# VideoMeter tool for YUV bitstreams*

Frank Fitzek and Patrick Seeling
acticom GmbH – mobile networks
R & D Group
Germany
[fitzek|seeling]@acticom.de

Martin Reisslein
Arizona State University
Department of Electrical Engineering
USA
reisslein@asu.edu

05. October 2002

Technical Report acticom-02-001

In this report we introduce the VideoMeter, a tool developed for the comparative evaluation of the quality of raw video data in the YUV format. The tool gives the differences in PSNR quality between two or three YUV video sequences and includes also a player for YUV video streams. In a typical application scenario, the tool is used for the quality assessment of videos that have been encoded with some lossy compression scheme and transported over a lossy network. The tool can be used to simultaneously play (*i*) the original YUV video sequence, (*ii*) the encoded (and subsequently decoded) video sequence, and (*iii*) the video sequence obtained after encoding, network transport, and subsequent decoding. The tool gives the quality differences in PSNR between the original video sequence and the video sequences obtained after encoding and network transport.

*Supported in part by the National Science Foundation under Grant No. Career ANI-0133252 and Grant No. ANI-0136774 as well as the State of Arizona through the IT301 initiative. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

# Contents

# 1 Video Quality Measurement

Video is expected to account for a large portion of the traffic in future wired and wireless IP–based networks. Despite its relatively low usage in ISDN networks in the form of videoconferencing, video is currently seen as one of the *killer applications* in 3G wireless networks, e.g. *video@mobile*. The enormous bit rates of raw (uncompressed) video and the limited bandwidths of wireless links necessitate the usage of adequate compression algorithms. In addition to the mere evaluation of the achieved compression (bandwidth reduction) of video coding algorithms, their error resilience behavior in error–prone environments (e.g. wireless networks) needs to be evaluated in order to assess the usability of the coding schemes on the underlying networks. This requires the evaluation of the video quality as it is provided by (*i*) the video compression schemes as well as (*ii*) the decoder after the transmission over a lossy network. One way to evaluate video quality is to have the videos viewed by an audience — which results in a so–called subjective evaluation. Having a audience view videos for evaluation is rarely practical. Therefore, several algorithms that provide an objective measure of the quality difference between two pictures have been developed [3, 8, 5, 7, 4, 2]. Despite these developments, the Peak–Signal–to–Noise–Ratio (PSNR) continues to be the most popular evaluation of the quality difference between pictures. Indeed, a recent study of the Video Quality Experts Group (VQEG) found that the PSNR still is a measure as good as the proposed alternatives [3]. We adopt the PSNR as quality metric in our VideoMeter.

Consider a *single* video frame (image) composed of $M \times N$ pixels (where $M$ is the width and $N$ the height of the image). Each pixel is represented by one luminance value. In addition, a set of pixels is represented by two chrominance values, as illustrated in Figures 1 and 2. Since the human eye is far more sensitive to the luminance than to coloring information [6], the YUV formats are typically sub–sampled (e.g. *compressed*). The two most common YUV sampling formats are 4:1:1 and 4:2:0, referring to the way the values per pixel are stored. Both formats have the same sub–sampling *compression* and differ only in the storing of values. As illustrated in Figure 1, in the 4:1:1 format, the hue and intensity values are stored for a row of 4 luminance values (e.g., pixels). On the other hand, for the 4:2:0 format, these two values are stored for a block of four pixels as shown in Figure 2. This latter format is widespread in the video research

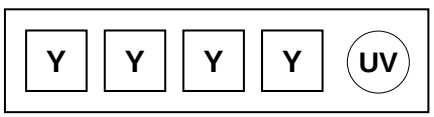


Figure 1: YUV 4:1:1 sub–sampling.

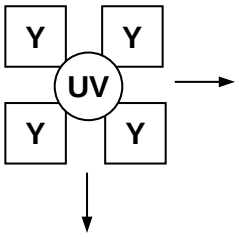


Figure 2: YUV 4:2:0 sub–sampling.

and codec–development community, and is therefore assumed by our VideoMeter.

We subsequently focus only on the luminance parameter for the calculation of the PSNR. Let $F$ denote the source image and $f$ a reconstructed (and possibly corrupted) image for the following calculations. The mean squared error (MSE) and the PSNR for a complete image are

given by [1, 2]:

$$MSE = \frac{\sum_{\forall M,N} [f(M,N) - F(M,N)]^2}{N \cdot M} \quad (1)$$

$$PSNR = 20 \cdot log_{10} \left( \frac{255}{\sqrt{MSE}} \right), \quad (2)$$

The PSNR in (2) is derived by setting the MSE in relation to the maximum possible value of the of the luminance (for a typical 8–bit value this is $2^8 - 1 = 255$). The result is a single number in decibels, ranging typically from 30 to 40 (and sometimes above 40) for medium to high quality video.

## 2 Overview of VideoMeter

In order to simplify the evaluation of the quality of video after compression and network transport, we developed the VideoMeter tool. In order to make the tool independent of the used video compression scheme, the tool takes concatenated YUV pictures (images) as input. The individual YUV pictures have to be in the YUV 4:2:0 format. We chose this format due to its widespread use in the video coding community as well as the independence from any particular video compression scheme. When we refer to frames (pictures) in this report, we always refer to the uncompressed frames in YUV format. Figure 3 illustrates the different spaces of either compressed video or YUV 4:2:0, which the VideoMeter utilizes. Additionally, items that may introduce errors $\epsilon$ are drawn in red. The tool supports up to three sequences that can be played back and their quality differences studied. A typical application of the tool in a wireless video streaming setting with these three sequences `original`, `encoded`, and `transmitted` is illustrated in Figure 4. The supported video formats are QCIF (176x144) and CIF (352x288). A screenshot of the tool with all possible options enabled is given in Figure 7 and Figure 8. In addition to the display of the YUV streams, the difference pictures for the Y component can be calculated to make the errors visible. These errors are also expressed in terms of the PSNR (in dB). The PSNR values and difference pictures are calculated for (*i*) the original and encoded (for the encoding difference), (*ii*) the encoded and the transmitted (for the transmission differences), as well as (*iii*) the original and the transmitted (for the complete difference). Both, the difference picture calculation and the PSNR calculation are done on a pixel basis and thus may cause a heavy processing load for slower systems. However, a pixel–based color space conversion has to be done in any case to display the YUV data on the RGB–based terminal screen. Therefore, the additional computational overhead for the calculation of the PSNR over the difference pictures is relatively low and thus done in any case the difference picture is displayed. The disabling of the PSNR switch only disables its graphical representation which uses some additional CPU time.

The information window shows some basic statistics about the currently displayed sequences. Additionally, the play–out modes are shown as are the source files. The information window further shows the PSNR values for the current frame, as well as the smoothed (i.e. mean) PSNR from the beginning of the play–out.

As errors occur, the comparison between the streams will possibly differ by one or several pictures as illustrated in Figure 5. This is due to the decoding algorithm that could drop frames that were too damaged during transmission or by rate–adaption schemes of the encoder. Frame drops would render a PSNR calculation no longer suitable, since the corresponding pictures of

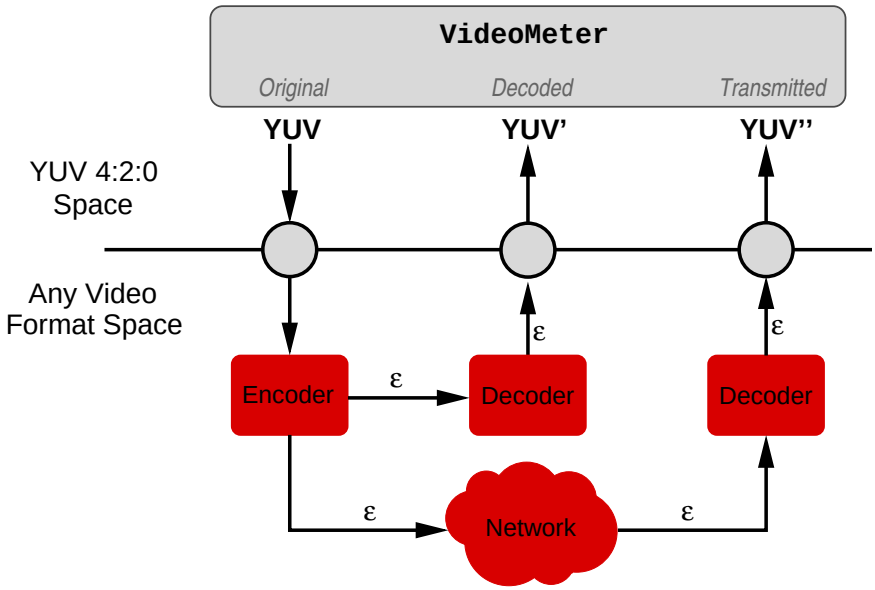


Figure 3: Overview of YUV and an arbitrary video compression scheme in the VideoMeter context.

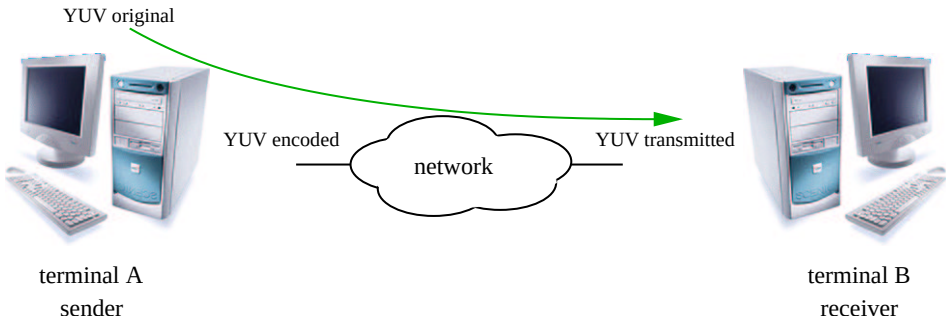


Figure 4: Relationship between `original, encoded, and transmitted` video files.

the streams would not be compared to each other, i.e., the synchronization would be lost. An example for this loss of synchronization and the resulting error–propagation due to frame drops in a typical MPEG–encoded sequence is illustrated in Figure 6. Consider the frame dependencies in this GoP. Since the B–frames are bi–directionally encoded, they rely on the preceding and the following P– or I–frames for successful decoding. Therefore a loss of either frame makes the decoding of the B–frames impossible. In the illustrated case, the I–frame of the next GoP was lost during transmission. As a result, the two preceding B–frames lost their reference and are not decoded. This shortens the decoded stream by three frames. In addition, the first two B–frames of the following GoP (not shown here) can not be decoded. To ensure the comparison of the corresponding frames in the face of these frame drops, we employ the widely used technique of *freezing*. With freezing, the last successfully decoded picture stays in memory until the next picture is successfully decoded. Since our tool does not include any knowledge about the decoding status, a *freeze file* (see Section 5 for details) has to be provided. The freeze file contains the

numbers of the frames that have to be frozen for comparison and play–out (i.e. the frames that were not decoded).

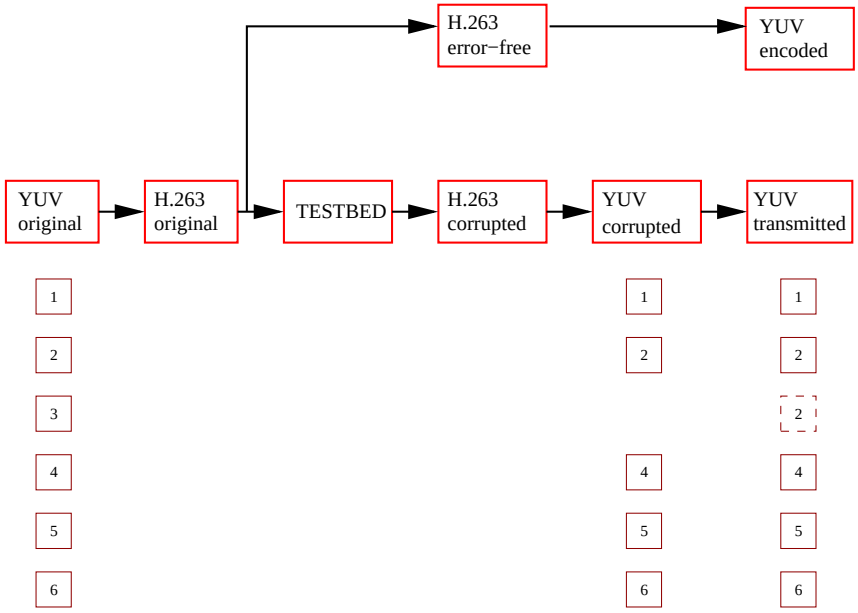


Figure 5: Methodology for PSNR calculation for video transmission over wireless links.

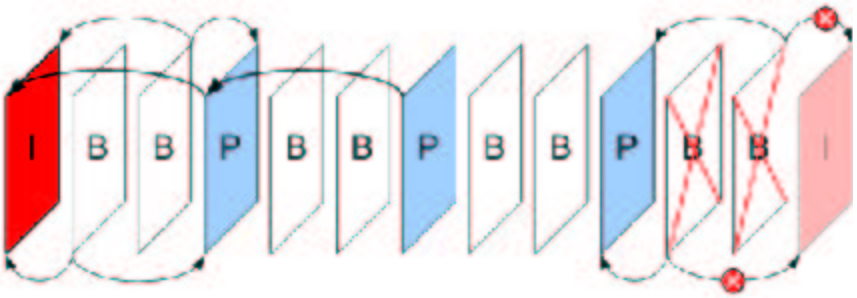


Figure 6: Lost references inside a group of pictures (GoP).

## 3 VideoMeter Usage

The tool is invoked from the command line. Several switches are used to enable or disable the features. The tool can be used as a mere YUV player, if only a single file is given and PSNR and difference picture generation are disabled. In the following we explain the usage of the switches in detail.

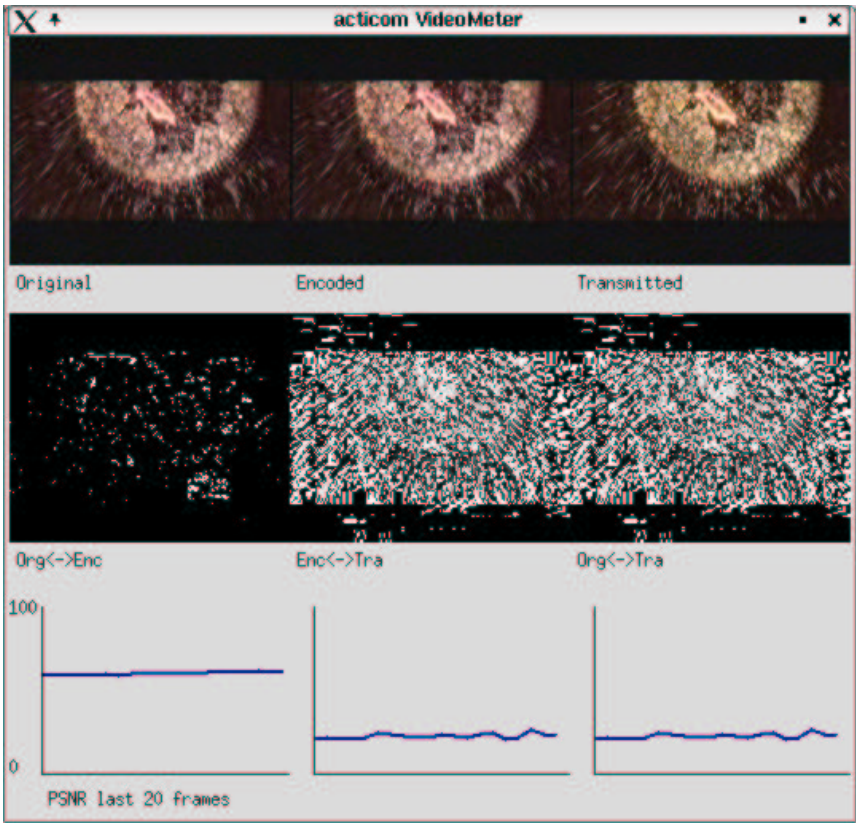


Figure 7: VideoMeter (main window).

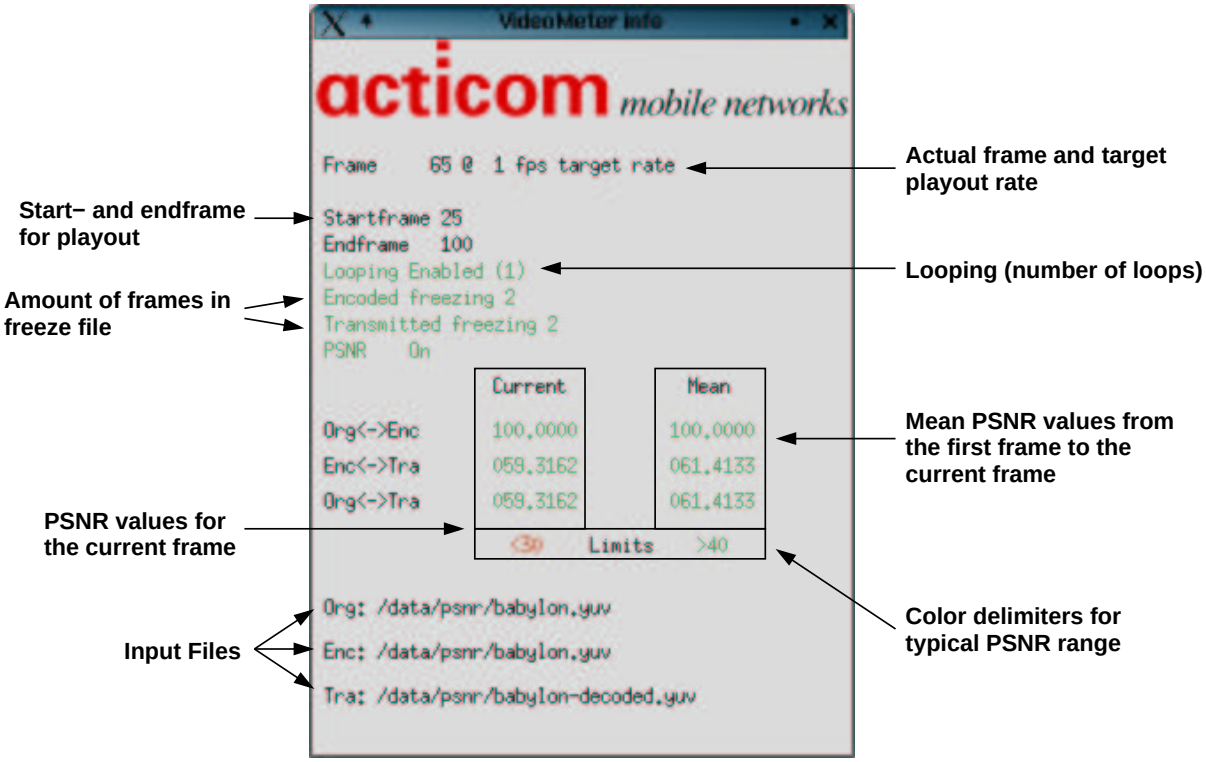


Figure 8: VideoMeter (info window).

# 4 Command Line Arguments

**Program Call:**

```
videometer [opt.switches] -f1 yuv-file [-f2 yuv-file] [-f3 yuv-file]
```

**Optional Switches:**

`-l`
Loops the playback. The number of looped playbacks is displayed in the info window.

`-nodiff`
Disables the output of the difference pictures. If PSNR is enabled, then the `-nodiff` switch is ignored and the PSNR values are calculated.

`-PSNR`
Shows a graphical representation of the last 20 PSNR values on the bottom of the main window. In addition, the current PSNR values and the mean from the beginning of the playback are shown in the info window.

`-fps n`
The program will target a play–out rate as specified with `n`. If the processing takes longer, it plays as fast as it can. The default value is 25 for PAL rate playback.

`-quiet`
Disables the info window. You can also simply minimize it during playback.

`-s n`
The playback starts with the frame `n` of the file `f1`. Default start is the first frame.

`-e n`
The playback ends with the frame `n` of the file `f1`. Default end is the last frame.

`-CIF`
Specifies the input YUV–sequence is in CIF–format. Default format is QCIF.

`-fr2 name`
Specifies the file that holds numbers of the missing frames in file `f2`.

`-fr3 name`
Specifies the file that holds numbers of the missing frames in file `f3`.

## 5 Freeze File

The *freezefile* contains the frames that are to be frozen for the specified stream during the playback for the difference picture and PSNR calculation. Freeze files can be provided for both the encoded as well as the transmitted YUV sequences. In the first case, a rate–adapting encoder setting may be forcing a frame drop, whereas in the latter case the damaged frame may be unrecoverable and will have to be discarded by the decoder. In both cases the resulting YUV streams will have fewer frames than the original sequence. In the following figures, red colored items represent errors and losses while green items represent fixed errors. Figure 9 shows an example of several losses in the YUV files. The *encoded* file misses frame $n$ due to rate adaption of the encoder while the *transmitted* file additionally suffers a second frame loss of frame $n+1$ due to transmission errors. Figure 10 gives an example of how the synchronization between the three YUV streams is lost due to the errors described above. The result is lost synchronization of the streams from the first error onwards. (We note that we envision that the tool is primarily used for the off–line comparison of YUV sequences. In an on-line experiment it would of course not be possible that the original frame $n$ aligns with the future frame $n+1$.)

As another effect, the resulting PSNR comparison is no longer valid in terms of comparing the quality losses thereafter. To fix this lost synchronization, we incorporate the possibility for the frame freezing. Figure 11 shows both the freeze file as well as the resulting comparisons of the YUV frames. The synchronization is regained by keeping the frame prior to a missing one in memory.

The freeze file should be in plain text format providing the numbers of the missing frames, starting from 0 for the first frame of the sequence. Each frame should be in a new line. No freeze file can be provided for the *original* sequence, since we assume that this is the one without any missing frames.

Example:

```
25
30
44
50
...
```

To generate the freeze file, one may utilize a bitstream parser. The parser needs to be customized for the particular video standard that is employed, such as MPEG4, H.263(+), and H.26L The parser reads the video frame number and the display time from the encoded bit stream. By comparing two parser output files, e.g., for the original and the transmitted video stream, the freeze file is generated. The parser is out of the scope of this technical report, especially since we want to emphasize the independence of the VideoMeter from video compression schemes.

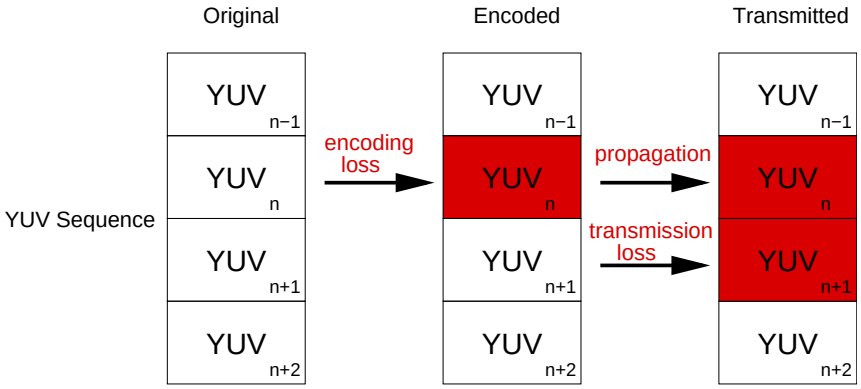


Figure 9: Example for missing frames in YUV streams.

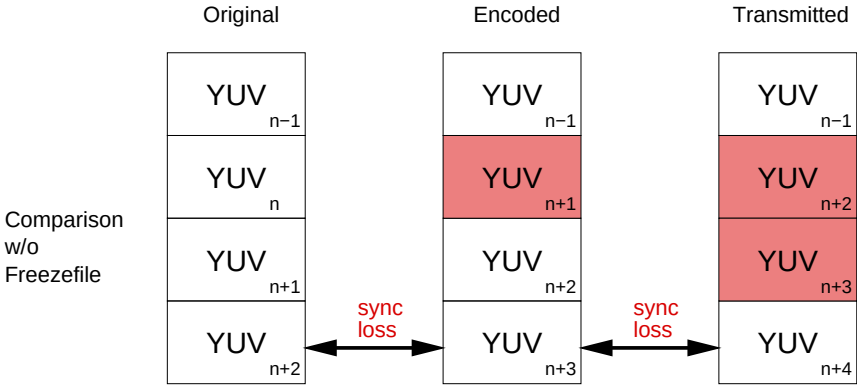


Figure 10: Example for lost sync while comparing erroneous streams.

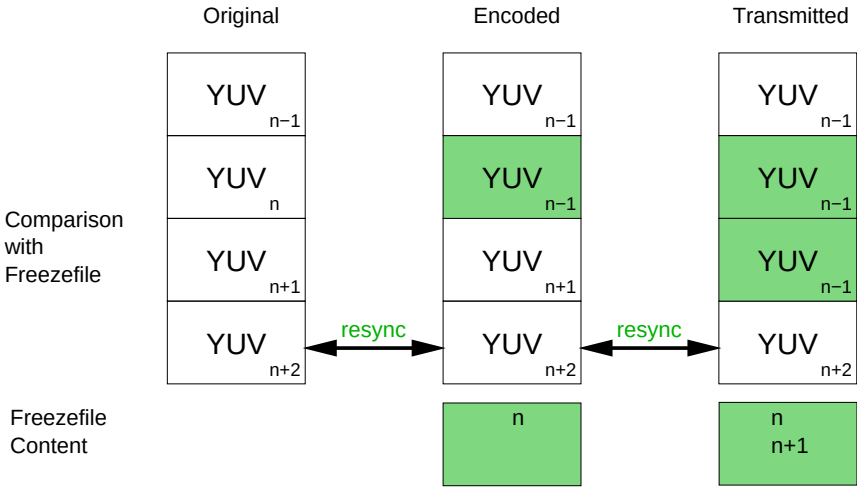


Figure 11: Example for freezefile utilization to regain synchronization.

## 6 Download Information

The VideoMeter tool can be downloaded from the following web pages:

**acticom GmbH**
P. Seeling, F. H.P. Fitzek
Research & Development
`http://www.acticom.de/videometer.html`

**Arizona State University**
M. Reisslein
Department of Electrical Engineering , Telecommunications Research Center
`http://www.eas.asu.edu/~mre/videometer`

## References

[1] A. Basso, I. Dalgic, F. Tobagi, and C. Lambrecht. Study of MPEG–2 coding performance based on a perceptual quality metric. In *Proceedings of PCS'96*, pages 263–268, Australia, March 1996. 4

[2] B. Girod and N. Färber. *Compressed Video Over Networks*, chapter Wireless Video. November 1999. 3, 4

[3] Video Quality Expert Group. Final Report From the Video Quality Experts Group on the Validation of Objective Models of Video Quality Assessment, 2000. 3

[4] J. Lu, V. Algazi, J. Robert, and B. Estes. Comparative study of wavelet image coders, 1996. 3

[5] W. Osberger, N. Bergmann, and A. Maeder. An Automatic Image Quality Assessment Technique Incorporating Higher Level Perceptual Factors. pages 414–418. 3

[6] Thrasyvoulos N. Pappas and Robert J. Safranek. Perceptual Criteria for Image Quality Evaluation. 3

[7] D. Schilling and P. C. Cosman. Image Quality Evaluation based on Recognition Times for Fast Image Browsing Applications. 3

[8] S. Winkler. Issues in vision modeling for perceptual video quality assessment. *Signal Processing*, 78(2), 1999. 3