VideoLat - an Extensible Tool for Multimedia Delay Measurements

Jack Jansen
CWI: Centrum Wiskunde & Informatica
Science park 123
1098 XG Amsterdam, the Netherlands
+31 20 5924300
Jack.Jansen@cwi.nl

ABSTRACT

When using a videoconferencing system there will always be a delay from sender to receiver. Such delays affect human communication, and therefore knowing the delay is a major factor in judging the expected quality of experience of the conferencing system. Additionally, for implementors, tuning the system to reduce delay requires an ability to effectively and easily gather delay metrics on a potentially wide range of settings. In order to support this process, we make available a system called *videoLat*. VideoLat provides an innovative approach to understand glass-to-glass video delays and speaker-to-microphone audio delays.

VideoLat can be used as-is to do audio and video roundtrip delays of black box systems, but by making it available as open source we want to enable people to extend and modify it for different scenarios, such as measuring one-way delays or delay of camera switching.

Categories and Subject Descriptors

B.8.2 [**Hardware**]: Performance and Reliability - Performance Analysis and Design Aids; H.4.3 [**Information System Applications**] Communication Applications - Computer conferencing, teleconferencing, and videoconferencing.

General Terms

Measurement, Performance, Experimentation, Human Factors.

Keywords

Delay measurement; video conferencing.

1. INTRODUCTION

Audio and video playout delays are a fact of life for conferencing systems and other multimedia processing systems. These delays are unavoidable: each and every component in a digital conferencing chain will introduce a non-zero delay, and even in an ideal world where all processing was instantaneous and all networks were infinitely fast, there would still be the delays introduced by time-slotted standards (such as frame rates), and by the physical distance between sender and receiver and the fact that the speed of light is finite.

Delays in videoconferencing have a large effect on the way people communicate [4][5][6] and are an important factor in the quality of experience of a conferencing system. Therefore, being able to measure this delay is a first step in evaluating the performance of a video conferencing system, and possibly optimizing it. Being able to measure audio and video delays is valuable in a wide range of scenarios and target audiences, such as

- evaluating closed commercial conferencing systems for their applicability to a certain use case (end users),
- determining the effect of the latest codec optimizations implemented (developers),
- comparing network protocol change implications (network operators).

The current standard way to measure user-perceived delay (as opposed to measuring only the network or codec delay, which is only part of the picture) is to presume that audio delay is less - or less important - than video delay and measure only the latter. Video delay is then measured by putting sending and receiving system side by side, capture input and output of the system being measured in a single shot with an extra camera, augmented by a timer. The footage of this extra camera is then examined offline, usually manually, after which the delay is computed. This is potentially a time-consuming and error-prone process. Xu et al [7] improve on this by using OCR for the offline matching, but their system still requires sender and receiver to be physically side by side. The vDelay [1] and AvCloak [3] tools go a step further and can do automatic measurements in realtime of a real conferencing setup, with the latter also measuring audio/video synchronization. However, neither can be used on turnkey systems, because they simulate hardware or require screen scraping to do their job. This method also means these systems do not include all camera and display delays as perceived by the end user.

In this paper we present videoLat, an open source tool that enables glass-to-glass video delay measurements, and sound-to-sound audio delay measurements. VideoLat is open source, and designed to be extensible, making it adaptable to other multimedia delay measurements. For example, measuring stream switching delays, or A/V synchronization requires only a minimal amount of code.

A previous version of videoLat was described in [2] but while the principle of operation is the same, this version is a completely modified code base to enable a better user interface, extensibility, and support for non-visual media (such as audio).

MM '14, November 03 - 07 2014, Orlando, FL, USA.
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-3063-3/14/11…$15.00.
http://dx.doi.org/10.1145/2647868.2654891

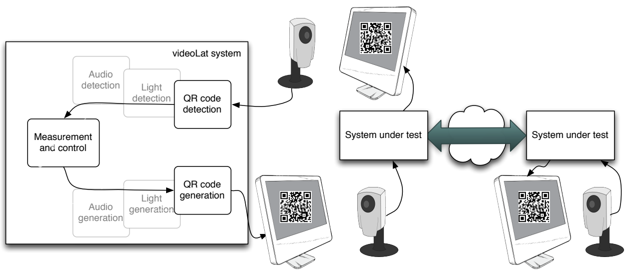


Figure 1 - Black Box Measurement

2. DESIGN

VideoLat is intended to be the "digital multimeter" for multimedia delay testing: a tool you pick up to get a quick idea of how a system performs, from an end user point of view, independent of the system under test, and easy to use. This has led to a number of design principles:

- results reported should be as close as possible to user-perceived delays,
- videoLat itself runs on a dedicated system, not on the system under test,
- setup and operation should not put much of a burden on the operator.

The general design is based on videoLat generating images (or sounds) that are computer-detectable. These are then fed through the system under test, which should reproduce them (after a delay) on their output device, from which the videoLat system will pick them up again. Figure 1 shows how this works for a video delay measurement: the videoLat display shows a QR-code which the conferencing camera picks up and transmits. The remote conferencing camera is pointed at the screen (or at a mirror so it can see the screen) and transmitted back. Here the videoLat camera picks it up again and the QR-code detector will trigger once the previous QR-code transmitted is received. The delay is computed, and the whole procedure repeated with a new QR-code. After a large number of measurements the average delay and standard deviation are computed.

There is one caveat: this delay includes the delay caused by the internal processing of videoLat itself. Therefore, before doing a real measurement, the operator should first do a calibration run with the exact same videoLat system, as shown in figure 2. This self-delay will then be subtracted from the real measurement.

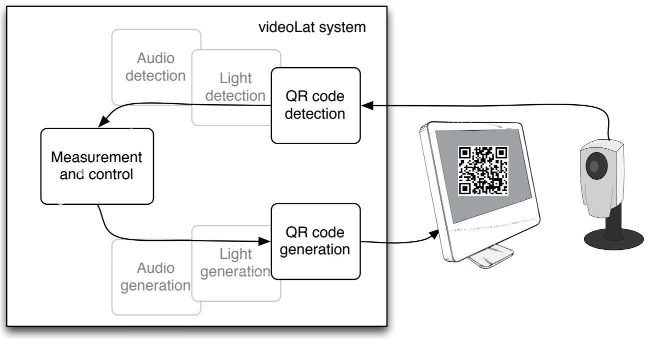


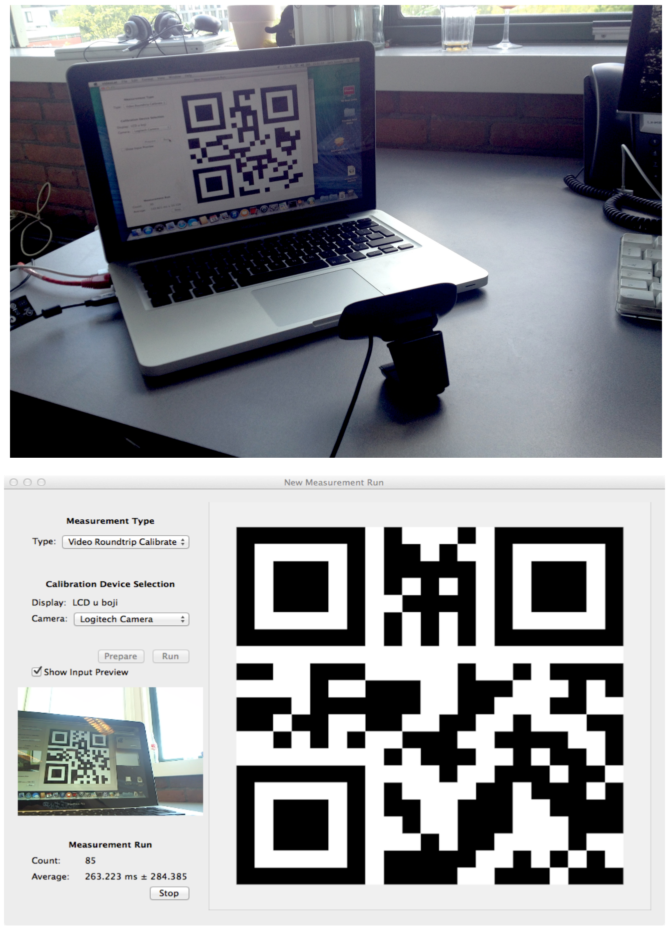
Figure 2 - Calibration Self-measurement

3. EXAMPLE USE CASE

Doing a video delay measurement of Skype (or any other conferencing videoconferencing tool) from scratch requires a MacBook and an external USB webcam, in addition to the machine used for Skype. Technically a desktop Mac or a builtin camera work, but then positioning the hardware becomes difficult. After downloading videoLat via http://www.videolat.org (or building it as explained in section 5) you do a calibration run by selecting the *New Measurement* command and selecting *Video Roundtrip Calibration*. Figure 3 show how this setup works in practice. Builtin help is available on the details. You point the camera at the videoLat screen and run the calibration, which shows the running average and standard deviation. Typically you would do about thousand measurements, but you can watch the average and standard deviation to guide this.

After you press *Stop* you see the results of your calibration as shown in figure 4 and if the distribution looks reasonable you save the calibration. Practically speaking, the calibration run shown in figure 4 is less than optimal: ideally the delays of the calibration run should be close to normally distributed because this will reduce the impact of measurement system idiosyncrasies on your real measurement.

You now setup a Skype call, and instruct the remote participant to either point their camera at their screen (if physically possible) or use a mirror so the camera can see the Skype window. Locally, you make sure the Skype camera see the MacBook screen and the MacBook camera sees the Skype screen. You then run videoLat again, this time selecting a *Video Roundtrip* measurement and



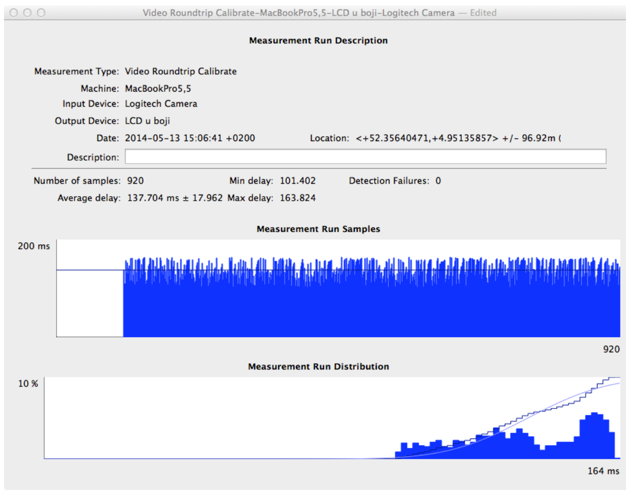
Measurement Type
Type: Video Roundtrip Calibrate

Calibration Device Selection
Display: LCD u boji
Camera: Logitech Camera

☑ Show Input Preview

Measurement Run
Count: 85
Average: 263.223 ms ± 284.385

Figure 3 - Calibration setup, picture (a) and screen shot (b)



Video Roundtrip Calibrate–MacBookPro5,5–LCD u boji–Logitech Camera — Edited

Measurement Run Description

Measurement Type: Video Roundtrip Calibrate
Machine: MacBookPro5,5
Input Device: Logitech Camera
Output Device: LCD u boji
Date: 2014–05–13 15:06:41 +0200 Location: <+52.35640471,+4.95135857> +/- 96.92m (
Description:

Number of samples: 920 Min delay: 101.402 Detection Failures: 0
Average delay: 137.704 ms ± 17.962 Max delay: 163.824

Figure 4 - Calibration Results

using the previous calibration measurement as the base. During the preparation phase you position cameras and screens such that the QR codes are best visible, and videoLat will determine whether mirroring is involved and determine an estimated value for the expected delay.

After running the measurement (which will probably take considerably longer than the calibration) you get a results display similar to the one for the calibration. You can also open older measurements, for comparison, and you can save the measurement for future reference. You can also export the results as three CSV files, which can be imported into a spreadsheet or graphing application for further processing.

4. IMPLEMENTATION

VideoLat only runs on MacOSX, with an iOS version expected in the near future. The reason for this is that initial experiments showed that AVFoundation, the Apple media capture framework, was the only framework that was able to produce timestamped video frames where the timestamps showed some semblance to the actual time of grabbing. Because videoLat intends to do user-perceived measurements this is an important feature: discrepancies between the actual capture time and the recorded capture time can skew the results, especially if these discrepancies are not normally distributed or dependent on some hidden variable (such as operating system scheduler policy).

VideoLat is structured around a plugin paradigm. For each measurement type there are four objects that need to be implemented: a measurement controller, a media generator, a media grabber and a detector. The controller handles the user interface of the measurement run (to allow selection of the camera to use, for example) and drives the other three components to do successive measurements.

Aside from the QR-code components mentioned previously there is also a complete set of audio components (generator, recorder, detector) to do audio delay measurements. In addition there are video components to do video delay measurements using simple black or white images (or light/no light conditions) and components that can generate and detect light/no light using dedicated arduino or (commercial) LabJack hardware. The video black/white modules are compatible with the hardware light/no-light modules, so these can be mixed to do a delay measurement of only a camera or only a screen.

Beside the measurement functionality there is code to display and compare measurement results, and to export to CSV-files so the data can be plotted or analyzed in a spreadsheet or another external tool.

The core of videoLat is implemented in Objective C, but the APIs are all available in Python as well, and because Python also has a good interface to Foundation and other OSX toolkits it is possible to create new components either in Objective C or Python. C++ is a third option.

VideoLat uses the open source libraries libzbar [8], libzint [9] and libpng [10] for QR-code generation and detection, and the freely available libLabJack [11] when interfacing to that specific hardware. VideoLat is licensed under the GPL license.

5. BUILDING AND EXTENDING

VideoLat can be built on MacOSX 10.7 or later. To build videoLat from source go to http://sourceforge.net/projects/videolat/ and download either the source tar file or clone the subversion repository. To build without any modifications open a Terminal window and run `scripts/build.sh` which will build videoLat after building the third-party libraries libzbar and libzint, and possibly warning you that you need to build libpng yourself (which is included with some OSX versions but not always). It will tell you where the application can be found after building.

If you want to extend, modify or debug videoLat you open the project file `videoLat.xcodeproj` and build in XCode. The Debug and Release targets are what you would normally use, the Distribution target you use if you want to create a signed copy for distribution (in which case you probably know about all the Apple-enforced rules for this).

The main APIs to use when extending videoLat are defined and documented in `Protocols.h`, and generic base class implementations for those are available. Detailed documentation on extending videoLat can be found on http://www.videolat.org.

6. EXTENSION AREAS

VideoLat is made available as open source because we feel there are many other types of delay measurements that can provide interesting insights into the operation of conferencing systems that cannot be handled by a turnkey measurement system. As an example, if the system under test has an API to switch cameras it is possible to extend VideoLat so that it emits such an API call. Then, by pointing the two cameras at different printed QR-codes videoLat can be used to measure the delay until a human viewer sees the result of a camera switch.

There are also a number of areas in which videoLat could use some improvement that are of general application. For example:

- one-way measurements are not supported as well as round-trip measurements, and some use cases, such as broadcasting, would benefit from one-way measurements,
- a mobile version (iPhone, iPad) would greatly enhance the "digital multimeter" role, especially with one-way measurements,
- audio delay measurements are currently painful because of audio feedback and echo cancellation and suppression,
- audio/video synchronization measurements as in [3] would be a great addition.

The author would be delighted to cooperate with people interested in these areas.

7. ACKNOWLEDGEMENTS

This work was supported in part by funding from the European Community's Seventh Framework Programme (FP7/2007-2013) Vconect project, under grant agreement no. ICT-2011-287760.

8. REFERENCES

[1] Boyaci, O., Forte, A., Baset, S. and Schulzrinne, H.. vDelay: A Tool to Measure Capture-to-Display Latency and Frame Rate. 11th IEEE International Symposium on Multimedia, 2009, pp. 194-200. DOI=10.1109/ISM.2009.46

[2] Jack Jansen and Dick C. A. Bulterman. User-centric video delay measurements. Proceeding of the 23rd ACM Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '13). ACM, New York, NY, USA, 37-42. DOI=10.1145/2460782.2460789

[3] Kryczka, A., Arefin, A. and Nahrstedt, K... AvCloak: A Tool for Black Box Latency Measurements in Video Conferencing Applications. IEEE ISM (2013) pp. 271-278. DOI=10.1109/ISM.2013.52

[4] Claire O'Malley, Steve Langton, Anne Anderson, Gwyneth Doherty-Sneddon and Vicky Bruce. Comparison of face-to-face and video-mediated interaction. Interacting with Computers (1996) vol. 8 (2) pp. 177-192. Elsevier, Amsterdam, the Netherlands. DOI=10.1016/0953-5438(96)01027-2

[5] Karen Ruhleder and Brigitte Jordan. Co-constructing non-mutual realities: Delay-generated trouble in distributed interaction. Computer Supported Cooperative Work (CSCW) (2001) vol. 10 (1), pp. 113-138. Springer, Heidelberg, Germany. DOI=10.1023/A:1011243905593

[6] Jennifer Tam, Elizbeth Carter, Sara Kiesler, and Jessica Hodgins. 2012. Video increases the perception of naturalness during remote interactions with latency. In Proceedings of the 2012 ACM annual conference extended abstracts on Human Factors in Computing Systems Extended Abstracts (CHI EA '12). ACM, New York, NY, USA, 2045-2050. DOI=10.1145/2223656.2223750

[7] Yang Xu, Chenguang Yu, Jingjiang Li, and Yong Liu. 2012. Video telephony for end-consumers: measurement study of Google+, iChat, and Skype. In Proceedings of the 2012 ACM conference on Internet measurement conference (IMC '12). ACM, New York, NY, USA, 371-384. DOI=10.1145/2398776.2398816

[8] Zbar bar code reader. URL=http://zbar.sourceforge.net. Downloaded at 2014-05-18.

[9] Zint Barcode Generator. URL=http://zint.github.io. Downloaded at 2014-05-18.

[10] Portable Network Graphics. URL=http://www.libpng.org. Downloaded at 2014-05-18.

[11] LabJack. URL=http://labjack.com/support/software. Downloaded at 2014-05-18.