Investigations Towards Low-Bandwidth Real-Time Video for Teleoperation

Stefan Kimmer, João Rebelo and André Schiele
Faculty of Mechanical, Maritime and Materials Engineering, Delft University of Technology, Delft, The Netherlands
Telerobotics and Haptics Laboratory, ESTEC, Noordwijk, The Netherlands
Email: {stefan.kimmer, joao.rebelo, andre.schiele}@esa.int

***Abstract*—This paper outlines an experimental setup to determine optimal transmission settings for compressed video-streams over a low-bandwidth video link (96kb/s) for direct telemanipulation. The experiments have been conducted in the frame of preparations for the multi-purpose end to end robotics network (METERON) project, which aims at bilateral telemanipulation from space (the international space station) to ground (the earth). In METERON, a 256kbit/s S-Band link is available for data up-link, which needs to stream video and control data. The experimental campaign aimed at identifying suitable transmission and compression characteristics, such as the compression format, the transport protocol and the belonging parameters for optimizing following performance metrics. These are the objective video frame quality like the peak signal to noise ratio, the delay and jitter of the point in time of the frame displaying, and frame dropping. The paper will outline the experimental implementation, which consists of a camera, a compression unit, a separate network shaper which simulates the characteristics of the S-Band link, a decompression unit and a display. Moreover, experimental results are shown that indicate optimum performance for telemanipulation resulting from the trade-off between having a video with less video frame quality and having a video stream with less jitter, delay and frame dropping.**

***Index Terms*—Video Compression, H.264, METERON, Telerobotics, Real-time Video, telemanipulation, teleoperation.**

I. Introduction

During the execution of experiments of the Multi-purpose End To End Robotics Network (METERON) project a robot needs to be controlled on the earth from an astronaut on the International Space Station (ISS) [1]. To do so the astronaut requires a video stream from the ground facility. This video stream must be transferred via a ground network and then be transmitted through an S-Band link with a bandwith of 256kbit/s in total.

The video stream is meant to be received for situational awareness by human operators controlling a robot in far distance. In contrast to film movies a subjective video quality measure is here not an important requirement. Such subjective quality measures may be e.g. the MOS (Mean Opinion Source) or the DSCQS (Double Stimulus Continuous Quality Scale). For a telemanipulation, the key measures are the jitter and the reliability of the video. Clear edges and low JPEG artifacts are more necessary than local error concealment, smoothing or de-blocking filters.

This paper first gives an overview of the data communication networks which are expected to be used in a METERON experiment (Section II) and how those can be simulated (Section III). A short survey of compression formats and packing a video stream into a network stream is thereafter provided while highlighting the special needs for the telemanipulation setup in METERON (Section IV). Finally results of experiments will be provided (Section V).

II. Expected Networks

An overview of the expected antennas and networks can be seen in Fig. 1. Video data has to be transmitted from

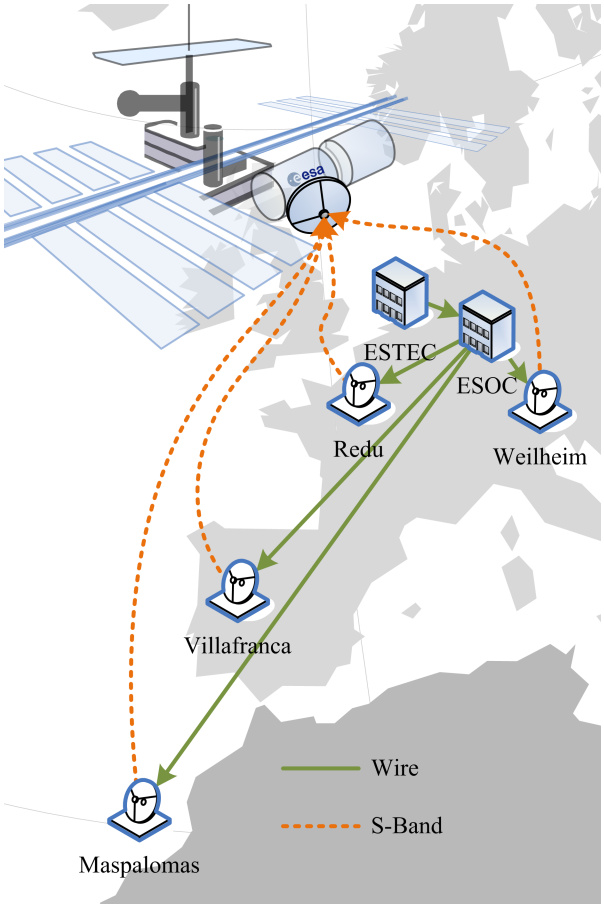

Fig. 1. The supposed network setup of the METERON project of the experiment where a robot has to be controlled at ESTEC. A video stream must be provided of the experiment set-up to an astronaut at the ISS.

the ground station ESTEC (European Space Research and

Technology Centre) to the ISS. During other phases in the METERON project, other ground facilities will need to transmit video data as well but the links are expected to have similar characteristics. The expected link from the ground stations to the ISS is explained first and then an overview of the expected ground network will be provided.

A. Ground to International Space Station link

For the data connection between the ground stations and the ISS a link will be re-used which is based on the ROKVISS (RObotic Components Verification on the ISS) experiment [2]. In this experiment an S-Band link is used which has a capacity of 256kbit/s (32kB/s) uplink and 4Mbit/s (500kB/s) downlink [3]. The data needs to be packetized according the Consultative Committee for Space Data Systems (CCSDS) Space Packet Protocol [4]. For the packetization a significant amount of bandwidth must be offered to headers and other additional data. In a METERON experiment robotics data must be transfered in addition to the video data through the data link to the ISS at the same time. This robotics data has the strict requirement of 500Hz as the frequency at which data packets must be sent. The link can be divided only into equally sized channels (see Fig. 2). One packet including the

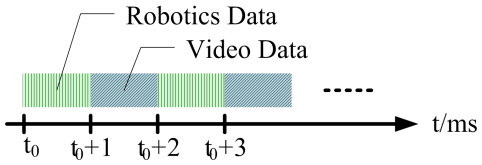

Fig. 2. Multiplexing of robotics and video data. Both blocks of data types have a layout like in Fig. 3 or modified as in Fig. 4.

header and other additional data is called a space packet. One or more space packets must further be packeted into one transfer frame. To achieve the required frequency just one packet will be in one transfer frame. As a consequence $\frac{32\text{kB/s}}{2} \cdot \frac{1}{500\text{Hz}} = 32\text{B}$ per channel each 2ms can be sent. This packet then includes a transfer frame primary header (5B), a space packet header (6B), an attached transfer frame start sequence marker (2B) and a transfer frame error control field (2B) (see Fig. 3). This makes 17B user data per packet per

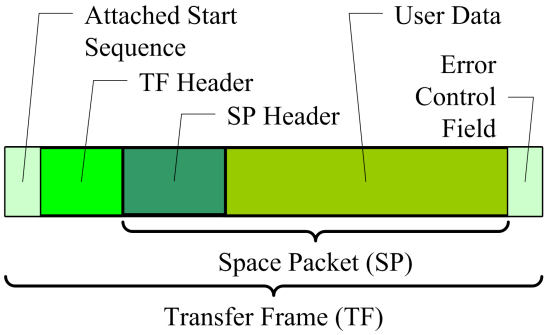

Fig. 3. The layout of one space packet in a transfer frame as in [3].

channel which is a data rate of 8.5kB/s. However the robotics data will require 22B per packet, therefore the space packet architecture is proposed to be modified. The transfer frame including one space packet will then have one packet header, one start sequence marker and one frame error control field (see Fig. 4). Thus 22B user data can be sent every 2ms per

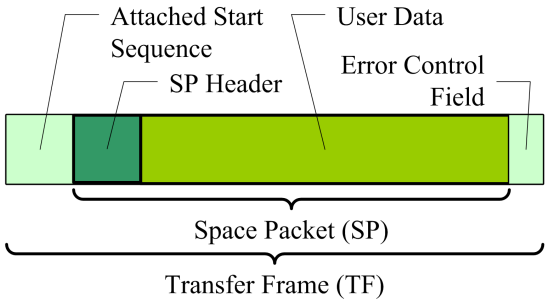

Fig. 4. A proposed modified version of one original space packet in a transfer frame. The transfer frame header is removed and other modification are made such that one sample of the robotics data fits into the user data field.

packet per channel. Considering the modification a data rate of $f_r = \frac{22\text{B}}{2\text{ms}} = 11\text{kB/s}$ for the video link are available.

The bit error rate of the S-Band link is estimated to be less than $e_b = 10^{-6}$ [3]. With a frame rate of $f_r = 12\text{Hz}$, an effective bandwidth of $b_r = 11\text{kB/s}$ and one packet per frame the size of that packet will be round $f_s = \frac{b_r}{f_r} \approx 0.92\text{kB}$. With this packet size the packet error rate is $e_p = \frac{f_s e_b}{1\text{bit}} \approx 7.36\%$ if a normally distributed probability of the bit errors is assumed. However the latter assumption does likely not reflect the real distribution. With strong confidence the bit errors will appear in bursts. This will reduce the packet error rate to 1.15% "assuming a packet size of 1024B and a mean distribution over 45 transfer frames" [3]. The jitter is guaranteed to be less than 1ms.

B. Ground Network

The video data needs to be transfered from ESTEC to the various antennas via a wire on ground (see Fig. 1). These wires are likely to be provided by dedicated Internet service providers such as NREN (National research and education network). These links will provide a significantly higher bandwidth as the S-Band link and hence a bandwidth limitation do not need to be taken into account. The jitter however is unpredictable and large with respect to the mean delay. In a test campaign [5] an existing link between ESTEC and the antenna stations has been examined at various conditions. In Fig. 5 a snapshot of the connection between ESTEC and Redu can be seen. The half of the round trip time is plotted as a function of time. In that test campaign only the round trip time was measured, but with the assumption that the link is symmetric, the half of the round trip time reflects the delay. A common pattern of the links between ESTEC and the other station are delay peaks. In figure Fig. 6 a close up of such a peak is visible. Such peaks may occur regularly as in Fig. 5 or randomly. It is import to know such peak occurrences beforehand since a data buffer must be established at the

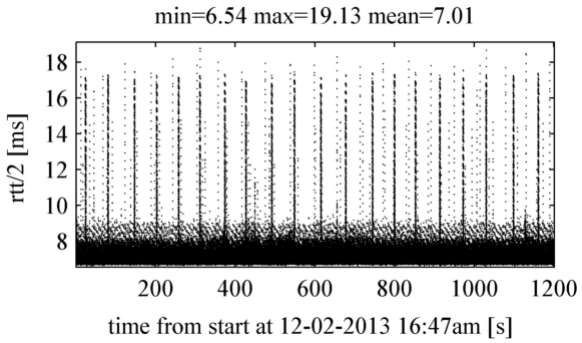

Fig. 5. This shows the half of the round trip time (delay) of the link between ESTEC and Redu as measured on 12 Feb 2013. Peaks of delays occur periodically. A close up of the highest peak in shown in 6.

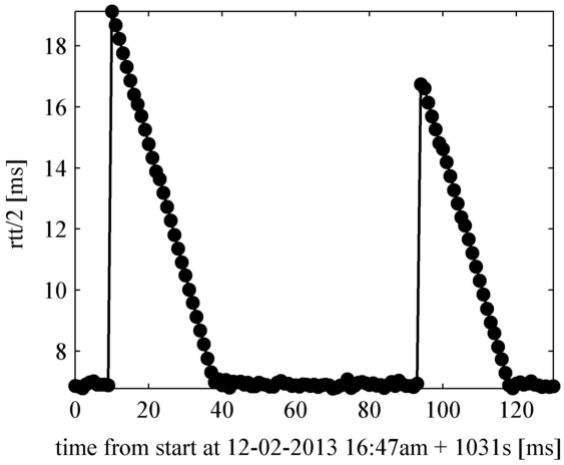

Fig. 6. Close up of the highest peak of the measurements of Fig. 5. If a buffer would be used of only 7ms all frames of that peaks would be dropped. This would increase the packet loss ratio dramatically. Hence the buffer should be chosen as a trade-off between the packet loss and delay.

receiving side to compensate for the jitter. Further, a packet loss of less then 0.1% can be assumed. In every performed measurement the loss was below 0.1% at 500Hz and even less at 10Hz.

III. Network Simulation

To emulate a sender and a receiver as realistic as possible a separate computer is used for each. For the network simulation a third computer is used which is called "Bottleneck" (see Fig. 7). In this way also IP (Internet Protocol) related issues can be examined as they will appear on the ground network. The Bottleneck is a Personal Computer where a FreeBSD operating system is running. The operating system has a build in network shaper "dummynet" [6]. Dummynet is part of the stateful firewall "ipfw" tool and can be assigned to a bridge between two network interfaces. A so called pipe can then be added for both directions of data flow. In List. 1 pipe 1 is shaping the traffic from the sender to the receiver and pipe 2 is shaping the traffic from the receiver to the sender.

```
ipfw add pipe 1 udp from 192.168.0.1/25 to 192.168.0.128/25
ipfw add pipe 2 udp from 192.168.0.128/25 to 192.168.0.1/25
ipfw pipe 1 config bw 88k plr 0.02 delay 7 queue 1
```

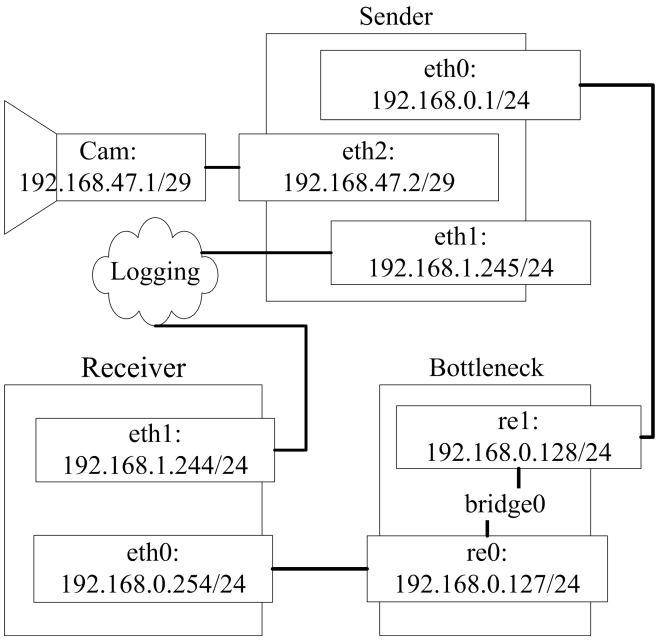

Fig. 7. Set-up of the network simulation as used in the experiments Section V. The sender, receiver and the bottle neck are all separate personal computers. The cam is a camera with a GigE Vision interface. The logging cloud symbolizes the ability to store experiment data on other communication ways as the actual video data. This allows for an entangled experiment.

```
ipfw pipe 2 config bw 4M plr 0.02 delay 7 queue 1
```

Listing 1. Commands for setting up a traffic shaper for a bi-directional connection. Pipe 1 emulates the up-link with a bandwidth limitation of $88\text{kbit}/\text{s}$ and pipe 2 emulates the down-link with $4\text{Mbit}/\text{s}$

Each pipe can be configured with a specified bandwidth (bw), packet loss rate (plr) and delay. The bandwidth is computed bit wise including all data above and including OSI (Open Systems Interconnection) model layer 3. This means that incoming bits are counted including the IP header and if the bits exceed the bandwidth the whole IP packet is queued until the bandwidth would allow further transmission. The incoming data will be buffered in a "queue" whereas the size of the queue can be configured to a specific slot size (as in List. 1) or to a specific size in kilo bytes. The transmission via the S-Band link (Section II-A) will most likely be done without an IP and UDP header. The bandwidth used for those headers may be added to the data rate f_r.

However the here proposed Bottleneck is not capable of dropping data from the head of the queue. It is only possible to establish a tail dropping queue which however can be set to the length of only 1 slot such that this limitation is irrelevant. The bottleneck is also not able to simulate delay, loss and jitter variations over time. This would be necessary so simulate the delay variations mentioned in Section II-B. An alternative Linux based network shaper could be "netem" (network emulator) [7]. With this tool network characteristics can be captured or defined over long time periods and "played back". Another alternative may be the exhaustive "ns" (network simulator) library [8].

IV. Compression and Packaging

A grayscale video with a resolution of 176×144 and with $1\mathrm{B}$ per pixel can only be transmitted with less than 1 frame every 2 seconds through the S-Band link with $f_r = 11\mathrm{kB/s}$. This is not sufficient for useful telemanipulations, hence the video stream needs to be compressed. A large variety of compression formats exists, of which some are suitable for the use in real time streaming.

The compression of digital video is maturing already since the beginning of digital imaging. Many methods and strategies arose which found their way into various compression formats and standards (Section IV-A). These compression formats are implemented as software (Section IV-B). Such an implementation supports specific video formats (Section IV-C). Various transport protocols exists which can transport video compressed with a certain standard.

A. Compression Standards

A large variety of compression standards are currently in use. A large amount of those are however designed for the compression of videos for storage purposes. For example an implementation of the Theora standard may result in a similar compression ratio as an implementation of the H.264 standard [9]. But advanced techniques for keeping the data rate constant which are indispensable such as a virtual buffer verifier are missing. However compression formats designed for teleconferencing targeting similar requirements as for telemanipulation. A widely used compression standard for video teleconferencing is the H.263 format which is a predecessor of the H.264 standard [10].

B. Software Implementation

For the ability to experiment with various video, compression and transport formats in a cross platform manner the GStreamer[11] library was used. Another powerful available tool is FFMpeg library. This library does not provide the direct usage of a transport protocol and was hence not used. Moreover includes GStreamer wide access to the FFMpeg functionality. An application ready possibility is the VideoLan Client [12]. However it turned out that it is not possible to use it for detailed analyzing of timings of an arriving video stream. GStreamer comes with implementation of the standards: Theora, Dirac, H.263, MPEG-4, VP8 and H.264 which all can be payloaded into a RTP stream. The implementation of the H.264 standard which is used by GStreamer for encoding is "x264".

C. Video Format

It is assumed that most telemanipulations can be performed with a grayscale video feedback to the operator. It is further sufficient to use a gray quantization with $1\mathrm{B}$ ($8\mathrm{bit}$ color depth). Although the H.264 Standard [13] specifies the support of a monochrome color format in the high profiles, a few software implementations are actually supporting it. It is thus necessary to feed an encoder with a chromatic video stream. The x264 encoder is only able to handle the chroma formats with the FourCC (Four-Character Code) "I420" and "YV12". Both formats are sub-sampled with the three part ratio of 4:2:0 but do have a slightly different memory layout. These formats are using $4\mathrm{B}$ for the luminance component and $1\mathrm{B}$ for each of the two chromatic components every 4 pixels.

a) Color: To perform the experiments with monochromatic video the chromatic components are made all zero. This increases the computational complexity but does increase the required bandwidth only marginal. That is because the chromatic components are encoded separated from the luminance component and remains the same across subsequent frames. In a keyframe (I-frame) not only the difference has to be transfered but since intra prediction is used such a uni-value component of the frame has the size of zero. Nevertheless headers indicating that nothing has changed need to be transmitted with the video stream. However the size is negligible in respect to the overall frame size.

b) Frame Size: The frame size is mainly determined by the computational power which is available for the compression and decompression. The data rate and quality does constrain the frame size much less respectively. However each macroblock[1] requires a small header ($< 1\mathrm{B}$), hence large frame sizes need to be avoided in a low bandwidth environment. In essence a trade-off between losing information at lower frame size and the increase of computational complexity has to be made. The information may get lost at the compression state in order to fulfill the desired data rate. The CIFs (Common Intermediate Formats) are a useful directive since they are in use for video telecommunication [10] for a long while. For the experiments (Section V) the frame size of 352×288 was used which is the resolution as specified in the CIF.

c) Frame Rate: Similar to the frame size the frame rate is primary constrained by the computational complexity. But each frame must be placed in a separate transport packet at the network layer. As a consequence the higher the frame rate, the more transport protocol overhead is generated. For the purpose of telemanipulation, a frame rate of $5\mathrm{Hz}$ is to slow and $12\mathrm{Hz}$ is a good consensus. At the moment and in the foreseeable future robots cannot be telemanipulated with a higher speed that would require a higher frame rate.

D. Transport protocols

Usually videos are embedded in a media container like the Audio Video Interleave format or the Matroska video format. For the transport through paketized networks specialized standards such as H.324/M, MPEG-4 (Moving Picture Experts Group-4) transport or RTP (Real-time Transport Protocol) exists [14]. Since the latter is widely used for video and audio streaming and has a small header the RTP is used for the experiments. The header size is $12\mathrm{B}$ per packet which gives an overhead of 1.31% at a frame rate of $12\mathrm{Hz}$ with a data rate of $11\mathrm{kB/s}$. It is also assumed that only one slice[2] per frame

[1] A macroblock is a region of 16×16px of one video frame

[2] A slice is a part of a video frame. Each slice will be placed into a separate network packed

is used. The RTP is also designed to compensate for jitter and reordering of network packets [15].

V. Experiments

Multiple common objective quality measurement for an evaluation of the received video are existing. One of those is the Peak Signal to Noise Ratio (PSNR) [10].

$$P_{SNR} = 10 \log_{10} \frac{(2^n - 1)^2}{M_{SE}}. \quad (1)$$

This evaluates the difference of the mean square error (M_{SE}) of the original image (the video at the sender side) and the impaired image (the video at the receivers side). Other objective quality metrics like the VQM (DCT-based video quality evaluation) [16] may be evaluated in addition.

For the experiments a test scene, showing the robotic arm which will be controlled from the ISS, is used. A video of the robotics arm with 121 images was shot with a frame rate of 12Hz. The robotic arm is moving 3 times up and down in front of a task board. Every 24th frame (that is every 2s) is shown in Fig. 8. This prerecorded video sequence is a raw video in I420

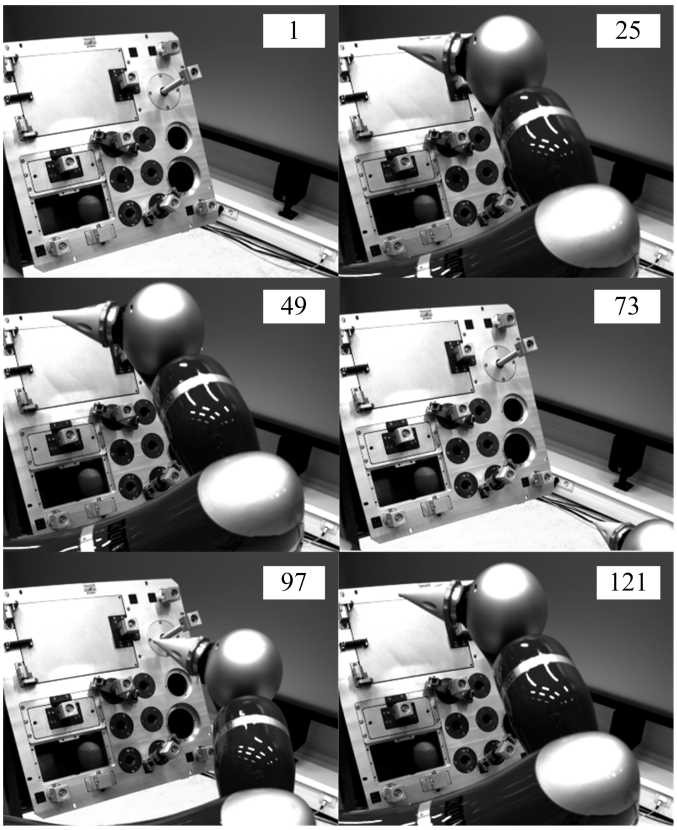

Fig. 8. Sample frames of the video scene which was used for the experiments. The frame number of each frame is printed on the top right. These frames correspond to the keyframes as set at the x264 encoder.

color format and has a resolution of 352×288. This video is pushed to a x264 encoder, then to a RTP payloader and further to an UDP sink witch sends it through the bottleneck. At the receivers side an UDP sink receives the packets which are then buffered in a RTP jitter buffer. The RTP packtes are then depayloaded and given to a H.264 encoder as a H.264 byte stream which decodes the stream into an I420 video. The just described pipeline can be seen in Fig. 9. The x264 encoder is

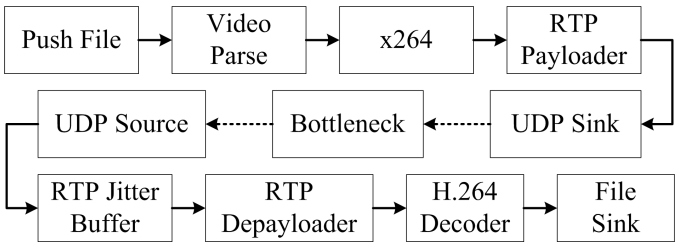

Fig. 9. Pipeline of the data flow during the experiment. The continuous lines show the flow inside GStreamer on a computer. The dashed lines show the data flow between the computers.

set to a constant data rate of 84kbit/s, a keyframe interval of 24 and the usage of one slice per frame. The data rate was chosen according to the bottleneck which is set to 88kbit/s while lowering it by the expected header load of $\approx 4\text{kbit/s}$. The latter approximation is derived by the frame rate of 12Hz and an IP/UDP/RTP packet header size of 40B.

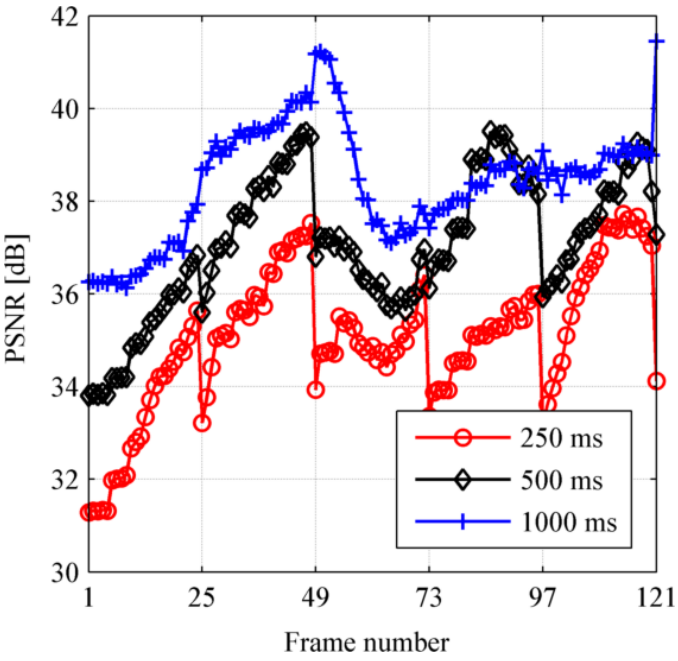

Fig. 10. The Peak Signal to Noise Ratio (PSNR) of the unimpaired stream (Fig. 8) and the compressed stream (Fig. 11) is plotted here. All three variants are encoded with the x264 encoder having the same parameter settings except the Virtual Buffer Verifier (VBV) which was set to three different values. The larger the VBV, the higher the PSNR and thus the higher the image quality.

As described in Section II, a packet loss of 1.15% at the S-Band link and 0.1% at the ground network (thus 1.25% in total) have to be expected. The compression is most effective if subsequent frames are predicted from the previous ones since only the differences need to be transmitted. If just these so called P-Frames are used, an occurring packet loss is affecting every subsequent frames. To avoid this permanent damage of

the video transmission keyframes need to be inserted. These keyframes do need more data to keep the visual quality, or the quality must be dropped significantly in this bandwidth constrained environment. Sudden quality changes need to be avoided to guaranty smooth telemanipulations. Note that bi-directional predicted frames (B-Frames) were not considered since they would introduce an additional delay of $\frac{1}{12\text{Hz}} \approx 83\text{ms}$. A way to keep the quality constant is to use more data for the keyframes and less data for the P-Frames in order to meet the data rate constraint. This comes with the price of a varying delay between the frames. In a telemanipulation these delays need to be avoided and thus the frames need to be buffered which introduces an additional delay. The H.264 standard specifies a "Hypothetical Reference Decoder" (HDR) that can be configured to adjust the trade-off between the quality and the delay. In the x264 encoder the Virtual Buffer Verifier (VBV) parameter can be used to configure the HDR. In Fig. 10 the PSNR (see eq.(1)) is plotted for three different settings of the VBV. For the smallest and the largest VBV setting three example frames are shown in Fig. 11. The higher

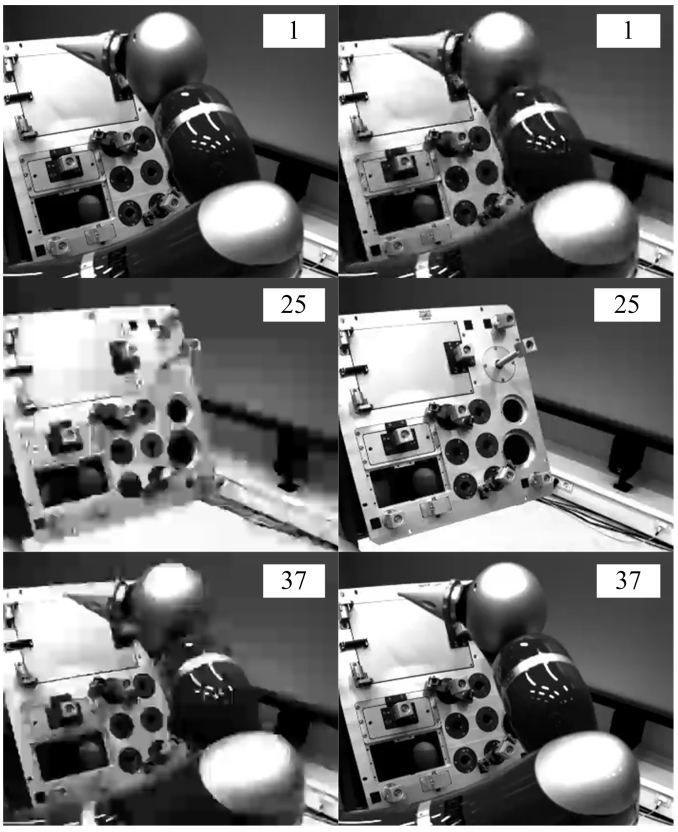

Fig. 11. Compressed frames of the unimpaired video stream of Fig. 8. At the left column the Virtual Buffer Verifier (VBV) was set to 250ms and at the right column the VBV was set to 1000ms. All other parameters remained the same. The top two rows correspond to the keyframes in Fig. 8. The last row shows an example of a frame in the middle of two keyframes.

the VBV the more the amount of data required for each frame may vary. Larger frames need more time to be transferred through the bottleneck. This introduces jitter of the arrival of frames in respect to the original frame rate. The jitter which is introduced by various settings of the VBV can be seen in Fig. 12. There the arrival time of each frame was measured

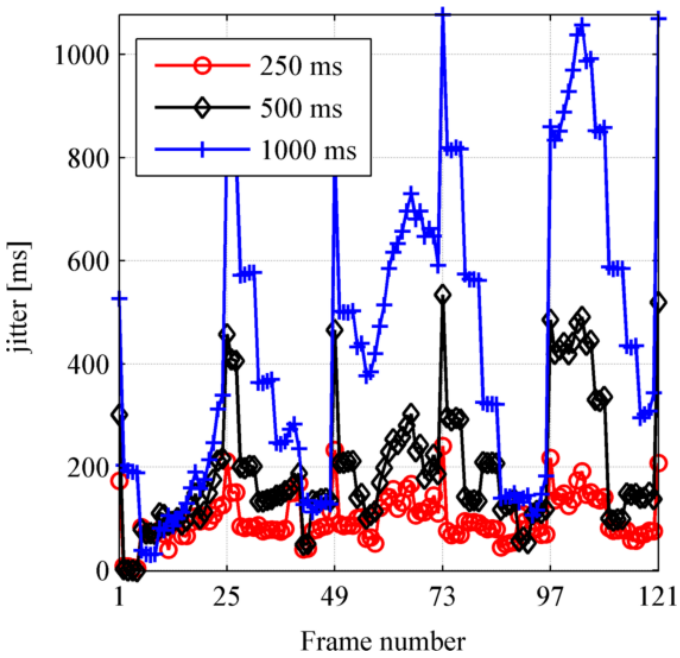

Fig. 12. This graphs show the delay of each frame in respect to the arrival time of the first frame (hence jitter). The three graphs correspond to a different size of the Virtual Buffer Verifier (VBV). As can be seen the larger the VBV the larger the jitter.

in respect to the first arrived frame. To compensate the jitter an additional buffer must be introduced which causes an additional overall delay of the video transmission. In the x264 encoder the size of the VBV must be given in milliseconds and corresponds to that overall delay which must be added in addition. The larger the VBV, the larger the delay must be chosen in order to avoid large jitter. A smaller VBV size will introduce a none optimal usage of the bandwidth of the bottleneck. That means in various phases of the transmission the complete bandwidth is not used to its full extend. This is a consequence of the implementation of the HDR in order to keep sudden quality changes low. In Fig. 13 the sent as well as the received bandwidth can be seen. The bandwidth is measured at discrete times as packets arrive. This introduces an amount of measurement noise to the original bandwidth which is computed bitwise at the bottleneck.

Another strategy of reducing the needed data increase at keyframes is to use an "intra refresh wave". If the intra refresh wave is enabled, a keyframe is transferred as a certain amount of macroblocks distributed across several frames. The intra refresh wave can be combined with the VBV.

VI. Conclusion and Future Work

It is possible to increase the quality of a video stream if error patterns like the packet loss of the network are known in advance. The more unknown the error pattern the more delay has to be added in order to compensate for those. Parameters such as the VBV can be adapted accordingly. If however the

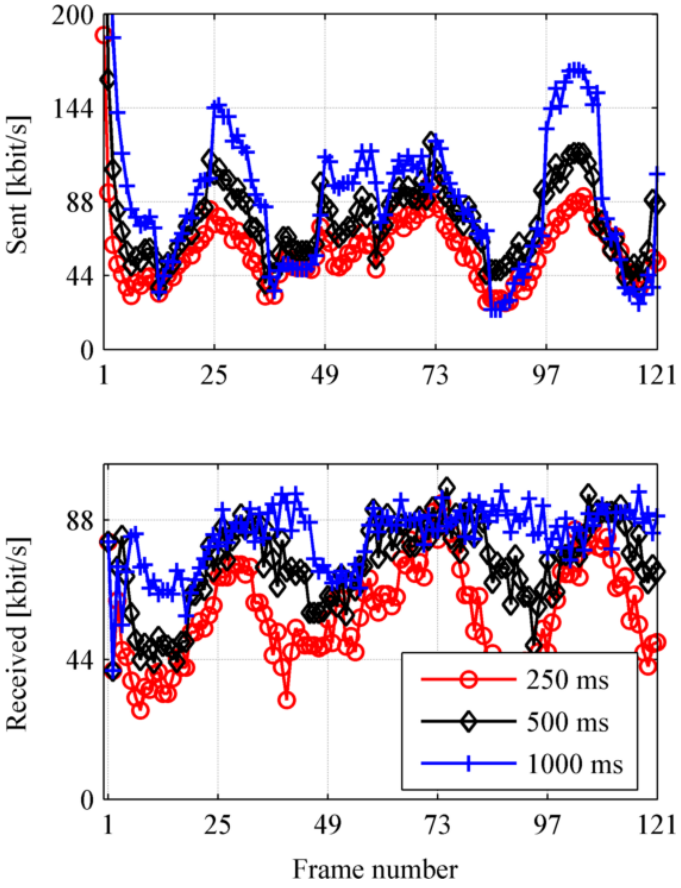

Fig. 13. This graph shows the bandwidth as transmitted by the sender and the bandwidth as received by the receiver. As can be seen the higher the Virtual Buffer Verifier (VBV) size the more varying the bandwidth at the sender but more close to the $88\mathrm{kbit/s}$ at the receiver.

error patters are worse than predicted and the system was optimized for those, fatal errors can occur which reduce the quality of the video far more than more relaxed parameters would do. The H.264 compression standard provides a huge variety of parameters which all have to be chosen carefully in order to provide a smooth video stream. One of those parameters is the VBV whose effects were described here in detail.

In the telerobotics and haptics laboratory the settings were successfully tested in a real telemanipulation scenario involving the control of the robot seen in Fig. 8. In future studies the influence of various parameter settings on the mean time to completion of a specific task to perform with the robot will be investigated in more detail.

References

[1] A. Schiele, "METERON - validating orbit-to-ground telerobotics operations technologies," in *11th Symposium on Advanced Space Technologies in Robotics and Automation, ASTRA 2011*, 2011.

[2] K. Landzettel, A. Albu-Schäfer, B. Brunner, A. Beyer, R. Gruber, E. Krämer, C. Preusche, D. Reintsema, J. Schott, B.-M. Steinmetz, H.-J. Sedlmayr, and G. Hirzinger, "ROKVISS verification of advanced light weight robotic joints and tele-presence concepts for future space missions," in *9ht Symposium on Advanced Space Technologies in Robotics and Automation, ASTRA 2006*, 2006.

[3] *ROKVISS - S-Band Communication Protocol Specification*, Issue 3 ed., Kayser-Threde GmbH, Feb 2003.

[4] *CCSDS Recommendation for Space Packet Protocol*, Consulative Committee for Space Data Systems Std., Rev. Issue 1, Sep 2003.

[5] T. Krueger, "METERON link test," Jan 2013.

[6] M. Carbone and L. Rizzo, "Dummynet revisited," *SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 2, pp. 12–20, Apr. 2010.

[7] A. Keller, *tc Packet Filtering and netem*, ETH Zurich, Jul 2006.

[8] F. Melakessou and T. Engel, "Network traffic simulator 2.0: Simulating the internet traffic," in *Open-source Software for Scientific Computation (OSSC), 2009 IEEE International Workshop on*, 2009, pp. 139–147.

[9] T. Halbach, "Comparison of open and free video compression systems a performance evaluation," in *International Conference on Imaging Theory and Applications (IMAGAPP)*, 2009.

[10] I. E. Richardson, *The H.264 Advanced Video Compression Standard*. Wiley, 2011.

[11] GStreamer, "Gstreamer," http://www.gstreamer.com/, Apr. 2013.

[12] VideoLAN, "Videolan client," http://www.videolan.org, Apr. 2013.

[13] *H.264 - Advanced video coding for generic audiovisual services*, ITU - International Telecommunication Union Std., 03 2010.

[14] S. Wenger, "H.264/avc over ip," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 13, no. 7, pp. 645 – 656, 2003.

[15] C. Perkins, *RTP Audio and Video for the Internet*, ser. Kaleidoscope Series. Addison Wesley Professional, 2003.

[16] D. Vatolin, A. Moskvin, O. Petrov, and N. Trunichkin, "Msu video quality measurement tool," 2009.