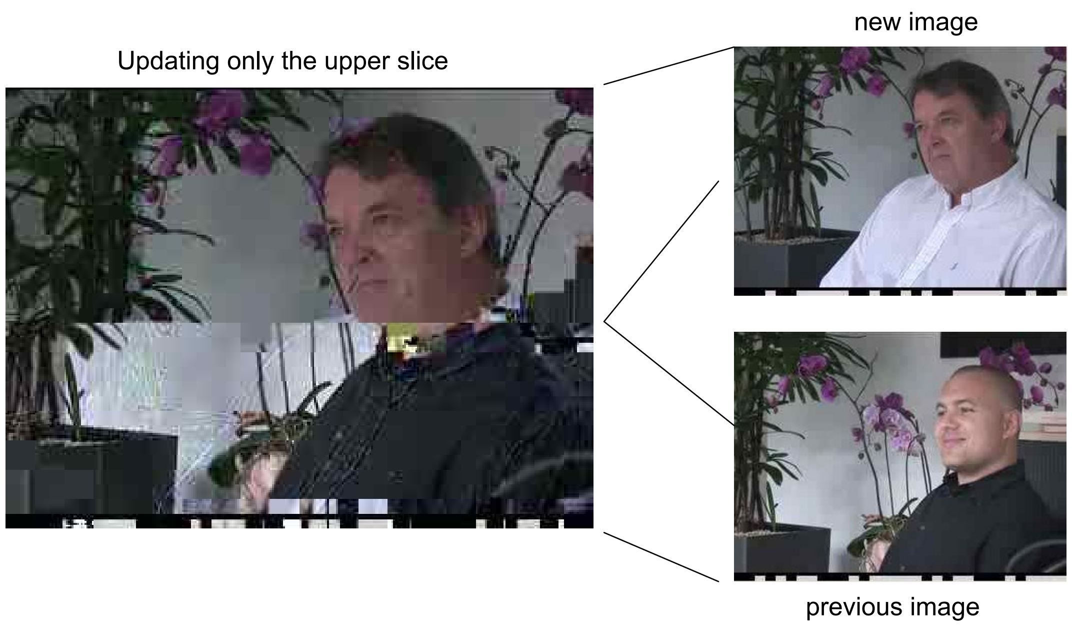


Figure 15: An example of Slicing

Slicing is caused by incomplete updates of the image information. Shadowing has similar reasons but lead to a different effect (Figure 16). The following image shows the person with the telephone. That is the last completely displayed picture information. For any number of reasons the update to another scene is missed (most likely by a missed I-frame). Now, only the update information to this (missed) new picture is received. This becomes visible as a kind of shadow showing a person with a base-cap. This mostly grey-step silhouette gives just difference information to (non-displayed) image.

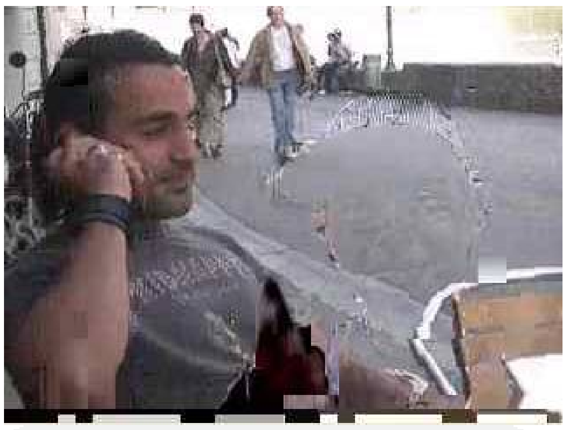


Figure 16: An example of Shadowing

Wrong update information usually leads to highly distorted images. The slight shadow as shown in Figure 16 is just an example for illustration. Usually, images become almost destroyed completely as shown in Figure 12. Also there information of previous images are overlapping with more recent information or become completely misplaced (as a shadow of the flowers suddenly appearing at the upper right side of the picture).

Whether such highly destroyed pictures become displayed at all depends widely from the strategy of error handling in the decoder and video player. No handling or a very simple handling of errors will display what is received independent whether it is erroneous or not. A simple concealment strategy is to stop the play out of the video when data is recognized by the decoder as erroneous. This stopping of playback is referred to as Freezing. More advanced players – especially for video conferencing – often try to mask errors by smart reconstruction of missed or erroneous information.

Jerkiness

All artifacts discussed above are related to spatial distortions that impact the image information. There are temporal side-effects such as Mosquito Noise or – even more visible – error propagation over a series of frames until the next complete update.

Jerkiness is pure temporal distortion. It describes how far a video is away from a perceptual fluent play-back. In case of a high frame-rate (i.e. 24 fps as for movies or 25 / 30 fps as in TV) video without any freezing events, the Jerkiness is almost zero. The video is seen as fluent.

In case the frame-rate becomes reduced, since frame rate reductions (even dynamic frame-rate reduction) are means for reducing the amount data to be transmitted, the video presentation becomes jerkier, coming closer to a 'slide show' effect.

On the other hand, a video with a sufficient frame-rate that is interrupted by intermittent pausing (freezing) events, no longer appears fluent. Both effects can be simplified and described as an unexpected long display time of an image. It might be regularly longer due to reduced frame-rates or just at individual positions as in Freezing.

Jerkiness is not a technically driven figure as freezing (i.e. measured in milliseconds) or frame-rate (measured in frames per second), it is much more perceptual.[4] The perceived fluentness of a video presentation depends highly on the content. The viewer does not recognize that the video does not move, he or she recognizes that there is a jump when the video starts again or that the still image does not fit to the surrounding amount of movement. Jerkiness is much more driven by the context of the surrounding video frames. In case there is a slow moving content, the perceived 'jump' between the longer displayed image and the surrounding ones is much smaller. Even though, the 'missed' information by a reduced frame-rate is much smaller. For high motion content such as a football game or a automobile race, any 'jumps' created by updating to next images are much larger and the perceived Jerkiness increases.

Jerkiness is a perceptual driven metric, since it takes into account context knowledge of the viewer. To properly determine Jerkiness, one needs to understand what is the viewer expects to see, or better, what the viewer is missing by un-fluent video presentation.

[4] Freezing is often called ‚pausing' too. There is 'pausing without skipping', where the video stops but starts playing again with the next frame and 'pausing with skipping', where the video stops and starts playing again at a later frames and frames in between become skipped.

4 Technical Background for Video Testing

Bitrate, Resolution and Scoring Time

In principle, the bitrate has nothing to do with the transmission capacity of a channel. However, in real-time streaming applications the channel capacity should exceed the bitrate; otherwise the video can't be delivered in time.

The term resolution describes the image size (i.e. QVGA or 240p is 240x320 pixels). There are various and not always consistent definitions of resolutions. There are the 'older' resolutions defined by ITU-T (or at that time CCITT) such as 'Common Intermediate Format' (CIF, 288x332) originally defined for video conferencing or QCIF (144x176) that is a quarter of CIF. There are even SQCIF (SubQCIF, 96x128) or larger formats 4CIF or 16CIF.

However, these resolutions are widely succeeded by the 'PC screen formats' such as the 'Video Graphics Array' (VGA, 640x480), which is also used as NTSC standard television format, or the well known QVGA (320x240) or even QQVGA (160x120), which are all following the aspect ratio of 4:3. The PAL and SECAM standard television formats also have an aspect ratio of 4:3 (768x576).

Newer formats are following the 16:9 aspect ratio like WVGA (853x480, 852x480), which is used also in NTSC 16:9 decoders. This aspect ratio is used as well for HDTV in either 720p (1280x720) or 1080i/p (1920x1080).

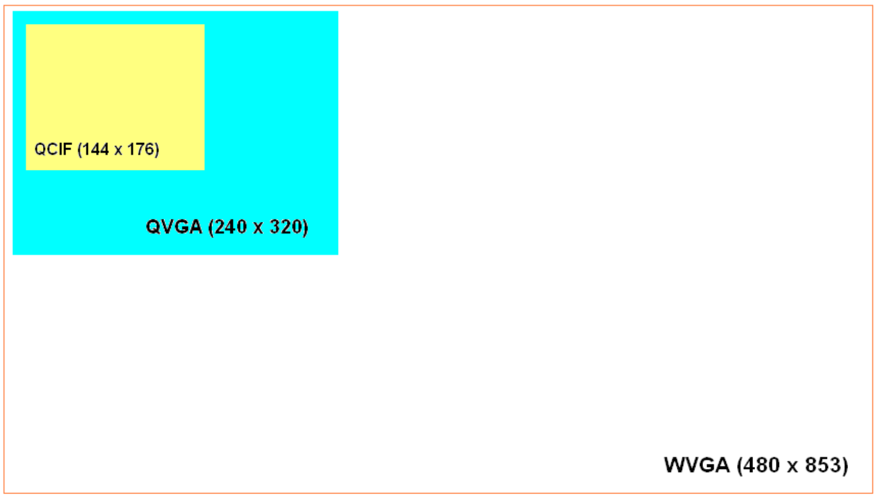


Figure 17: Evolution of multi-media formats in video

The resolution and the frame rate define the 'number of pixels per time'. The higher the resolution and/or the frame-rate, the more information (pixels) have to be transmitted at the same time. A compression ratio 'just' reduces the amount of data by a factor. Compressing a higher resolution video to the same bitrate as a lower resolution video would require a higher compression ratio. Consequently, that video has a lower quality (compared to its original) due to the higher compression and more visible loss. Generally spoken, a higher bitrate is needed for higher resolutions in order to obtain the same spatial and temporal quality.

Irrespective of the bitrate used for transmission, the video player decodes (decompresses) the compressed video stream into plain RGB for displaying. Even though there is a loss of information (due to compression we lose spatial information); the resulting bitmaps have the same size in data than the original ones. That way, the amount of data to be handled for video scoring is independent of the bitrate in the transmission chain; it only depends on the resolution and on the number of frames.

VQuad as a quality predictor analyses the de-compressed video, the same the user will see on the screen. As explained, it can be imagined as a series of RGB bitmaps. This amount of data to process is independent of the bitrate of the compressed video, and only depends on the resolution and frame rate (i.e. the number of pixels per time). A QVGA/25fps video clip can be scored on Diversity in around 6 to 8s regardless of the bitrate used for transmission. A VGA/25fps video clip results in an amount of data that is four times larger, with the result that the scoring time increases to more than 20s.

Resolution and Video Quality

If we talk about Video Quality we consider the quality in terms of 'Mean Opinion Score' (MOS). It is usually obtained in visual tests with human viewers under controlled experimental conditions. The common MOS scale ranges from 1.0 (bad) to 5.0 (excellent). The MOS is just the average over all individual scores for a video clip obtained with e.g. 24 test persons.[5] Examples for visual experimental setups can be found e.g. in ITU-T Rec. P.910.

Video quality is defined only for individual resolutions. It means a Video MOS obtained for a QVGA video can only being compared to another QVGA video MOS. It can't be compared to a SD video MOS for example. The underlying subjective tests are always presenting only a single resolution (since it is usually tied to an application and a device, i.e. a smart phone or a TV set). Of course, SD videos can be scored in an HD context for example, but the SD video will be re-sized to HD resolution for that. There are no test procedures for providing inter-resolution quality scores taking into account the actual resolution (i.e. presenting different image sizes on the same screen). This fact has to be considered in the test scenario defined later on.

In subjective test procedures, human viewers always score the quality relative to the perfect high quality video in the same resolution. The loss of information is scored relatively to this original or reference video, independently of how much information is lost due to the lowering of the resolution from e.g. HD to QVGA.

These elements lead to the fact that a QVGA/25 fps video that is compressed to 1200kbps might be scored with a MOS of around 4.0 or above depending on its complexity, since there are almost no spatial or temporal compression artifacts to be seen. In case a WVGA (480x852)/25fps video is compressed to 1200kbps too, the loss of spatial information in relation to the original perfect video is much higher. Consequently, this video may get a MOS of e.g. 3.0 only compared to the >4.0 a QVGA video would get at the same bitrate.

Objective Video Quality Measures

So-called objective measures predict quality through signal analysis. They predict video quality as perceived in visual experiments by a group of viewers. Usually, those measures give additional information about the signal too. Such objective models are commonly distinguished by the information they get for analysis.

Basically, there are 'no-reference' and 'full-reference' models. A no-reference model only analyzes the received / recorded video signal. A full-reference model gets the received signal and the reference signal for comparison. This reference signal is the input signal in the transmission chain. Logically, full-reference signals have much more information due to the comparison to the reference signal than no-reference signals and are therefore more accurate in their quality prediction.

[5]For practical reasons, the maximum quality reachable in visual experiments is in the range of 4.5, since all viewers never score a perfect video with 5.0. There are always some people giving a lower score such as 4.

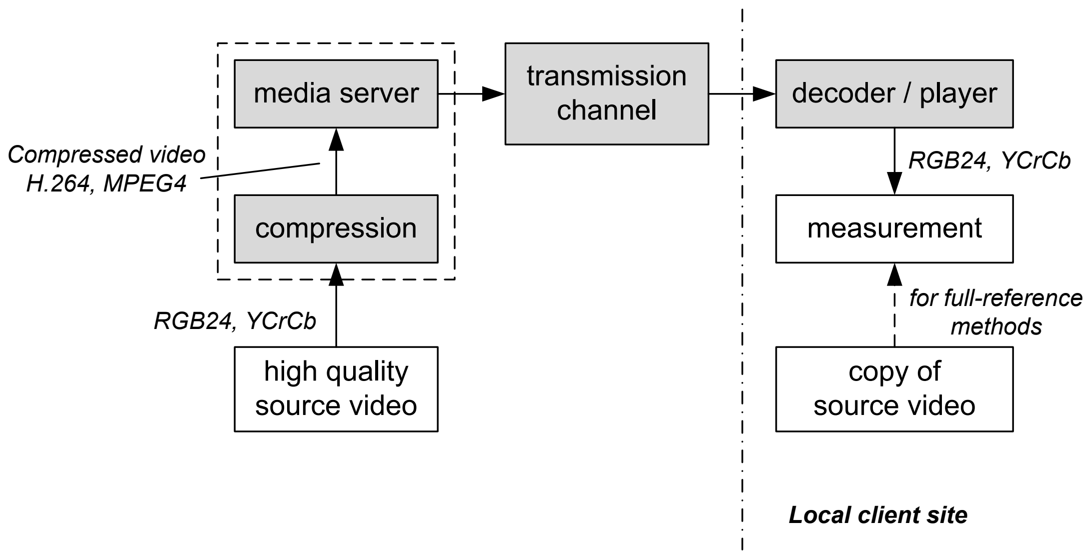


Figure 18: Principle of the video flow in streaming quality measurements

Figure 7 shows the basic flow of streaming and a related measurement approach. At the beginning there is always a high quality video source available. For test and measurement purposes this source video is in uncompressed format, i.e. RGB or YCrCb. The compression can be done outside or inside the media server.

In SwissQual's measurement approach there is separate compression in case of using the Darwin media server. That way SwissQual has full control of the bit-stream to be stored at the server. In case of YouTube the high quality video (after a practically lossless compression with H.264) is uploaded to the YouTube server and this server itself compresses the video for further streaming purposes.

All grey filled components in the above figure have an influence on the quality: The compression inserts artifacts due to information loss, the channel might lose or delay packets and the player may have different strategies for concealment of lost or skipped packets and can even try to smear compression artifacts. The screen outcome of the player is captured (as bitmaps) and transferred to the scoring algorithm.

A no-reference model will just evaluate the incoming and captured video; a full reference algorithm has access to a local copy of the uncompressed source video for comparison purposes.

Although full-reference models are more accurate, they cannot be used for all application scenarios, as they require the reference signal. Usually, that reference is the high quality video that is later compressed and fed into the channel (i.e. stored at a media server and streamed on request). This test approach is possible in case the user a) has access to the high quality source clip and b) has access to the server and can install (compressed) video clips there.

This is usually not possible in live TV test applications. Here no-reference methods are the only way of assessing the video quality.

However, as long as the methods only refer to the video signals, the channel or system to be analyzed can be considered as 'black box'. There are also approaches that analyze the IP bit-stream, either alone or in conjunction with the video signals. The channel or system to be analyzed can then be considered as a 'grey box', i.e. some information (codec type, bitrate, etc.) describing the system can be read out.

There are pure 'bit-stream models' and combinations with common no-reference / full-reference models in so-called 'hybrid models'.

Pure bit-stream models have a limited scope; they can't consider the video decoder/player and concealment techniques. They also have limited knowledge of content characteristics and are limited to protocols and video codecs they were designed for.

Conversely, hybrid approaches can improve a video (image) analysis by information extracted from the bit-stream. There are currently standardization efforts for these methods.

VQuad as an Example of Video Quality Measures

VQuad is a full reference method scoring video quality. It uses the reference video signal for a frame by frame comparison with the recorded signal. It recognizes differences in the individual images (spatial distortions) as well as missed or skipped frames (e.g. by reduced frame rates) or a different display time of the frames (e.g. Freezing, where a frame is displayed much longer than in the original video).

VQuad must successfully align the two videos, the reference and the recorded clip, in order to provide a video MOS. Therefore, for each recorded frame the corresponding frame in the reference video is searched. VQuad makes use of a special watermarking system to find the corresponding frames by a unique ID. VQuad uses the watermarks to speed up the scoring process. The OEM version of VQuad can also find corresponding frames by content analysis.

Once a corresponding frame pair is found, global parameters such as luminance are aligned to avoid differences just by a global luminance differences. This procedure is often called temporal and spatial registration. The aligned frames will be analyzed with regard to individual distortions such as Blurriness, Blockiness or similar. The amount of these individual distortions is compared between the two frames. In addition a pixel-wise comparison is made too (so-called PSNR). All the calculated spatial and temporal indicators are than weighted and aggregated to an overall quality prediction.

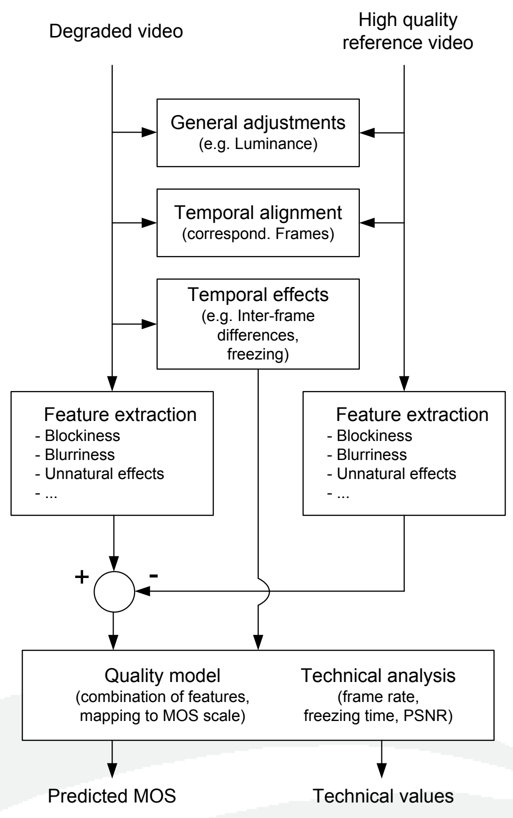

Degraded video
High quality reference video
General adjustments (e.g. Luminance)
Temporal alignment (correspond. Frames)
Temporal effects (e.g. Inter-frame differences, freezing)
Feature extraction - Blockiness - Blurriness - Unnatural effects - ...
Feature extraction - Blockiness - Blurriness - Unnatural effects - ...
+
-
Quality model (combination of features, mapping to MOS scale)
Technical analysis (frame rate, freezing time, PSNR)
Predicted MOS
Technical values

Figure 19: Principle of VQuad as example of a full-reference method

For interpretation of the results it is very important to make sure that display effects (e.g. brightness, contrast or color changes) are not considered. Video enhancements (e.g. color booster or sharpened edges) are only partially considered as improvement. In principle it can be stated: The closer the recorded video is to the reference signal, the higher the predicted quality will be.

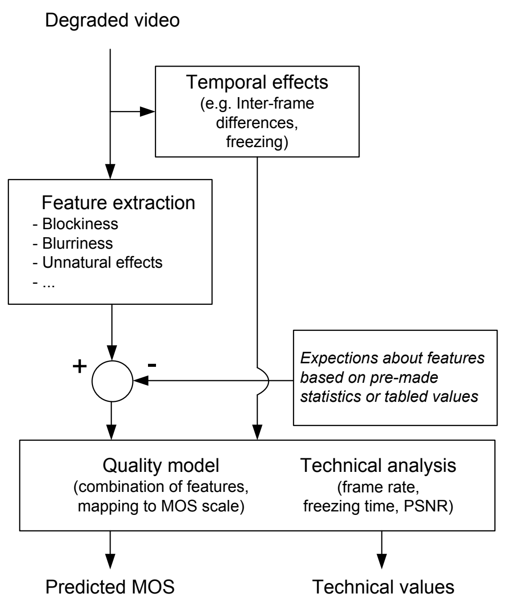

Degraded video
Temporal effects (e.g. Inter-frame differences, freezing)
Feature extraction
- Blockiness
- Blurriness
- Unnatural effects
- ...
+
-
Expections about features based on pre-made statistics or tabled values
Quality model (combination of features, mapping to MOS scale)
Technical analysis (frame rate, freezing time, PSNR)
Predicted MOS
Technical values

Figure 20: Principle of VMon as example of a no-reference method

Compared to a full reference approach, a no-reference measure such as VMon has no access to the reference video. It only analyzes the recorded video with respect to known artifacts and the fluency of the video. However, it can recognize severe problems with confidence but – since the input is unknown – there is a content dependency on the scores and their accuracy.

VMon as an example of a no-reference method compares the extracted features (mainly known artifacts) with ‘expectations’, means how much of those artifacts could be considered as normal in an image and corrects the measured values by that.

5 Test Scenarios for Video Services in Drive Tests

Testing progressive download

Diversity allows the use of Flash Player along with YouTube in a PC emulating test scenario. In this scenario, the mobile device is configured as a modem connected to a Windows 7 based PCM (PC Module). In case the transmission bandwidth is sufficiently high, the video is downloaded in a shorter time than the video duration.

How do different technologies and networks behave here? A high transmission capacity will fill the initial buffer in a shorter time, thus the playback of the video starts earlier. It will also enable a higher buffering time and may be less affected by interruptions, as these are then bridged more efficiently. However, due to the use of HTTP no transmission errors will be visible.

The main discriminative factor between different technologies will be ‘time to first picture’ and the amount and duration of freezing caused by re-buffering.

The main disadvantage is the non-synchronous relation between the individual channels tested or between operators. For one operator the download could be finished just before a bad coverage area is reached. This will not be recognized, since the transfer is complete. Another operator may have a just slightly different bandwidth and will be affected dramatically by losing the stream at that place. There is a strong non-synchronicity with the actual completeness of the physical data transfer in the network. It strongly depends on the available bandwidth and its continuity. Despite starting at the same time, the transfer might end earlier or later (despite of the ongoing playback of the video). In addition there is absolutely no relationship between the occurrence of a visible problem in the video (i.e. re-buffering) and the actual geographical location of the stream loss. Since the buffer is played out at first; the actual problem in the network may have occurred seconds or even minutes earlier.

In principle, testing progressive download only gives information on how fast a given amount of data can be transmitted and the only criterion is whether this time is shorter than the video duration itself. More generally, a plain FTP or HTTP download test delivers more information about channel capacity than a progressive download video test. However, ‘time to first picture’ might be a useful parameter that is related to the perception of a user of such a service.

Testing real-time streaming

Contrary to testing progressive download, all RTP based streaming services are real-time orientated. SwissQual Diversity supports RTP streaming using 3GP transport streams along with Real Player and QuickTime. Such streams can be accessed via YouTube (emulating a mobile phone) or by a media server delivering real-time streaming.

Compared to testing progressive download, there can be transmission errors affecting the quality as well as interruptions resulting in visible freezing of the video. Due to the very short buffer time, real-time streaming tests are more sensitive to transmission problems in general, since only short interruptions can be bridged by the very limited buffer size. Furthermore, there is an acceptable synchronization between the occurrence of a problem in the video and its cause in the network for drive tests. One main advantage is the largely absolute synchronicity between the tests of individual operators or channels. The start of the test is synchronized and since the streaming itself lasts for the entire duration of the video, the end of the test is synchronized too and corresponds almost to the end of the physical data transfer.

A real-time streaming test not only gives information on whether the channel capacity is sufficient for a certain bitrate, but it also provides information about the continuity of the data stream.

Media and video servers

Commercial media servers are becoming more and more intelligent. They are not simple storage servers, where pre-compressed video files can be streamed from. They are capable of performing on-the-fly transcoding or synchronized switching between streams in different bitrates based on returned network information or access routings. The individual handling of the video is highly vendor dependent and can furthermore be adjusted by a wide set of controls. It has to be noted, that the server is part of the entire transmission chain and should be considered in an end-to-end measurement. In case a customer is interested in the perceived quality of its subscribers, the test video samples should be streamed from a server that is used in its real field setup. This might be different for individual operators and content providers.

A standard media server such as Apple's Darwin server streams follows a more simple approach. It can stream pre-compressed video clips on demand in all of today's formats. There are no latest technologies integrated that make use of dynamic bitrate adjustments controlled by network information. The user itself has to request a stream that may fit its expected channel bandwidth. Using such an approach takes the sever dependency out of the measurement. It is kept constant for all operators or settings to be measured. However, it has to be accepted that the quality is not optimized for a particular wireless network technology or setup.

Apple's Darwin server even supports progressive download in conjunction with QuickTime as the video player. Other servers, e.g. YouTube, support this as well, especially for Flash video. However, YouTube in this environment will limit the amount of video inside the buffer to a specified amount of time.. This approach is driven by the fact that most YouTube users don't watch the entire video stream. Thus a download of the entire stream would waste network capacity in most of the cases.

YouTube in general is a simple example of a more real field application. The user can in principle choose between different qualities (i.e. different bitrates and image resolutions). That is close to a simple approach as mentioned. But YouTube also considers the access technology used. In case a client connects via a mobile network it reaches a different set of streams (means a set of different bitrates). The client is offered streams with lower bitrates (e.g. 300kbps) and in 3GP format for the typical Real Player installed on mobile phones.

Test applications and emulating end user's experience

In principle there is the same challenge in testing video services as there is for testing other services offered by operators. Videos can be compressed in a way that they can almost always be transmitted, even with non-optimal coverage or on older data technologies. Here the channel capacity is often not fully used and the user gets a medium quality, but at least for most of the time. In case the bitrate is set higher, the quality increases but the availability of the service drops. Of course, dynamic bitrates using returning information solve that problem but there are not always in place. In addition there are many third parties content providers who don't get or use such information.

The same situation occurs for testing and especially benchmarking. Of course, the ideal solution would be for each operator in the test campaign to store the test clip on its media server used in its network and to choose the same settings he usually applies for his video services. This approach is not applicable in practice.

One possible approach could be to use a stream that results on average in the best trade-off between compression quality and 'being transmitted'. This compromise has to be found for each operator individually, depending on its status of infrastructure. The second and more applicable case is the emulation of the same test case for all operators to be tested. This test case should be realistic but also avoid any disadvantaging by intent. It would simulate the video streaming from a third party provider.

For testing video streaming the real-time idea is important. Even if there are a lot of users who may use video on demand services, they are not describing the video delivery capability of a network. It is closer to a pure file transfer. Therefore a real-time application should be used in test campaigns. It would emulate watching live TV or live video over a data link.

In principle, we can emulate two scenarios. One would be a mobile client using its phone; the second one would emulate a user at a (mobile) PC being connected via a data modem to a mobile network.

Since we are talking about real-time video, both scenarios would utilize RTSP and a corresponding video player. On a mobile phone it is usually Real Player or QuickTime (for the iPhone). In case of PC use, these players are also quite common, however there are a wide set of free players such as VLC or others, partially based on Real Player or QuickTime components in the background. The difference is mainly the displaying size and the available hardware resources.

In a measurement approach, both can be emulated on a measurement PC. The display size can be chosen to emulate a PC user or to emulate the size of video displayed on a typical mobile phone. In addition, mobile phones can be further emulated by shortening the video buffer and switching off hardware video accelerators. For emulating PC use cases the player will be used in a 'normal setting' as a PC user would do it as well. Finally, the differences in the settings don't have so much influence on the quality but rather on the CPU performance required by the player.

Another important aspect of testing in a mobile environment is the type of connection made to a media server. When a PC is connected through the internet to a media server, information about the operating system and media player type is transferred to the server. Based on this information, the media server will stream the most appropriate content. SwissQual Diversity allows the user to configure the data connection to look like a standard PC or a mobile device. Depending on the individual user's testing needs Diversity can then be used to emulate a mobile device or a USB modem, The media server will then stream information based on this connection type.

Video streams to be used for measurements

Under the assumption that the user or the measuring party has the possibility to choose the video for measuring, the contents and complexity can be chosen freely.

It makes sense to keep the number of test clips in a measurement campaign low so that variability between video clips does not contribute to differences reported between carriers. The dependency of the quality on the chosen video clip is quite high, higher than the dependency of a voice MOS on the speech sample. In case only the clip is changed, there might be visible differences in the observed quality just by changing the clip. On the other hand it doesn't help to 'standardize' a video clip for testing. By nature, compression algorithms and measurement methods would be optimized for this clip.

However, there are differences between the selection of clips for codec standardization and automated measurements. In video testing for standardization usually different content types are used such as 'head and shoulder', 'movie', 'cartoon' or 'home video'. For each content type a many examples are used. The selected excerpts are often chosen as critical for compression algorithms to evaluate these codecs under extreme conditions and are less meant to represent 'real field' sequences.

For testing services this is not required or even wrong, since the encoding part is less of interest and it does not reflect the user's experience. However, for service testing too some basic considerations of the content should be made. It is wise to utilize different clips, the content itself is less of interest but they could be characterized by spatial and temporal complexity. SwissQual currently provides three video clips

- Head and shoulder, slow moving, recorded with fixed camera position, static and spatially less complex background, no scene cuts
- Head and shoulder moving, fixed camera position, slight moving and spatially more complex background, scene cuts
- Persons, fast moving, hand held camera position, moving and spatially complex background, fast scene cuts

All videos are 'natural', there are no graphical objects or still scenes and they are available in different image sizes. As explained above the quality after compression depends on the complexity of the video. That means, for a target quality the first clip would require a lower bitrate than the third one. In other words, for a given bitrate, the first clip would achieve the highest quality and the third one a lower one.

In addition, a faster moving clip is more sensitive to freezing. Even short freezing or just a lowered frame rate will lead to a perceived loss of information. An almost not moving clip will be perceptually less affected by those jerky presentations. However, this difference is only true for very short freeze-ups of upto a few hundred milliseconds. Larger frozen areas will be perceived as very distorted and will be scored very low regardless of the temporal complexity of the content.

Another important question is the length of the streamed video. If it is too short, it can happen that mainly the initially-filled video buffer is played out. If it becomes too long the drop probability increases and there are less ‘per-stream’ KPIs possible in a given measurement time.

From a user’s perspective, mobile streaming will emulate e.g. news watching, weather forecast or YouTube clips. For this approach we typically see video duration between 30 seconds and a few minutes. Of course, there might be other use cases like watching a football game or even a movie. However, we should keep in mind that results from measuring shorter streams can be mapped to qualitative statements for longer videos.

In principle it can be stated, that it makes more sense to use shorter streams as long as they are not too short. A good practice are streaming durations of 60s to 180s. It may even make sense to have a look to common tests in voice telephony, whereas a typical test call has a duration of around 120s.

We should consider the influence of the stream length to basic KPIs. In principle, the longer a stream, the higher the probability of losing or dropping the stream at a certain time, since we may lose coverage or experience a bad handover.

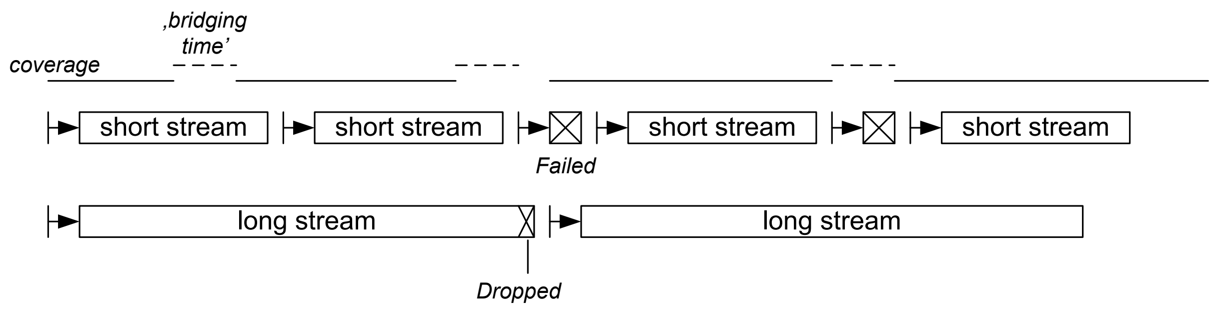


Figure 21: Failed and Lost/Dropped behavior depending on stream duration

On the other hand, the shorter a stream is, the lower the loss probability. However, there is a slight increase of the failed probability for cases where a small gap in coverage results in a failure of the setup, while an established stream can be kept (maybe with some freezing). Comparing Failed and Lost counts only makes sense if the desired streaming duration is comparable. Figure 10 illustrates this point in a very simplified manner.

Testing video streaming services serves to describe the network’s ability to continuously deliver a video with high quality and less to score compression artifacts introduced before streaming. It is important to utilize or to emulate the streaming process. Therefore it is important to:

- use compression algorithms used in practice (e.g. MPEG4, H.264),
- use protocols, transport containers and streams as used in the application (e.g. MPEG-TS, 3GP)
- apply bitrates as used in the target scenario (i.e. ~300kbps for mobile YouTube, ~600kbps to 1500kbps for high quality multimedia, ~3000kbps for home SDTV and perhaps >5Mbps for HDTV

As a consequence, the test application should use the characteristics of the target application. It is less important whether the displayed image size is the same; this is just a question of power consumption on the measurement platform. Artifacts, especially those from compression and even more from transmission can be recognized also in reduced image resolutions.

For testing SDTV or even HDTV on a measurement platform as used in drive test tools, it is more important to use the correct codecs, protocols and bitrates required for those formats rather than grabbing the video in full native resolution. Otherwise the grabbing process may exceed the resources available on the platform. Strategies can be the grabbing of a center region of the video or a down-sizing before capturing.

In a simplified but efficient way, smaller images can be transferred directly by using a high bitrate as usual for larger images, e.g. QVGA at 1200kbps can be imagined as a ‘common’ application for WVGA in a multimedia context or even as SDTV transmission in moderate quality. HVGA (360x480) at 3Mbps can be considers as an SDTV transmission case.

This approach even has the advantage of being highly discriminative. QVGA at 1200kbps can be considered as almost perfect quality. It means each transmission problem is immediately visible as a drop in quality. When using ‘real’ WVGA videos, the quality will not considerably exceed 3.0 at that bitrate. That means that smaller transmission artifacts are easily smeared by the compression artifacts and the range for scoring the transmission problems is lowered to the range from 1.0 to 3.x compared to QVGA where we have 1.0 to 4.x.

A quality score as MOS is not a linear addition or combination of the individual distortions. There are 'inter-distortion' masking effects, i.e. slight blockiness, truly visible if there is nothing else, will have less or even no influence if there is also a heavy slicing effect.

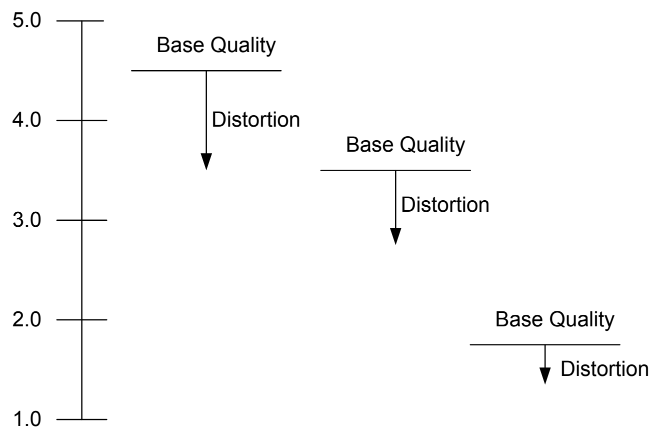


Figure 22: Non-linearity and non-additivity of the MOS scale

This principle is shown in Figure 11. The MOS scale is non-linear, i.e. an 'objective' amount of distortion that leads to a drop from e.g. 4.5 to 3.5 would lead e.g. only to a drop from 3.5 to 2.8 in presence of other distortions. The illustration refers to different 'base qualities' e.g. given by compression artifacts and the 'distortion' stands for the same objective amount of added distortion, e.g. a certain amount of freezing or similar.

Independent of the displayed and captured image size is the application of the internal cognitive model of the objective measure as in VQuad. Here the perceptual weightings are used as they will be perceived by watching a video in native resolution. SwissQual has the expertise to utilize scoring strategies where larger image sizes can be scored in both native and in down-sized formats. Using down-sized videos becomes interesting for evaluating images >QVGA on portable measuring platforms like Diversity. Current investigations are analyzing those strategies for VGA/WVGA, SDTV and even HDTV in 720p.

How to describe video streaming quality?

Up to now the question about quality in a video streaming service was focused on the quality of the video itself. This is only a part of the perceived quality of a streaming service. Of course there are other KPIs such as Failed Access or Lost Streams but also 'Time to First Picture'. These are detectable indicators and mainly defined by the IP layer as the server can't be reached or the IP stream is lost. The time for connecting and initial buffering until the first displayed picture can be measured too.

The following diagram gives a good overview about today's video testing and main parameters at this service level.

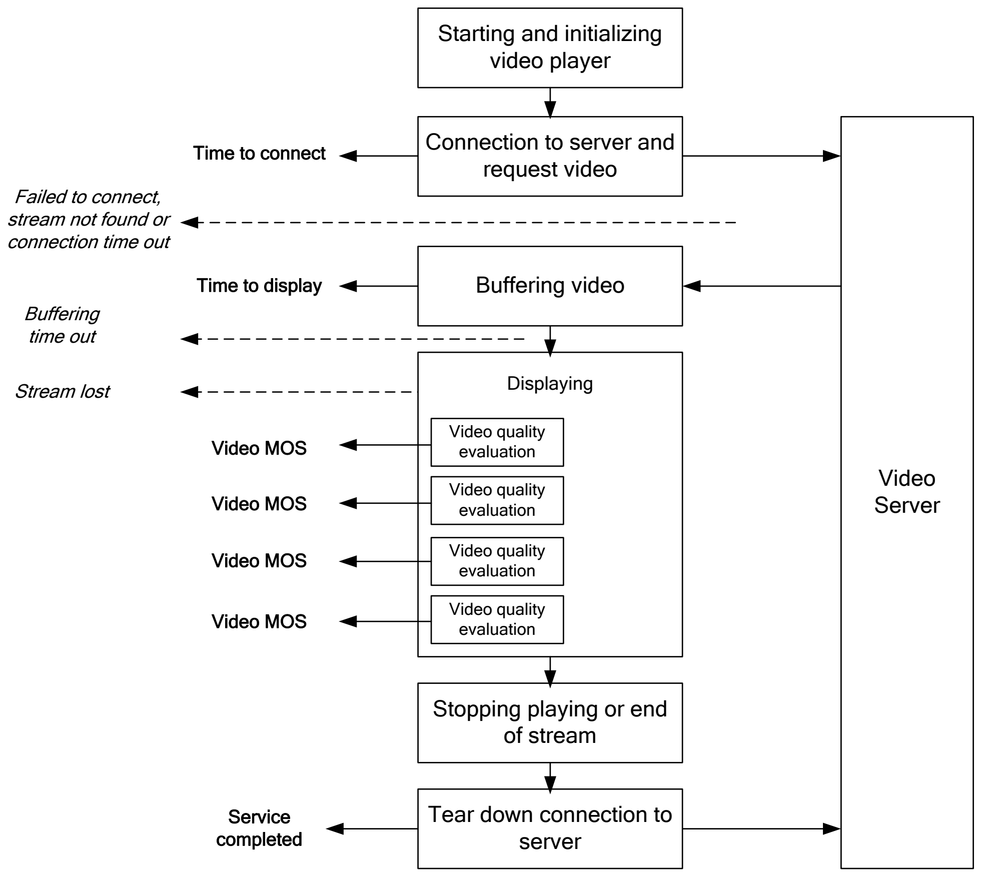


Figure 23: Flow of a typical video service measurement

However, there is a grey zone where neither the IP-based indicators nor the common video quality give a clear indication of the perceived service quality.

One simple example is that of a very long connection and/or buffering time before the video is displayed. Here a time-out emulates the user's 'patience', after a user defined time-out the test is aborted and defined as failed even if an IP connection to the server was established.

Another example is a stopping of the video. It is displayed but after a while it freezes constantly. Also here this long freezing is detected and is used for considering the stream as lost after a given time-out.

More complicated are cases where there are heavy distortions or long freezing but not exceeding the time-out. Those clips are analyzed by the objective measure and it may happen that a score cannot be computed due to e.g. too few moving frames. The clip is almost completely frozen, by definition there is no 'video' quality, since it isn't a video but rather one or a few individual images. Here a qualitative problem information about that individual clip is often more helpful than an artificially given very low quality score.

There are also cases where the video is heavily corrupted and there is no frame-by-frame analysis possible by the models since the frame comparison fails.

In general it makes more sense to give a general quality indicator for the entire stream that includes this information in addition to the video quality scores. It can even be discussed to consider heavily corrupted or widely frozen streams as failed from the user's perspective.

6 Real Field Testing of Video Services

Recommended test cases for video services

One test concept would be of course to stream one video clip with a constant bitrate. This would emulate a simple case and seems to make the comparison between different networks or locations easy. However, this approach should only be preferred if different operators or locations should be evaluated, where the deployed data technology is comparable.

In case of a wide range of technologies in the field, the situation becomes more difficult. A stream that fits well for high speed networks will be hardly transmitted in previous generation networks. The video quality will be quite high for the high speed locations but rather poor – if measurable at all due to consistent freezing – for older technologies. Finally, there will only be MOS scores of e.g. 4.x and 1.x that lead to a bi-modal distribution. The average over all MOS values would then just provide an indication of how often real problems were encountered (counted with 1.x) rather than a good overview that includes medium qualities too. Such an approach disadvantages networks with lower capacity, since streaming high rate videos isn't a typical demand there. On the other hand the use of a medium or low rate video stream disadvantages all operators or locations equipped with high speed technology by not using available capacities.

If we are talking about high and medium rates, we should not count too much on the theoretical capacity of the data technology. Loaded cell sites will provide much lower throughput and there might be higher cell change rates and handovers. We have to consider that a high quality multimedia video stream at e.g. 1200kbps (corresponding to a gross rate including audio and header of about 1500kbps) may be difficult to transmit in loaded HSPA or CDMA Rev. A cell sites. It is no problem to transmit it in unloaded cell sites of these technologies or higher ones like WiMAX or LTE. However, it will for sure exceed EDGE capabilities. Nevertheless, in those technologies or in demanded cell sites, a moderate quality multi-media stream at e.g. 300kbps can be transferred. At least it may result in quality scores in the medium range instead of very low scores for higher bitrates that cannot be delivered to the subscriber in time. Compared to voice calls, video streaming requires much more capacity and is much more affected by the technology and the load of cell sites. This also leads to a more visible peak / off-peak difference.

Furthermore, for benchmarking or evaluating live networks in the real field, the subscriber's usage and the availability of user devices should be considered. Especially after launching a new technology, almost no subscriber owns the required devices yet, leading to widely unloaded cell-cites for an initial period of time.

How can we get a real picture of quality? As discussed, the usage of operator's media servers for test purposes is almost impossible. Furthermore, video servers that adapt the bitrate to the channel capacity automatically are rare and require return information from the network and/or the player. This is not always supported for third party servers.

The idea of benchmarking is mostly about getting a view of the transmission channel; the compression quality is less of interest. In other words, how is a subscriber disadvantaged by network problems in cases where he or she expects high quality? Using a Darwin media server, it is a reasonable idea to test different bitrates in series (i.e. 300kbps and 1200kbps) to verify channel capacities and continuity. This would – to some extent – emulate a dynamic bitrate as well.

At first glance, it seems like a good idea to utilize different image resolutions for the two bitrates (i.e. QVGA and WVGA, test case 1). A higher available bitrate may force a user to access larger images. Let's assume a clip in QVGA / 300kbps. Here we may get a (QVGA) Video MOS of 3.0. Under the assumption that a WVGA clip is accessed at 1200kbps, a (WVGA) Video MOS of 3.0 is realistic too, since larger image sizes require higher bandwidth to get the same quality. Please remember that there is no inter-resolution Video MOS.

Another idea to reflect the quality advantage of a higher bitrate is to use the same resolution (only QVGA or only WVGA, test cases 2 and 3). Under ideal conditions we may get for these three cases:

Ideal case	Test case 1	Test case 2	Test case 3
Video at 300kbps	QVGA ~3.0	QVGA ~3.0	WVGA ~1.5
Video at 1200kbps	WVGA ~3.0	QVGA ~4.0	WVGA ~3.0

But what is the purpose of the test at all? It is to quantify the different networks, i.e. if a network has problems transmitting the video it should result in a lower MOS.

Let's imagine what happens in case of a network that can transmit 300kbps but not 1200kbps constantly:

Low capacity network	Test case 1	Test case 2	Test case 3
Video at 300kbps	QVGA ~3.0	QVGA ~3.0	WVGA ~1.5
Video at 1200kbps	WVGA ~1.5	QVGA ~2.0	WVGA ~1.5
Drop in quality to ideal	*0.0 / -1.5*	*0.0 / -2.0*	*-0.0 / -1.5*

The drop in quality to the ideal case as in the first table is the strongest in test case 2. This case will react the most sensitively to transmission problems.

An even harder case is really low transmission capacity or discontinuities in transmission in general. Let's imagine what happens in case of a network that cannot even transmit 300kbps constantly:

Bad network coverage	Test case 1	Test case 2	Test case 3
Video at 300kbps	QVGA ~2.0	QVGA ~2.0	WVGA ~1.0
Video at 1200kbps	WVGA ~1.0	QVGA ~1.5	WVGA ~1.0
Drop in quality to ideal	*-1.0 / -2.0*	*-1.0 / -2.5*	*0.0 / -2.0*

It can be seen, that test case 2 is the most discriminative one as well, since the 'reachable' maximum quality is less limited by the compression quality. This way a high value can be reached for ideal conditions, and there is enough space for it to drop down in case of problems. In case of a video that shows only medium or even low quality already due to the compression, a problem in transmission will hardly decrease the quality any further. That would lead to a lower discriminative power of different transmission scenarios.

The test cases are well chosen for these bitrates; however the video MOS values are examples only. In case of bitrates like 300kbps and 1200kbps QVGA is still the most discriminative resolution. If higher bitrates, e.g. 700kbps and 2500kbps should be tested, a larger resolution (e.g. 360x480) might be a good choice for a similar discriminative power as QVGA in the example cases.

Measuring at the user's interface

Streaming video as a subscriber and measuring quality in those applications should be as close as possible. The measuring application should emulate the user's behavior and perception.

However, an interface to the measurement equipment is required. The measurement has to be based on exactly the same images as seen by the user. Creating own decoding components in the measurement software itself therefore is not an option. While the player 'just' displays the decoded video as a bitmap to the screen, the measurement equipment has to capture and to process these 'images'.

We have to consider that the processing resources required for handling the uncompressed bitmaps exceed the resources required by the video player by far. It can often make use of low-layer components and hard-ware accelerators, even on the graphics card.

It is important that neither the capturing interface nor the processing effort affect the streaming process or block a fluent display of the video.

SwissQual's capturing approach is implemented in an extremely efficient way. The capturing interface itself is exactly adjusted to the corresponding video quality algorithms and performs the required transformation of the color planes and re-sizing procedures already during the capturing process. The video images are directly transferred to the evaluating algorithm without needing to be pre-saved in a file. In addition, the evaluation algorithm is able to apply most of the image analysis in real-time before the next video frame is captured.

Of course, the image resolution is the final limiting factor and depends on the available resources of the measurement platform, such as an 'off the shelf PC', an industrial CPU board, a high-end PC or even only a mobile operating system on the phone itself like Symbian, Android or Windows Mobile.

As an example, Diversity PCM can safely display, capture and analyze QVGA (240p) videos independent of the bitrate used for transmission. PCs equipped with Dual Core CPUs are even capable of scoring HVGA (360p) videos with a slightly increased scoring time and without parallel measurement tasks. Diversity PCM cannot capture VGA, WVGA or even SDTV in native size.

Diversity uses the approach of streaming QVGA videos at bitrates usual for VGA or WVGA for describing the network capacity and the streaming quality. In the near future this approach will be extended to SDTV and HDTV 720p. Here videos in native resolution can be streamed too but captured by a down-sized player window.

7 Conclusion

Video streaming services, be they video on demand or live video services, are being used more and more. These services are enabled by highly efficient compression algorithms, sufficient bandwidth even in cellular networks and of course by powerful end-user devices equipped with brilliant high-resolution displays.

However, video streaming is a resource demanding application, much more demanding than voice telephony. In addition, real-time video streaming requires a consistently available and sufficient bandwidth, compared to file download or progressive download scenarios where this consistency is less important.

Finally, the quality of the video streaming service is a very discriminating indicator; it reacts very sensitively to coverage or capacity problems in the network. It also relies on an application and bandwidth range that is easy 'to perceive' by a user. There is a difference for a subscriber whether he or she has consistently available bandwidth of only 300kbps or 2Mbps. In case of HDTV even 5Mbps to 8Mbps may be required. However, having 15 or 50Mbps available will not make a difference in the subscriber's experience.