

Video Tester – A multiple-metric framework for video quality assessment over IP networks

Iñaki Ucar*, Jorge Navarro-Ortiz†, Pablo Ameigeiras† and Juan M. Lopez-Soler†

* Dept. of Automatics and Computer Science

Public University of Navarra, Campus Arrosadia, 31006 Pamplona, Spain

i.ucar@unavarra.es

† Dept. of Signal Theory, Telematics and Communications

University of Granada, C/ Periodista Daniel Saucedo Aranda s/n, 18071 Granada, Spain

{jorgenavarro,pameigeiras,juanma}@ugr.es

Abstract—This paper presents an extensible and reusable framework which addresses the problem of video quality assessment over IP networks. The proposed tool (referred to as Video-Tester) supports raw uncompressed video encoding and decoding. It also includes different video over IP transmission methods (i.e.: RTP over UDP unicast and multicast, as well as RTP over TCP). In addition, it is furnished with a rich set of offline analysis capabilities. Video-Tester analysis includes QoS and bitstream parameters estimation (i.e.: bandwidth, packet inter-arrival time, jitter and loss rate, as well as GOP size and I-frame loss rate). Our design facilitates the integration of virtually any existing video quality metric thanks to the adopted Python-based modular approach. Video-Tester currently provides PSNR, SSIM, ITU-T G.1070 video quality metric, DIV and PSNR-based MOS estimations. In order to promote its use and extension, Video-Tester is open and publicly available.

Index Terms—Performance evaluation, objective and subjective measures, traffic and performance monitoring, networking and QoS, IPTV & Internet TV.

I. INTRODUCTION

During the last years, there has been a continuous growth of video traffic over packet-switched networks [1]. Users' demand is determinant in the proliferation of a wide range of video-based services and applications such as video on demand (VoD), live streaming, video calling, video instant messaging, video monitoring, webcam traffic, Internet video to TV, live Internet TV, mobile video, etc. All these heterogeneous services are designed to fulfill different purposes and requirements [2]. In spite of their dissimilarities, they all share a common need. Namely, to make use of tools for video quality evaluation.

The issue of video quality assessment (by defining the proper metrics) is of paramount importance for codec design,

different transmission methods evaluation and network planning among others. Indeed, this problem has generated a lot of contributive research efforts [3] along with the specification of several video quality measurement standards [4].

However, public available tools that systematize the video quality assessment over IP networks are scarce. In this respect, it is also valuable that the designed assessment tool will be highly extensible and reusable since it eases the quick inclusion of any metric.

In this work, we propose Video-Tester, an open-source modular framework [5] which provides a complete and flexible solution to the problem of video quality assessment over IP networks. Video quality evaluation is a multidimensional problem that requires to take into consideration multiple criteria, sometimes located at different levels. Our tool works at any of the three layers involved in any video over IP service—the bitstream level, the packet level and the upper picture level [6]—. As a remarkable benefit, Video-Tester is able to support any existing and future video quality metric.

The rest of the paper is organized as follows. Section II provides the state of the art of the addressed problem. Section III summarizes the proposed framework. It explains our Video-Tester design and also reviews the current supported metrics. Section IV outlines some experimental results that validate the proposed framework and, finally, Section V concludes the paper and discusses future work and extensions.

II. RELATED WORK

Several quality evaluation toolboxes have been recently proposed, like the implemented by Murthy and Karam [7]. However, in general, these applications are mainly restricted to codec testing and metric comparisons, because they do not consider multimedia transmission methods and the degradation and quality impact derived from this process.

On the other hand, it is also remarkable the seminal work (EvalVid) proposed by Klaue *et al.* [8], a widely used tool-set for video transmission evaluation. This tool-set does actually include transmission, some quality of service (QoS) parameter estimation (packet loss, delay and jitter) and some bitstream parameter evaluations (frame loss rate and frame jitter). EvalVid video quality evaluation techniques include

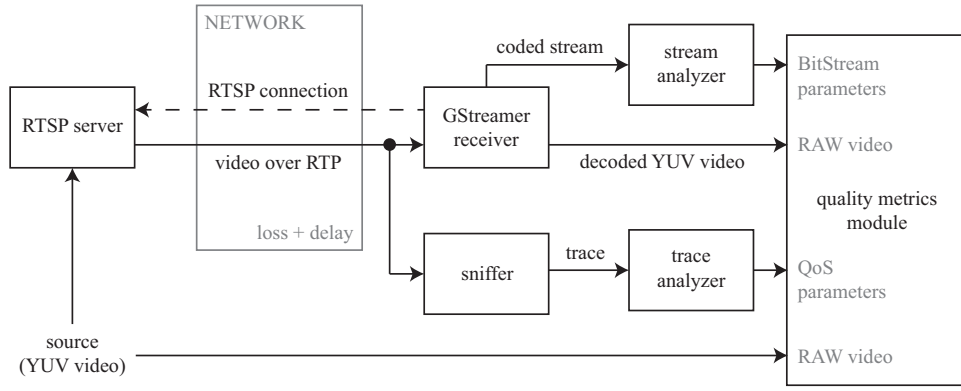


Fig. 1. Operation scheme.

PSNR, Structural Similarity index (SSIM) [9], Mean Opinion Score (MOS, mapped from PSNR) and Distortion in Interval (DIV, mapped from MOS as described in [10]).

However, this is a multi-tool framework that requires 3rd party tools like *tcpdump* (packet capture), *ffmpeg* (video encoding and decoding), *netcat* and *MP4Box* [11] (video multiplexation). This fact makes the automatization of its use a bit difficult. Furthermore, the use of MP4 multimedia containers restricts the available codecs. Added to that, the election of ISO-C programming language is definitively appropriate for code efficiency (not so for porting because of the necessary 3rd party tools), but if new quality metrics are needed, the source code must be modified.

Additionally, a common impairment in video transmission evaluation is related to frame losses. The decoded RAW video at the receiver end will have fewer frames than the original one if packet losses occur. As a consequence, we have to deal with frame misalignment in order to apply picture metrics like PSNR. In [8], it is stated that Fix-Video (FV) tool solves this problem. Nevertheless, to the author's knowledge the actual version of this framework [12] does not include the FV tool.

III. PROPOSED FRAMEWORK

Video-Tester comprises a single command-line application that works with a number of configuration files that ease its script-based execution. Once Video-Tester is launched at the two particular end-points, one side acts as the server and the other one as the client. The client side can optionally be launched with a user-friendly graphical user interface (GUI) in order to plot metrics automatically.

Video-Tester is written in Python in order to promote its extensibility. Video processing (specifically encoding, decoding and transmission) is performed by using the valuable GStreamer library [13] due to its broad and sustained support by the research community. As a beneficial consequence, the codec support is subjected to the GStreamer support and, hence, it is not constrained to a given container format.

A. Video-Tester design and operation

Figure 1 shows the general operation scheme. The server side is concurrent and talks to clients using the XML-RPC

protocol. Both the server and the client are able to work behind a NAT router (with port redirection at the server side). Each client is able to select the following parameters in order to start a new test:

- The uncompressed video file to transmit, selected from a common local database (Video-Tester accepts lossless encoded videos like [14]).
- The codec (H.263 [15], H.264 [16], MPEG-4 Visual [17] and Theora [18] are currently available).
- The bitrate (not all bitrates are available with some codecs).
- The frame rate.
- The transmission method (RTP over UDP unicast or multicast, and RTP over TCP are supported).

Then, the server side sets up a GStreamer-based RTSP server (see [19]) instance with the above parameters. At the client side, a GStreamer-based client receives the video signal. Remarkably, it is programmed to keep the frame rate constant. That is, if transmission losses cause the frame rate to fall, the receiver end duplicates properly the available frames.

At the same time and executed in a separate thread, the packet-sniffer saves concurrently a network trace. So that whenever the transmission is completed, the following files are provided:

- The network trace (PCAP file [20]).
- Received compressed video (probably impaired with losses).
- Received uncompressed video (probably impaired with losses).
- Reference video (from the local database).

At this point, the client side performs the offline analysis. From the RTP trace, we extract the following QoS parameters:

- Packet size.
- RTP sequence number.
- RTP timestamp.
- Packet arrival time.

From the coded stream, we extract the following bitstream parameters:

- Frame type (I, P, B).

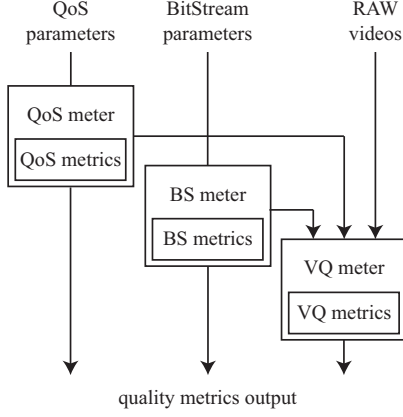


Fig. 2. Quality metrics module.

- Frame size.

The quality metrics module receives all this information. Figure 2 shows a detailed overview of this module. It contains three submodules called *meters* that isolate the implemented metrics from the rest of the application logic. Meters keep track of the available metrics and they communicate with those metrics through a standard interface. Therefore, implementing a new metric is as easy as writing a new Python class and registering it at the proper meter, without modifying the application core.

B. Supported metrics

The *QoS submodule* provides the following metrics:

- Latency (L), averaged from several (N) round-trip time (RTT) values.

$$L = \frac{1}{N} \sum_{n=1}^N \frac{RTT(n)}{2} \quad (1)$$

- Packet inter-arrival time (Δ), where R_i is the arrival time for i -th packet.

$$\Delta(i) = R_i - R_{i-1} \quad (2)$$

- Jitter (J), as described in [21], where R_i, R_j are arrival times and S_i, S_j are RTP timestamps for packets i, j .

$$\begin{aligned} D(i, j) &= (R_j - R_i) - (S_j - S_i) \\ &= (R_j - S_j) - (R_i - S_i) \end{aligned} \quad (3)$$

$$J(i) = J(i-1) + \frac{|D(i-1, i)| - J(i-1)}{16} \quad (4)$$

- Clock skew (T), the relative offset between packet arrival time and RTP timestamp, where R_i is the arrival time and S_i is the RTP timestamp for packet i .

$$T(i) = S_i - R_i \quad (5)$$

- Instantaneous bandwidth (B) for packet i , where $Size_n$ is the size of packet n and N is the number of packets

in the last second.

$$B(i) = \sum_{n=i-N}^i Size_n \quad (6)$$

- Packet loss rate (PLR), where Seq_n is the RTP sequence number of packet n and N is the total number of packets.

$$PLR = \frac{1}{N} \sum_{n=2}^N Seq_n - (Seq_{n-1} + 1) \quad (7)$$

- Packet loss distribution (PLD). If we divide the transmission time in K intervals, we can apply (7) to every interval k .

$$PLD(k) = \frac{1}{N_k} \sum_{n=2}^{N_k} Seq_{n,k} - (Seq_{n-1,k} + 1) \quad (8)$$

The *bitstream (BS) submodule* provides the following metrics:

- Video stream framing structure (at the receiver end, probably impaired with frame losses).
- Reference video stream framing structure (with no losses).
- Group of pictures (GOP) size, averaged from measured GOP sizes once atypical values has been discarded.

$$GOPsize = E[X] \quad (9)$$

with $X = \{ GOP_n :$

$$GOP_n \in [E[GOP] - \sigma, E[GOP] + \sigma]\}$$

- I-frame loss rate. We count an I-frame loss when

$$GOP_n > E[GOP] + \sigma \quad (10)$$

The *video quality (VQ) submodule* provides the following metrics:

- PSNR.
- SSIM index [9].
- MOS, mapped from PSNR [10].
- DIV, mapped from MOS (as proposed in [10]).
- As an illustration, we provide ITU-T G.1070 video quality metric [22], because it provides an example of gathering information from other submodules, specifically the packet loss rate.

IV. EXPERIMENTAL RESULTS

A. Frame misalignment

To test the effectiveness of the Video-Tester frame rate sustainer procedure, a particular scenario with 2 % of packet loss probability was chosen. The video `akiyo_cif.264` [14] was encoded with H.263 (GOP size of 15) at 128 kbps and 25 fps, and transmitted over UDP unicast. 600 video transmissions were performed. For each transmission, we get the received video with and without the Video-Tester frame rate sustainer element.

We choose 5 frame positions (70, 120, 170, 220 and 270 of 299 total frames) from the reference video in order to compare

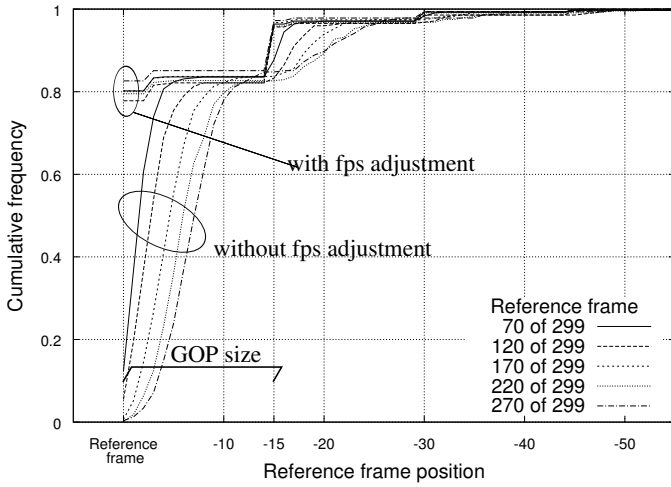


Fig. 3. Frame misalignment test.

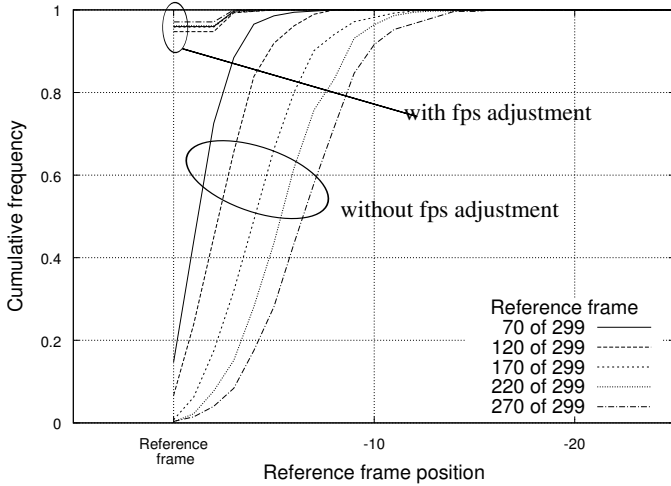


Fig. 4. Frame misalignment test (once the videos without the first frame were removed).

misalignments through all the tests. Figure 3 shows the cumulative frequency of the position occupied by the reference frame. The received video presents severe misalignments in absence of the frame rate sustainer. Instead, our method highly increases the probability that the frames are aligned. However, note that there is an anomaly at position -15 .

This anomaly can be explained attending to the GOP size. When the loss occurs at the first frame of the first GOP (an I-frame), the remaining frames (P or B) have no reference to be decoded. Therefore, the loss of this first frame implies the loss of the whole GOP. To the receiver, the video transmission starts one GOP later, so the frame rate sustainer has no work. This point can be proved with Figure 4. It shows the same analysis, once the videos without the first frame were removed.

Accordingly, in order to do reliable Full-Reference calculations (like PSNR) with our frame adjustment method, we can dismiss the first GOP if the received video lacks the first frame.

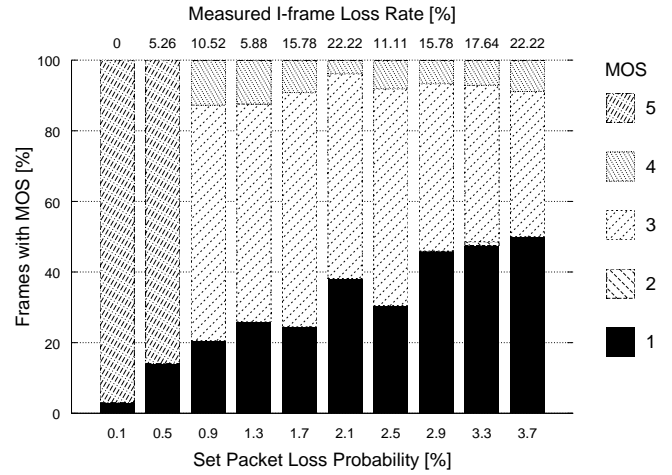


Fig. 5. MOS ratings measured in scenarios with different packet loss probability.

B. Exemplary tests

A bank of tests was designed in order to show the kind of results that can be achieved with our Video-Tester tool. The video `akiyo_cif.264` [14] was encoded with H.263 at 300 kbps and 25 fps, and transmitted over UDP unicast. 10 tests were launched with increasing packet loss probability, from 0.1 % to 3.7 %. Within each test, we measured the I-frame loss rate and the percentage of frames with certain level of MOS.

Results are shown in Figure 5. Each stacked bar belongs to one test. The bottom x-axis shows that tests are ordered from left to right with increasing packet loss probability. The top x-axis shows the measured I-frame loss rate for each test. As expected, we can observe a progressive deterioration of both I-frame loss rate and MOS ratings.

V. CONCLUSION

This paper proposes Video-Tester, an extensible and reusable single framework for video quality assessment over IP networks. On the basis of two end-points (public or private IP addresses), it comprises all the procedures involved in video over IP communications. More specifically, it accounts from capturing to rendering video, which involves encoding, sending, receiving and decoding the video signal.

Video-Tester estimates the proper parameters and characteristics required for the overall system evaluation, and due to its modularity, it can evaluate the impact of any of the components—at any level—on the final video quality.

Notably, Video-Tester is a Python-based tool and, hence, it benefits from Python key distinguishing features. It also uses the valuable GStreamer library [13]. More precisely, to achieve the greatest flexibility and for making it maximally configurable, a GStreamer RTSP server [19] is designed. Moreover, a procedure to solve the frame misalignment problem in lossy scenarios is proposed.

Summing up, Video-Tester includes the main features of previous works and adds further improvements in terms of

usability, extensibility, codec support, support of transmission methods and metric robustness (frame alignment) in case of losses. The wide range of extracted parameters allows the implementation of virtually any kind of video quality metric: Full-Reference, No-Reference or Reduced-Reference metrics; as well as any level picture-, packet-, bitstream-based or even hybrid metrics.

ACKNOWLEDGMENTS

This work was partially supported by the “Ministerio de Ciencia e Innovación” of Spain under research project TIN2010-20323.

REFERENCES

- [1] (2012) Cisco visual networking index: Forecast and methodology, 2010-2015. [Online]. Available: <http://www.cisco.com/> 1
- [2] W. Simpson, *Video Over IP*, 2nd ed. Focal Press, 2008. 1
- [3] S. Chikkerur, V. Sundaram, M. Reisslein, and L. Karam, “Objective video quality assessment methods: A classification, review, and performance comparison,” *Broadcasting, IEEE Transactions on*, vol. 57, no. 2, pp. 165–182, Jun. 2011. 1
- [4] S. Winkler, “Video quality measurement standards – Current status and trends,” pp. 1–5, Dec. 2009. 1
- [5] I. Ucar, J. Navarro-Ortiz, P. Ameigeiras, and J. Lopez-Soler. (2012) Video Tester – A video quality assessment tool. [Online]. Available: <http://code.google.com/p/video-tester/> 1
- [6] S. Winkler and P. Mohandas, “The evolution of video quality measurement: From PSNR to hybrid metrics,” *IEEE Trans. Broadcasting*, vol. 54, no. 3, pp. 660–668, Sep. 2008. 1
- [7] A. Murthy and L. Karam, “A MATLAB-based framework for image and video quality evaluation,” in *Quality of Multimedia Experience (QoMEX)*, 2010 Second International Workshop on, Jun. 2010, pp. 242–247. 1
- [8] J. Klaue, B. Rathke, and A. Wolisz, “EvalVid – A framework for video transmission and quality evaluation,” in *Modelling Techniques and Tools for Computer Performance Evaluation, 13th International Conference on*, Urbana, Illinois, USA, Sep. 2003, pp. 255–272. 1, 2
- [9] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: from error visibility to structural similarity,” *Image Processing, IEEE Transactions on*, vol. 13, no. 4, pp. 600–612, Apr. 2004. 2, 3
- [10] J. Gross, J. Klaue, H. Karl, and A. Wolisz, “Cross-layer optimization of OFDM transmission systems for MPEG-4 video streaming,” *Computer Communications*, vol. 27, pp. 1044–1055, Jul. 2004. 2, 3
- [11] (2012) GPAC, an open source multimedia framework for research and academic purposes. [Online]. Available: <http://gpac.wp.institut-telecom.fr/> 2
- [12] J. Klaue. (2012) EvalVid – A video quality evaluation tool-set. [Online]. Available: <http://www2.tkn.tu-berlin.de/research/evalvid/> 2
- [13] (2012) GStreamer – Open source multimedia framework. [Online]. Available: <http://gstreamer.freedesktop.org/> 2, 4
- [14] (2012) YUV CIF reference videos (lossless H.264 encoded). [Online]. Available: <http://www2.tkn.tu-berlin.de/research/evalvid/cif.html> 2, 3, 4
- [15] *Video coding for low bit rate communication*, ITU-T Recommendation H.263, Jan. 2005. 2
- [16] *Advanced video coding for generic audiovisual services*, ITU-T Recommendation H.264, Mar. 2010. 2
- [17] *Information technology – Coding of audio-visual objects*, ISO/IEC Standard 14496-2, Dec. 2001. 2
- [18] (2012) Theora specification. [Online]. Available: <http://www.theora.org/doc/Theora.pdf> 2
- [19] (2012) GStreamer RTSP server. [Online]. Available: <http://gstreamer.freedesktop.org/modules/gst-rtsp-server.html> 2, 4
- [20] (2012) Programming with pcap. [Online]. Available: <http://www.tcpdump.org/pcap.html> 2
- [21] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, “RTP: A transport protocol for real-time applications,” RFC 3550, Jul. 2003. 3
- [22] *Opinion model for video-telephony applications*, ITU-T Recommendation G.1070, Apr. 2007. 3