Stereoscopic Video over IP Networks

Eva Hladká
Faculty of Informatics,
Masaryk University,
Botanická, 68a 602 00 Brno,
Czech Republic
eva@fi.muni.cz

Miloš Liška
Faculty of Informatics,
Masaryk University,
Botanická, 68a 602 00 Brno,
Czech Republic
xliska@fi.muni.cz

Tomáš Rebok
Faculty of Informatics,
Masaryk University,
Botanická, 68a 602 00 Brno,
Czech Republic
xrebok@fi.muni.cz

Abstract

Transfers of high-quality multimedia content pose new demands on capacity and services provided by the contemporary high-speed computer networks. Transfer of stereoscopic video is a specific example, as it needs synchronization between two separate data streams. We have set up a stereoscopic video capture system and studied synchronization of two separate Digital Video format streams sent over packet networks. We have adapted application tools to support the synchronization and used an active element working as a synchronizing UDP packet reflector to explicitly synchronize the streams if they are desynchronized in the network. We have experimentally studied the quality of achievable synchronization and the relationship between the amount of desynchronization and the additional latency overhead posed by buffering of the data on the synchronizing reflector. The results prove our assumption that even high-quality DV streams can be successfully synchronized using the simple packet reflector running on common IA32-based computer.

1. Introduction

Multimedia transfers are becoming one of the most important applications for current high-speed computer networks. New services, that are being developed for high-performance multimedia processing and transport, are often used to create virtual collaborative environments, where people can interact regardless of geographical distance between them. Most of the contemporary collaborative environments work in 2D only, meaning that the reality, which has three dimensions (3D), is reduced to 2D picture with inevitable loss of some information connected with the depth of the space. To simulate 3D environment by stereoscopic image, two streams must be transmitted over the network synchronously, one for each eye. To provide natural perception, the quality of individual streams must be rather high and thus high-resolution image with low compression needs to be deployed resulting in high data rate. Such requirements are satisfied e. g. by Digital Video format. Transmission and synchronization of stereoscopic video streams in DV format over the high-speed network are studied in this paper.

Processing of multimedia content usually comprises three phases: acquisition, transport, and presentation. The acquisition of stereoscopic video capture uses two cameras that mimic two human eyes. The two resulting streams need to be transmitted over the network and received synchronously by the display system. However, common computer networks can't enforce directly this kind of synchronization. One possibility is to synchronize and multiplex data at the source and send both video streams in one data (packet) stream. However, processing both streams on a single machine might not be feasible depending on video format used for the transmission. Our solution is thus to synchronize otherwise independently transmitted streams at some point in the network close to the display nodes, where suitable active element is placed. For purpose of our evaluation, the general purpose active element has been implemented using simple synchronizing UDP packet reflector.

2. Digital Video

In order to create highly realistic and information-rich environment, we have opted for using Digital Video (DV) format as the basic video format for our experiments with stereoscopic video. It provides very good image quality (with limits given by PAL resolution) while having reasonable compression ratio (approx. 5:1), low latency compression and decompression process based on I-frames only, and sustainable requirements on processing infrastructure.

The DV video transmission over IP networks has been standardized and implemented by DVTS project [1] for

Proceedings of the Joint International Conference on Autonomic and Autonomous Systems
and International Conference on Networking and Services (ICAS/ICNS 2005)
0-7695-2450-8/05 $20.00 © 2005 IEEE

several operating systems (e.g. Linux, FreeBSD, Windows 2000/XP, and MaxOS X). The DVTS project originally included (i) IEEE-1394 kernel driver for FreeBSD, (ii) `dvsend` for sending the DV video captured from IEEE-1394 to the network as a RTP stream, (iii) `dvrecv` for receiving the DV stream from the network and saving it locally to the disk, and (iv) `dvplay` for sending any DV data to the IEEE-1394 device. The tools produced by the DVTS project are stable and mature with the exception of the `xdvshow` prototype tool for displaying video locally using X-Windows interface. We have reimplemented the `xdvshow` tool, as its functionality is crucial for high quality display of stereoscopic video.

The original `xdvshow` implementation used just one thread for all operations on the DV video. The mutual exclusivity of the reading, decoding and displaying of one video frame was handled by busy waiting. The CPU consumption was unacceptably high, while CPU was spending most of the time in the busy waiting loop. The stereoscopic video display needs two DV streams synchronously and that means it should be possible two running instances of `xdvshow` at least on a single high-end computer.

The multi-threaded implementation solves the busy waiting problem sparing CPU time—this may be up to 50% of the total CPU time on modern CPUs. The multi-threaded architecture also allows separation of the reading process from decoding and displaying of the DV video. The multi-threaded `xdvshow` implementation uses three primary threads: One thread reads the DV stream from selected input, the second one is used for decoding of the stream and displaying the decoded video data and the third one is used for decoding and playing the audio part of the DV stream. There may be an additional thread used for interacting with the packet reflector, when it is the source of the DV stream. System of semaphores is used to thread cooperation and mutual exclusion for shared buffers' access. The `xdvshow` uses two shared buffers, one for DV video frames and the other for corresponding audio frames.

3. DV over IP

RTP protocol [2] is the major real-time transmission protocol used for multimedia distribution in IP networks. RTP is non-reliable and non-guaranteed service over the underlying UDP protocol. Also, QoS parameters of the transmission are not guaranteed by the protocol and must be handled on the application level if needed. The data transport by the RTP protocol is complemented by the RTCP protocol that provides support for delivery monitoring and also for control and identification functionality.

Standardization of DV over IP transmission using RTP protocol has been introduced by K. Kobayashi *et al.* in [3, 4]. A video frame in the DV format is divided into several "DIF sequences." A DIF sequence is composed of 80-bytes long DIF blocks. A DIF block is a primitive and atomic unit for all operations over the DV stream. Each RTP packet starts with an RTP header and no additional header is required for DV over IP transmission. The atomic DIF blocks are placed directly after the RTP packet header. It is possible to place any number of DIF blocks representing one distinct frame into one packet. The DIF blocks belonging to the next frame must be transmitted in a new RTP packet to facilitate frame detection. Since the RTP payload contains an integral number of DIF blocks, the length of RTP payload is divisible by 80.

The DV video transmission extensively uses Timestamp and Marker bit arrays in the RTP header. The time when the first data in a particular frame has been sent is stored in the timestamp array. All RTP packets in one videoframe must have the same timestamp according to the above mentioned standards. The timestamp increment for 25 fps PAL and 29.97 fps NTSC video are 3600 and 3003 respectively. The marker bit, left for any user-defined data by the RTP standard, is used to mark the last packet of the frame. When such packet is received, the whole frame can be immediately displayed, instead of waiting for the next packet to recognize the end of the frame. This mechanism reduces the total latency. However, the detection of end of frame must not rely on the marker bit presence only as the corresponding packet may be lost, and the check on the timestamp change in the RTP header must be always performed, too.

It is possible to transport the audio and video data in the same stream or separately. The choice must be done once and forever for one stream. It is also reflected in the dynamic Payload type and thus it must not change until the end of the RTP session.

The DV format video uses bandwidth of 25 Mbps. When audio data and header overhead is added, the resulting stream uses approximately 30 Mbps per each stream.

4. Stereoscopic Video

Nowadays, most of the video material transmitted over the network depicts scenes in two dimensions only. While human eye or better human brain is able to recognize quite lot of depth cues even in two dimensional picture to create some idea of space, realistic perception can not be achieved without taking the human eye anatomy into accounts. In real-world conditions, each eye perceives independently under a little bit different angle and the brain superimposes both images to get full 3D perception. This means we need to capture two independent video images for each eye to provide perception of the third dimension. The schematics of such a 3D video capture system is depicted in Fig. 1. The theoretical model of such system is given in [5] and [6].

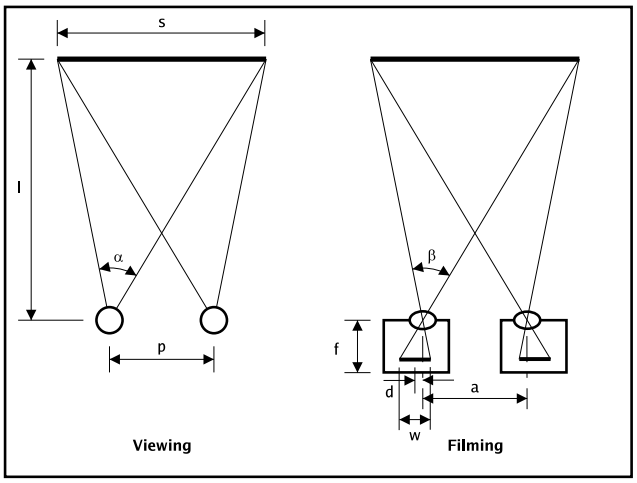

Figure 1. Setting up the cameras

To fulfi ll at least some of the preconditions, like the distance of the camera lenses focal points should be the same as the distance between human eyes focal points, a suitable camera mount needs to be used. After experimenting with a simple solution suggested in [7], we decided to use more sophisticated dual camera mount by Apec called "Parallax Setting Device" (PSD), as shown in Fig. 2.

Figure 2. Parallax setting device

Stereoscopic effect created by two cameras can be optimized by proper setting of stereoscopic base and convergence angle set between two cameras. PSD is capable of sliding and rotating cameras for setting stereoscopic base and convergence angle respectively. Device allows perfect alignment of optical axes of the two cameras along both vertical and horizontal directions.

5. Synchronization

Two cameras mean two independent video streams are created and must be transferred over the network. To remove any unwanted effects on human observer, both streams need to be synchronized when displayed. Synchronizing reflector solves this problem explicitly using timestamps in RTP and RTCP packets. RTP packets include relative time-stamping information which may differ both in time base and time increment for streams with different sources coming even from several applications running on one client computer. Conversion between relative time and absolute time can be performed using information sent in RTCP packet that are sent with much lower frequency for each stream. RTCP packets contain both relative time-stamp and absolute time-stamp in NTP format. Therefore after receiving two RTCP packets it is possible to calculate both relative time base and increment. To synchronize streams coming from two different machines, they must have their clocks synchronized, e. g. using NTP protocol. Conversion between "real" absolute time and relative RTP time based on RTCP information is depicted in Fig. 3.

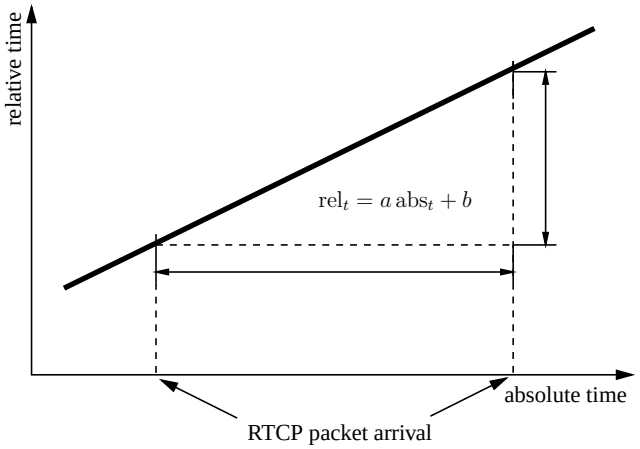

Figure 3. RTP time conversion

For the UDP stream, with no guarantee on delivery, two steps are necessary for the actual synchronization: (1) reorder and/or discard out-of-order packets and (2) match the packets using the RTP/RTCP time information from different streams.

The synchronization and packet reordering is not without penalty. The overall latency increases, which is not desirable for interactive applications like videoconferencing. In such cases even small latency in order of hundreds of milliseconds induces communication problems and disrupts the reality illusion (e. g., when one person tries to interrupt the other one to express his/her opinion).

In order to be able to synchronize RTP streams each RTP packet must have its own timestamp. However, for the DV transmission the timestamps are increased once per a videoframe, which is not acceptable for fi ne-grained synchronization. To provide synchronization support on a per packet basis, we had to change the protocol and to put individual timestamp into each RTP packet with the DV payload.

Proceedings of the Joint International Conference on Autonomic and Autonomous Systems
and International Conference on Networking and Services (ICAS/ICNS 2005)
0-7695-2450-8/05 $20.00 © 2005 IEEE

6. Implementation

While it is possible to synchronize two independent streams only at the point of delivery (before they are displayed), we opted for a network support where the streams are synchronized every time they are retransmitted through a reflector. This solution not only reduces the overall latency due to synchronization and possible reordering, but also provides support to synchronize several sites in one step.

We have enhanced our reflector implementation [8] to support synchronization of multiple reflected RTP packet streams for synchronized, timely, and optionally also in-order delivery to the connected clients. The reflector uses multi-threaded model in which several threads are used as network listeners that place packets coming from different streams into ordered buffers for each stream. Then the sending thread takes packets out of these queues and sends them to the connected clients in synchronized way.

Data processing within the reflector called `rum` proceeds as follows. The reflector starts N separate threads, where N is the number of ports placed as arguments. The main thread is now used as sender and the N threads are used as receiving listeners. Each receiving thread initializes particular socket the reflector is listening to (there are actually two sockets initialized for each RTP session—one for RTP and other one for RTCP packets).

When RTP packet arrives to the listener thread, the RTP header is extracted and parsed to obtain packet relative creation time, which is in turn converted into absolute time. Then the packet is stored into time-sorted buffer—oldest packets are on the top and wait to be sent. Information on stored and dropped packets is kept for statistical purposes.

After receiving an RTCP packet the data for conversion between relative RTP time and absolute time is updated, taking into account previous conversion data by computing sliding average. We assume linear dependence of relative and absolute time. If abrupt change occurs program waits for at least three consecutive RTCP packets carrying consistent time information to achieve stability and avoid short time fluctuations.

The main function of sending thread is to send packets which are saved on top of all the buffers (doing that synchronously if requested). It is also possible to specify that all the late arriving packets are dropped from the buffers. Sending is performed using round robin method for all buffers. Packets on the buffer top are sent to the connected clients when their absolute time is smaller or equals to absolute time of packets in other buffers. When all buffers are empty, the sending process stops and reflector waits for incoming packets. If requested by the user, it is possible to make the reflector stop for random time period after the completion of each cycle (until all buffers are full enough), reducing thus processor load (and naturally increasing overall latency). This option is needed when the reflector is synchronizing low bandwidth data because all buffers have to contain some packets to perform correct synchronization.

7. Testbed and Measurements

The reflector implementation, as an implementation of the network supported synchronization of the stereoscopic video streams, has been subjected to a series of tests to evaluate its performance. The goal of these tests was to confirm the synchronization capabilities of the reflector and also to evaluate the additional latency induced by the synchronization effort. DV format was used for both video streams.

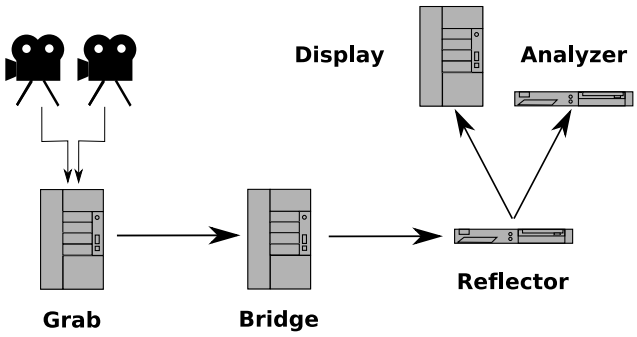

Figure 4. Reflector testbed

The evaluation testbed depicted in Fig. 4 was composed of the following components:

Sender FreeBSD 5.4-RC4, CPU: VIA C3 (1200 MHz), 512 MB RAM

Bridge Dell 1600SC, FreeBSD 5.3-RELEASE-p8 with two network interfaces, CPU: Intel Xeon (2800 MHz), 1 GB RAM

Reflector Linux 2.6.8-2-686-smp, CPU: 2× Intel Xeon (3000 MHz), 4 GB RAM

Analyzer Linux 2.6.8-2-686-smp, CPU: 2× Intel Xeon (3000 MHz), 4 GB RAM

Display Dell Precision 530, Linux 2.4.27-smp, CPU: 2× Intel Xeon (2400 MHz), 1 GB RAM

Switch HP Procurve 6108

All the computers used Intel PRO/1000 MT network interface cards, the network link capacity (available bandwidth) has been always 1 Gbps. The testbed used dedicated machines and links.

All measurements were done on two streams where one stream has been sent without any delay and the other has

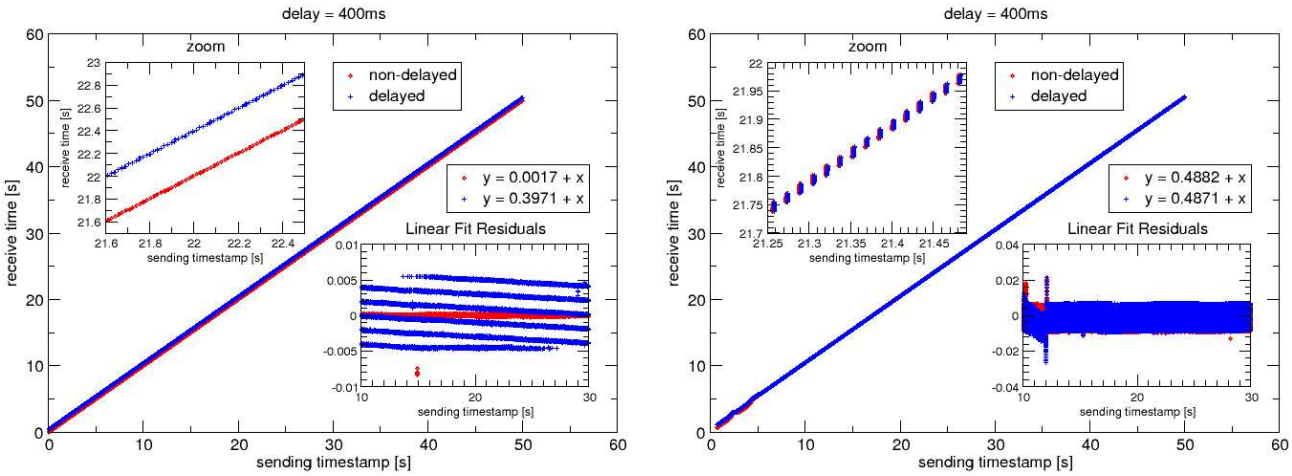

Figure 5. Graph with delay 400 s without/with synchronization

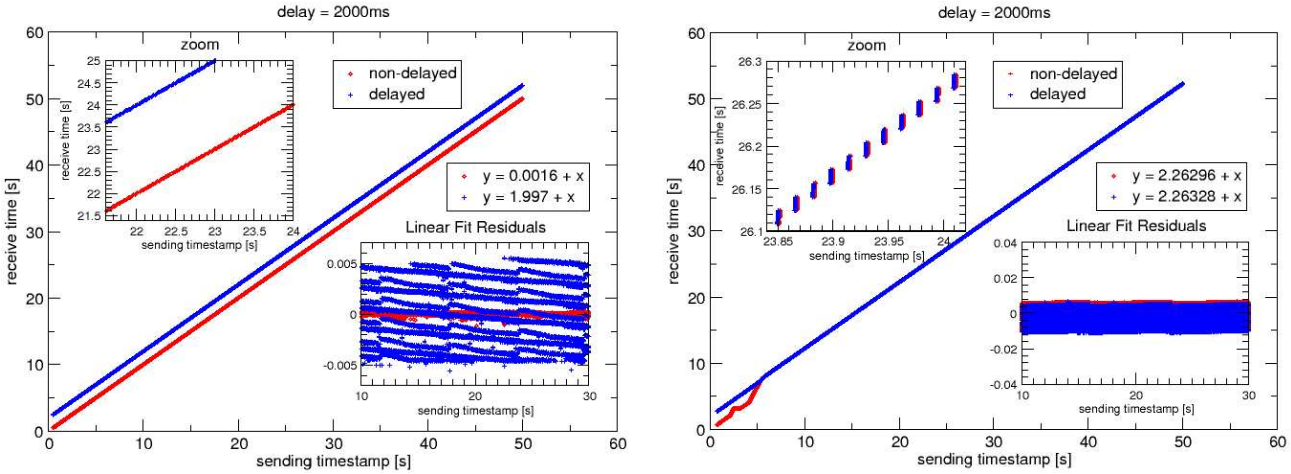

Figure 6. Graph with delay 2000 s without/with synchronization

been delayed for a specified time using traffic shaper available as a part of `ipfw` firewall in the FreeBSD kernel. Both streams arrived to the reflector, has been synchronized there and sent to the analyzer (and simultaneously also to the displaying device so we were able to watch the effect also subjectively, but this evaluation is not presented). To cover wide range of possible delays in real production networks, we choose the following delays: 0, 20, 40, 80, 100, 200, 400, 600, 800, 1000, 1500, 2000 and 5000 ms. The choice of measured delays corresponds to the timing of individual fields and frames in 25 fps PAL video. The measured total delay after synchronization is presented in Tab. 1.

We see that the reflector internal delay is around 68 ms, which is only slightly increased if the inter-stream delay is just 20 ms. However, with increased inter-stream delay the total penalty sharply increases. The worst cases are around 100 – 200 ms. With further increase of inter-stream delay the absolute and especially the relative penalty decreases, as the reflector has time to process both streams practically independently. The final penalty for very high inter-stream delay is around 10% of this delay, which is probably acceptable overhead for fully software-based solution.

All the output streams has been fully synchronized. To demonstrate it, we selected two measurements, with inter-stream delay of 400 and 2000 ms, depicted in Figs. 5 and 6 respectively. All graphs show the relationship between the receiving and sending time of RTP packets. The graphs on the left side show both streams without synchronization—we can see two separate streams (the smaller picture on the left side provides a zoomed view of the relationship). The graphs on the right side show the resulting synchronized stream. The linear-fit residual graphs show the difference

Link delay [ms]	Synchron. delay [ms]
0	68.3
20	71.1
40	142
80	287
100	357
200	368
400	487

Link delay [ms]	Synchron. delay [ms]
600	642
800	901
1000	1130
1500	1690
2000	2240
5000	5550

Table 1. Link and synchronization delays

between the actual time when a particular RTP packet has been received by the target (the analyzer) and the ideal time of its reception. Again, the results confirm that the synchronization is practically absolute. The initial time synchronization, which needs several RTCP packets, is visible as the small nonlinearity at the beginning of the measurements.

8. Conclusions

When stereoscopic video is sent over IP network in two independent streams, they must be synchronized before the they are is displayed. If multiple sites are receiving the same stereoscopic video, the synchronization is best done in the network, otherwise each site may be exposed to different latency, unacceptable for interactive applications.

The idea of overlay network with active elements capable of providing new functionality to computer networks has already been shown as a successful foundation of controlled multicast transmission. The same idea has been used for the stereoscopic video streams synchronization. A simple software implementation running on commodity hardware is able synchronize the two streams in DV format successfully even when the original streams are highly de-synchronized. The penalty of the synchronization is increased latency, as the "faster" stream must wait for data in the other stream, plus some processing latency is added to the final perceived delay. While this delay may be problematic in interactive implementation, the reflector based synchronization element can be easily used for synchronized unidirectional stereoscopic streaming to multiple end users even in highly adverse and desynchronizing network conditions.

9. Acknowledgments

This research is supported by a research intent "Optical Network of National Research and Its New Applications" (MŠM 6383917201) and a research intent "Highly Parallel and Distributed Systems" (MŠM 00216224419). We would also like to thank to Petr Holub for helping with methodology and interpretation of measurements and Luděk Matyska for proof reading this text.

References

[1] Akimichi Ogawa and Katsushi Kobayashi and Kazunori Sugiura and Osamu Nakamura and Jun Muri. *Design and Implementation of DV based video over RTP*. November 2004. `http://www.sfc.wide.ad.jp/DVTS/pv2000/index.html`

[2] H. Schulzrinne and S. Casner and R. Frederick and V. Jacobson. *RFC3550–RTP: A Transport Protocol for Real-Time Applications*. July 2003. `http:///www.zvon.org/tmRFC/RFC3550/Output/index.html`

[3] K. Kobayashi and A. Ogawa and S. Casner and C. Bormann. *RFC3189–RTP Payload Format for DV (IEC 61834) Video*. January 2002. `http://www.zvon.org/tmRFC/RFC3189/Output/index.html`

[4] K. Kobayashi and A. Ogawa and S. Casner and C. Bormann. *RFC3190–RTP Payload Format for 12-bit DAT Audio and 20- and 24-bit Linear Sampled Audio*. January 2002. `http://www.zvon.org/tmRFC/RFC3190/Output/index.html`

[5] V. Griberg and G. Podnar and M. Siegel. *Geometry of binocular imaging*. in Stereoscopic Displays and Applications. February 1994. `http://www.ri.cmu.edu/pub_files/pub1/grinberg_v_s_1994_1/grinberg_v_s_1994_1.pdf`

[6] V. Griberg and G. Podnar and M. Siegel. *Geometry of binocular imaging II: The augmented-eye*. in Stereoscopic Displays and Applications. February 1995. `http://www.ri.cmu.edu/pub_files/pub1/grinberg_v_s_1995_1/grinberg_v_s_1995_1.pdf`

[7] Rhys Hawkins. *Digital Stereo Video: display compression and transmission*. February 2002. `http://escience.anu.edu.au/research/papers/02_Rhys_Hawkins/thesis-small.pdf`

[8] E. Hladka, P. Holub and J.Denemark. *User Empowered Virtual Multicast for Multimedia Communication*. in ICN'2004 Conference Proceedings. March 2004.