Web-based Stereoscopic Collaboration for Medical Visualization

Dissertation

zur Erlangung des mathematisch-naturwissenschaftlichen Doktorgrades

"Doctor rerum naturalium"
der Georg-August-Universität Göttingen

im Promotionsprogramm in Computer Science (PCS)
der Georg-August University School of Science (GAUSS)

vorgelegt von

Mathias Kaspar
aus Gotha

Göttingen, 2013

Betreuungsausschuss

Prof. Dr. Otto Rienhoff, Abt. Medizinische Informatik, Universitätsmedizin Göttingen

Prof. Dr. Jonathan C. Silverstein, Center for Clinical and Research Informatics, NorthShore University HealthSystem, Evanston, IL, USA

Mitglieder der Prüfungskommission

Referent:

Prof. Dr. Otto Rienhoff, Abt. Medizinische Informatik, Universitätsmedizin Göttingen

Korreferent:

Prof. Dr. Xiaoming Fu, Institut für Informatik, Universität Göttingen

Weitere Mitglieder der Prüfungskommission:

Prof. Dr. Jens Grabowski, Institut für Informatik, Universität Göttingen

Prof. Dr. Dieter Hogrefe, Institut für Informatik, Universität Göttingen

Prof. Dr. Ulrich Sax, Geschäftsbereich Informationstechnologie, Universitätsmedizin Göttingen

Prof. Dr. Ramin Yahyapour, Institut für Informatik, Universität Göttingen

Tag der mündlichen Prüfung: 23. August 2013

Summary

Medical volume visualization is a valuable tool in medical practice and education to virtually explore volume data of the human body. Real-time interaction, stereoscopic presentation, and collaboration are required for its full comprehension in all its three dimensional complexity. Such visualization of high-resolution data, however, is due to its high hardware demand almost only available on special imaging workstations. Whereas remote visualization systems are used to provide such visualization at peripheral locations, they still require complex software deployments. Since these are barriers for an universal and ad-hoc availability, the following hypothesis arose: A high performing remote visualization system, specialized for stereoscopy and ease of use, can provide access to real-time interactive, stereoscopic, and collaborative medical volume visualization.

The most recent work about remote visualization utilizes pure web browsers, but without emphasizing high performing usability by any participant nor essential functionalities to support various stereoscopic display systems. The web browsers familiarity, their ease of use, and wide availability led to following main research question: Can we evoke a solution that fulfills all aspects by only using a pure standard web browser at the client side?

A proof of concept was conducted to verify the hypothesis, including a prototype development, its practical application, its performance measurement, and comparison.

The resulting prototype system (CoWebViz) is one of the first web browser based systems without added software that provides fluid interactive remote visualization in real-time. Performance tests and comparisons show the superiority of the approach to the tested existing applications, including a native application. Its support of various stereoscopic display systems, which are simultaneously usable in a single collaborative visualization session is currently unique via such a lightweight client. Its usage for an usually resource intensive stereoscopic and collaborative setup for anatomy teaching shared with inter-continental participants shows the approach's feasibility and simplifying character. The feasibility of the approach has also been shown by its further successful usage in high-performance computing, and in surgery.

Zusammenfassung

Medizinische Volumenvisualisierung ist ein wertvolles Werkzeug zur Betrachtung von Volumendaten in der medizinischen Praxis und Lehre. Eine interaktive, stereoskopische und kollaborative Darstellung in Echtzeit ist notwendig, um die Daten vollständig und im Detail verstehen zu können. Solche Visualisierung von hochauflösenden Daten ist jedoch wegen hoher Hardware-Anforderungen fast nur an speziellen Visualisierungssystemen möglich. Remote-Visualisierung wird verwendet, um solche Visualisierung peripher nutzen zu können. Dies benötigt jedoch fast immer komplexe Software-Deployments, wodurch eine universelle ad-hoc Nutzbarkeit erschwert wird. Aus diesem Sachverhalt ergibt sich folgende Hypothese: Ein hoch performantes Remote-Visualisierungssystem, welches für Stereoskopie und einfache Benutzbarkeit spezialisiert ist, kann für interaktive, stereoskopische und kollaborative medizinische Volumenvisualisierung genutzt werden.

Die neueste Literatur über Remote-Visualisierung beschreibt Anwendungen, welche nur reine Webbrowser benötigen. Allerdings wird bei diesen kein besonderer Schwerpunkt auf die performante Nutzbarkeit von jedem Teilnehmer gesetzt, noch die notwendige Funktion bereitgestellt, um mehrere stereoskopische Präsentationssysteme zu bedienen. Durch die Bekanntheit von Webbrowsern, deren einfach Nutzbarkeit und weite Verbreitung hat sich folgende spezifische Frage ergeben: Können wir ein System entwickeln, welches alle Aspekte unterstützt, aber nur einen reinen Webbrowser ohne zusätzliche Software als Client benötigt?

Ein Proof of Concept wurde durchgeführt um die Hypothese zu verifizieren. Dazu gehörte eine Prototyp-Entwicklung, deren praktische Anwendung, deren Performanzmessung und -vergleich.

Der resultierende Prototyp (CoWebViz) ist eines der ersten Webbrowser basierten Systeme, welches flüssige und interaktive Remote-Visualisierung in Realzeit und ohne zusätzliche Software ermöglicht. Tests und Vergleiche zeigen, dass der Ansatz eine bessere Performanz hat als andere ähnliche getestete Systeme. Die simultane Nutzung verschiedener stereoskopischer Präsentationssysteme mit so einem einfachen Remote-Visualisierungssystem ist zur Zeit einzigartig. Die Nutzung für die normalerweise sehr ressourcen-intensive stereoskopische und kollaborative Anatomieausbildung, gemeinsam mit interkontinentalen Teilnehmern, zeigt die Machbarkeit und den vereinfachenden Charakter des Ansatzes. Die Machbarkeit des Ansatzes wurde auch durch die erfolgreiche Nutzung für andere Anwendungsfälle gezeigt, wie z.B. im Grid-computing und in der Chirurgie.

Acknowledgments

I would like to express my deepest thanks to Prof. Otto Rienhoff for his unremitting support during the past years, who helped me getting deeper into working at an university in general and, specifically, in helping me to conceptualize and finish my dissertation, and in successfully arranging the visit at The University of Chicago very shortly after I started my work.

I would like to express my deepest thanks to Prof. Jonathan C. Silverstein, who welcomed me in his working group at The University of Chicago for a short research stay that ended up in a long collaboration, who provided an environment that allowed us to advance our initial ideas with a very practical meaning, and for his enduring optimism and confidence.

I would like to thank all committee members, Prof. Otto Rienhoff, Prof. Xiaoming Fu, Prof. Jens Grabowski, Prof. Dieter Hogrefe, Prof. Ulrich Sax, and Prof. Ramin Yahyapour for any discussion, comments, and hard questioning, which undoubtedly helped me to refine my thesis. I would also like to thank Prof. Nigel John for his external review of my work.

I would like to thank Nigel M. Parsad and Fred Dech for all the countless hours they helped me amongst others with my prototype developments, tests, manuscript improvements, and for giving me an idea about the US american life.

I would also like to thank Nestor J. Zaluzec, Eric Olson, Joseph Insley, and Thomas Uram from the Argonne National Laboratories, Mike W. Daley from the Cardiff School of Computer Science and Informatics of Cardiff University, and Benjamin Löhnhardt from the Georg-August-Unviversity of Göttingen for their discussions about and testing of CoWebViz.

I would like to thank the German Academic Exchange Service (DAAD) for their support via a short-term fellowship for Ph.D. research studies.

Thanks to my parents Martina and Detlef Kaspar for always believing in me.

Lastly, my wife Lea K. Seidlmayer deserves my eternal gratitude for all of our discussions and, especially, her endurance during the last years.

Contents

1 Introduction

Direct volume visualization is an increasingly important technique to visualize volume imaging data as acquired for instance by Computed Tomography (CT) and Magnetic Resonance Imaging (MRI). Volume data contains detailed information about the three-dimensional (3D) structure of a part or the whole human body. In contrast to the direct presentation of volume data as multiple cross-sectional two-dimensional (2D) images, direct volume visualization enables the presentation of a whole volume dataset in a single rendered image. Such condensed information is inevitable to be understood by the people who use it, physicians and other medical staff [1]. They considerably depend on the presentation of highly processed and compressed data as created by medical visualization algorithms (see Section 1.1) in order to optimize the time involvement in the professional daily life (see Section 1.2). However, compared to other data that is commonly acquired by medical systems, e.g. high-dimensional structured data for administrative and treatment documentation, quantitative data for laboratory results, and sensor data in the scope of Ambient Assisted Living [2], volume data is unstructured and can become very large in size [3]. Because of its high compute hardware and potential network requirements, high quality volume visualization is bound to special imaging workstations or remote visualization approaches (see Section 1.3). Easy useable systems, however, have a positive effect on being used [4]. It therefore is necessary to provide and evaluate a lightweight system that enables high-performing volume visualization with least requirements on the client side (see Section 1.4).

1.1 Aspects of medical visualization usage

Understanding complex data by utilizing advanced visualization algorithms can be eased by providing interactive functionality, stereoscopic presentation, and shared visualization for discussion.

Interactive functionality. To understand virtual 3D objects as a whole and in all its details, it is necessary to modify the visualization interactively and in real-time for a comprehensive exploration of all regions of interests. Yi at al. [5] sub-divide interactive functionality of information visualization into seven categories, which are partially applicable to volume visualization: A user needs to zoom into or pan a visualization (*explore*), to rotate the visualized object (*reconfigure*), to show it in another colorization (*encode*) [6], to mark a Point of Interest (POI) (*select*), and to change the windowing level to highlight for instance bones or muscles (*filter*). Further categories described by Yi at al. are *abstract* and *connect*, which have no direct counterpart in volume visu-

alization. For *abstract*, the addition of an overlaid emphasizing illustrative visualization technique is thinkable. *Connect* could for instance be represented by the linkage of 3D coordinates (POIs) to descriptive data (e.g. labels). Essential for any interactive functionality is the modification of visualization in real-time, which requires a timely presentation, e.g. few milliseconds, of a modified visualization after it was being requested by the user [7].

Stereoscopic presentation. The presentation technique is another important aspect to enhance the understanding of the visualized data [8]. The simplest and likely most often used technique is the presentation on a 2D display (monoscopic visualization), which requires a mapping of the three-dimensional object (the medical volume data) to a two-dimensional image (e.g. direct volume visualization). More advanced techniques provide the viewer with a depth perception and/or a better visual immersion [9] (see Section 3.2). Thus, such systems provide more assisting information with two or more views of the three dimensional object by using stereoscopic or even volumetric displays [10]. Stereoscopic visualization has not yet been used extensively in practical medical scenarios except for special disciplines (e.g. robotic surgery [11]). Its success in other scientific disciplines and studies about its medical benefit however indicate its possible future importance [8, 12, 13]. Stereoscopic visualization techniques provide the viewer with a depth perception by providing separate views (eye perspectives) for each eye, which can eminently support the process of understanding data [14, 15]. The movie industry pushed the usage of stereoscopy with the wider dissemination of stereoscopic techniques to movie theaters and homes in the past years. This resulted in the common availability of high definition stereoscopic consumer grade 3D TVs for low costs and a general easier availability of stereoscopic technologies.

Collaborative usage. Another and closely connected aspect that is important in a treatment workflow is the consultation between physicians within the scope of telemedicine [16]. Such a consultation session is about a specific patient, including the patient's data and images. They vary from a simple asynchronous exchange of single or stacked annotated images and might potentially result in real-time discussions with interactive and shared advanced visualization in the future. The need for collaborative consultations between physicians, who are remote to each other, might get especially interesting with the development of larger hospital corporations and increasing specialization of physicians [17, 18].

1.2 Usage scenarios for medical volume visualization

Scenarios that could benefit from the provision of volume visualization with the previously described aspects are manifold and exist in different caregiving medical divisions as well as for education.

Medical practice. In practical medicine, volume visualization is usually based on data that is directly connected with the treated patient. Such is e.g. used in a *surgical planning*, which is done by surgeons before, during or after a surgery to get accustomed with the specific anatomy of the patient, to define the individual steps prior to an invasive procedure, and to evaluate the procedure afterwards [19, 20]. An easy access to the visualization might be important for this case, especially if the surgeon wants to do the planning independently of the location by using mobile devices before and during the procedure. A use case that could be closely connected is the *remote consultation* with multiple professionals who discuss a medical case based on a shared visualization [21] (e.g. a surgeon questioning a radiologist about more details pre-operatively). Remote could mean that both physicians are in different hospitals, but it could also mean that both are in different departments of the same hospital. The latter case could be useful to reduce the time that would be necessary to do the consultation in person. Besides the before mentioned use cases, a wider availability of *visualization in general* can also provide a benefit. An example is the presentation of visualization to a patient to explain a planned treatment. Enzenhofer et al. [22] for example showed that the patient's knowledge and satisfaction increases, when the physician-patient dialog that is necessary to inform a patient before an invasive procedure is combined with visualization.

Medical education. Whereas a new technology needs to undergo a complex process prior to its practical application in patient care, educational scenarios without direct patient contact are less restrictive since they only require anonymized data. There are many different levels of using visualization of patient data to educate students, which is essential to the understanding of complex structures of the human body. The most basic but likely the most widely spread variant of using medical visualization for education is the presentation of static pictures, screenshots or short videos from real patients and their data taken from productive systems. They are presented in the class, books or in a further advanced version distributed via the Internet for pre- or post-lecture self-studies. Examples have been shown that use recorded videos and presentation slides [23, 24]. There are also anatomy self-study tools based on anatomical atlases that can be used to understand the human body part by part, which, however, are usually based on surface visualization [25] (see Section 3.1.2). On a next level, this data is provided as interactive instead of static media on a web page for self-study. Sophisticated viewers are necessary to provide functionality that can be used to examine interactive pre-processed media. Examples have been shown that allow the viewing of multiple image types with varying levels of detail and zoom [26] and to provide multiple viewpoints of 3D objects [27–34]. Such tools are easily accessible with a web browser, making them deployable almost everywhere, but usually with the necessity of additional software deployments. Compared to the presentation of pre-processed visualization, stereoscopic visualization is rarely used for anatomical education. However, some projects where initialized that utilized pre-rendered stereoscopic visualization [13, 35].

The usage of interactive volume visualization in real-time on real patient data is rarely used during the lecture session itself nor on web pages, especially not in stereoscopy. But real-time interactive visualization is also interesting in a classroom scenario [36], because it allows to provide the students ad-hoc with the visualization they require for a deeper understanding. Just as important could be the usage of advanced and interactive visualization for self-directed learning.

1.3 Necessity for easier accessibility of volume visualization

All previously described use cases require a system with fewest usage barriers as possible. Physicians for instance do already need to work with a multitude of systems in their daily life. The provision of applications that are similar to already known applications and/or require fewer steps in order to be used, might reduce the initial learning phase and, thus, increase its acceptance. Departments and institutions more often have stereoscopic hardware deployed in special rooms (e.g. conference rooms). But these might not be in the full control of the people who want to use their own stereoscopic visualization. Thus, even in the case of stereoscopic visualization, where a usage almost always requires stereoscopic hardware, it might be of advantage to reduce the need for additional software deployments. In the case of collaborations with stereoscopic visualization, remote participants might require different stereoscopic content for their setup.

There are basically two visualization types that can be used to render volume data, extracted surface visualization and direct volume visualization (Section 3.1.2). The former can often be rendered on standard computer equipment, which might also be powerful enough to render volume visualization of small datasets with low interactivity [37]. But the rendering of high-quality volume visualizations of large high-resolution volume data sets in real-time and in stereoscopy is only available on special imaging workstations, e.g. tethered clients of imaging modalities. Technical progression of imaging modalities and computer technology leads to increasing compute power but also larger and more detailed data. Because of this and the current necessity to use high-performing hardware, it is likely that full quality high-performing volume visualization will also require more than standard hardware in the future. Access to high-performing medical visualization is therefore limited.

Accessing volume visualization collaboratively by multiple remote participants can basically be solved by rendering the visualization locally on every participating client (client side rendering) or solely on a single server (server side rendering). As described in Section 3.3.2 in more detail, a client side rendering approach requires each participant to have adequate hardware to render the desired visualization and a local copy of the data set. In the case of a 0.5-2 GB CT dataset, the data transfer may result in an initial waiting time of 1-3 minutes on a fast 100 Megabit per second (Mbps) network, but 7 to 27 minutes on a standard connection of 10 Mbps. A server side rendering does not require a specific initial data transfer to each client, but instead a continuous stream of the images that are rendered on the server (see Figure 1.1). A single initial data

transfer might be necessary, but only between the data server and the visualization cluster, both are likely at the same location and, thus, likely have a fast interconnect. Each single image requires about 10-100 ms in order to be transferred on a 10 Mbps connection. Thus, server side rendering requires a higher network load during the usage, but is the only technique that provides an ad-hoc usability on lightweight client systems.

The research communities that use high-performance and grid computing have lots of experiences with centralized server-based computing and visualization. They already use approaches that could change the typical visualization usage in hospital environments. Many groups experimented with grid computing for medical use, e.g. for managing medical imaging data [38], for image data analysis [39], or for providing a central point to access visualization [40]. Also the parallel volume visualization was tested within a grid, however, without real-time interactivity [41]. A suitable solution for hospital environments might therefore be the centralized rendering of medical visualization on high-performing single or clustered computer systems.

Developments of the past years show the increasing importance of web-based Hospital Information Systems (HISs) [42]. These provide remote access to centrally stored patient data and only require a web browser on the client side. Web browsers exist for almost all devices and operating systems. They are already installed on most computers and, most importantly, are already known to most users. Thus, they usually do not require any special deployments, which makes web-based systems very easy to access. The advantages for hospitals are high, especially for large hospital corporations that operate a multitude of scattered hospitals and other health care centers.

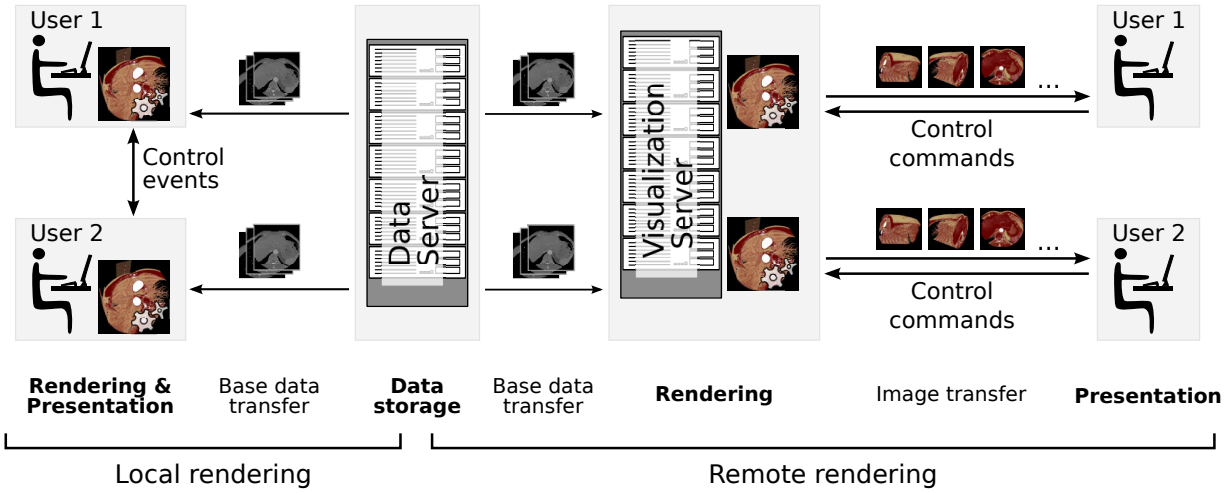


Figure 1.1. Local vs. remote visualization rendering. A data server initially holds the base data that is to be visualized (e.g. a hospital PACS), which needs to be transferred to the rendering computer. With a local rendering the visualization is created on the presenting device (left side). A remote rendering requires a visualization server that creates the visualization and exchanges it with the presenting device.

The idea of providing access to remote rendered, interactive medical visualization via web browsers is therefore straightforward. Many academic projects and medical technology vendors worked on this, as discussed in greater detail in Section 3.3. Liu at al. for example state that "techniques for supporting web-based interactive applications of high-resolution 3D Medical Images are highly desirable" [43]. However, it is always a trade off between providing simple access to the visualization and providing enough functionality. Web browsers with added software (e.g. Java or Flash) can provide functionality that is almost identical to native applications. Such added software provides simple to use development environments and the required functionality, which are therefore used for most existing systems. But their usage for remote and collaborative visualization systems results in a higher complexity on the client side due to additional deployments. Also the ad-hoc usage of such systems might not be possible, because of missing user rights on the local computer to install the added software.
With the recent development of Hypertext Markup Language (HTML) 5 technologies, it is now more easily possible to develop comprehensive applications for pure web browsers. It allows for example for a direct usage of the local 3D hardware by using JavaScript (Web3D), which was previously only possible by using added software. But secondly, there are also new and more efficient techniques to transfer data from the server to the web browser and vice versa, which can be used for remote rendered 3D visualization. Web technologies make use of a very modular architecture with multiple distinct technologies, each with a specific purpose. These modules can be combined as needed to produce a joint comprehensive application. Some projects already worked on the topic of providing remote rendered visualization as described in Section 3.3.

1.4 Rationale and Objectives

Current medical visualization systems provide comprehensive functionality for interactive image processing, collaboration, and partially also for stereoscopy in high quality. However, their usage is either bound to special imaging workstations and, thus, to specific locations or as described in Section 3.3 to remote visualization applications that require special deployments. None of them conjointly support all of the above described and desired visualization aspects with minimum user involvement (Section 1.1). This leads to following hypothesis:

> A high performing remote visualization system, specialized for stereoscopy and ease of use, can provide access to real-time interactive, stereoscopic, and collaborative medical volume visualization.

This hypothesis can be sub-divided in the following four research requirements, each highlighting a specific aspect that needs to be considered for such a system:

1. Minimum user knowledge and involvement during setup and usage of interactive remote visualization in real-time.

2. Support of multiple participants at different locations with different stereoscopic systems simultaneously.
3. Generic support of any existing and future visualization application.
4. Automatic quality adjustment during the runtime to optimize the balance between performance and quality on a given network condition.

Many applications, including modern Hospital Information Systems (HISs) and recent related work systems, are of pure web-based nature, which results in a wide availability of web browsers and people being familiar with it. Thus, an equally simple provision of highly interactive and stereoscopic volume visualization via pure web browsers would be beneficial. The following main question arose:

> Can we evoke a solution that fulfills all requirements by only using a pure standard web browser at the client side?

As described in Section 3.3.3, none of the related work systems can be used simultaneously with multiple stereoscopic setups nor do the pure web-based systems provide an optimized performance for the real-time usage on different network conditions. Thus, the focus of this work lies on fulfilling all requirements, but especially on requirement 2 and requirement 4. It is expected that such a system is used by not more than four simultaneously collaborating remote groups (e.g. classrooms) as discussed in Section 8.2.

2 Methodology

A proof of concept conduction is a suitable method to test the feasibility of using web browsers without added software to remotely access stereoscopic volume visualization in real-time and collaboration. A proof of concept is "taking an idea and investigating to see if the idea has merit" [44, Ch. 2.2.1]. Thus, the basic idea of this thesis was the development of a prototype application that implements the approach introduced in Chapter 1 and its usage in a real practical scenario. The overall methodology is illustrated in Figure 2.1 by referencing the corresponding methodology and result sections.
The first task was to identify existing approaches that allow to answer the given problem of distributing visualization by only requiring a pure web-browser. This was done via a literature analysis and resulted in a state-of-the-art description of related work and enabling techniques, which is described in Section 2.1.
Several of these techniques were potentially usable for the development of a prototype system. It was therefore necessary to evaluate these techniques in order to find the most promising technique for an efficient data transfer. This evaluation was done by developing, testing, discussing, and grading several simple rapid prototype applications based on the requirements, which is described in Section 2.2.
This evaluation resulted in a single technique selection, which was then used to develop a more sophisticated prototype that implements the whole approach as described in the introduction (see Section 1.4). This system was then used for an inter-continentally shared medical anatomy class and further scenarios, which resulted in observations that led to further prototype improvements. The optimized prototype was tested in detail to compare it with other visualization transfer methods and related work (see Section 2.3).

2.1 Literature analysis

The literature analysis was done on two levels: *(1.)* Finding literature about projects with the same topic and *(2.)* finding methods and techniques that enable such a system.
In order to get a quick overview about the topic in the beginning, the databases Pubmed [45] and Google Scholar [46] were searched. Pubmed is ideal to find publications in the field of medicine and medical informatics. Google Scholar in contrast has indexed a much broader scope of scientific and other publications (e.g. patents) and, thus, is a good addition to Pubmed. Google Scholar resulted in a very fast acquisition of popular literature ranked by search algorithms, but also in a

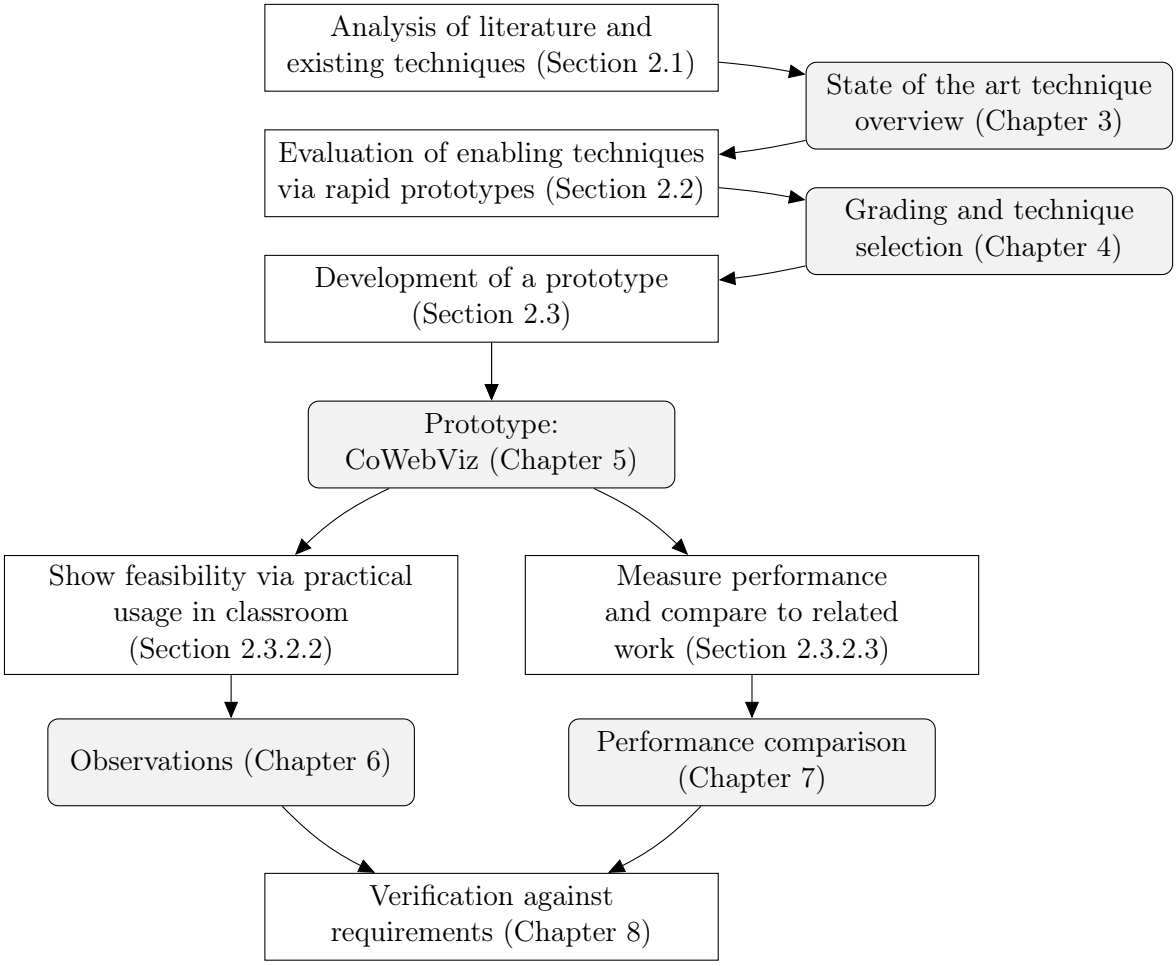


Figure 2.1. Flow diagram of the overall methodology (white boxes) and subsequent results (gray rounded boxes), each linked to the corresponding sections and chapters.

very large amount of publications that were off-topic. Among others, following search terms where used on both databases: "shared collaborative visualization", "shared medical visualization", "stereoscopic classroom", "state of the art visualization", "web visualization education", and "web-based remote visualization". In case of Pubmed, it was also tested to find literature via the MeSH terms "Education" and "Depth Perception" [47], which, however, only resulted in medical articles that were not relevant for the very technical problem of this thesis.

A systematic literature analysis was done afterwards to get a profound state of the art description of current systems, which is illustrated by a flow diagram in Figure 2.2. Pubmed was used again to find the usage of the desired systems in the medical discipline. IEEE Xplore digital library (IEEE DL) [48] and ACM digital library (ACM DL) [49] were used to find related work in technical disciplines. The search term that was used on each database was "remote AND visualization AND (browser OR web based)", which was adapted to each specific database search

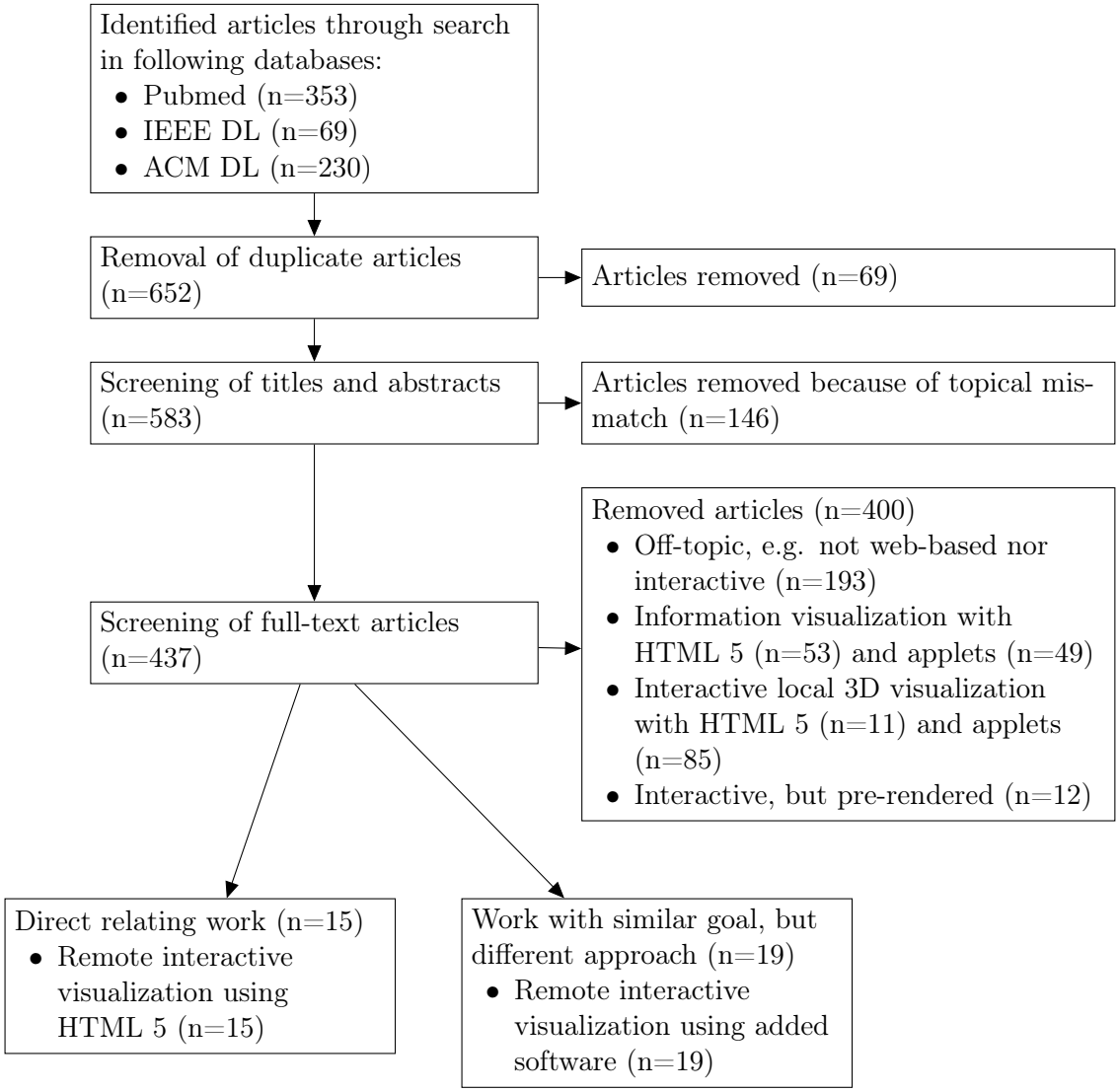


Figure 2.2. Flow chart of the literature analysis.

engine. The search was conducted in December 2011 without constraining the publication date. However, due to the search term, the first publications were not older than the early 1990's. The search was updated in February 2013 by constraining the publication date to the years 2011 to 2013. The process resulted in 583 publications after duplicate removal. These publications were screened on the basis of their title and abstract, which led to the removal of 146 publications. The resulting 437 articles were screened and categorized based on their full text. Categories are *1)* "off-topic", *2)* "information visualization"[1], *3)* "interactive local 3D visualization", *4)* "pre-rendered visualization" (such was often used for web-based medical education), and *5)* "interactive remote 3D visualization". Categories 2, 3, and 5 were further sub-divided into *a)* "utilizing

[1]Information visualization is a field of research about the visual representation of "non-spatial abstract data" [50], which is frequently used on web-browsers.

additional software" and *b)* "utilizing a pure web-browser". All articles categorized under 5 utilize a remote visualization rendering approach, but only the 15 articles in 5b ("interactive remote 3D visualization utilizing a pure web-browser") are considered as directly relating to this work [51–65], since they have the same scope of simple visualization access. The other 19 articles [66–84] were categorized under 5a ("interactive remote 3D visualization utilizing additional software").
To find the very technical methods of distributing the visualization, it was also necessary to find state of the art products and technologies that allow to distribute visualization via a network to a pure web-browser. The literature analysis described above was important for that, since the resulting articles most often describe utilized techniques. But it was also necessary to do standard web searches to find products and specifications. The HTML specification [85] was an important source to find potential web-based technologies.
The techniques found are presented in Section 3.3 and the most promising were evaluated in Section 4.

2.2 Techniques evaluation via rapid prototypes

The literature analysis resulted in the description of several projects, which utilize different techniques with potentials to transfer data from a server to a web browser client and vice versa (see Section 3.3). This section describes the evaluation process that was necessary to identify a single best performing technique suitable to develop a sophisticated prototype for the proof of concept conduction. This step was necessary, because web browsers were initially not being developed for real-time interactive data transmissions, which resulted in the development of many data transmission techniques, partially built around very basic connectionless techniques.

2.2.1 Method

The desired technique that is to be selected for the proof of concept conduction needs to fulfill the research requirements defined in Section 1.4. It needs to allow for interactive remote visualization in real-time (requirement 1) with potentials of being most performing (requirement 4). The prototype evaluation described in Section 2.2.2 was based on more granular aspects of these requirements, which are described in Table 2.1. Requirements 2 and 3 are not related to the basic visualization transfer technique and therefore not considered in this evaluation.
A rapid prototype is a prototype that implements a basic aspect of a system to show its feasibility [86]. In the case of this evaluation, the basic aspect is the transfer of visualization to a web browser and, vice versa, the control commands to the server. Several rapid prototypes were developed to test visualization and command event transfer techniques on the web browsers Firefox (Version 3.5.9 and 3.6.3), Google Chrome (Version 5), Safari (Version 4.0.5), and Internet Explorer (Version 8) on Ubuntu 10.04, Windows XP/7, and Mac OS 10.6. To keep the development effort low, they were based on existing software and, only in some cases, additional programming.

Table 2.1. Success criteria that need to be fulfilled by the desired technique to transfer visualization to a web browser.

Number	Criteria
1	The technique does only need a web browser without added software.
2	The technique shall provide remote visualization in real-time at the client side.
3	The technique should provide a high frame rate with a possible high efficiency.
4	The technique is useable with as many web browsers as possible.

Following rapid prototypes were tested:

Prototype 1: HTML 5 video steaming: HTML 5 video streaming is the streaming to and playback of stored or real-time video by pure web browsers [85] (see background in Section 3.3.3.2, Video streaming).

The HTML 5 video streaming solution was tested in two setups. *(1.)* The first setup was based on VLC[2] (Version 1.0.2), which was configured to capture a desktop metaphor and to provide it as video stream via its built-in Hypertext Transfer Protocol (HTTP) interface on a specified Uniform Resource Locator (URL). This setup was tested with the video codecs H.264 and Ogg/Theora on Firefox, Safari and Internet Explorer. *(2.)* The second setup was similar to the first, but used an additional Icecast2 streaming server[3] (Version 2.3) as middleware between VLC and the web browser.

Prototype 2: Pulling single images: Continuous pulling JPEG (pJPEG) is the continuous requesting and loading of the most recent image via a standard HTTP GET request [89] (see Section 3.3.3.2, Single image transfer).

The distribution of visualization by consecutively pulling single images from the web browser was tested using VLC (Version 1.0.2) and an HTML page with JavaScript. VLC was configured to consecutively capture images from a desktop metaphor and to store each as Joint Photographic Experts Group (JPEG) images on a web server directory. A JavaScript was executed simultaneously to the server software on the client side web browser, which consecutively updated the image source URL and, thus, the visualization image.

Prototype 3: Pushing/pulling Base64 images: The continuous pulling or pushing of Base64 images is the transfer of JPEG images that are converted to Base64 (see Section 3.3.3.2, Single image transfer).

[2]VideoLAN Client (VLC) is an open source media player and streaming platform that supports various streaming protocols and file formats [87].
[3]Icecast2 is an open source streaming server [88].

Only the pushing approach was tested via a comet style design pattern [90], which is a specific technique that allows to push data from a server application to a web browser by only requiring client side requests[4]. VLC (Version 1.0.2) was used to capture the desktop metaphor and store the images on a server side non-web-accessible directory. A JBoss server [91] (Version 5.1) was used to run a simple Java web application that continuously captured the newest image, to encode it to Base64, and to send it to the web browser by utilizing the Direct Web Remoting Ajax and comet programming library [90] (Version 3.0).

Prototype 4: Pushing motion JPEG: Pushing motion JPEG (mJPEG) is the concept of concatenating multiple JPEG images in one file or stream. Web browsers support mJPEG streams in the format of a Multipurpose Internet Mail Extension (MIME) multipart message [92] as shown in Listing 5.2 on page 57 (see Section 3.3.3.2, Single image transfer).

The mJPEG version was initially tested by including the streams of already existing webcams to web browsers, e.g. from the TelePresence Microscopy Collaboratoration (TPM) (see Section 3.3.3.2). The web browser only requires a very simple web page without JavaScript to be viewable. Afterwards VLC (Version 1.0.2) was used to capture a desktop metaphor and to stream mJPEG via its built-in HTTP server.

Another important aspect of real-time interactivity is the well performing transfer of the control commands from the client to the server. There are only two basic techniques qualified to transfer data from the web browser to the server: the Representational State Transfer (REST) style design paradigm and WebSockets. However, the real practical importance of optimizing the event transfer was only discovered after experiencing delays with the REST interface in the classroom. The comparison of both techniques was therefore done retrospectively by utilizing the proof of concept prototype as described in Section 5. It was extended on the server- and client side in the following way to measure the timing and rates of the command event transfers on different network types: On the client side, a JavaScript method sends a sequence of control events one after another to the server, which answers each command with an acknowledgment. The client measures the round-trip transfer times of each event during test runs of 30 seconds. Multiple tests were conducted for each network type and method, which are presented as mean.

2.2.2 Verification

Each of the visualization transfer rapid prototypes was tested to demonstrate its technical feasibility. This was done between a Desktop PC and a Laptop via a fast 90 Mbps connection (see test environment in Section 2.3.2.1). However, since these prototypes were very simple implementations utilizing different existing software, they were not comparable in terms of detailed performance measurements. The evaluation in Section 4.1.1 is therefore based on observations

[4]Server pushing via a persistent connection was not feasible without the usage of added software before the development of WebSockets.

made during the prototype testing and the discussion with the success criteria (see Table 2.1) and related work. The resulting evaluation is summarized in Table 4.1 (page 47) sub-divided by the success criteria. Each prototype's support of the criteria is graded on a scale from *1* (bad support) to *3* (good support) and "/" (exclusion), which is summed up as a simple score in the table's right-most column. Two of the techniques resulted in an almost equal score and were further compared analytically in Section 4.1.2.
In contrast to the visualization prototypes, the event transfer techniques were tested retrospectively by elevating real test data. These tests were conducted several times on networks with different bandwidth conditions between two cloud computing instances, which are described in Section 2.3.2.1. The results are summarized and scored equally to the visualization prototypes and presented as mean values in Table 4.2 on page 50.
The results were discussed within the working group and external scientists at the Argonne National Laboratories.

2.3 Proof of concept conduction

The rapid prototypes of the previously described technique evaluation did not have the required functionality to access interactive stereoscopic visualization in real-time. A more sophisticated prototype development was therefore necessary to answer all research questions described in Section 1.4, based on the rapid prototype technique evaluation. This prototype is called *Co*llaborative *Web*-based *Vi*sualiz*z*ation (CoWebViz) and was used in several practical scenarios and its performance tested.

2.3.1 Method

A proof of concept is to put a new idea or an approach into practice of a specific scenario, examine its feasibility and show whether it has merit [44]. Hence, a sophisticated prototype was developed that implements the approach defined by the hypothesis and the research requirements and, afterwards, utilized in different practical scenarios. The prototype development consists of following single steps, based on Agarwal et al. [93]:

a) Communication – an idea is discussed and leads to a quick plan: The foundational idea was the extension of the virtual anatomy class [36] by the usage of a much simpler to use client environment for the stereoscopic visualization (see introduction in Chapter 1).

b) Quick design – a quick design is modeled that represents the idea: An architecture draft was created and discussed in the working group. This included illustrations and descriptions of necessary functionalities, which were refined to the architectural illustrations in Chapter 5 and the requirements in Section 1.4. A further direct consequence of this phase was the

analysis of techniques in Section 2.2, which resulted in further discussions within the working group and external visualization experts (see Section 4.4).

c) Construction – the prototype is developed: C++ was selected as programming language as it allows for the development of a high performing system with all required external libraries. CoWebViz was developed as a web-based front-end for possibly every visualization application and therefore relies on applications that provide the visualization. In the tests and the proof of concept conduction this system was mainly vl3 and *Med*ical *Vol*ume *Vi*sualiz*z*ation (MedVolViz) (see Section 2.3.2.1). CoWebViz was developed for the main usage on Linux systems, since most scientific visualization applications are available on Linux, if not even developed mainly for Linux. The development and test environment is described further down in Section 2.3.2.1. The final prototype architecture is described in Chapter 5.

d) Deploy and use – the prototype is deployed and used to obtain experience: The resulting CoWebViz version was deployed on a visualization cluster and used in the virtual anatomy class in 2010 and subsequently also in other scenarios. The usage/test environment is described in the following Subsection 2.3.2. Its results are described in Section 6.

e) Verification – the results are verified against all research requirements: The verification method is described in the following Section 2.3.2.

f) Repeat – if necessary, conduct another prototype iteration: Two iterations of the prototype development have been conducted according to these steps. CoWebViz version 0.1 was early taken to a practical usage, after having an adequate usage opportunity in the virtual anatomy class of 2010. Since this version did not fulfill all research requirements, the prototype architecture was further extended to CoWebViz version 0.2, which was used for performance tests and further scenarios.

2.3.2 Verification

The developed prototype was constantly improved along with its practical application. Nevertheless, the development process can be sub-divided into the two versions CoWebViz 0.1 and 0.2. The most extensive practical application was the usage of CoWebViz 0.1 in the virtual anatomy class. This version was the state of the art available at the time the class started and is described in the beginning of Chapter 5. It did not have all the optimizations of CoWebViz 0.2, but had all the necessary enhancements in a basic version to ease access to the visualization by still requiring some manual technical involvement. The overall class procedure is described in the following Subsection 2.3.2.2. The specific steps of using CoWebViz, observations made during its usage about its technique und usability, and the advancements it provides compared to the previous class setup are described in the results (see Section 6.1). CoWebViz's class usage shows

the feasibility of the approach, but does not show the fulfillment of all research requirements defined in Section 1.4. The prototype architecture was therefore further extended to CoWebViz 0.2. Subsequent utilizations of CoWebViz 0.2 in other scenarios are described, additionally to the class observations in Section 6.2 in order to provide a hint of its feasibility and its importance for other scenarios.

CoWebViz's usage shows the feasibility of most research requirements in part or completely. These results, however, cannot be compared with other tools or projects, which led to the conduction of detailed performance tests of CoWebViz 0.1 and 0.2 presented in Chapter 7. The tests were conducted as described in the following paragraphs and subsections in order to verify the hypothesis by verifying the proof of concept conduction against the four requirements.

1. To present an overview of CoWebViz's performance and to verify the interactive usability in real-time (requirement 1), CoWebViz was tested and measured in monoscopic visualization mode. These tests show the different optimization steps of CoWebViz itself, but also provide a hint to the performance of other existing tools that utilize these patterns. These results also show the improvements originated by the automatic quality adjustment algorithm (requirement 4). In the test conduction, the visualization was transferred via different methods or rather design patterns to the client (e.g. send all processed images in static quality) as described in the first paragraph of Section 2.3.2.3. The tests were conducted on a visualization cluster (lightsaber) as server and a Laptop as client connected via different network types (1 Mbps, 3 Mbps, 11 Mbps, and 90 Mbps) using Google Chrome. The resulting data about frame rate, network throughput, quality, and CPU usage are presented as mean and standard deviation in Section 7.1. The data represents all recorded data entries that were marked as "in modification phase" by the test script.
2. The feasibility of providing different stereoscopic techniques (requirement 2) is verified by providing performance data of the supported stereoscopic visualization content types, which is then compared to the monoscopic test data. The test environment and data presentation is equal to the test environment described previously. The results are presented in Section 7.1.2 and shall be used to verify the first part of requirement 2 about the support of multiple stereoscopic systems.

   The second part of requirements 2 is the verification of supporting multiple simultaneously accessing users. This is done by a scalability test of using CoWebViz with 1 to 6 simultaneously accessing clients. These tests were conducted on a visualization cluster (lightsaber) as server and 1 to 6 cloud computing instances as client connected via different bandwidth types (1 Mbps, 3 Mbps, and 10 Mbps) using Google Chrome. The resulting data about JPEG quality, image file size, frame rate, and network throughput are presented as mean and standard deviation in Section 7.1.3. The data represents all recorded data entries that had a frame rate greater than 0. However, the selected cloud computing instance type was not equipped with a comprehensive connection of a continuously high bandwidth. This led

to phases without any data transfer rate at all while transferring larger amounts of data to multiple clients during this scalability test. Thus, the data represents all data entries with a frame rate above its median with the aim to represent the peak performance usage.

3. The generic support of any existing and future visualization application cannot be tested practically, but is a direct consequence of the system architecture. It is discussed in Chapter 8 by relating CoWebViz's approach to existing work.
4. In order to compare CoWebViz's performance and the automatic quality adjustment algorithm directly to the related work, two applications of the related work were chosen, tested and compared with CoWebViz (see Section 7.2). The first tool is the Virtual Network Computing (VNC) client Vinagre version 3.4 [94]. VNC [95] was chosen, because of its frequent usage for remote desktop access. VNC is a protocol that is implemented in numerous specific applications, mostly as native application. The second tool was screenleap.com [96], a recently developed purely web-based remote desktop sharing service. Screenleap.com allowed for no remote control at the time of testing, but its technique is similar to other related work and, in contrast to others, is freely available for testing. The tests were done by executing and equally measuring the frame rate of each application (in the case of screenleep.com by using CoWebViz's WebSocket event transfer). The frame rate was measured on the client side via a special application described in the second paragraph of Subsection 2.3.2.3. These tests were conducted multiple times on a visualization cluster (lightsaber) as server and a cloud computing instance as client connected via different bandwidth types (see Section 2.3.2.1) using Google Chrome. The resulting data about frame rate are presented in Section 7.2 as mean and standard deviation of the whole test session data and divided by quartiles in order to show highest and lowest performing phases.

The final verification of this proof of concept conduction is the verification of all observed and measured results described in Chapter 6 and 7 against the research requirements, which is done in the discussion (Chapter 8).

2.3.2.1 System environment

This section describes the environment of the proof of concept conduction, which includes the development, testing, and usage of CoWebViz.

Development environment. CoWebViz is a server application developed in C++ using the Vim editor [97] and Eclipse C/C++ Development Tooling [98] to write the source code and the GNU GCC 4.6.3 [99] for the compilation. It is build on top of following additional libraries:

- The Boost C++ libraries [100] provide a comprehensive collection of well performing and helpful functions and constructs, which are published under the permissive Boost Software License. CoWebViz requires Boost as a dependency for Pion, for the thread management and shared pointer.

- The Pion network library [101] provides the functionality of a lightweight HTTP server that is included into CoWebViz to manage most HTTP transactions. It largely utilizes the Boost asio library [102] and is therefore also published under the Boost Software License.
- The X library (Xlib) [103] is provided by the X-Server to use its functionality programmatically. It is published under the MIT License and required by CoWebViz to screen-scrape the visualization from a visualization application and to send control commands to the visualization.
- libjpeg-turbo [104] is a library that has the same interface and functionality as the Independent JPEG Group's standard library [105] with the addition of being 2-4 times faster. It is published under the wxWindows Library License and is required for its faster JPEG compression.
- FFmpeg [106] is a cross-platform library with various functionalities for multimedia handling. It is published under LGPL and GPL and required for its image scaling functionality and its various input and output formats, e.g. to integrate webcam streams.
- jQuery [107] is a JavaScript library that provides a common interface to the functionality of most existing web browsers. It is published under the GPL and required for various client side functionalities, especially for capturing mouse and keyboard events.

Server environments. CoWebViz currently depends on other applications to render the visualization, which was MedVolViz during the proof of concept conduction. MedVolViz is a medical volume-rendering engine based on the parallel-processing volume rendering engine vl3. vl3 was developed by collaborators of the working group at the Argonne National Laboratories. MedVolViz was mainly developed by Nigel M. Parsad at the University of Chicago's Department of Surgery. Together, it is a visualization engine for the usage on clustered computers in order to render high-resolution volume visualization with high-performance. Besides the high-performing nature, MedVolViz provides all the basic manipulation mechanisms to explore a medical volume dataset, as for example rotating, panning, zooming, clipping, and transfer function manipulations. MedVolViz supports multiple colorization modes, creating either standard grayscale or automatic realistic, spectral, and thermal colorized visualizations in a default or perceptual mode [6]. MedVolViz can render high-resolution stereoscopic images displayed as side-by-side stereoscopic content. [108]

CoWebViz was mostly used on lightsaber (including all performance test runs), which is a visualization cluster located at the University of Chicago's Computation Institute. Lightsaber had 8 worker nodes and 1 head node, each with an Intel Core i7-920 quad-core processor, 6GB of DDR3 1066 RAM and two NVIDIA 275GTX GPUs running in SLI mode. The head node merged the visualization rendered in parallel by the worker nodes. CoWebViz ran on the head node to capture the final visualization.

Client environment. CoWebViz's client was tested and used on following computer types:

1. Laptop: with a 2.4GHz Intel Core 2 Duo CPU, 4GB of DDR3 1067 RAM and a NVIDIA GeForce 9400M graphics card, running Mac OS X Lion
2. Desktop PC: with a 2.8GHz Xeon Nocona CPU, 1GB RAM and a Radion 9250 graphics card, running Windows XP
3. Cloud instance: Amazon EC2 Micro Instances with a 613MB RAM and no graphics card in zone us-east-1b, running Ubuntu 12.04

Most tests were conducted within the University of Chicago network, having following download conditions at the time of the test conduction:

1. LAN at the Crerar Library of the University of Chicago: ∼40 Mbps
2. LAN at the Computation Institute: ∼90 Mbps
3. WI-FI at the Computation Institute: ∼10 Mbps
4. LAN at the EC2 instance: ∼80 Mbps (The available throughput varied heavily during the test conduction using the micro instance)
5. Lower connections were tested via bandwidth throttling using wondershaper[5]

2.3.2.2 Evaluation methodology for the immersive virtual anatomy class

Since 2006 an immersive virtual anatomy class was held to undergraduate biology students at the University of Chicago [36]. The class evolved over time from a simple setup (with a direct visualization cluster connection) to a class with "multi-location, multi-direction and multi-stream sharing of video, audio, desktop applications and cluster based stereo volume rendering" [108]. Since this thesis' results affect the class procedure, the previous class usage is described in this section and the resulting modifications related to the usage of CoWebViz in Chapter 5.

The class environment. The class was lectured live at the University of Chicago (Chicago, IL, USA), but was setup to be completely shareable with remote classrooms. In recent years, it was shared with the Cardiff School of Computer Science and Informatics of Cardiff University (Cardiff, UK).

The local setup of a single classroom is illustrated in Figure 2.3. Each classroom had two projection screens: The first was used for the group-to-group videoconferencing system Access Grid [110] to display streams of the remote group (Fig. 2.3–1) and the shared web browser (Fig. 2.3–2). The latter was used to share 2D illustrations and photographs of human anatomy, which were stored on a web-accessible HTTP server directory. The second projection screen was part of a two-projection stereoscopic setup (see GeoWall in Section 3.2.2.4) and was used to present 3D stereoscopic visualization based on CT data (Fig. 2.3–3/4).

The class' collaborative functionality completely relied on Access Grid, which was used to stream video streams from the lecturer and the audiences to and from a remote location. Consecutively,

[5]The Wondershapter is a traffic shaper that allows to reduce the bandwidth connection [109].

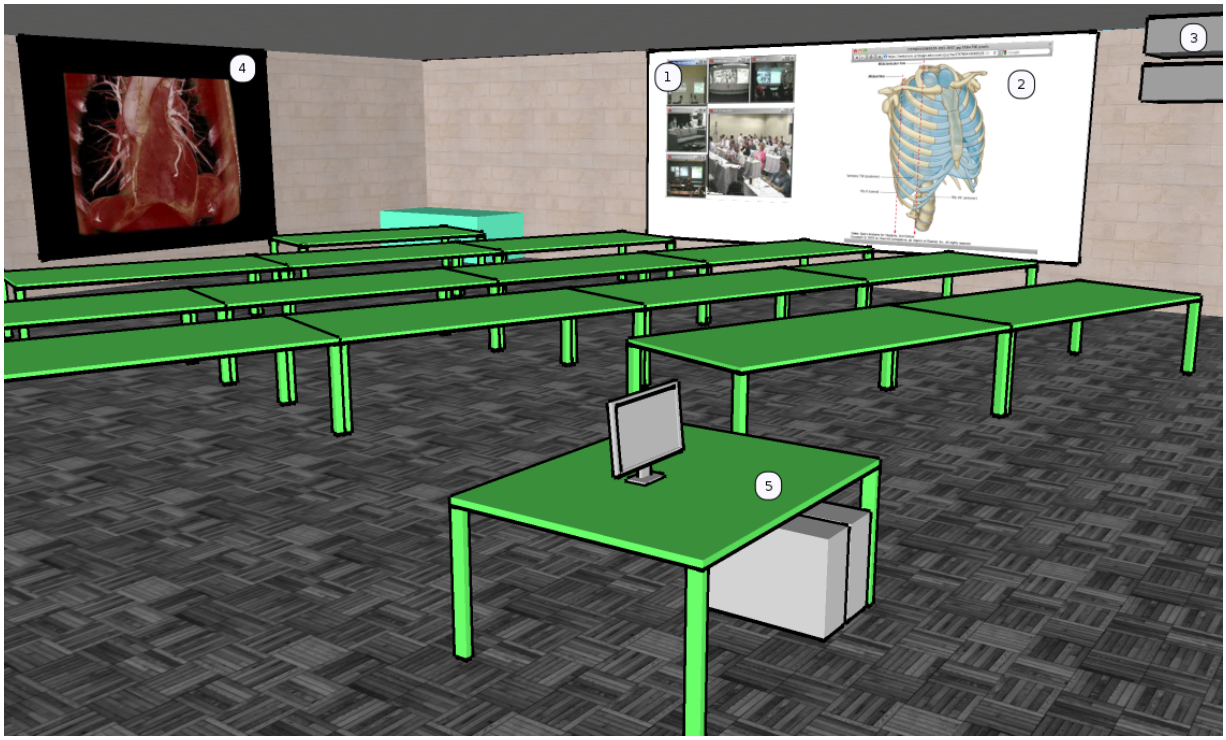


Figure 2.3. The class had two projection screens: The first was served by a single projector and presented video of the remote audience (1) and 2D illustrations (2), the second (4) was served by two projectors (3) and presented the stereoscopic visualization. The technical setup was controlled from a dedicated workplace (5).

each classroom required an Access Grid client deployment (see Section 3.3.3.2) along with several connected cameras, one directed to the lecturer and others to the audience.

The parallel rendering of medical visualization on a visualization cluster provides high-quality, high-performance and high-resolution images at all times, whether the visualization is modified or not. The rendered volume visualization was based on standard non-processed medical CT volume data, partially taken with contrast agents. The visualization was distributed by a specialized system that was closely developed with Access Grid.

Teaching procedure. Two types of media were used in synchronization during the lecture: at first drawings and photographs of human anatomy and, afterwards, stereoscopic volume visualization. A new lecture topic was typically introduced by using the labeled drawings. Afterwards, the same body region was presented in stereoscopy to clarify the spatial relations of the most important anatomical parts. The students had to wear polarized glasses whenever the 3D stereoscopic visualization was used for teaching. The lecturer controlled the stereoscopic projection on a laptop via a VNC connection to the computer that was part of the stereoscopic setup.

2.3.2.3 Performance test procedure

All performance tests were conducted via following methodology, which was required for a consistent data collection.

Testing CoWebViz and its specific visualization transfer methods. The test data described in Section 7.1 was created by a broad metadata logging functionality on the server- and client side. The data on the server side were recorded by CoWebViz's internal logging functionality that allowed to record data of any functionality, especially the sending of new images to the client. The visualization output channel was monitored to collect the JPEG quality, the image resolution, the current file size and other internal values. The control input channel was monitored to collect arriving command events. On the operating system level, CoWebViz was monitored by a process that continuously collects the following performance values using the Linux command top[6]: CPU load in percent and the real and virtual memory usage. During some tests, also the client's web browser was monitored by collecting the same performance values via top[6] as on the server side. The data was collected as summary of one second of the test conduction.
The tests itselves were conducted via a strict protocol with alternating sequences of modifying and not modifying the visualization for 180 seconds, with each sequence being 30 seconds long starting with a non-modification phase. The modification was conducted manually on the web browser by moving the mouse as fast as possible in order to demonstrate the peak performance usage. The mouse modification did not need to be automated because the event rate was higher than the visualization rate.

Testing and comparing CoWebViz and related work applications. The performance comparison of CoWebViz with related work applications described in Section 7.2 required black box testing on the client side. A simple Java application was written that simulated a visualization usage and measured the resulting visualization changes. As input, it sends mouse events in a defined area on the test system's desktop, where the visualization application was located during the test session. It keeps track of the colorization changes in a desktop's single pixel within the same desktop area as previously described and logs the delay between these changes. The output is a list of frame rates for each second of the test session. For each application, the test was conducted for 30 seconds at two different daytimes.

[6]Top is a standard Linux tool to display information about current processes.

3 Background and related work

This chapter starts with the presentation of background information about medical visualization (Section 3.1) and visualization with depth perception (Section 3.2). Both are essential parts of this proof of concept conduction and necessary to fully understand the thesis' overall approach.

Related work about techniques that enable collaborations of remote groups with shared visualization – the key interest of this thesis – is described in Section 3.3. It starts with an overview about the different sharing approaches and ends with the specific state of the art of sharing visualization.

3.1 Medical Visualization

The early beginnings of medical imaging technologies – technologies that image the interior of a living human body without surgery – lie in the end of the 19th century. At this time the discovery of X-ray beams led to first products that were capable of producing flat two-dimensional (2D) images of the three-dimensional (3D) body [111]. This development was followed by other 2D imaging modalities, as e.g. sonography and nuclear medical imaging. But still today, X-ray is one of the most important imaging modalities and builds the foundation of medical imaging [42].

The 3D data acquisition of a living body was not possible until about 1972 [112], when the development of computed tomography (CT) allowed the first time the recording of 3D image representations of the living human body. 3D/4D Sonography, MRI, Positron Emission Tomography (PET), and Single Photon Emission Computed Tomography (SPECT) are examples for further important developments that acquire 3D data [113, Ch. 3].

The new possibilities given by the acquisition of CT volume data stimulated medical imaging technologies, which concluded in the beginning of medical visualization as new scientific field in the late 1980s. Medical visualization is a specialty of scientific visualization and deals "with the analysis, visualization, and exploration of medical image data" [113, Ch. 1].

The following sections describe the basic concepts of 3D visualization in order to highlight the need for volume visualization and the need to use it via remote visualization approaches (Section 3.1.2), based on an overview about 2D and 3D data types (Section 3.1.1).

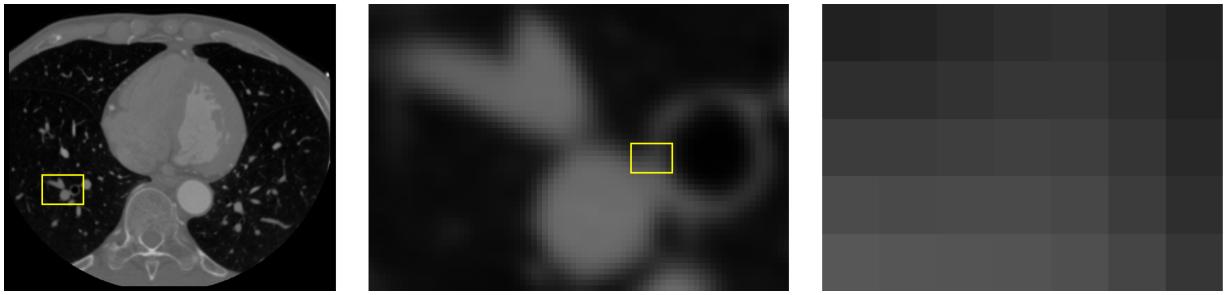


Figure 3.1. A single cross-sectional CT image on the left and cutouts on the right. Each cutout is magnified to show its pixel structure.

3.1.1 Imaging data types

This section provides a basic overview about 2D and 3D imaging. Both imaging types are also used for more advanced imaging techniques, as e.g. time-resolved, multi-modal imaging, or a combination of both [114].

2D imaging. Technically, a 2D image is a 2D array of pixels (a raster graphic). The pixel structure is illustrated in Figure 3.1, with a CT image slice on the left and two magnified cutouts on the right. Pixels have an image-specific bit size that may be further divided into sub-pixels. In the case of grayscale, the whole image pixel bit size is used for a single gray value, which e.g. allows in the case of a 12 bit image one of $2^{12} = 4096$ distinct gray values for each pixel. In the case of color images, the bit rate is divided by the number of sub-pixels. An RGB[1] True-Color[2] image allows for instance for 256 distinct values for each color, which results in 16,777,216 distinct color values for a pixel. The bit rate values and image resolution directly results in a specific file size of an uncompressed image. Compression algorithms are used to reduce the image size by multiples of the original. [3, Ch. 3]

Digital Imaging and Communications in Medicine (DICOM) is the main standard for image processing (e.g. storage and transfer) in medicine [115]. It supports a built-in native image format without compression and various external encapsulatable standards. Among them are Run Length Encoding (RLE), JPEG file interchange format (JIF, JPEG), JPEG lossless, JPEG 2000, and MPEG2 [116].

Over 60% of all diagnostic imaging procedures are based on the 2D projection of X-rays [3]. An X-ray image is the result of the projection of X-ray beams on a film cassette or a digital detector array, which are created in a vacuum tube. The resolution of X-ray images differs depending on the application. A typical X-ray resolution is 2048x2048 to 1780x2160 pixel with 12 bit per pixel. The image resolution for a mammography is higher with up to 4000x5000 pixel.

[1] RGB is a color model consisting of the three colors red, green, and blue.
[2] True-Color uses 24 bit per pixel, with 8 bit per sub-pixel that leads to $2^8 = 256$ color values per sub-pixel.

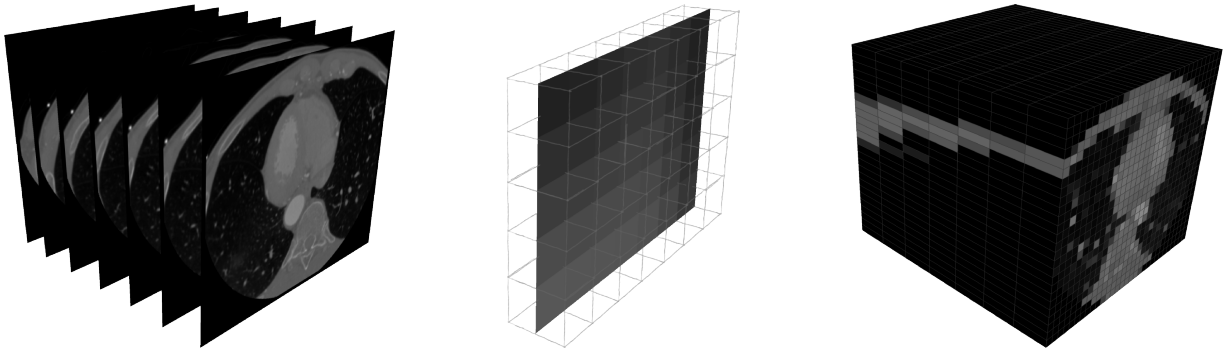


Figure 3.2. In medicine, a volume cube of CT or MRI consists of a multi-plane image stack (left), where the voxels are represented by the pixels of each plane (x and y coordinate) and the distance between the planes (y coordinate, center). The resulting volume cube on the right is shown with a very low resolution to illustrate single voxels.

Subtraction angiography and nuclear medicine use a lower resolution of 512x512 and 128x128 pixel, respectively. [3, Tbl. 2.1]

3D imaging. A volume dataset is a 3D array of values (e.g. color or gray/intensity values). In contrast to the pixel of 2D images, each volume data value is called a voxel[3]. Medical volume data is often based on multiple cross-sectional images of the body as illustrated in Figure 3.2. The x- and y-axis of volume data is typically represented by the layers of the cross-sectional 2D images; the z-axis by the shifted images (see Figure 3.2).
Volume data is a discrete three dimensional set of measured data values. Each voxel is a scalar value on a 3D vector, which can be imagined as a cuboid (voxel) with the value in its center. The width and height of each cuboid is typically equal. Since medical volume data is usually put together of cross-sectional images, the voxel height depends on the z-axis distance between the images. Thus, the voxel height is typically larger than its width and depth. If the voxel is a cube, the dataset is called isotropic, otherwise anisotropic.
Multiple cross-sectional images of one dataset are typically stored as multiple DICOM images with same identifiers. Besides the storage as 2D images, DICOM also allows for 3D compression of volume data using JPEG 2000 [117].
The data size strongly varies by modality and the specific examination. CT imaging slices are typically 512x512 or 1024x1024 pixel with 12 bit per pixel, but newer devices can provide a resolution of up to 2048x2048 pixel [118]. Depending on the resolution and slice thickness, a typical dataset with a resolution of 512x512 pixel and a slice thickness of under 2 mm can vary between 75 MB to 400 MB, but can reach, in the case of a full body scan or higher resolutions, more then 1 GB. [3, Ch. 4]
A typical MRI examination results in a volume dataset with a resolution from 64x64 up to 320x320 with 12 bit per pixel [3, Tbl. 2.1]. Functional Magnetic Resonance Imaging (fMRI) and

[3] Voxel is an acronym for volume element in the naming style of pixel.

Diffusion Tensor Imaging (DTI) techniques need to acquire multiple volume datasets for a single examination. In case of fMRI, this can be up to 400 volumes per examination. [3, Ch. 4]

3.1.2 Basic visualization techniques

This section described the two fundamentally differing medical visualization techniques. Whereas volume visualization is the direct rendering of the volume dataset into a 2D image, surface visualization requires additional processing steps via geometrical objects to get a 2D image.

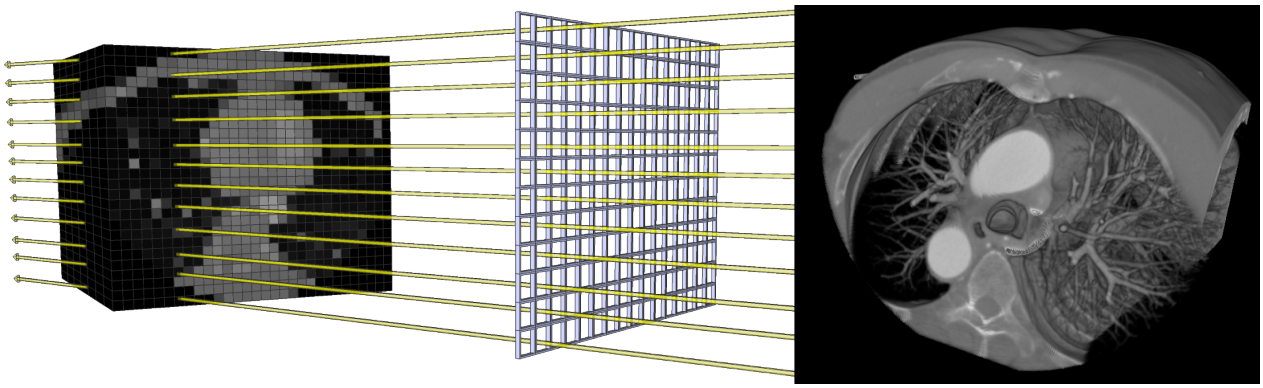
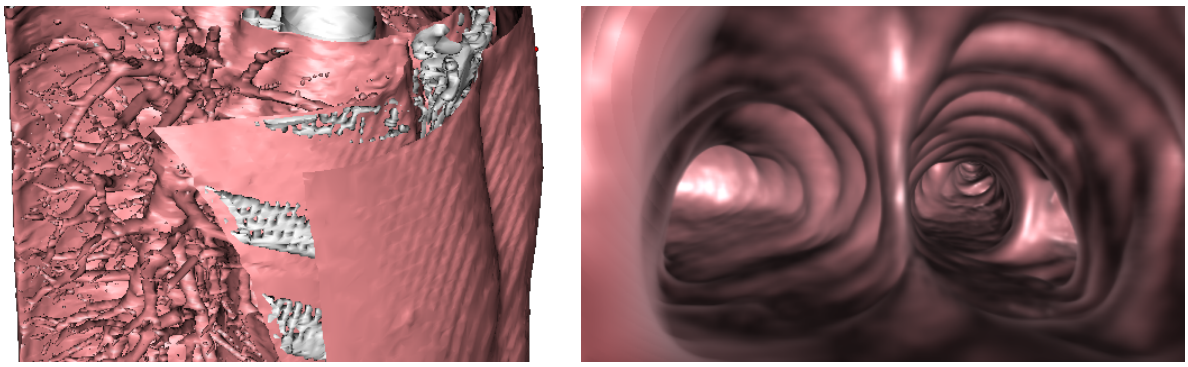


Figure 3.3. The principle of volume visualization: The volume cube introduced in Figure 3.2 is shown again with a low voxel resolution. Virtual rays traverse the volume, each directed from one pixel of a virtual image plane (center) and result in a grayscale volume visualization on the right (Created by MedVolViz [108]).

Volume visualization. Most display devices (e.g. standard displays) require a 2D image as input. Volume data, in contrast, is a 3D array of values, which needs to be converted into a 2D raster graphic to be displayable. This process, of directly mapping a complete set of volume data to a raster graphic is called (direct) volume visualization. It is a mapping of all volume data voxels to raster graphic pixels, without the usage of intermediate formats.
A very basic algorithm to create such a volume visualization is volume ray casting. In a simplified version, ray casting works as illustrated by a virtual scenery in Figure 3.3. This virtual scenery has a camera that is positioned in front of the volume dataset. In-between the camera and the data is a virtual image plane, a raster with the resolution of the resulting image. For each image plane pixel, a virtual ray is sent from the camera that traverses the whole volume. Each ray accumulates every voxel value on its path, which results in a calculated single value for its pixel. Transfer functions are used to select and change the representation of each voxel as the ray traverses the volume, e.g. used to colorize or hide specific voxels. [113, Ch. 8]
Every modification of the scenery (e.g. zooming or rotating) requires a whole new volume rendering. Alternative algorithms have been developed to render a volume with less hardware requirements, e.g. shear-warp [119] and splatting [120].

Surface visualization. A surface visualization is the direct rendering of geometric objects (polygons) into 2D images. A 3D dataset of medical imaging, however, is initially only available as voxels. Hence, this data needs to be transferred into polygonal data by applying manual, semi-manual or automatic segmentation algorithms. The aim of the segmentation is to determine compartment edges (e.g. of vessels and bones) in the volume dataset, on which the polygons are created. Compartment edges are represented by a gradient of intensities over several neighboring voxels (e.g. from skull to scalp). Multiple voxels along a compartment edge have the same or a similar intensity, since they represent the same material. This intensity is called isovalue, which could also be a value range instead of a single value. The combination of all isovalues along a compartment edge creates a surface, which is called isosurfaces in correspondence to an isobar and isoline in meteorological weather and geographic elevation maps, respectively. The resulting surface data (polygons) can directly be rendered to a raster graphic or stored via intermediate file formats, e.g. Virtual Reality Modeling Language (VRML), for a later rendering, potentially on another computer. [113, Ch. 7]
A surface visualization, in contrast to the scalar value grid of a volume visualization, is based on polygons that only represent about 10% of the data of a volume visualization. Furthermore, the choice of the segmentation algorithm might influence the computational data interpretation, which might cause different visualization images and, subsequently, misinterpretations of the data. [121]



(a) A shaded surface display of the abdomen in a two-color colorization.

(b) A virtual bronchoscopy with the view from the trachea into the two primary bronchi.

Figure 3.4. Two surface visualization examples (Created by OsiriX [122]).

3.2 Display techniques with depth perception

Still the most common way to present visualization is by just presenting 2D raster graphics on a 2D display device, as for instance standard computer displays, projectors, handheld devices, and even printouts. This is done independently of whether the visualization is a 2D line drawing or a 3D volume rendering.

A real-life scenery has, in contrast to such simple display devices, a depth that the human vision system perceives via multiple cues, some of which can also be experienced in simple 2D images (a monoscopic image). These depth cues are described on following example with a city street scenery of cars and houses: A car usually has a *known size*, which helps to identify the *relative sizes* of objects that are close by, as for example a house wall. Another car, which is covered by the first, is clearly farther away from the viewer then the first (*interposition*). A strong *light* source in the back that leads to long *shadows* in the front helps to give a cue of the distance of the light and the distance of the car to the viewer. These and other cues are some of the ones that exist in all monoscopic images, which are not solely bound to photographs but also to medical visualization types. But if the viewer would have multiple 2D images of the same 3D scenery taken from different positions in the scenery, it would see all objects from different positions and, thus, would get a much broader impression of the three-dimensional positioning of all objects. Another way to get this impression is by moving through the scenery via monoscopic images. Such a *motion-parallax* can be experienced while moving a virtual or real camera through a virtual scenery or by manipulating (e.g. rotating, zooming) these objects in front of the camera. [123, Chap. 11]

Stereoscopic images additionally allow for *stereopsis* (also called retinal disparity or stereo parallax [124]), which simultaneously provides the brain with two images of the same scenery from a slightly different angle (based on distance between the human eyes). As in real-life vision, the brain can use these images to calculate the depth of all objects and, thus, provide an intuitive perception of depth. Not all existing depth cues are already served by currently popular stereoscopic systems, which therefore might still lead to personal discomfort while viewing stereoscopic visualization [10]. Such cues are for example the ones that are based on eye muscle movements, which are *accommodation* (the eye's adaption to the distance) and *convergence* (the lateral and nasal eye movement) [123, Chap. 11].

Two or more 2D images of the very same object have to be provided for a single stereoscopic visualization. These images have to be created (e.g. rendered or taken by a camera) in one point in time from a slightly different angle [125]. A specific stereoscopic content is often necessary to be used on specific stereoscopic display devices, which is the format in which these images are provided to the system. An overview of common stereoscopic systems and often supported content types is therefore provided in Section 3.2.2. These devices have in common that they provide a specific method to lead a left and right image to one or multiple user's left and right eyes, respectively. But they particularly differ in the way, they achieve this. The most basic and existing principles are therefore described in Section 3.2.1. The list and description of these techniques is not complete and is rather provided to give an overview of different techniques and ideas.

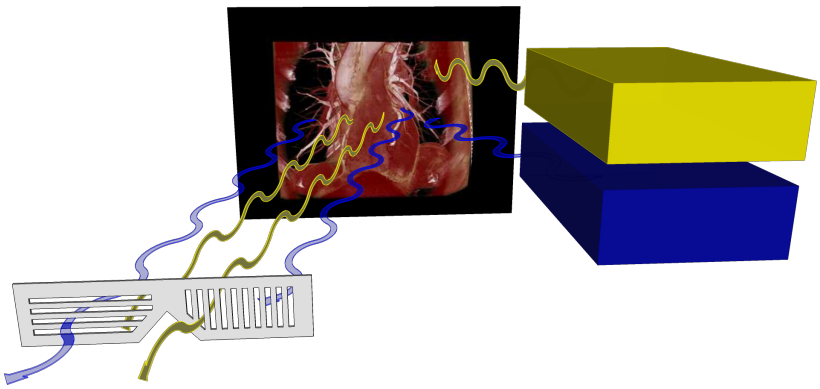


Figure 3.5. The illustrations shows two projectors on the top right, each projects one of the two stereoscopic views through polarized filters on a projection screen. The left view (blue) is projected using a linear horizontally aligned filter and the right view (yellow) using a linear vertically aligned filter. The viewer can view the stereoscopy using a pair of polarized glasses.

3.2.1 Stereoscopic display principles

The following sections describe the three basic principles of providing a stereoscopic presentation: passive, active, and auto stereoscopy. Each stereoscopic principle depends on some kind of display device that simultaneously presents stereoscopic content with two or more images of the same scenery. The two images that an user can see at one point in time are called left and right view for the left and right eye perspective. If the system provides more than two views, these can e.g. be viewed after head movements. The following sections provide an overview about the stereoscopic principles by referencing common examples.

3.2.1.1 Passive stereoscopy

Passive stereoscopic methods are all methods that allow the viewer to experience the depth perception by using some type of utility, which does not require any electronic control or synchronization with the display device. [126]

Polarization. The light's property of electromagnetic waves allows for polarization by using the fact that field vectors are always rectangular to the direction of the light's distribution. Therefore, light can be polarized linear (all field vectors are directed in one or the opposite direction) or circular (all field vectors circulate left or right handed to the distribution direction). [127, Chap. 10.2]
This can be used to provide stereoscopy by polarizing each image in a distinct way. A linear polarization encodes the left and right view with light that has all field vectors aligned in a horizontal and vertical direction, respectively. A circular polarization encodes the left and right view with left and right handed circulating light, respectively. Stereoscopic setups utilize equal

polarized filters on the projector and viewer side, which are distinct for each eye perspective [128]. A viewer can tilt its head without impairment in the case of circular polarization, but not in the case of a linear polarization. Polarized glasses do not interfere the visible light spectrum and are retaining the view's correct colorization.

Spectral division. The visualization views can also be encoded into the visible spectrum of the light. A long existing technique is anaglyph stereoscopy, which utilizes two views that are encoded in one of two complementary colors superimposed in a single image. A typical color combination with a good color representation is red-cyan[4] [128, 129]. A viewer wears glasses with colored lenses, the left and right lens are colored in correspondence to the colored views. However, through to the color encoding, anaglyph images have the disadvantage of providing a stereoscopic visualization with colors that are not equal to the original image colors. This principle has the advantages of being perceivable on almost any display devices (including printouts) by only requiring low cost glasses and, therefore, is widely available.
A similar variant with a better color experience is INFITEC (INteferenzFIlterTEChnik) [130]. This method divides the visible spectrum of the light into six spectral bands, with two bands for each of the three colors red, green, and blue.

3.2.1.2 Active stereoscopy

Active stereoscopic methods require the viewer to use a special utility (e.g. shutter glasses), which is synchronized with the display device. [126]

Shutter systems. Shutter systems require each viewer to wear glasses that are synchronized with the display device. The left and the right view is shown on a single display device consecutively one after another, each for a very short time frame. The synchronization is necessary to prevent the left eye from seeing the right view and vice versa. It is done by blackening the opposing lens of the currently presented view (e.g. the left lens is transparent when displaying the right view) [126]. The consecutive presentation of both views has to be done with a frequency that is higher than used on standard displays (e.g. 120 Hz, 60 Hz for each view) to keep up the impression of a movie and to avoid flickering.

3.2.1.3 Auto stereoscopy

Auto stereoscopic methods provide a stereoscopic perception without any additional tools on the viewer side. A wide range of auto stereoscopic techniques already exist, but many are still in development. [124]

[4] In the RGB color model, the left view is often encoded in the red channel and the right view in the green and blue channels [129].

Spatial multiplexing. Spatial multiplexing is the presentation of the whole stereoscopic content simultaneously, with all views being merged into a single image (pixel or sub-pixel wise). It requires a special display that is usually based on standard high-resolution display panels, which is overlaid with an additional optical component directly in front of the panel. This component is responsible to lead the viewer's eyes to the proper view's pixels. Such systems can provide one stereoscopic view (a left and a right view), but also multiple stereoscopic views as e.g. 5 or 8 [131]. Multi-view systems allow the viewer to see different stereoscopic views of the same scenery during a head movement. [124]

Frequently used techniques are *parallax-barrier* and *lenticular* displays. The first for instance utilizes opaque stripes that block the viewer's eyes from seeing some pixels from specific viewing positions [132]. The second uses half-cylindrical lenses to lead the view from specific viewing positions to the according view pixels [133]. Integral displays are similar to lenticular displays, but use a 2D array of small hemispheric lenses, instead of vertically aligned lenses [134]. They allow the viewer to move the head not only to the left and right, but also up and down [135, 136]. Spatial multiplexing devices also exist in variations with head tracking [137] to provide specific viewing zones that automatically adapt to one or multiple viewers. Other systems are described that use some kind of display or projection technique in correlation with various types of mirrors to lead the views to the viewer's eyes [138].

All methods mentioned above have in common that they have optimal viewing zones in front of the display and that the resolution of the underlying display is virtually reduced with more views. In the case of the parallax-barrier and lenticular displays the resolution is only reduced horizontally, but in the case of the integral display also vertically.

Volumetric. Volumetric systems provide an almost real-life representation of virtual objects, which allows the viewer to walk around the device and view the object from all sides.

Swept volume displays require a moving display device and many single views from all around the object that is to be presented. While the rendering of the views is complex but straightforward, the presentation needs to be done by some mechanism that allows to present the views at the right position, time, and angle in reality. "The key to this technique lies in interpreting the input images as 2D slices of a 4D function – the light field" [139]. Such a presentation could e.g. be done with a display panel mounted on a rotating plate [140] or by projecting the visualization on a rotating projection surface [141, 142].

A static volume display is a technique that presents any visualization pixel statically at all times at the right location. One example is the DepthCube, which is presented by Sullivan et al. [143]. It consists of a stack of 20 transparent liquid-crystal panels, each presenting a single volume slice. An early stage development to display volumes statically is the usage of infrared lasers to create plasma dots right in the air. Saito et al. [144] describe such a system that can create 1000 dots per second in a 50x50x50 cm space. Hoshi et al. [145] describe such a system with colorization that can even be "touched".

Semi-volumetric. Modern handheld devices allow for a completely new way to explore data. Such devices are small and have many built-in sensors that provide enough data to get it's exact spacial orientation. Subsequently they can be used as a movable window to a virtual dataset in the real world. They do not necessarily provide stereoscopy, but still a depth perception via motion-parallax. J. Francone et al. [146] describe for instance a system that constantly re-renders a 3D scenery towards the viewers current viewpoint. This allows the viewer to see other sides of the visualization after a head or device movement. A built-in front camera is used to get the viewer's position and distance to the device, which is then used to render the specific visualization. Subsequently, the viewer can explore a virtual object with a tablet PC as if the tablet is a real world representation of this object itself in the viewer's hands.

3.2.2 Common visualization setups

The main principles of providing stereoscopic visualization were described in the previous Section 3.2.1. This section has its emphasis on the technical setups that allow to use stereoscopic visualization by utilizing these principles.

3.2.2.1 Standard monoscopic displays and projectors

Some stereoscopic principles can be used on standard 2D displays and projection screens. The simplest and most easily deployable method is the usage of *anaglyph stereoscopy* (see Section 3.2.1.1), which can be displayed on almost any display type, including printouts. To allow the best possible quality with fewest ghosting and other stereoscopic artifacts, the display is ideally color calibrated. Each viewer does only require to wear glasses that correspond to the color encoding of the anaglyph images. Such are of low cost and may even be already at hand, because they are often shipped with magazines, books, and others. The number of possible viewers depends on the display size and quality, but is not limited by the technique. Its image quality is low, due to the color encoding.
Shutter systems can also be used on standard displays (see Section 3.2.1.2), because the left and the right view is presented one after the other. However, these systems require additional technical equipment for the synchronization between the computer and each pair of glasses. Such is available as external hardware for standard displays [147] or included in stereoscopic displays. Such systems are of relative low cost, as they are based on existing hardware. As previously, the number of viewers and the quality depends on the display.

3.2.2.2 Stereoscopic displays

Various companies provide stereoscopic displays using passive, active, and auto stereoscopy. Especially *active and passive stereoscopic displays* are already widely available on the consumer market, after the movie industry pushed stereoscopic movies in recent years. Some of the dis-

plays present both views simultaneously at the same time (e.g. passive displays). This, however, leads to a final horizontal resolution, that is divided into halves. The stereoscopic content that is directly being presented on the display is e.g. half-wide side-by-side, half-high top-to-bottom, interlaced or checkerboard. Such displays usually have a specific range in front of the display, from where the visualization can be experienced best. Thus, the number of viewers is limited by the space in front of the display and the number of available glasses.
Auto stereoscopic displays are also available on the market, but are compared to the passive and active displays of higher cost. Auto stereoscopic systems also provide two views, but their full capabilities can only be experienced with multi-view visualization. Multi-view systems, however, are more complex in terms of creating, processing, especially transferring, and displaying the content than in the case of the other two types. Auto stereoscopic displays usually can only be viewed from specific viewing areas in front of the display, which limits the amount of viewers. The visualization content that is directly presentable with depth perception on such a display is a single multi-view image, with all images merged into one (often on the sub-pixel level).

3.2.2.3 Tiled displays

A matrix of multiple horizontally and/or vertically arranged displays is called a tiled display. Simple variations of these are widely used at offices to span an increased desktop across two or three displays or projectors. More complex systems were developed to display high-resolution monoscopic visualization on up to 72 displays [148–150]. In contrast to a single projection screen that might have the same horizontal and vertical dimensions, a tiled display accumulates the resolution of each display. Tiled displays can be used to present visualization with a resolution above 300 Megapixel [150, 151]. Such displays system are not only very expensive, they additionally need a clustered computer to manage the high definition desktop metaphor and may need additional processing power to calculate such a high resolution visualization [152].
Tiled displays are also build by using active, passive, and even auto stereoscopic displays. Sandin et al. [153] describes e.g. a tiled autostereoscopic display with passive parallax barrier screen tiles, which allows to view a stereoscopic visualization with a resolution of 11,200 x 6,000 pixel. Febrettia et al. describe the CAVE2 [150], a passive tiled display with 36 Megapixel per eye, which are about 98,000 x 55,000 pixels.

3.2.2.4 Special projection screens

Special projection screens can be used to provide various types of large scale stereoscopic visualizations. Such systems exist that are usable by single persons or small groups as for example the CyberDome [154] or the ImmersaDesk [155].
A GeoWall [156] is a passive stereoscopic projection system with consumer grade hardware and polarized filters (see Section 3.2.1.1). It is one of the cheapest possibilities to build-up a large projection-based stereoscopic setup for a larger audience. It uses a polarization pertaining silver-

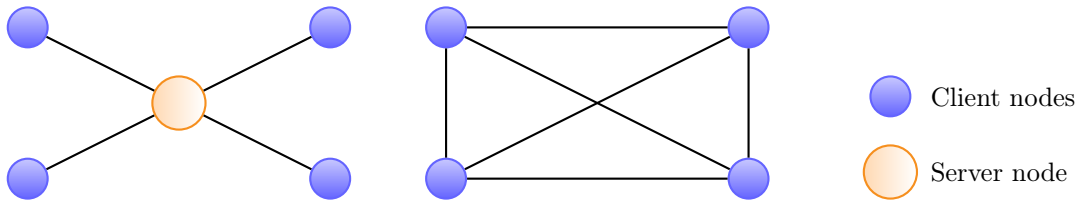
screen and two aligned projectors. Each projector projects either the left or the right view, both overlaid on top of each other on the screen. The similar principle is also used in many movie theaters, however with improved technology [157]. Even auto stereoscopic projection systems were described [158].
A Cave Automatic Virtual Environment (CAVE) is a cube like room with stereoscopic projection screens as walls [159]. Either rear or front polarization pertaining projection screens are used, each is typically served by one or two projectors. Each viewer requires polarized glasses and stays inside the cube to fully immerse into the visualization. There are many variants of CAVE projection systems, that for instance do include all walls, the ceiling, and the floor (full CAVE) or only include three screens around a corner. A CAVE also allows for improved immersive remote collaboration types [160].
There are even systems that provide a similar environment as a full CAVE, but on a large scale. The Allosphere [161] for instance is a large-scale sphere that combines large surrounding stereoscopic visualizations together with a 3D sound system.

3.3 Remote sharing of interactive visualization in real-time

Sharing interactive visualization in real-time is the viewing and modifying of visualization jointly by multiple participants at different locations for the specific time of a visualization session. It is an extension of a pure local visualization usage and requires its sharing, with all of its state changes between all participants. Thus, the most basic design principle of a remote collaborative visualization system is the choice of where the visualization that is to be shared is to be rendered. This may be done on a single central location (server side rendering) or on every client computer (client side rendering).
Advantages and disadvantages of client- and server side rendering are described in Section 3.3.2. This foundation is then used to describe the state of the art techniques that enable a shared visualization in Section 3.3.3. The related work that is directly related to this thesis is described in Section 3.3.3.2. A summary of it and a selection of additional prominent projects is presented in Table 3.2.
This section starts with the description of basic network characteristics that are important for a shared interactive visualization system in following subsection.

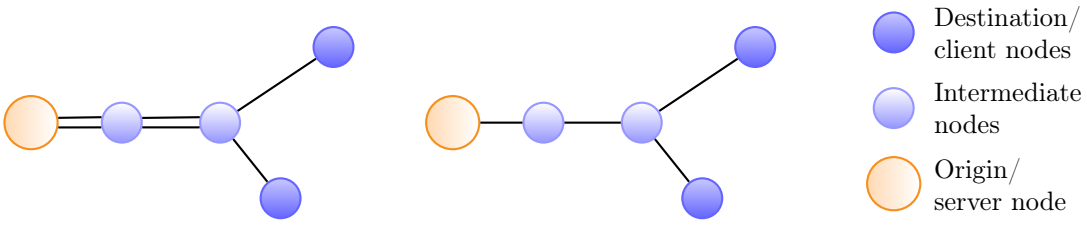
3.3.1 Network aspects of interactive systems

The network type, on which a remotely shared system is based, has a substantial influence on the visualization performance (e.g. frame rate) and network load (e.g. bandwidth usage). In general, following basic network aspects are to be considered for such a system: *(1)* the network characteristics (connection-oriented/less, routing schema, latency, and bandwidth) and *(2)* the network topology [162]. However, for an application that purely operates on a high level of the



(a) Star topology **(b)** Mesh topology

Figure 3.6. Network topologies, well usable for collaborative shared systems.



(a) Unicast scheme **(b)** Multicast scheme

Figure 3.7. Network schemes, well usable for collaborative shared systems.

Open Systems Interconnection Model (OSI) [163] as HTTP [89] in the case of this thesis, the network characteristics between any two participants are mostly pre-defined, especially with an ad-hoc usability in mind. To influence the network characteristics, an application needs to work on a lower level protocol (e.g. TCP or IP) and/or requires special hardware adjustments or resource reservations.

A characteristic with a high influence on the bandwidth usage of a collaborative system is the routing schema (unicast or multicast), which describes the method that is used to traverse data through the network. With unicast any data is sent multiple times in a separate transmission for each participant (see Figure 3.7a). Thus, a data transfer to n computers requires a server bandwidth of n-times the bandwidth of a single connection at the client side. Multicast is usually used, whenever the data needs to be shared with a group of participants. In this case, only a single set of the data is transferred for the longest common path and is divided into multiple transmissions to be transferred over the last network hops to the specific recipients (see Figure 3.7b). Since the data is only sent once at the server side, it results in a high scalability with a server bandwidth need of a single client connection. Multicast is especially useful for audio and videoconferencing techniques that need to send a high amount of equal data to different recipients. However, the usage of multicast networks depends on specific protocols or network adjustments and is not usable ad-hoc with a standard unconstrained network, as e.g. the Internet.

The network topology describes the interconnection of the system nodes, which basically is the network-related architecture. In a partial or full mesh topology, the data is transferred directly

from one computer to one, multiple or all others, each via a separate connection (see Figure 3.6b). A hierarchical or a star topology is used to transfer data via one or several central servers (see Figure 3.6a). [164]

Another aspect that is not directly related to the network characteristics, but influences any data transfer of client/server architectures using HTTP is the way the data transfer is initialized, whether it is a push or a pull mechanism [164]. Using *pull* requires every client to initiate the data transfer itself. In the case of using *push* the server decides when to transfer the data [165]. HTTP provides several techniques that allow to use push and pull, where push is often based on pull, e.g. using long-lived HTTP connections. When comparing these, the push approach results in a higher coherence and network performance [166].

3.3.2 Client and server side rendering

Another aspect that highly impacts on the performance and network load of such a system is the choice between rendering the visualization on the client or server side.

As described in Section 3.2, almost any current display device is based on 2D display panels or projection screens to display monoscopic (one image for one eye-perspective) and stereoscopic visualization (two or more eye-perspective images). Subsequently, the final result of a visualization creating process usually is a single or multiple 2D images. If these images are being presented at the location where they are being rendered, it is called a local or client side rendering. Vice versa, if the images are being presented at a remote location, it is called a remote or server side rendering. In the case of remote rendering, the visualization needs to be transferred via a network to a remote client computer, where it is then being displayed.

A natural but in terms of performance most determining functionality of interactive visualization systems is the modification of the visualization, as for instance to rotate, clip, pan, and zoom it. Any modification command results in a modified image that needs to be presented. In the case of a local visualization, every newly rendered image is directly being displayed on the display device via a fast connection in full quality. But in the case of a remote visualization, each modified visualization needs to be updated to the remote computer via a disproportionally slower network connection almost always in reduced quality after compression. For a real-time access to the visualization, the user requires to view the modified visualization within a very short time frame after the control command was given. This time should be as short and unrecognizable as possible, which is usually within few milliseconds [7].

Both principles are illustrated in Figure 1.1 on page 5 with the data provisioning as central service. The main advantages and disadvantages of client and server side rendering are summarized in Table 3.1. The following paragraphs describe the problems and prospects acompanied with client and server side rendering methods for sharing visualization with remote participants.

Client side rendering. A client side rendering requires every client to have a local copy of the base data and an application that handles the visualization rendering and synchronization with all other clients. Each client eminently requires proper graphics hardware that is sufficient to render the required visualization. If all participants need to work on identical visualization, each control command (e.g. requests to modify the visualization and system configuration) needs to be managed and synchronized.

The most apparent task of a client side visualization is the initial base data transfer to each client. A fast 100 Mbps network connection allows to transfere a dataset of about 0.5 to 2 GB within about 1 to 3 minutes. But it may take about 7 to 27 minutes on a slower but more commonly available connection of 10 Mbps. Whereas a participant might wait 3 minutes in the first case, an initial waiting time of 30 minutes is unfeasible.

A multi-user access control management is necessary to allow only one participant to modify the visualization at any point in time. The synchronization of all clients is therefore unavoidable to provide every participant with the exact same view. Technically, this could be solved generically

Table 3.1. Server side versus client side rendering.

Type	Advantages	Disadvantages
Client side rendering	◦ Only control commands need to be transferred on a continuous basis ◦ The server does only have to provide data and manage the synchronization (in specific cases)	◦ A visualization session starts with a possibly large and long-lasting data transfer from the server to the clients ◦ Every client needs adequate graphics hardware ◦ The synchronization between multiple clients can become complex
Server side rendering	◦ The visualization session has no time-consuming initialization phase (e.g. base data transfer) ◦ The clients do not require special hardware ◦ The data stays on the server (more secure) ◦ State synchronization is less complex (client to server)	◦ The resulting raster graphics have to be transferred on a continuous basis to each client as soon as they are created; Therefore the network traffic of the whole visualization session is larger ◦ The delay between giving a control command and seeing the resulting visualization is larger (depending on the longer round-trip time to the visualization renderer)

on the level of the operating system by recording and distributing the mouse and keyboard commands of the modifying participant to any other client. But in order to have more control about the visualization, it could also be solved on the level of the specific visualization application, by keeping track and distributing all visualization parameters (e.g. camera positions and filters). On principle, the data transmission could be done with both, star and mesh topology. If restricted to HTTP, however, any transmission needs to be done via the web server, which is a clear decision for a star topology. The absence of such synchronization would allow every client to interact in parallel on the same data, but different visualization views.

A real-time system requires the visualization algorithm to render the visualization result within few milliseconds to provide the user with the desired frame rate. Whereas the transfer of the control commands to the visualization application is negligibly short, the transfer to the remote clients causes a delayed rendering at the remote client. However, the data that is to be transferred is very small and, thus, a minor issue.

Transferring only the control commands to synchronize the visualization and not the visualization itself results in a low network load and allows for high frame rates on according workstations. In summary, a client side rendering has the main disadvantages of requiring a long lapse of time for the initial data transfer and high-performing computers for each participant.

Server side rendering. A server side rendering requires the server to have the data and the visualization application. The latter handles the rendering of the desired visualization and its continuous transfers to every client. Every client only requires a lightweight application to continuously receive and present visualization images and to capture and send the control commands to the server. As a consequence, only the server requires to have access to the base data, which is a very basic kind of data security. The server requires to have proper graphics hardware to render the desired visualization and the client only basic hardware.

Whereas client side rendering requires a long initial time to transfer the base data, a server side rendering only needs about 8 to 80 ms to transfer each modified image with a size of 10 to 100 KB on a slow 10 Mbps network. These file sizes allow for a theoretical frame rate of about 125 to 12.5 frames per second (fps).

The base data might initially be stored on a data server (e.g. a PACS) and therefore might need to be transferred to the visualization server. However, both servers will likely have a static location (e.g. the same data center) and, thus, likely have a much faster interconnect. In any case, only the host of the visualization session (the participant who initiates it) in opposition to any participant might need to wait for this initial data transfer.

Whereas the client side rendering almost only depends on a fast local rendering, the server side rendering depends on a fast transfer of the control commands from and the visualization to the client. Therefore, the time for a single visualization modification is a summation of the time to transmit the command from the input device to the visualization application, the time to

render the visualization, and the time to transmit the visualization to and display it at the display device. By nature, the time to transfer the data from and the visualization to the display device is much higher with a remote than a local visualization rendering. Real-time Interactive visualization systems therefore highly depend on a fast transmission of control commands and the subsequently modified visualization.
A server side rendering also requires the management of the currently controlling participant, which is clearly done in the same network topology as the visualization transfer (star topology). From using one visualization session with server side rendering follows that each client displays the exact same visualization. A second server instance with a second visualization session would have to be initialized, if one user needs to view a specific visualization asynchronous to the other clients.

3.3.3 State of the art of shared visualization techniques

This section describes the state-of-the art of techniques and projects usable and used for shared visualization. Due to the specific research hypothesis, the server side techniques are described in greater detail. A recent review of such work is also provided by Mouton et al. [55].

3.3.3.1 Client side rendering techniques

Until recently, all existing research projects that use client side 3D rendering were about the provision of special software that runs natively on the operating system or as added software within a web browser. Some of these systems were limited to specific web browsers. Tongleamnak et al. [167] for instance describes the distribution of ActiveX controls to render volume visualization at the client side, using an Internet Explorer. Also web browser extensions of standard web browsers were provided for such collaborations, e.g. the provision of custom sidebars to the Internet Explorer to list objects used for a collaborative annotation [168]. An often used framework is Java [169], which takes advantage of having an interpreter that is available on many platforms. Gerth et al. [170] for instance describe a Java web start application that is launchable via a web browser and can be used to show 3D body models with gene expression patterns. Java also supports 3D renderings via Java3D [171], which can directly being used within web browser Java plugins [172]. More general visualization plugins are used to load and render intermediate file formats, like VRML and Extensible 3D (X3D) files [37, 173]. In this respect, VRML is a very often used file format for medical visualization as for instance for remote image segmentation [174], exploring anatomical models [175], and many more [176–181]. Melzer et al. described a system that provides X3D files of medical data that is interactively viewable via plugins on a big variety of clients [182]. Such X3D visualization can also be combined with interactive data acquisition from the server [183].
With the recent introduction and broader support of HTML 5, there are now technologies to present 3D visualization with pure web browsers to the user. Interactive 3D visualization can be

rendered within a web browser by using WebGL, which allows to utilize 3D graphics hardware with OpenGL commands by just using JavaScript and the HTML 5 canvas element [184]. WebGL can directly be used for an interactive rendering. Congote et al. [185] implemented for instance a ray casting algorithm in WebGL to provide interactive and locally rendered volume visualization within the web browser. WebGL can also be used to render X3D files without the use of plugins. Such is e.g. demonstrated by Birr et al. [33], who presents a surgical teaching application that renders anatomical surface visualization. It stores data as X3D file, which is loaded and rendered with the X3DOM framework [186], a JavaScript library that allows the rendering of X3D and Scalable Vector Graphics (SVG) files via WebGL.

3.3.3.2 Server side rendering techniques

Various techniques can be used to distribute interactive visualization rendered on a central visualization server to multiple clients. The basically differing approaches are to transfer multiple distinct images consecutively one after another by using image compression standards and to transfer video streams compressed by a video codec with inter frame compression. The related work of each technique is introduced by describing the systems that deploy native applications, which were typically introduced earlier and are capable of providing the most advanced functionality in a rather unrestricted way. Such projects describe applications that transfer the visualization via any approach, with single images compressed via JPEG [104, 105], JPEG2000 [187] or Portable Network Graphics (PNG) or with video codecs compressed via MPEG or H.264 [188, 189]. Projects that describe remote visualization approaches via web browser added software and pure web browsers are presented afterwards.
Table 3.2 presents an overview of the direct related work and a selection of additional prominent projects, each is graded by its support of the research requirements (see Section 1.4).

Video streaming. Videoconferencing systems were developed to take a video from a video camera, encode it via a video codec and stream it to remote locations. Video codecs, e.g. H.264 [190] and H.265 [191], utilize similarities in the same and succeeding images with intra and inter frame compression to reduce the size of the resulting video stream or file [192].
Standard videoconferencing systems (e.g. H.323 supporting systems [193]) are often used for participant-to-participant conferences in an unicast network topology. More efficient and better performing videoconferencing systems are used for group-to-group conferences in a multicast network topology, e.g. EVO [194] and Access Grid [110]. Videoconferencing systems of each category usually provide the functionality to stream a participant's desktop metaphor, e.g. including a visualization application.
Access Grid was initiated by the Argonne National Laboratories (ANL) in 1994 and developed closely together with the Globus Toolkit architecture [195]. In the first place, it is a collaborative room based group-to-group conferencing software, but it additionally provides an infrastructure of

videoconferencing rooms (in May 2013: 326 nodes in 32 countries of the world), and at last it also is an access point to Grid middleware [195]. Access Grid is based on the multicast tools vic and rat [196, 197] and provides extensive functionalities in order to interconnect many different groups of people. It was and is being used between scientific project partners and for education [36], but also for other purposes as for instance during the SARS outbreak in Taiwan [198].

The streaming of stereoscopic video content can be realized by using the straight forward way of compressing a side-by-side or top-to-bottom video. More sophisticated codecs, as e.g. the H.264/MVC (Multiview Video Coding) [199], compress stereoscopic video with inter frame and inter view compression. The latter utilizes similarities of two or more related views. Two-view (stereoscopic) [192, 200, 201] and multi-view compression [202] are already introduced in several projects.

Special remote visualization applications can be used to interconnect standard computer up to high-end display systems. The access to monoscopic visualization via special software clients and devices is shown by many projects [204–206]. An example that generically extends visualization applications is described by Lamberti et al. [207], which parses the visualization application's Graphical User Interface (GUI) and provides it in a reconfigurable version for the remote client. Special high-end video streaming applications and network reservations can be used to transfer high-resolution monoscopic visualization of up to "12,800 x 4,800 pixels or an astonishing 61.44 Megapixels per frame" [208] over hugh distances, as shown on conferences [208–210]. Also stereoscopic displays can be used with remote visualization as described for an auto stereoscopic setup by Christodoulou et al. [211]. In this case, however, the stereoscopic views are transferred separately to the client, where they are then transformed to the specific auto stereoscopic content. Kurillo et al. [212] described the remote interconnect of CAVEs by using special applications and video streaming.

Special medical video streaming solutions are for instance being used in operating theaters to record sessions of endoscopic procedures [213–215] in order to stream them to a remote audience for educational use [216, 217]. Also stereoscopic videos of invasive procedures were already used for surgical education [218, 219].

The streaming of video is also being widely used on web browsers, as for instance on news sites, video clip sharing sites or commercial movie streaming services. Most commercial video streaming provider use added software to provide their content, but common video clip sharing sites increased the support of HTML 5 video considerably from 10% in 2010 to over 60% in 2011 [220], which likely will further increase. HTML 5 video streaming is a rather new alternative, defined by the W3C that does not require any added software [85], which by now is supported by most web browsers. However, the specific video container and codec was not defined by the HTML 5 specification, which results in a variety of supported formats by different web browsers.

Table 3.2. Overview of related work. This table includes the work that is directly related to this thesis (as defined by the literature analysis in Section 2.1) and other prominent examples. Each is graded by its support of any aspect of the requirements (defined in Section 1.4) into three categories: Yes (full support), Partly (partial supported), and No (no support).

	Remote & interactive	**Multiple simultaneous users**	**Web-based**	**Generic integration**	**Different stereo-systems**	**Auto quality adjustment**
Generic and scientific visualization tools						
1998 Richardson (VNC) [95]	Yes	Yes	Partly	Yes	No	Partly
2003 Bethel (VisPortal) [78]	Partly	No	No	No	No	No
2005 Ichimura [57]	Partly	Yes	Yes	Yes	No	No
2007 Zhao [56]	Partly	Yes	Yes	Yes	No	No
2008 Zhu [54]	No	Yes	Yes	No	No	No
2008 Mc Lane [61–63]	Partly	Yes	Yes	Yes	No	No
2009 Johnson (Envision) [69]	Yes	Yes	Partly	Yes	No	Partly
2010 Gedda (Guacamole) [58]	Yes	No	Yes	Yes	No	No
2010 Zhou [64]	Yes	No	Yes	Yes	No	No
2011 Wessels [60]	Partly	No	Yes	No	No	No
2011 Jourdain (ParaViewWeb) [59]	Yes	Yes	Yes	No	No	Partly
2011 Truong (Screenleap) [96]	Partly	Yes	Yes	Yes	No	Partly
2012 Śniegowski (Vitrall) [65]	Yes	Yes	Yes	No	Partly	No
Tools used for medical visualization						
2002 TPM [203]	No	Yes	Yes	No	No	No
2002 Temkin [32]	No	No	No	No	Partly	No
2006 Parvati [35]	No	Yes	No	No	Partly	No
2007 Silverstein [36]	Yes	Yes	No	No	Partly	No
2008 O'Byrne [26]	Partly	No	Partly	No	No	No
2009 Petersson [30]	No	No	Partly	No	No	No
2011 Suwelack [66]	Yes	No	Partly	No	No	Yes

There are some web browsers that only support the freely available video codecs Theora and/or WebM, while others only support H.264 [221]. HTTP live streaming [222] is an alternative that was developed to better integrate video streaming with web-based applications. It provides an environment to store a video on a web-accessible directory of a standard HTTP server by splitting the video file into multiple parts that are combined via a playlist file.

Video streaming is used for web-based learning systems via the provision of pre-recorded video and visualization [23]. Wiecha et al. [24] for instance described a web-based course system that provides pre-recorded video files and other materials for self-study. Another technique to provide video-like but interactive visualization is a Quicktime VR movie. This is a special format that stores multiple images of all around a 3D object, which is then explorable on the web browser client using a Quicktime plugin [29]. There are also medical system that utilize plugins to stream volume visualization as e.g. by using Silverlight [66].

Single image transfer. A computer-generated visualization does not change, if the scenery does not change. This stays in contrast with a video recorded by a video camera, which always leads to a different image on the pixel/colorization level, e.g. due to little movements of the objects. A visualization that is merely modified on an irregular basis can therefore also be transferred as single intra frame compressed images.

In current medical environments, *standard non-interactive 2D images* are mainly retrieved and transferred by using DICOM and Health Level Seven (HL7) standards. These standards can for instance be used with the Integrating the Healthcare Enterprise (IHE) workflow integration profile Cross-Enterprise Document Sharing (XDS) [223] to allow an interoperable document transfer. A newer addition to these standards is the Web Access to DICOM Persistent Objects (WADO), which is part of DICOM and can be used to access DICOM images on a server by using an HTTP REST style interface [224]. WADO also allows to convert a requested image to JPEG prior to it's transfer to be directly displayed via the web browser. It further allows to request image parts to minimize the necessary bandwidth usage, if a specific point of interest is known. Noumeir et al. [187, 225] showed the interconnection of XDS, WADO, and JPIP to access 2D DICOM images using interactive JPEG. This system allows the user to zoom and pan the visualization on the 2D layer. There are other medical systems that provide interactive viewers for DICOM images that can be integrated into web-based clients, e.g. of a PACS or HIS [226]. A possible scenario with these techniques is the viewing of volume data as single cross-sectional 2D images with [227] and without web browser added software [226, 228]. In the latter case, the viewers provide only very basic functionality.

Exceptionally large image files, which are too big in file size and resolution to be transferred and displayed as a whole, can be viewed with different resolution and zoom levels by using special applications and pure web browsers. Such is for example of interest in digital microscopic imaging, as demonstrated for parasitology [229] and pathology [230, 231]. Both are presenting large images via pure web-based image viewers, which enable the presentation of multi-layered images and different resolutions.

Interactive 3D visualization requires a faster interactive image-reloading functionality than it is provided and required by the previously described techniques for 2D image transfers.

The most often used application type that allows to access GUI applications on remote computer systems likely are *general screen sharing applications*. A popular example is Virtual Network Computing (VNC), which was introduced in 1998 and allows for an interactive usage of a remote desktop metaphor [95]. It constantly captures the desktop metaphor and transfers the changed image data to the client. Further improvements have been made to use remote 3D graphics hardware with VNC [232, 233]. VNC supports different kinds of reducing the data transfer, as e.g. sending only updated parts of the screen and having special window managers at the client that even prevent the transfer of pixel data from the server. VNC clients are developed for various systems, which also includes Java web browser plugins [72, 76, 234]. Large data centers have used such VNC plugins to make high-performing visualization accessible via web browsers [69, 71]. Alternative systems with a similar principle have been developed by various companies, which are also being used for research [73, 235, 236].

Many *special applications* have been described that utilize single image transfers for remote interactive visualization, based on web browser added software [77, 83]. Most often, they are built on top of Java [70, 74, 75, 79–82, 84], but also utilize other techniques, e.g. Silverlight [60]. Albu et al. [237] and Vazhkudai et. al [67] for example describe systems with Java-based clients for Computer-aided Design (CAD)-based liver cancer analysis and neutron scattering experiments, respectively.

Compared to the literature of interactive visualization systems that use web browser added software, the *pure web browser based solution* is rare (see comparison in Table 3.2). Whereas the previous paragraphs only provide examples of technique usages, the following paragraphs provide a full overview of the pure web-based systems as defined by the literature analysis.

The TelePresence Microscopy Collaboratoration (TPM) [203, 238, 239] is a project that allows to monitor remote resources (e.g. machinery) using remote web cameras via pure web browsers. It is for example being used to provide remote monitoring for earthquake researchers of the NEESgrid [240]. The system provides the functionality to modify the camera's point of view (tilt, pan, and zoom) from remote. The TPM cameras use a very simple but effective technique that pushes single images in a multipart HTML message to the client. This technique is called motion JPEG (mJPEG) and is most widely used for web cameras and, thus, supported by many web browsers.

A very early remote visualization system was demonstrated by Bethel et al. [78], who describe the VisPortal, a Grid-computing based rendering system that provides multiple renderings of a single dataset based on a pure web browser as client. This system, however, did not provide real-time interactive visualization, but rather allowed to render the visualization multiple times at once, which was afterwards downloadable as video files. Bohne-Lang et al. [241] described

one of the first systems that allowed for interactive visualization on a pure web browser via continuously pulling single images, which however is not optimized for a real-time usage. Similar applications with no real-time interactive functionality (low frame rate) have been developed for medical volume visualization [242] and other scientific visualization fields [54].
Pure web-based systems were also already being demonstrated to be usable for general screen sharing applications. Ichimura et al. [57] describe a very scalable system for remote education, which however can not be controlled from remote. It screen scrapes the desktop metaphor of the lecturer with the resulting images being stored as tiled images on a web server, from where the clients download the images after polling. Guacamole [58] is a special server application that is a combination of a web server and a VNC client which is accessible from HTML 5 web browsers. The system uses a pulling mechanism to fetch full Base64 encoded images. Base64 images, however, are about 30% larger than its binary counterpart. Screenleap.com [96] is freely usable service that allows to share the local desktop metaphor via a Java applet with remote participants. In contrast to the server, the remote participants just require a pure web browser, but do not have interactive control of the desktop.
Web services have also been used to provide interactive visualization as a service. Zhao et al. [56] describe a system that generically integrates visualization applications and can be used on a pure web browser. However, they also describe an image delivery delay of more than 500 ms per image, which is not highly interactive. A web service based framework is also used by Mc Lane and Greensy et al. [61–63]. It runs on various web browsers and transfers Base64-encoded JPEG images. However, the latency of 1 second and 500 ms (in a later version) for a single image results in a low interactivity [64]. Jourdain et al. [59] describe ParaViewWeb, a web-based access point to the ParaView rendering service that provides interactive visualization in real-time. Besides a Java and Flash applet, ParaViewWeb also provides a pure JavaScript-based client interface that uses long-polling to inform clients about a newly modified image, which then is instantaneously loaded as a binary image. Its web server utilizes web service interfaces to connect to the rendering service, where the rendered visualization is statically compressed with a JPEG quality of 50 in order to decrease the image delivery time. Wessels et al. [60] suggest to use web sockets to transfer a server side rendered visualization, by pushing an image as soon as available. They also describe the necessity of using Base64 encoding to transfer visualization via WebSockets. One of the most recent tools that allows for a pure web based remote visualization is Vitrall [65]. It provides remote and collaborative access to visualization, mainly surface visualization rendered with OpenSceneGraph [243]. Vitrall was also already being used with a remote access to a single stereoscopic content.

3.3.3.3 Mixed rendering on client and server

Some projects aim to combine the best properties of client and server side rendering, by sending as less data as possible and/or by processing as much of the performance intensive tasks at the

server to keep the requirements on the client low. However, most of these systems currently require special software deployments.
One method to do this is to reduce the data size by subsampling it for the specific client system [244]. Engel et al. [245] shows for instance a shared visualization system that emphasizes this aspect by reducing the data on the server side. It cuts and converts the initial DICOM files into VRML textures, which are then renderable with much lower computer power on the client.
The usage of surface visualization based on volume visualization is also a type of mixed rendering, where the computation intensive part, the procession of the volume data to geometrical objects (see Section 3.1.2) can be done on the server. The resulting objects can be stored in an intermediate file format (e.g. X3D) to be transferred and rendered on the client. The surface visualization is usually based on only about 10% of the volume data and, thus, requires less transfer time and performance than the volume visualization [121].
Turner et al. [246] describe a client-server architecture, which provides an instantaneous access to a remote volume rendering, but in background also transfers the dataset to the client. This setup allows for an instantaneous useable server-based visualization and a fast client side visualization, after the base data is available. Another example solution is to utilize the compute performance of any client and server by dividing the workload as for instance described by Grimstead et al. [68]. They describe a parallel rendering system that renders on the client, if enough graphics power exists, but utilizes remote server side rendering, otherwise.

3.3.3.4 Utilization of compute infrastructures

Many projects have been using grid- and cloud computing infrastructures for various tasks, e.g. with the focus on compute and/or data management [195, 247]. Both infrastructure types provide the user with the capability of using remote storage, CPUs, Graphics Processing Units (GPUs), and General-purpose Computing on a Graphics Processing Unit (GPGPU).
Projects in the medical environment also utilized such infrastructures for various image related use cases. Grid computing can for instance be used for server discovery [248], image retrieval [38, 248], and analysis [39]. It was also demonstrated that these infrastructures can be used for distributed visualization and remote rendering [41, 249, 250]. The main problem of such infrastructures is to simplify the initialization of and access to the visualization session. Such resource reservation systems have often been solved via web-based systems. The subsequent interactive visualization usage, however, is mostly provided by special applications, e.g. VNC-like applications [251]. But in the case of Paraview [252, 253], it can be seen that this paradigm is changing. Paraview is a powerful visualization system that allows to utilize remote visualization resources with local native client applications. As described earlier, Paraview was recently also being extended by a client that is usable on pure web browsers.

4 Evaluation of data transmission techniques using web browsers

The related work projects and applications described in the previous sections utilize different techniques to transfer data from and to pure web browsers. This chapter describes the evaluation of these techniques in order to find a single technique selection for the proof of concept conduction. Section 4.1 is about the transfer of images from the server to the client and Section 4.2 is about the transfer of control commands as single events from the clients to the server.

4.1 Visualization transfer

The evaluation of the visualization transfer techniques is based on rapid prototypes, which are described in Section 2.2.1. The results are summarized in Table 4.1 with an overall grading in the right-most column. Two of the techniques resulted in a similar grading and where further evaluation in Section 4.1.2.

4.1.1 Prototype observations and evaluation

Prototype 1: HTML5 video steaming. HTML 5 video is an increasingly often used method to stream video and audio content to pure web browsers and will likely be the future direction of streaming video to web browsers. However, any test to load a real-time generated HTML 5 video stream on a pure web browser resulted in a delayed client side presentation of more than one second. The server side test system (VLC) was extensively adjusted by modifying most parameters in order to optimize the encoding and transfer, which, however, did not result in a faster presentation. It was therefore assumed that the delay is caused by a buffering delay added by the web browser.
This observation was later verified by the work from Metzger et al., who state that "the time scale on which streaming applications buffer content lies in the range of seconds" [254]. They conducted detailed tests about HTML 5 video streaming and show that the delay of a client side video playback increases after network delays occurred. This behavior can be explained by the HTML 5 video specification [85], which recommends a video playback algorithm for web browsers that only starts the playback "when it can be ensured that the video can be played without interruption" [254]. The HTML 5 specification currently has no option to control this behavior. This means that a short transmission delay that might even occur on a very fast connection may

already result in a buffer increase. Such a buffering behavior is very useful for the continuous presentation of video content without the need of a user intervention. But it follows that it can not be assumed that a web browser continuously presents video frames with low delay and, thus, it currently prevents the usage for a real-time interactive visualization system.

Table 4.1. Comparison of visualization transfer techniques (rows) by the criteria defined in Table 2.1 (columns). Each cell shows the summarized main arguments and a grading of 1 (bad support) to 3 (good support), and "/" (excluded), which is summed up to a total score in the left-most column.

Technique	**Criteria 1:** Added software	**Criteria 2:** Real-time transfer	**Criteria 3:** Data size	**Criteria 4:** Browser support	**Sum**
Streaming HTML 5 video	3 No added software necessary	/ Currently not useable for real-time streaming due to a long delay between sending and playback	3 Useable with high-end video-codecs	2 Multiple codecs need to be used to reach all web browser	/
Pulling single images	3 No added software necessary	1 Requesting images requires more communication than pushing, which is theoretically slower	2 Quality and resolution can be switched between frames, high image quality can be transferred	2 Works with JavaScript enabled browsers	8
Pulling / Pushing Base64 images	3 No added software necessary	1 Pushing and pulling of single or multiple concatenated images	1 Image is ca. 33% larger than a binary image, resulting in higher data rates and a lower frame rate	1 Only some JavaScript enabled web browser support large Base64-encoded images	5
Pushing image streams (motion JPEG)	3 No added software necessary	3 Server push image after as soon as it is created	2 Quality/ resolution changes between frames, high image quality, theoretically more efficient than the other non-video streaming techniques	1 No JavaScript necessary, but it does not work with all current generation browsers (e.g. not with Internet Explorer and Opera)	9

Another disadvantage of HTML 5 streaming is the codec support of the web browsers. No single supported video codec exists that is supported on any browser [221]. Thus, it would be necessary to provide specific streaming methods for different web browsers.

Prototype 2: Pulling JPEG (pJPEG). Pulling single images from the client side is a very straightforward method to transfer the visualization, because it is the default technique to load images of web sites. It results in a fluent frame rate with a subjective low delay on the client side. On the contrary, it requires a communication overhead between the client and the server, because any new image needs to be request separately via a new connection.

Prototype 3: Pushing Base64 images. The continuous pushing of Base64-encoded images resulted in a subjective fluent playback. Testing it on a single computer did not result in a noticeable longer delay than the pJPEG image approach. But Base64 image files are on average 33% larger than its binary version [255], which results theoretically in either a lower frame rate or a higher network load factor. Transferring Base64-encoded images seems therefore less performant than the continuous pulling of single JPEG files.

Prototype 4: Pushing motion JPEG (mJPEG). Pushing mJPEG provided a fluent sequence of images on the client with a subjective very fluent playback. In contrast to the other single image transfer techniques, it utilizes a single persistent connection to transfer all images binary. Each new image is just pushed to the client via the opened connection. Therefore, it does not have the additional communication overhead of prototype 2 and 3 of continuously establishing new connections and/or larger file sizes.

4.1.2 Motion JPEG vs. pulling JPEG

Whereas pulling JPEG (pJPEG) requires a separate establishment of a network connection for each image, motion JPEG (mJPEG) only requires a single connection establishment in the beginning of a visualization session. Due to this architectural difference, each technique's performance capabilities can be easily compared analytically. In a setup with

$$t_{init} = \text{time (in milliseconds) required to initialize a new HTTP connection}$$

and the time (in milliseconds) to transfer an image with a file size s (in MB) on a network with a bandwidth b (in Mbps) of

$$t_{img}(s, b) = \frac{s \ 1000}{\frac{b}{8}} \quad ,$$

mJPEG would have a theoretical maximum frame rate of

$$fps_{mJPEG} = \frac{1000}{t_{img}(s,b)}$$

and pJPEG a frame rate of

$$fps_{pJPEG} = \frac{1000}{t_{img}(s,b) + t_{init}} \quad .$$

The comparison of control event transfer techniques in Section 4.2 provides a good reference value for t_{init} with 20 ms (the overhead of REST compared to WebSockets). This is true, because both of REST and pJPEG utilize HTTP GET to load data and both of WebSockets and mJPEG utilize a persistent connection for any transfer. Thus, with the additional $t_{init} = 20ms$, pJPEG always results in a lower frame rate than mJPEG. Figure 4.1 illustrates the frame rates of pJPEG (red) and mJPEG (blue) based on these equations with a file size between 1 and 100 KByte on a 3 Mbps (dashed line) and 10 Mbps (solid line) connection. It shows that the difference between mJPEG and pJPEG gets less important with larger files, because of the steady decrease of the t_{init}/t_{img} ratio. But it also shows that there is a large performance difference when using low image file sizes.

4.1.3 Image format choice

Transferring single images to pure web browsers requires the web browsers to support the image format. Currently, the most widely supported image formats on web browsers are PNG [256] and the JPEG file interchange format (JIF) [257]. JPEG is a lossy image standard (which also exists

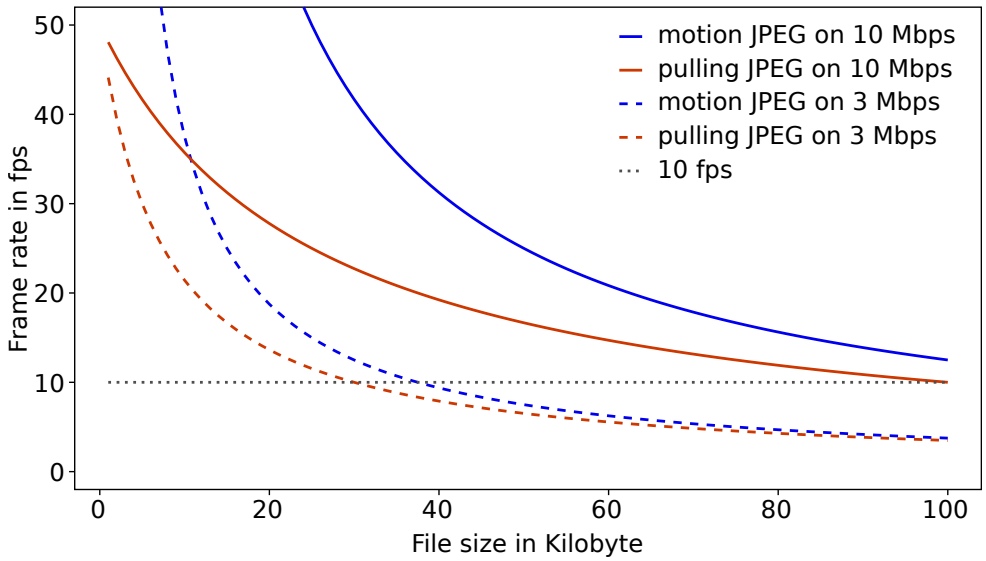


Figure 4.1. Frame rate comparison of mJPEG and pJPEG.

in a lossless compression version), which means that an image differs on the pixel level after a single compression and decompression cycle, even with the highest possible image quality of 100. In return, JPEG provides a good overall image quality with a compression ratio of more than 10:1 and is optimized for smooth transitions, as they usually appear in medical visualizations. In contrast, PNG is a lossless image compression standard, optimized for hard transitions as e.g. line drawings [258]. A more efficient image compression format is JPEG2000 [259], which uses wavelet compression in contrast to the discrete cosine transformation of JPEG, resulting in a better compression ratio [260]. Unfortunately there is almost no web browser available that supports JPEG2000 by default. The same is valid for other newer formats as WebP [261] or H.264 intra frame compression [262].

4.2 Event transfer

There are fewer techniques that allow a data transfer from the client to the server than vice versa. Typical techniques are HTTP GET and POST [89], which both are HTTP requests that can be utilized via a separate short lasting connection for each data transfer (e.g. via a REST-style interface). Whereas, browser plugins were developed to circumvent this issue by providing long lasting connections, only WebSockets [263], a rather new technology, allows for a similar transfer without added software.

Both, REST and WebSockets, where tested as described in Section 2.2. The results of their comparison are presented in Table 4.2. It shows that each technique requires an almost equal transfer time on the different tested networks, with an exception on the very slow 1 Mbps connection. Thus, it seems that the available bandwidth is of minor importance to transfer control events of very small size. But, whereas the REST-style interface requires on average 46 ms for each control event, WebSockets only requires about 25 ms. Thus, WebSockets are on average 21 ms faster, which results in a faster average event transfer rate of 35 events per second (eps), which are 16 events more compared to the 19 eps of the REST-style interface.

Table 4.2. Test results of using REST and WebSockets to transfer control commands on different network conditions. Presented is the average time to transfer a single event (time in ms) and the average event rate (rate in events per second).

	1 Mbps		3 Mbps		90 Mbps		Average	
	time	rate	time	rate	time	rate	time	rate
REST	51.3	18	44.2	20	43.0	21	46.2	19
WebSockets	34.9	26	21.0	43	20.5	38	25.5	35
Difference	16.4	8	23.2	23	22.5	17	20.7	16

Table 4.3. Comparison of event transfer techniques (rows) by the criteria (columns) defined in Table 2.1. Each cell shows the summarized arguments and a grading of 1 (bad support) to 3 (good support), and "/" (excluded), which is summed up to a total score in the left-most column.

Technique	**Criteria 1:** Added software	**Criteria 2:** Real-time transfer	**Criteria 3:** Data size	**Criteria 4:** Browser support	**Sum**
REST-style	3 No added software necessary	2 Event transfer rate is low	1 Large overhead due to continuous connection establishment	3 Supported on all web browser	9
Web-Sockets	3 No added software necessary	3 Event transfer rate is high	3 Low overhead	1 Supported on many browsers, which will likely improve	10

4.3 Expert discussion

The possible techniques that are evaluated in this chapter were also discussed with two collaborating working groups, who had similar aims of high-performing remote visualization and/or a simpler access to it.

The first working group[1] had extensive knowledge with remote visualization [188, 195, 242, 264]. They told us about the problems and solutions they already experienced and linked us to new projects and thoughts. A key point of the discussion was that they already developed a system for a pure web-based remote access that utilized pJPEG [242]. This system, however, resulted in a slow visualization update on the client side. Therefore, they also suggested the usage of mJPEG to provide a faster transfer.

The other collaborator[2] was already using mJPEG to stream the video of webcams in the TPM (see Section 3.3.3.2). This system was used to stream video in various projects. The most interesting approach, however was the usage in a hospital, which is the same environment as CoWebViz was intended for.

4.4 Discussion

HTML 5 video streaming would be the most desired solution, due to its best performance. However, it is currently be excluded, because it can not be assured that the visualization is instantaneously transferred and presented on the web browser client.

[1]Personal communication with Joseph Insley, Eric Olson and Tom Uram, Futures Laboratory, Argonne National Laboratory, Argonne, IL, USA.

[2]Personal communication with Nestor J. Zaluzec, Electron Microscopy Center, Argonne National Laboratory, Argonne, IL, USA.

The two best graded techniques are pulling JPEG (pJPEG) and pushing motion JPEG (mJPEG). Both provide a feasible approach to transfer visualization to a web browser. The main disadvantage of the first is that it requires a separate request/connection for each image, but on the contrary is capable of providing visualization on the widest range of web browsers. mJPEG's advantage is the lower transfer overhead for a fast image transfer, but is not supported on all web browsers. Since mJPEG seamed to be the most efficient of the tested techniques, it was chosen to be used for the proof of concept prototype. This decision was further promoted by the personal discussion with collaborating working groups (see previous Section 4.3).
The selection of a control command transfer technique was straight forward, since this evaluation was done after the prototype conduction already started. The first implementation only supported the REST-style interface, and WebSockets were additionally implemented after this evaluation. The tests show that both techniques are feasible in terms of transfer rate. Due to the possibly higher event rate of WebSockets, it is more preferable than the REST-style interface.

5 A system design for remote collaborative web-based visualization

The previous section describes the evaluation of possible techniques to transfer visualization and command control events to and from web browsers, which resulted in a preferred technique selection for each type. This technique selection was the foundation of a more sophisticated prototype developed for the proof of concept conduction, which is called *Co*llaborative *Web*-based *Vi*sual*iz*ation (CoWebViz). It was developed to fulfill all research requirements defined in Section 1.4 by utilizing a pure web browser as client system for collaborations with remote, interactive, and stereoscopic volume visualization. The primary goal of this prototype was to ease the deployment and conduction of the virtual anatomy class (see Section 2.3.2.2). The prototype usage experience in the class is described in Chapter 6 and detailed performance tests in Chapter 7.

This chapter describes the system design and necessary functionality of the prototype, which was partially already published in [51, 265, 266]. The development and test environment is described in Section 2.3.2.1. The final prototype is CoWebViz 0.2 and includes the entire functionality that is described in this chapter. CoWebViz 0.1 is a subset of CoWebViz 0.2, which was already capable to access monoscopic and two-view stereoscopic visualization from remote, but did not have the current visualization and event transfer optimizations nor the parallel architecture that are described in Section 5.2 and 5.3. The image transfer of CoWebViz 0.1 was done via a fixed quality and resolution as described in configuration A of Section 7.1.1.

5.1 System design

The anatomy class relied on MedVolViz to create interactive medical volume visualization (see Section 2.3.2.1). Thus, CoWebViz required to meet all requirements necessary to be used with MedVolViz in the first place. But on the same time it also required to be generically usable with other applications to meet future changes of MedVolViz or the class. The system was therefore designed to incorporate visualization of an external base application (see Section 5.1.1) and to provide a server side HTTP interface (see Section 5.1.3) that is easily accessible by a web browser. This basic principle makes CoWebViz to a web-based front-end for potentially every visualization application.

CoWebViz is developed as multi-threaded application in C++ with an architecture that shall provide the greatest possible freedom of use, by providing interchangeable modules. The appli-

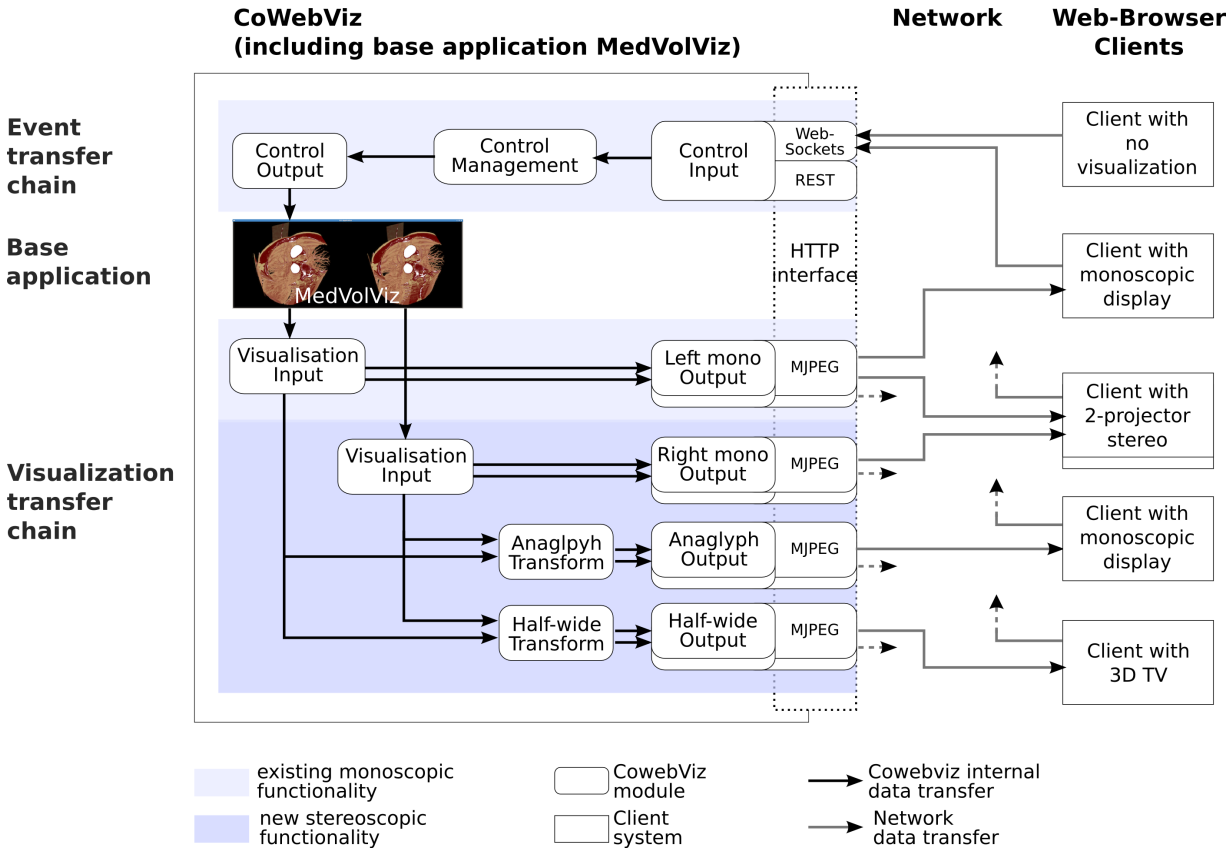


Figure 5.1. This data flow diagram shows CoWebViz's system architecture on the left side and its web browser client instances on the right. The top part shows the event transfer and the bottom the visualization transfer chains. In contrast to the light blue part, the dark blue part of the architecture is new, compared to the related work.

cation fulfills two basic tasks, the visualization transfer from the server to the client and, vice versa, the transfer of control commands as single events from the client to the server. Both tasks are implemented as following separate process chains [265]:

Visualization transfer chain: This chain directs from the the server to the client. It is attached to a base visualization application, from where it continuously takes rendered visualization images, forwards them to an HTTP interface that provides functionality to send the images to any currently connected client. After receiving a new image the client displays it instantaneously.

Event transfer chain: This chain directs from the client to the server. It captures mouse and keyboard events at the client, transfers them to the server side HTTP interface, where they are being processed and transferred to the base application.

Each chain consists of several modules that fulfill different subtasks, which are illustrated for the event and visualization transfer chain on the top and bottom part of the Figure 5.1, respectively. Each module is represented by a virtual class, which is an implementation blueprint

for various specific implementations that provide the same class methods. Thus, modules are exchangeable with other implementations in order to provide an architecture with customizable and interchangeable functionality. Each module runs in a separate thread, which are connected via shared memory.
Due to its objective, the system requires modules for data input, data output and intermediate processing, which are called input, output, and process channel, respectively. An input channel takes data from a system-external source, processes it, and puts it to a shared memory. An output channel takes data from a shared memory and provides the data via an interface to an external application. Process channels take data from a shared memory and put the processed data back to another shared memory. The consumer/producer pattern is used for the shared memory implementations.
The modular architecture enables the parallel execution of multiple modules of the same and different types. The parallel execution of a single module or a whole chain of modules is for instance necessary to handle multi-view stereoscopic formats (e.g. one for each, the left and right eye view). Multiple modules of the same type are for instance necessary to provide two parallel visualization output channels, e.g. to simultaneously stream images to a web browser and to record them to a file.
The whole network communication between the application and the web browsers is realized by HTTP interfaces (REST, multipart/x-mixed-replace, WebSockets) on a single integrated HTTP server, which only requires a single network port.
All currently implemented modules (introduced in Figure 5.1) are described in the following subsections.

5.1.1 Integration of visualization applications

Visualization input channels are designated to take visualization from external sources into CoWebViz, where it is converted into an internal uncompressed RGB image format.

The *InputX11* module provides functionality to screen scrape visualization from a single base visualization application that runs on X11. It connects to the specified application and automatically recognizes its GUI window size, which is then screen scraped from the desktop metaphor via a shared memory with the X11 server. Thus, any X11 application can be used as a CoWebViz base application. Many scientific visualization applications just provide a visualization window with no GUI (e.g. MedVolViz). But other applications provide extensive GUIs, which can be handled in two ways, either by screen scraping the whole application window or only the part of the window that contains the visualization, as illustrated in Figure 5.2. The latter case, would prevent a client user to directly access the GUI-elements and would therefore require the addition of a specific client interface that makes the hidden functionality available (e.g. similar to the client GUI shown in Section 5.6). A sophisticated example is demonstrated by Lamberti et al. [207]. The third method is to connect programmatically to the base application, which is only hinted in

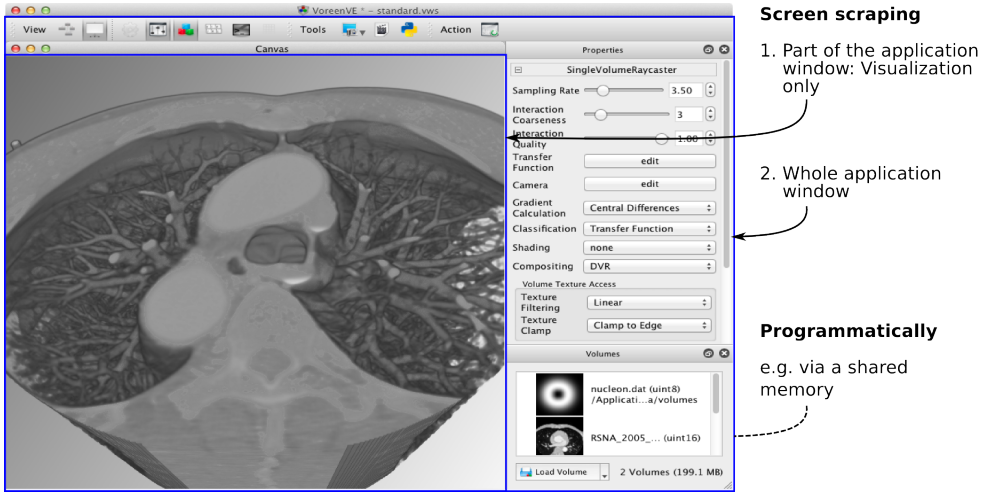


Figure 5.2. Possibilities to integrate existing visualization applications (e.g. Voreen [267]) into CoWebViz via screen scraping or a direct programmatical connection.

Figure 5.2, because it is enabled by CoWebViz but requires specific additions. Two instances of this module can be used to integrate stereoscopic content in the form of a side-by-side or half-wide side-by-side visualization, with a left and a right view displayed in full or halved resolution next to each other. Each instance is then used to separately screen scrape one of the virtually halved application windows. Side-by-side is the default stereoscopic content provided by MedVolViz. Other types that present full resolution images next to each other, e.g. top-to-bottom or two views in separated windows, could be integrated with few modifications.

The *InputFfmpeg* module provides functionality to integrate visualization from any FFMPEG source [106], which among others includes most video file formats and video cameras.

Event output channels are designated to send or write control events from CoWebViz to an external destination.

The *ControlOutputX11* module provides functionality to send control command events to the connected base application on an X11-Server. It gets CoWebViz internal events from a shared memory, converts each to an X11 event and sends it to the connected application. It is the counterpart to the InputX11 module.

The *ControlOutputScriptingFile* module provides functionality to write events to a file, which is necessary to record a whole visualization session for later playback (see Section 5.6). It gets events from a shared memory and stores each in the format described in Listing 5.1.

5.1.2 Internal procession

Visualization process channels are designated to take one or multiple images from one or multiple shared memory instances, which are processed and put to another shared memory.

Listing 5.1. Format of the event recording file, including the time to the most recent event, the event type (k or m for key or mouse event), the event number (a CoWebViz internal event definition), and the x and y coordinates of a mouse event.

```
<DelayInMs> <EventType> <EventNumber> <MouseXcoordinate> <MouseYcoordinate>\n
...
```

Listing 5.2. Specification of a motion JPEG (mJPEG) stream based on a multipart message [92] as it is used to stream to web browsers [106, 268]. The boundary name is a string and the image size a positive whole number. A mJPEG stream requires the MIME type to be "image/jpeg".

```
HTTP/1.0 200 OK\r\n
Content-Type: multipart/x-mixed-replace;boundary=<boundary-name>\r\n
\r\n
--<boundary-name>\r\n
Content-Type: <MIME-type>\r\n
Content-Length: <image-size-in-bytes>\r\n
\r\n
<image-data>\r\n
--<boundary-name>\r\n
....
```

Such channels were implemented to create anaglyph and half-wide side-by-side stereoscopic content of two separated stereoscopic views in the *ImageFilterAnaglyph* and *ImageFilterHalfWideSbS* modules, respectively (see Section 5.5).

Event process channels are designated to take control events from a shared memory, which are processed and put to another shared memory.

The *ControlManagement* module provides functionality to handle every event that comes into the system from multiple participants before forwarding it to the base application. It keeps track of the most recent events and filters them based on their sources (see Section 5.4 for more).

5.1.3 Interfaces to access and control the visualization

Visualization output channels are designated to take images from an internal shared memory, to forward it to an external destination.

Each of the following output channels require an *output format* module, which takes the internal RGB image and transforms it to the required output image format. Such are currently implemented for JPEG (*OutputJpeg*) and mJPEG (*OutputMjpeg*).

The *VisualizationOutputHttp* module provides extensive functionality to distribute visualization to multiple participants via a MIME multipart message shown in Listing 5.2. The module is listening for new client connections and starts a new *visualization destination servant* as soon

as a connection request is stated. Each destination servant handles the data transfer of a single view for a single participant. It is running in its own thread and manages the connection and any data transformation using the output format. At first the destination servant sends a simple HTTP header (Listing 5.2, line 1) to initiate the HTTP connection. Afterwards, a specific header is sent for the output format, which is OutputMJPEG in this case (Listing 5.2, line 2-3). Any consecutively sent image is transferred as shown in line 4-7. The web browser loads the mJPEG stream via a given URL, as for instance `http://<URL>:8080/left`. The web browser receives the simple HTTP header, recognizes the mJPEG multipart message and consecutively displays the following images. The mJPEG stream can be accessed by web browsers, but also by network-enabled video players (e.g. VLC [87]) or other applications. In order to provide the user with a control mechanism, a web page is necessary as described further down.

The *OutputFile* module provides functionality to write each processed image to a single or multiple files, which is necessary to get a session recording (e.g. in mJPEG) or single visualization snapshots.

Event input channels are designated to take control command events (e.g. mouse and keyboard events) into the system.

Any network data transfer to and from the system is managed by an integrated HTTP server, which provides either services for a web page delivery or data connections. Each service is accessible via a specific URL, as e.g. `http://<URL>:8080/ctrl` for REST-style control events.

The *HttpServiceControlInput* and *HttpServiceWS* modules were developed to integrate control events via a REST-style and WebSockets interface, respectively. Both are continuously listening for new control events, which are transferred as described in Listing 5.3 and 5.4. Each event is converted into an internal event format and put on a shared memory. These modules are also listening for other defined data transfers, which are necessary to configure the system.

The *visualization control service* provides the web pages that provide the interactive functionality to a web browser. Different web pages are dynamically created and delivered for specific functionalities. They include a superimposed active layer on top of the mJPEG stream with the exact same size of the stream that captures the events via JavaScript. The visualization is included into the web page via the specific mJPEG stream URL. The events are captured and sent to the REST or WebSockets HTTP interface (see previous paragraph). The required URLs (e.g. of the mJPEG stream) are already pre-defined in the delivered web page. The user can instantaneously start modifying the visualization using the base application's control functionality (see Figure 6.3a and 6.3b).

The *control service* is basically the same as the visualization control service, but without showing the mJPEG stream (see Figure 6.3c).

The *ControlInputScriptingFile* module provides functionality to load events from a recorded text file (see Listing 5.1), which is used to playback a recorded visualization session (see Section 5.6 for more).

The *file service* provides basic functionality to transfer files to the client, as necessary to access JavaScript libraries or Cascading Style Sheets (CSS) files.

Listing 5.3. Transfer format of a single WebSockets event, including the event number (a CoWebViz internal event definition) and the x and y coordinates of a mouse event.

```
1 <EventNumber> <MouseXcoordinate> <MouseYcoordinate>
```

Listing 5.4. Transfer format of a single REST event, including a client user ID, the event number (a CoWebViz internal event definition), and the x and y coordinates of a mouse event. In addition to the WebSocket message, REST requires a user ID, because of its connection-less'.

```
1 http://<URL>?<EventType>=<UserID>,<EventNumber>,<MouseXcoordinate>,
      <MouseYcoordinate>
```

5.2 Visualization transfer optimization

Following methods were utilized to provide each participant with a fluent visualization on its specific network conditions.

5.2.1 Bandwidth optimizations

JPEG was chosen as image format, because it still is the one format with the best compression ratio supported on all web browsers (see Section 4.1.3). Its streaming via mJPEG was firstly done to access web camera videos from the web browsers. In contrast to video, which requires the constant sending of new images, an interactive visualization only changes after a control command was given. Thus, the most significant method to reduce the network throughput is to send only the images that have been changed. CoWebViz utilizes a simple algorithm that compares consecutive images on the sub-pixel level, which starts at the image center and proceeds to the beginning and afterwards to the end. The image center is a good starting point for the comparison, because a volume visualization is typically centered, which means that most modifications while have an effect on the image center. The algorithm, which is located in the visualization input channel, stops with the first occurrence of an unequal sub-pixel and marks the image as modified or not modified. Since this image might still be relevant for some specific output channels (e.g. to record a session movie), it is in the responsibility of the output channel to handle the image accordingly (e.g. send or not send). [51]

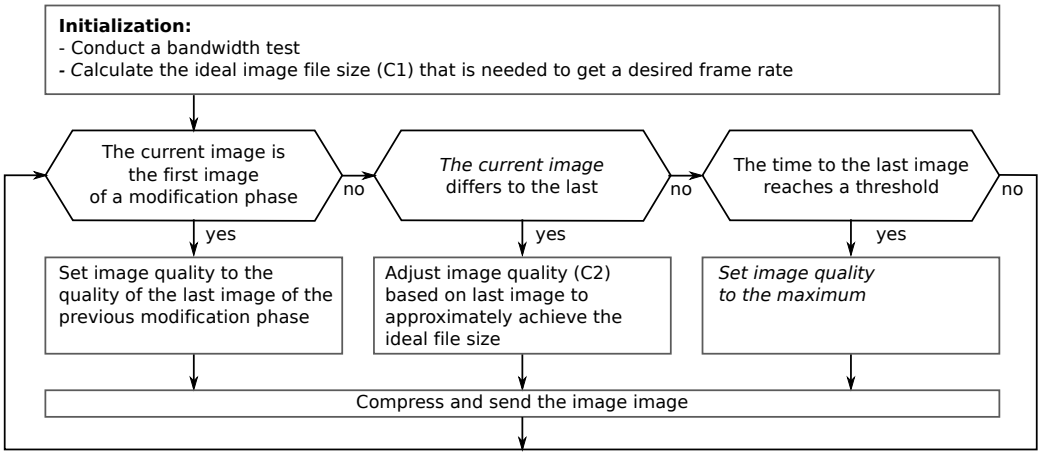


Figure 5.3. This flow chart describes the adjustment of the quality to the specific bandwidth conditions. The calculation of C1 and C2 is illustrated in Figure 5.4a and 5.4b, respectively.

Using mJPEG for the continuous transfer of single images results in an image presentation issue that we experienced on some web browsers while sending only modified images, which would be of no concern if using mJPEG for its original designation (video transfer). Of two sent images, the first is only displayed if the second is transferred to the web browser. A simple work-around would be to send each image twice, which, however, would result in a large overhead of twice the necessary data transfer rate. The solution that is implemented in the system is to send the same image a second time, after no modification occurred for a specified time (e.g. 500 ms).
The usage of this system for a visualization session might lead to a long time without any modifications. We never experienced a connection loss during long class visualization sessions, which even included periods of up to 30 min with no modification.

5.2.2 Quality and performance optimization

While providing real-time remote interactive visualization, not only the image quality is important, but also the fluidity in which consecutive images are being presented. The perceived quality that a user experiences therefore is a combination of the visualization's image quality and the fluidity of its presentation.
A straightforward method to increase the performance is the reduction of any procession time on the server side as for instance done by using the improved JPEG compression library libjpeg-turbo [104]. However, most time of a remote visualization process chain is required for the image transfer over the network. While sending JPEG images, the image quality is mainly based on the JPEG quality[1] and the image resolution. Rendering and compressing a specific image by applying a specific JPEG quality and resolution results in a specific image file size. Transferring

[1] Another variable is the JPEG quantization matrix, which can also be modified to create smaller files [269]. However, the standard matrix is recommended for best quality.

multiple images over a specific network condition results in a frame rate, which is a common metric for the image presentation fluidity. The network condition, however, is mostly pre-defined by external circumstances (in the case of an ad-hoc usability). An automatic adaption of the image file size towards the specific network condition of each single accessing client is therefore necessary to provide the best possible perceived quality. Thus, the frame rate and the image quality needs to be balanced with the current bandwidth condition. The parallel architecture enables such a specific provision for every connecting client.

CoWebViz's automatic quality algorithm, implemented in the OutputMjpeg/VisualizationOutputHttp modules, is illustrated as flowchart in Figure 5.3. It basically adjusts the JPEG quality and image resolution of each consecutive modified images $i : 1 \dots n$ to meet an initially calculated ideal image file size (ifs) that is sufficient to get a desired frame rate ($dfps$). The algorithm utilizes an abstract quality value ϕ on a scale from 0 to 100, which is defined as

$$f : \phi \mapsto \{JPEG\ quality, image\ resolution\} \quad .$$

The image file size is increased or decreased by increasing or decreasing ϕ in the following way [51]:

1. The initial step of the algorithm is the bandwidth measurement of bw in Mbps. bw is then used to calculate the ideal file size ifs as

   $$f : \{bw, dfps, sv\} \mapsto ifs \quad ,$$

   where ifs is the maximal file size that enables the system to achieve a desired frame rate $dfps$ on the specific network bw. ifs is basically calculated by dividing the available bandwidth by the desired frame rate (see Figure 5.4a). sv is the amount of stereoscopic or other visualization streams the participant has opened. sv is important, since multiple open visualization streams result in a multiple of a single stream's throughput. The single stream throughput therefore needs to be reduced.
2. A modified image i that is to be sent needs to be compressed using a ϕ_i, which results in a compressed image with the file size cfs_i (current image file size). This ϕ_i is set to a default value, if the modified image is the first image of the first visualization modification phase of a visualization session. If the modified image is the first image of a follow-up modification phase, ϕ_i is set to ϕ_{i-1} (the last non-maximized quality setting of the previous modification phase).
3. Every following image is compressed with a ϕ_i that is calculated based on the a cfs_{i-1}/ifs ratio (see Figure 5.4b). This difference ratio, is a comparison of the most recently sent image's current image file size with the ideal image file size:

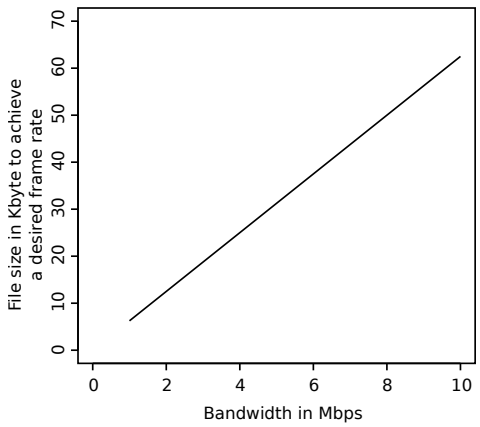
   $$f : \{ifs, cfs_{i-1}\} \mapsto diff_i \quad .$$

The file size of the last instead of the current image has to be taken for this calculation, because the current file size is only available after compression. $diff$ is a positive or negative value that is used to increase or decrease ϕ as

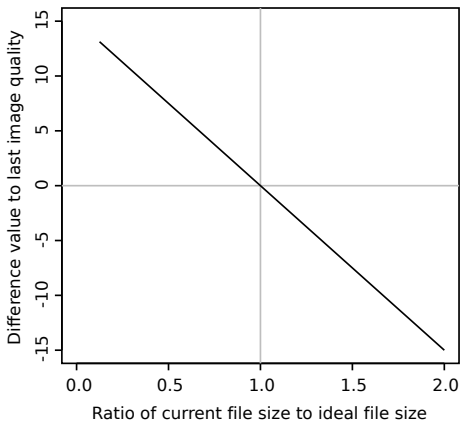
$$f : \{diff_i\} \mapsto \phi_i \; with \begin{cases} \text{increase } \phi_{i-1} & diff_i < 1 \\ \phi_{i-1} & diff_i = 1 \\ \text{decrease } \phi_{i-1} & diff_i > 1 \end{cases}$$

4. If no modification of the visualization occurs for a specified time (start of a viewing phase), ϕ is set to its maximum value. Otherwise the loop repeats with step 2.

The CoWebViz client allows to select between this algorithm and two other methods to specify the image compression handling as shown in Figure 5.5. The automatic quality adjustment further allows to slightly balance between quality and fluidity by manually adjustment (see Figure 5.5.A). This is implemented by increasing or decreasing the ideal file size dynamically after request. The second method is a *semi automatic adjustment* that provides a single slider at the client side to specify a required quality by modifying ϕ directly (see Figure 5.5.B). The third method is a *complete manual setting* that allows to specify every value separately (see Figure 5.5.C). This latter method allows for the most control, but is only useable for development.



(a) The algorithm's initial calculation to achieve a desired frame rate during the usage: The ideal image file size (y-axis, in Kbps) is calculated based on the users bandwidth connection (x-axis). [51]



(b) The algorithm's ongoing quality calculation: The comparison of the current and the ideal file size (x-axis) maps to a quality difference value (y-axis, greater/lower 1), which results in a quality value change for the current image.

Figure 5.4. The graph in Figure 5.4a illustrates the approach of calculating an ideal file size based on the bandwidth. During a modification phase, this visualization's ideal file size is approximated by modifying the ϕ based on the current and ideal file size. As shown in Figure 5.4b, a current file size that is larger than the ideal file size (ratio > 1) results in a quality reduction.

Another minor method that should increase the frame rate is to limit the resolution of the images to the maximum resolution that the client can display. By default, the system sends the same resolution to the client that the base application provides. But collaborations might also include devices with smaller screens. Thus, the client's maximum resolution (the dimensions of the web browser window) is exchanged to the server directly after the connection was established or the browser window was resized, which is then used as maximum resolution.

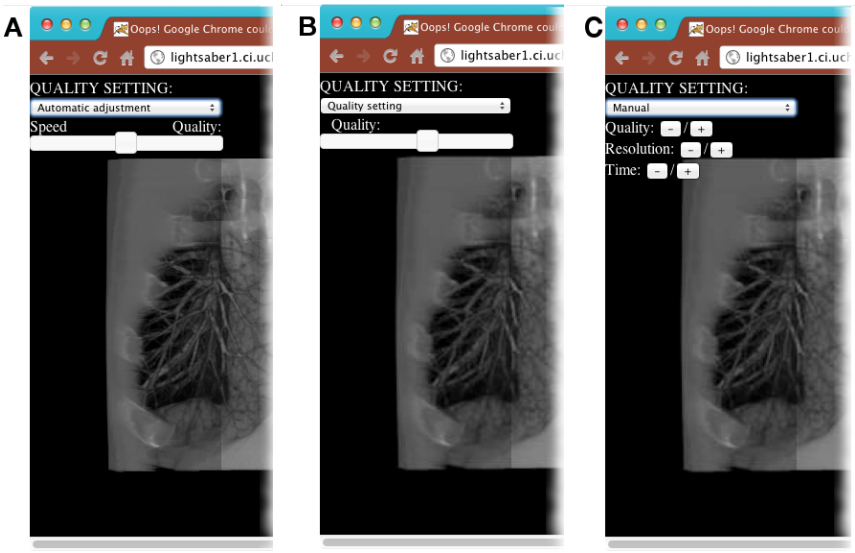


Figure 5.5. Illustration of all quality handling methods supported by CoWebViz: A) a complete automatic quality setting that allows for an optional manual speed adjustment, B) a manual quality setting of a single value that is used for any image without considering the frame rate, and C) a complete manual setting of all parameters.

5.2.3 Visualization smoothness optimization

The modification of visualization using CoWebViz resulted in an observable lag, if it was modified at a high pace. This lag could potentially be a very specific issue of using mJPEG on TCP networks (defined by the HTTP basis) and could correlate with the non-controllable process of decompressing the JPEG images on the client in the following way: If sending multiple images that are completely transferred (because of TCP), each image is decompressed and displayed with a speed that is too slow to be presented on time. This could finally result in a stepwise decompression of all images and, thus, a delayed presentation.

CoWebViz's approach to reduce this observed lag is to set a specific delay after an image has been sent, which the system has to wait for this participant and visualization view before it can sent the next image. It is implemented in the OutputMjpeg/VisualizationOutputHttp module and is

based on the bandwidth test and the file size of the image that is to be sent (see Section 5.2.2):

$$f : \{bw, cfs_i\} \mapsto delay_i \quad .$$

The delay basically is the time that is theoretically necessary to completely transfer the image on the specific bandwidth condition from the server to the client. [51]

5.3 Event transfer

A visualization can only be re-rendered by the base application, after it received according control commands from the user. The event transfer therefore is an equally important network related aspect of interactive remote visualization as the visualization transfer. The specific implementation is described in Section 5.1.3.
The initial implementation of the event transfer was the REST interface. An example of this interface as implemented by CoWebViz is `http://<URL>:8080/ctrl?m=1,1,40,50`, which has 5 specified variables that are the event type (m), the user ID (1), the left button down event (1), and the x- and y position of the mouse event (40,50). The class experience (see Section 6.1.2), however, showed that this interface led to a slow and lagging visualization. The only other newly available technique with a potentially faster transfer was WebSockets, which was implemented as a consequence of this lag. It was later proofed by the analysis in Section 4.2 that it actually is faster than the REST interface.

5.4 Collaborative multi-user access control

If multiple participants need to have access to a single visualization applications, the access to its control mechanism needs to be managed in order to avoid confusions on the programmatical level (illegal event states) and the user level (notification of who is modifying). Other collaborative systems utilized various different types of access control mechanisms. A most simple one is to provide only one participant with the right to control, where all others are only passive viewers (e.g. supported by [95, 270]). Another method is to allow every participant to get dedicated control, but only after an explicit control request (e.g. supported by [271]). Subjectively, either provides too few flexibility or management overhead.
For collaborative usage, CoWebViz requires a parallel running system for personal communication between all participants (e.g. audio and/or video conferencing, or chat systems). Since all users can communicate with each other, it seems to be natural to just talk about who is using the visualization next. Leigh et al. [160] suggested to minimize the application management for such collaborative cases.
The specific implementation of CoWebViz is realized in the ControlManagement module. This control mechanism lets every participant freely control the visualization instantaneously as long as wanted, if no other participants is modifying in the beginning. But as soon as the user stops to

modify the visualization for a specified amount of time (e.g. 3 sec), every other participant is able to gain control as before. In order to let everybody know the current usage state, one of three message is displayed at every client saying if somebody modifies the visualization as illustrated in Figure 5.6.

5.5 Stereoscopic visualization support

Another collaborative aspect is the provision of distinct stereoscopic content types for different display devices. Thus, multiple monoscopic and stereoscopic visualization types were implemented in CoWebViz, which allows each remote user to choose a specific type that is best suited for its local environment. Supported are standard monitors, larger scale projection based stereoscopic setups and consumer grade 3DTVs.
The specific adaptions of the system design (see Section 5.1 and Figure 5.1) to support each stereoscopic methods is shown in Figure 5.7. Each relies on two input images from the base visualization application, one for the left and right eye perspective. The event transfer chain is not affected by the stereoscopic visualization provision.

Stereoscopy using a two-projector solution. Two-projector stereoscopic systems are introduced in Section 3.2.2.4. They require a left and a right view, each projected via a separate projector. In CoWebViz, both views are provided as separate visualization streams on separate web pages. The implementation is basically the duplication of the visualization transfer chain of a monoscopic visualization as shown in Figure 5.7b (compare to Figure 5.7a). Both visualization chains are processed completely parallel. The resulting two mJPEG streams are provided

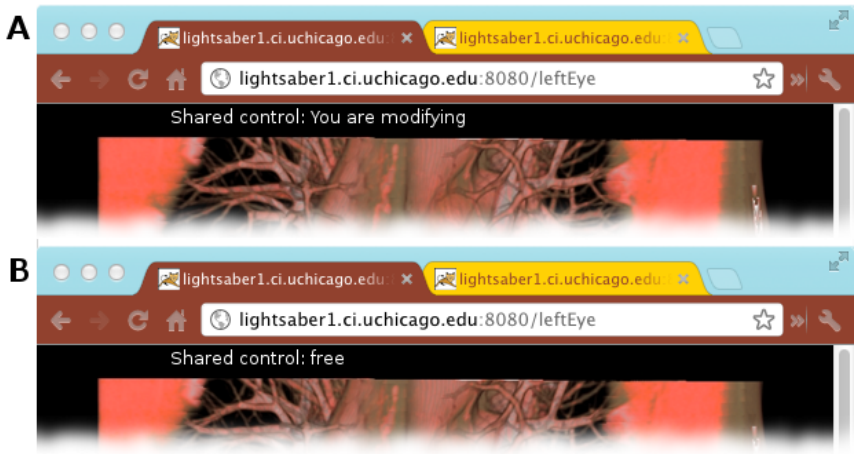
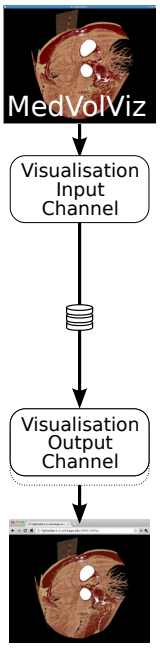
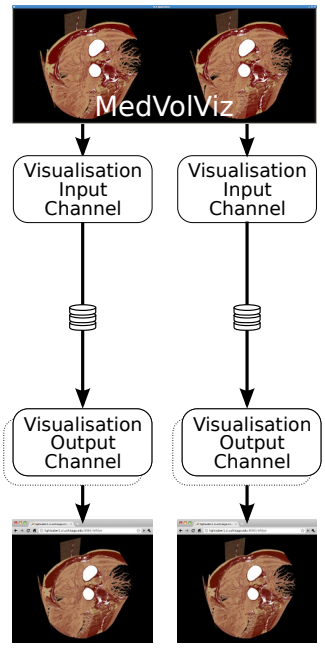


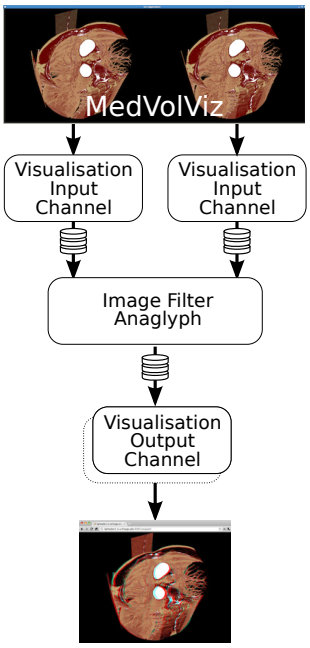
Figure 5.6. Types of notification each user gets during a multi-user session: A) The user modifies the visualization itself (all others will see "Shared control: Somebody is modifying") and B) Nobody is modifying the visualization.



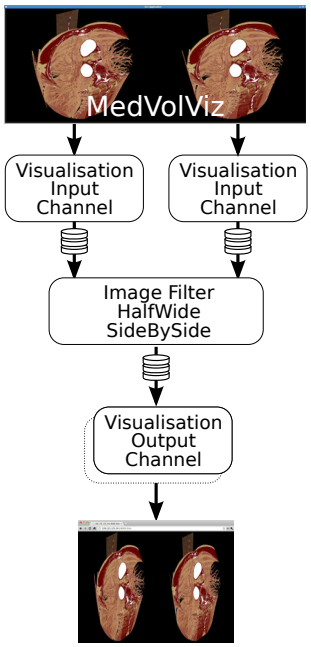
(a) Monoscopic transfer chain.



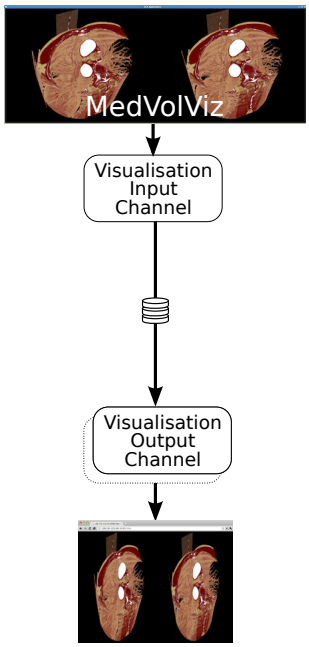
(b) Two-view stereoscopic transfer chain with a duplicated monoscopic chain.



(c) Anaglyph stereoscopic transfer chain with an image filter to create the anaglyph image of two input images.



(d) Half-wide side-by-side transfer chain with an image filter to create the formatted image of two input images.



(e) Half-wide side-by-side transfer chain with an image filter to create the formatted image of one input image.

Figure 5.7. CoWebViz's monoscopic (a) and stereoscopic visualization transfer chains and formats (b-e).

via an HTTP interface on specific URLs, e.g. `http://<URL>/left` and `http://<URL>/right`. They are accessible with control mechanism via web pages with very similar URLs, e.g. `http://<URL>/leftView` and `http://<URL>/rightView`.

Stereoscopy using a 3D TV. 3DTVs are introduced in Section 3.2.2.2. They are capable of taking various kinds of stereoscopic content as input. A widely supported content type is half-wide side-by-side, which has the advantage of requiring both stereoscopic views in a single image, with both views being separated on the image's left and right half. The separation allows to compress and transfer such images to a web browser without the need for a client side post-processing before it is displayed. This content type is implemented in two types of visualization transfer chains. The first is only usable if the base application provides a side-by-side visualization. It just screen scraps the whole visualization view in one image and scales the horizontal aspect ratio down to its half width (see Figure 5.7e). The second separately takes the left and right view, scales each view down, and merges them afterwards (module FilterHalfWideSbS, see Figure 5.7d). The second is more hardware intensive, but is also the variant that is usable in a more versatile way with different base applications and/or collaborative scenarios with multiple display devices. The half-wide side-by-side image is provided via a specified URL, e.g. `http://<URL>/3dtv`, which is accessible with control mechanism via a specific web page, e.g. `http://<URL>/3dtvView`.

Anaglyphic stereoscopy. Anaglyph stereoscopic content is usable on standard non-stereoscopic displays as introduced in Section 3.2.2.1. CoWebViz provides this content via the ImageFilterAnaglyph module, which takes two input images and creates a single anaglyph image. The content is transferred as single mJPEG stream, e.g. `http://<URL>/anaglyph`, with the control mechanism available via a specified web page, e.g. `http://<URL>/anaglyphView`. The chain is illustrated in Figure 5.7c.

This content type eminently influences the perception of the visualization's original colorization. Different algorithms were tested in order to get a good perception of red, especially the ones described by Wimmer [129]. Red is one of the encoding colors, but is also important for volume visualization. It is for instance being heavily used in the colorization modes that highlight muscles in different shades of red, but also in the perceptual colorization algorithm [6]. The current implementation is based on Wimmer's optimized algorithms with a further brightening of the red tones.

5.6 Special classroom additions

Some additional functionality was implemented in CoWebViz in order to make the remote usage of the visualization easier for a classroom setting.

The first addition is a *screenshot functionality* in order to allow students to take a picture of the current visualization state for their records. Each user can take a screenshot by clicking

on the according button, as shown in Figure 5.9. The server stores the picture that is being presented during the button click in a specific web-accessible directory. Afterwards, all pictures are downloadable via a specific web page, which is shown in Figure 5.8. It is implemented by combining the visualization output channel OutputFile with the output format for JPEG. This implementation allows to store images of any visualization type, even if the system currently shows another stereoscopic content type. This means that it allows to take for instance an anaglyph picture, while watching a two-view stereoscopic visualization in the classroom.

The second addition is a *client side GUI*, implemented to ease the visualization manipulation of the base application as shown in Figure 5.9. CoWebViz is mainly developed for the proof of concept usage with MedVolViz as base visualization application. MedVolViz, however, does not provide any GUI by itself, as e.g. for data exploration, but instead is completely controlled via keyboard events. This allows for a fast visualization-centered usage by people who know the application, but slows down people, who are not familiar with it. The GUI itself is implemented as HTML/CSS controls. Each button is assigned with a JavaScript function, which are collections of those mouse and/or key events that a user would have to give via the mouse and/or keyboard. On button press, these events are sent to the server.

The third addition is a *record and playback* functionality of a visualization session with the aim to guide a user (e.g. a student) through different states of a visualization without any knowledge of the application functionality. In detail, a user can access CoWebViz as usual, start a recording by clicking on a button, use the visualization as usual and eventually stop the recording by clicking on a button (see Figure 7.2: 1). It was implemented by storing all mouse and keyboard events that the user raises during this time (see ControlOutputScriptingFile module), each with the time difference to the previous event (see Listing 5.1). Multiple of such event sequences can be combined and stored as a lesson (see Figure 7.2: 2). A lesson can be played back, sequence by sequence (see ControlInputScriptingFile module). At every time the user has the opportunity to stop the automatic playback for a manual manipulation of the visualization (see Figure 7.2: 3-5).

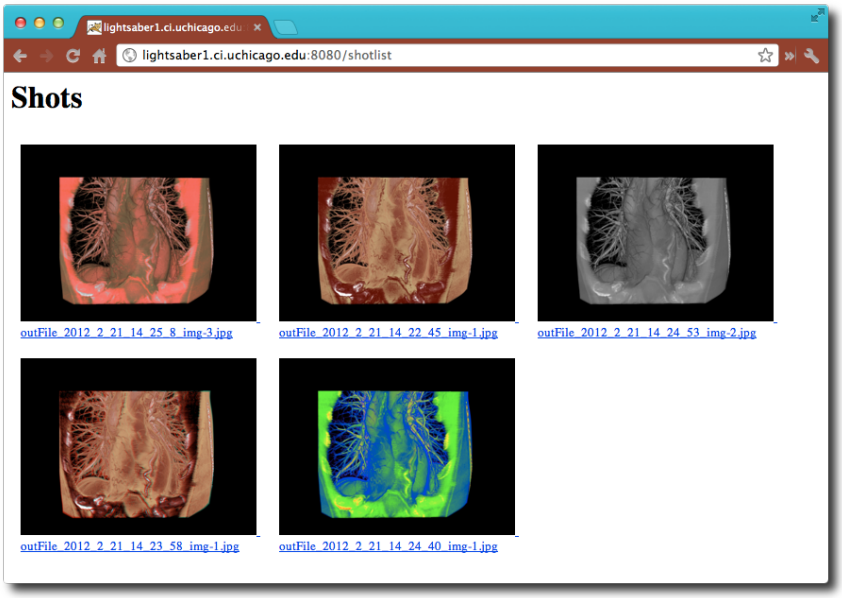


Figure 5.8. Presentation of screenshots taken during a visualization session.

Figure 5.9. This illustrations shows a CoWebViz v0.2 client with GUI buttons for CoWebViz's and MedVolViz's most important data exploration functionality: *(1)* Presentation mode switch, *(2)* Persistent control bar, *(3)* Visualization exploration modes (e.g. rotation), *(4)* Windowing, *(5)* Clipping, *(6)* Volume mouse pointer, *(7)* Reset all variables to default, *(8)* Screenshot taking, and *(9)* Manual rotation, panning, and zooming with arrow buttons.

Figure 5.10. This figure illustrates the recording functionality at the client side: *(1)* Start, stop, and name a new recorded sequence, *(2)* list recorded sequences for playback and lesson creation, *(3)* store a new lesson created by the + buttons of (2), *(4)* select, and *(5)* play a lesson.

6 Proof of concept conduction results

This chapter describes the proof of concept conduction: the practical application of CoWebViz in a collaborative anatomy class with shared stereoscopic visualization. This most eminent utilization of CoWebViz and its advantages compared to the previous class setup are described in Section 6.1. During the development phase and after the initial classroom usage, CoWebViz was also used for other projects by our own working group and collaborators. These applications are briefly described in Section 6.2 and 6.3 to show that the approach is also interesting for other scenarios. An overview of all usages is given in Table 6.1.

6.1 CoWebViz usage for collaborative anatomical education

CoWebViz was introduced in the collaborative virtual anatomy classroom for undergraduate biology students in spring 2010. The overall procedure and previous technical setup of the virtual anatomy class are introduced in Section 2.3.2.2. As before, it was a collaboration for education between two groups. It was held at The University of Chicago's John Crerar Library in Chicago (IL, USA) with remote participants at the Cardiff School of Computer Science and Informatics in Cardiff (UK) (see Section 2.3.2.2, [108, 266]). The University of Chicago room setup was similar to the setup of the previous years, it is illustrated in Figure 2.3 on page 20. An Access Grid client with two cameras was used for the videoconferencing, which was projected on a single projection screen, together with the 2D illustrations (see Fig. 2.3, right side). The stereoscopic visualization was rendered by MedVolViz (Section 2.3.2.1) on a visualization cluster at The University of Chicago's Computation Institute. It was presented via a permanently installed stereoscopic two-projector setup (see GeoWALL in Section 3.2.2.4). The remote classroom had a very similar setup.

The specific procedure of using CoWebViz 0.1 (see Section 5) for the stereoscopic visualization in the classroom is described in Section 6.1.1. Usage experiences and observations that led to a well performing system are described in Section 6.1.2. In Section 6.1.3, it is compared with the previous remote visualization system.

6.1.1 Procedure of using CoWebViz in class

On the server side (the visualization cluster), CoWebViz and MedVolViz required to be directly started on the command line via a ssh connection (Figure 6.1, step 1). Afterwards the left and right eye view were accessible via previously configured URLs.

Table 6.1. List of practical usages of CoWebViz.

Purpose	**Endpoint**	**Rendering location**	**Number of groups** (server distance)	**Details in Section**
Teach anatomy to biology students	Stereoscopy on 2-projector in classroom	Computation Institute	2 (180m, 6180km)	6.1
Teach anatomy to medical students	Stereoscopy on 2-projector in classroom	Computation Institute	1 (180m)	6.2.1
Demonstration/ Discussion	Stereoscopy on 3DTV in conference room	Computation/ Research Institute	1 (180m)	6.2.2
Inform surgeon during plastic surgery	Stereoscopy on 3DTV in operating room	Research institute	1 (20km)	6.2.3
Ad-hoc collaboration	Monoscopic	Computation Institute	2 (1km, 30km)	6.2.4
Integration in TPM	Monoscopic	Computation Institute	1 (n/a)	6.3.1
Integration in high-performance computing	Monoscopic	Argonne National Laboratories	unknown	6.3.2

The client was a single computer with two desktop metaphors, each was attached to one of the two projectors of the GeoWALL. A separate web browser window was opened on each of the two desktop metaphors. The specific URL of the view that corresponded to the desktop metaphor/ projector was loaded on the browser window and set into fullscreen mode (Figure 6.1, step 2). Google Chrome [272] was used, because of its splendid mJPEG support and fullscreen mode. After each web page and its visualization was loaded successfully, the visualization was usable (Figure 6.1, step 3).
Whereas the remote participants only require to perform step 2 in order to have interactive access (step 3), only the initiating user needs do step 1.
The lecturer used an additional laptop, without any connection to the stereoscopic presentation setup, on which a web browser window with the control view was loaded (see Figure 6.3c). Whereas the lecturer also required this second computer to control the 2D illustrations, a tablet PC could also have been used for this control view. [266]

6.1.2 Experiences of using CoWebViz in class

CoWebViz was gradually introduced into the class over the time of the first sessions. In the beginning, the class environment was only used to test CoWebViz in order to get an impression about its stability and real-time interactive performance. After assuring that it was stable for multi-participant usage, it was used to replace the previous setup. Nonetheless, manual configurations where necessary in the beginning, which were reduced stepwise over the time of the class.

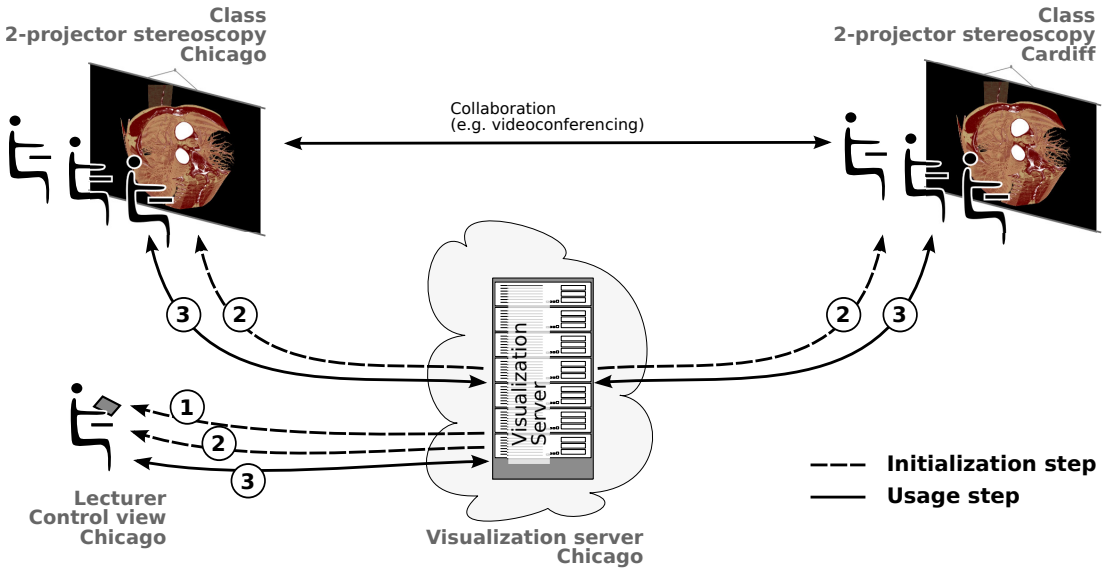


Figure 6.1. Steps to use CoWebViz in class: *(1)* Server session initiation by the lecturer or a technician, *(2)* loading the visualization web pages, and *(3)* use it on each device. The lecturer controls it via a dedicated control view from a laptop or tablet-PC.

The first tests were conducted from a single destination at the Chicago classroom using a non-threaded image streaming, which sequentially served each view and participant. The loading of the left and right view resulted in a fluent and responsive usability. However, a short lag between the presentation of the left and the right view became faintly recognizable. These observations slightly intensified, after increasing the number of active clients to three, each loading a left and right view. Still, the visualization was fluid and had little loss in the directness while modifying it. But the lag between the views became more recognizable.

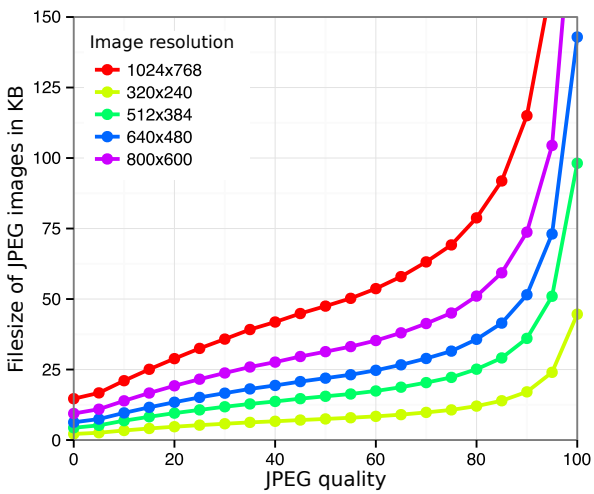
The next step was to test the visualization usage from two remote locations. At first both views were only loaded at the Chicago classroom, which resulted in a frame rate of about 15 fps, and afterwards in Cardiff. However, after Cardiff started the visualization, the frame rate dropped to about 3-6 fps at both classrooms. The lag between the left and right view was also recognizable. Both issues, the inter view presentation lag and frame rate drop, can be explained by the sequential system architecture of CoWebViz 0.1. The sequential image procession and distribution was no issue on a fast network connection. The Chicago classroom had a distance of 180 m to the visualization cluster and therefore had a low round-trip time of about 1 ms (measured with ping[1]). But the Cardiff classroom was about 6180 km away and, thus, had a higher round-trip time of about 133 ms. The result was that the Chicago client needed to wait for the image transfer to Cardiff in order to get an image update. Whereas, the later developed parallel system

[1]ping is an utility to check the availability of a remote host with a simple timing functionality.

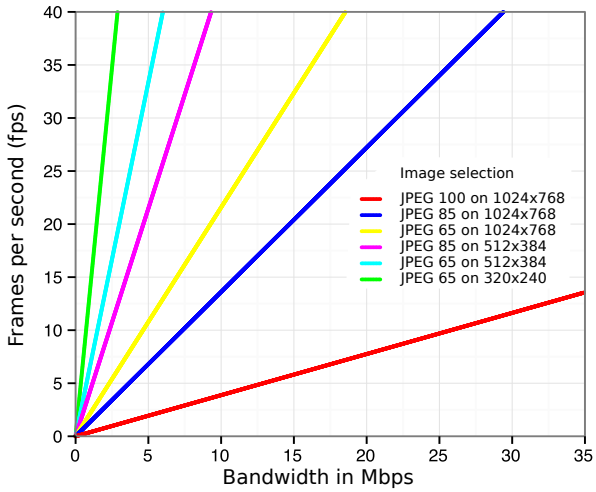
architecture improved the scalability, including the improvement of the low frame rate with multiple connections, a temporary solution was necessary for the immediate class usage. This was mainly done by providing two CoWebViz server instances, which allowed to provide a specific configuration for Chicago and Cardiff. This resulted in a fluent visualization at both classrooms, which however was not scalable and based on manually defined configurations and, thus, was not ad-hoc usable. The inter view lag was due to the high bandwidth connection almost not recognizable in the class and therefore did not require any immediate solution, but was overcome by parallelizing the system architecture later (see Section 7.1.3). [265]

The next test was conducted to get an impression of the directness of the visualization as observed from remote. A mJPEG stream is not directly testable from the web browser client, which prevents the speed measurement of transferring single images in the productive environment. Thus, only a simple test could be done, by pointing one of the Access Grid cameras on the visualization at the Chicago classroom. Our collaborators in Cardiff were then able to binary compare both visualization streams as provided by CoWebViz and the video stream. Objectively observed, both were similar fast. [265]

The traversal speed of a single image can obviously be reduced directly by reducing its file size. The automatic image quality algorithm (Section 5.2) is a direct method to increase the traversal speed by reducing the file size via reducing the quality (see Section 7.1).



(a) Comparison of different JPEG quality settings with the resulting file size of five different image resolution settings.



(b) Comparison of the anticipated frame rate of specific image resolution and JPEG quality settings on different bandwidth conditions.

Figure 6.2. Effect of JPEG quality, image resolution, and bandwidth conditions on frame rate in the configurations used for the class.

With a remote visualization system, there always is a tradeoff between a good image quality and a fluent usability. Thus, the next task was to find the best pair of a JPEG quality and image resolution for a usable system at both classrooms. A matrix of JPEG images with 5 different aspect ratios (from 320 x 240 to 1024 x 768) and 21 different JPEG quality levels (from 0 to 100) were created and visually compared next to each other. These images were produced with CoWebViz and show a CT volume visualization rendered by MedVolViz. This graph is too large for this thesis, but a small selection is presented in Figure 7.2a. Subjectively, they showed few loss of detail between a JPEG quality of 70 to 100. Compression artifacts were noticeable between a JPEG quality of 50 to about 70 and strongly noticeable below 50. The file size of these images was compared to each other in Figure 6.2a, which shows a strong file size increase with a JPEG quality above about 85.

Figure 6.2b presents the bandwidth usage that some of the previously defined image quality/ resolution settings (see Figure 6.2a) would require in order to get a desired frame rate. The Chicago classroom had a very high bandwidth connection to the visualization cluster and, thus, was provided with the best quality (image resolution: 1024 x 768, JPEG quality: 85). This resulted in a very high throughput of about 25 Mbps for both streams, each with 17 fps (see Figure 6.2b). Cardiff had a much lower, but still decent network connection. After several test sessions we ended up in sending them a quarter of the original resolution and a lower quality (image resolution: 512 x 384, JPEG quality: 65), which resulted in about 11 Mbps for both streams, each with 7 fps. The previous class system only allowed to stream a static aspect ratio of 704 x 576 pixels (also see Section 6.1.3). Its peak bandwidth usage during modification phases was about 2.8 Mbps. Extrapolated to the aspect ratios 512 x 384 and 1024 x 768[2] as used by CoWebViz, it resulted in 1.4 and 5.4 Mbps, respectively. [266]

The previous system utilized inter frame compression, which naturally has a much lower network utilization. Consequently, the usage of CoWebViz 0.1 resulted in a significantly higher bandwidth usage. The later developed automatic quality adjustment optimizes the frame rate as well as the image quality, but still has a high bandwidth usage compared to video codecs (Section 5.2). Compared to the static resolution of the previous class system, these optimizations also improve the presented quality during viewing phases, which is then maximized to full resolution and full JPEG quality. [266]

In order to test the collaborative functionality more extensively, a test session with three remote clients was conducted. The visualization was rendered as usual on the visualization cluster in Chicago and loaded at clients in Chicago, Cardiff (UK) and Göttingen (Germany) with an aspect ratio/JPEG quality of 1024 x 768/85 for the Chicago class and 512 x 384/65 for Cardiff and Göttingen. The visualization was fluid at all locations, but lagged few milliseconds up to 1 to 2 seconds behind, if controlled from Europe. Each client had different frame rates of 4 fps, 6 fps, and 15 fps, depending on the specific network connections in Göttingen, Cardiff, and Chicago,

[2]This is valid, under the assumption that the bandwidth usage increases linear with the resolution.

respectively. It seemed that the frame rate was higher in Cardiff and Göttingen, when controlled from Chicago compared to the controlling from their clients. [266]
The lagging visualization could therefore be explained by a slow control event transmission by the REST interface. This issue was optimized by implementing WebSockets for the event transfer (Section 5.3), which brought a further speedup to the interactivity shown in Section 4.2. Another observation during this session was that it was not always clear for everybody, who was modifying at any point in time if multiple participants started to modify. This issue was solved by additionally implementing the multi-user access control module as well as providing status notifications of who is currently modifying (see Section 5.3). [266]

6.1.3 Comparison of CoWebViz with the preceding class setup

While the previous section describes the observations made during the usage in the classroom by referencing associated improvements, CoWebViz's application in the class itself already improved the situation compared to the class' previous remote visualization system.

The most basic and most significant difference is the usage of a web browser as client application instead of a special native application. The previously used application was a special development, which was tied to the server application. It required to be specifically deployed with additional dependencies (e.g. Python [273] and vic [196]) and managed on the client computer. Associated with this is the continuous development overhead due to the client provision for multiple client environments (MacOS and Windows). Such is necessary, because every little change of the base environment may break the system, as happened with Python updates on the previous client application. The usage of an external application as the web browser in this case, may also result in the necessity of maintainance. But this only affects the server side, since web browsers are usually already being maintained.

The initial steps that were necessary to perform a class session were reduced by switching to CoWebViz. Now, the usage is as simple as using any other web application after the server is started: open a web browser and load a specified URL. A stereoscopic two-projector visualization only requires to open a second window and to set both into fullscreen mode (see Figure 6.3a and 6.3b). Previously, it was additionally necessary to manually align the left and right stereoscopic view exactly on top of each other (each provided via a single application window). This was necessary, because the application did not support full screen mode and no automatic centering. But a web-application is much simpler extendable and thus simply allowed to center a visualization stream in a web browser window with fullscreen functionality.

Previously, a specific H.261 video streamer was included in the visualization application and used to stream the visualization to the classrooms. Meanwhile, H.261 is a rather old video format, but is still frequently used in Access Grid. H.261 only supports a restricted set of aspect ratios with

the largest being Common Intermediate Format (CIF), which has an aspect ratio of 352 x 288 pixels. The previous remote visualization system therefore utilized a complex system of four single CIF streams for each eye perspective, resulting in a static aspect ratio of 4x Common Intermediate Format (4CIF) (704 x 576 pixels) and 8 streams [108].
CoWebViz allows to use a single stream, in contrast to the previous four streams, with an arbitrary choice of the aspect ratio. This allowed for a much easier handling and, thus, simpler client system. This can for instance be seen in the differentiation between the destinations during the first class sessions, where it was easily possible to send a specific higher aspect ratio to the Chicago classroom and a lower to the Cardiff classroom. But still, our collaborators empirical opinion at Cardiff was that the CoWebViz usage resulted in a better overall visualization quality in terms of compression artifacts and colorization. [265]

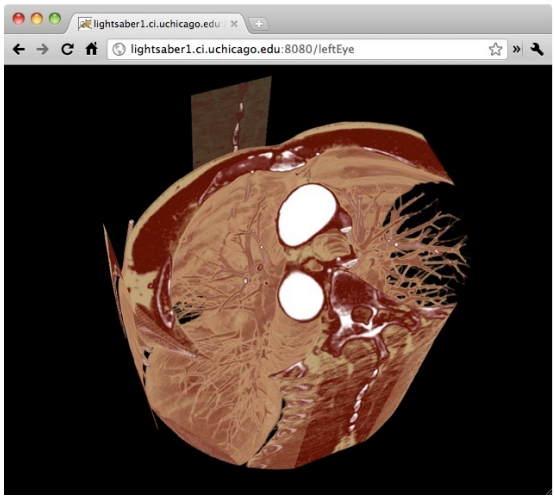
The usability of the base visualization application's remote usage via CoWebViz is on principle equal to its direct local usage. An obvious difference of both remote applications (CoWebViz and previous system) is the slightly slower performance, because there always is a noticeable lag between the finished rendering on the server side and its presentation on the remote client. But whereas the previous remote system was only able to exactly stream the two stereoscopic streams to the client, CoWebViz also allowed for other usage patterns. Previously, both stereoscopic streams needed to be loaded on the computer that was responsible for the stereoscopic projection. The lecturer, however, required to sit in front of the class with a separate computer to control the 2D and 3D visualization. It was therefore required to connect the lecturer's computer via a VNC connection to the computer of the stereoscopic setup in order to control the stereoscopic visualization.
CoWebViz in contrast simply allowed to open a specific control view (see Figure 6.3c) that shows no visualization but allows to fully control the visualization without further deployment.

6.2 Other applications of CoWebViz

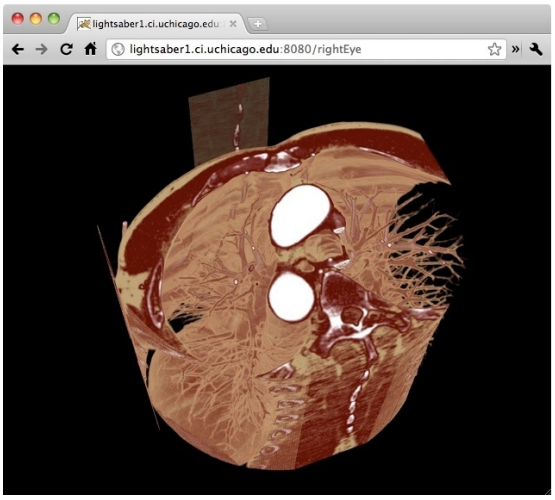
The previous section describes the utilization of CoWebViz 0.1 on a two-projector stereoscopic system. The optimized CoWebViz 0.2 was developed und used for other scenarios and stereoscopic setups, which are described in this section. Whereas, each of these stereoscopic content types can be used solely, they can also be used simultaneously in a single collaborative session as enabled by the parallel architecture (see Section 5.1) and illustrated in Figure 6.4.

6.2.1 Anatomical education in medical school

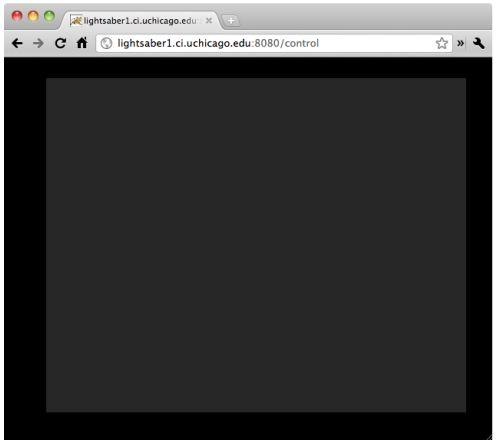
CoWebViz was used to teach anatomy to medical students at The University of Chicago's Pritzker School of Medicine in August/September 2011. The class was conceptualized as an optional session additional to the standard medical anatomy class. The visualization was presented on a



(a) Client instance: stereoscopic left eye view.



(b) Client instance: stereoscopic right eye view.



(c) Client instance: control view.

Figure 6.3. CoWebViz's distinct client instances (web pages) as used in the classroom. Each can be used to control the visualization, but the control view is dedicated to this functionality.

stereoscopic two-projector system, which required two CoWebViz client instances (one for each view). Teaching assistants held the class and controlled the visualization via a CoWebViz control client instance (see Fig. 6.3c) on a Laptop. They did not have any experience with this setup before the class started. We therefore developed several additions in order to help them using it: the client side GUI controls and the visualization scripting, both described in Section 5.6. Whereas the scripting was not used, the GUI was supportive for an instantaneous system usage. But since they used the system the first time in class, they quickly switched to take our direct personal support with the keyboard based control. The class was only realized for two sessions.

This class was based on the same setup as the previously described anatomy class for biology students (Section 6.1), but had no second participating remote group.

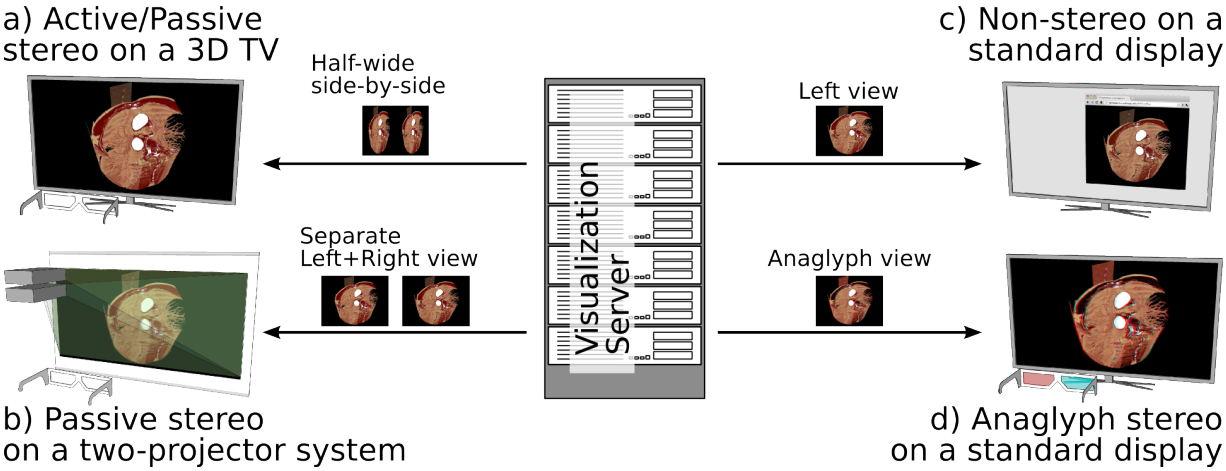


Figure 6.4. Stereoscopic visualization types for a sole or simultaneous usage as provided by CoWebViz (based on [51]).

The usage of the optimized CoWebViz version 0.2 in this class showed the advantages of the revised design in a practical setting. The main observations where that the inter view lag (lag between the presentation of the left and right view) was not recognizable anymore. The system was also usable without specific server side configurations, but delivered a fluent usable visualization during modification phases and maximum quality when viewing.

6.2.2 Demonstration in conference rooms

CoWebViz was and is occasionally used to present stereoscopic visualization for various reasons to current or potential future collaborators. The visualization was presented on state of the art 3DTVs at two locations, The University of Chicago Hospital and the NorthShore University HealthSystem Research Institute. These sessions usually required several client instances of CoWebViz, one for the 3DTV and others to control it on a Laptop. Such TVs are much easier deployable than two-projector setups. It was therefore also possible for a colleague to use CoWebViz instantaneously on his private 3DTV.

The technical environment was in principle and apart from the stereoscopic presentation identical to the class setup. Due to its simplified system architecture, few effort was necessary to extend CoWebViz to support a stereoscopic content type that is supported by 3DTVs (see Section 5.5). The visualization was rendered by MedVolViz either on a visualization cluster at The University of Chicago or a high performing single computer at NorthShore University HealthSystem.

The usage in conference rooms was CoWebViz's first usage on 3DTVs and therefore showed its feasibility.

6.2.3 Informing surgeons in the operating room

In 2013, CoWebViz was used in a pilot study[3] to present stereoscopic visualization to a plastic surgeon during reconstructive surgery in the operating room, e.g. for craniofacial and free-flap breast reconstruction. While the diagnosis is raised on other systems, this stereoscopic presentation is an additional resource for a fast real-time guidance for the surgeon. The visualization of the patient specific imaging data is presented in stereoscopy on a 3DTV, which was specially deployed in the operating room. The system is currently not controlled by the surgeon, who is too much involved into the procedure, but by another person on his command.

A single CoWebViz client instance is used on a Laptop that is connected to the 3DTV. The visualization is remotely rendered on a high-performing computer at NorthShore University HealthSystem by MedVolViz. The distance to the operating room is about 20km.

Since this is an ongoing study, the results will follow. However, the current state of the study already shows that real-time visualization via CoWebViz is usable in the operating room of a single health corporation's hospital with the visualization being rendered in a remote located data center.

6.2.4 Monoscopic ad-hoc collaborative usage

CoWebViz was used a single time for a truly unscheduled collaborative ad-hoc demonstration. It was used to present MedVolViz's visualization and CoWebViz to collaborators at The University of Chicago with the visualization being rendered at the University of Chicago, but us being at NorthShore University HealthSystem. The demand for this session arose during a telephone conference. It was basically done during the phone call by starting CoWebViz and sending the URL to the collaborators via email. Both participants loaded the URL and modified as needed after oral communication.

6.3 Integration of CoWebViz into external web-based systems

CoWebViz's system design involves its integration into external systems on the server and the client side. The server side integration of base visualization applications is already described in Section 5.1.1. The client side integration is already shown by CoWebViz's client itself, which integrates the MJPEG visualization stream into a web page that adds the control functionality (see Section 5.1.3). But it also allows to be integrated into external client side systems, preferable into web-based systems. The HTTP interfaces provided by CoWebViz is optimized to be accessed with web browsers, but is also accessible by any other application.

[3] This pilot study is conducted by Nigel M. Parsad and Jonathan C. Silverstein, NorthShore University HealthSystem, Evanston IL, USA

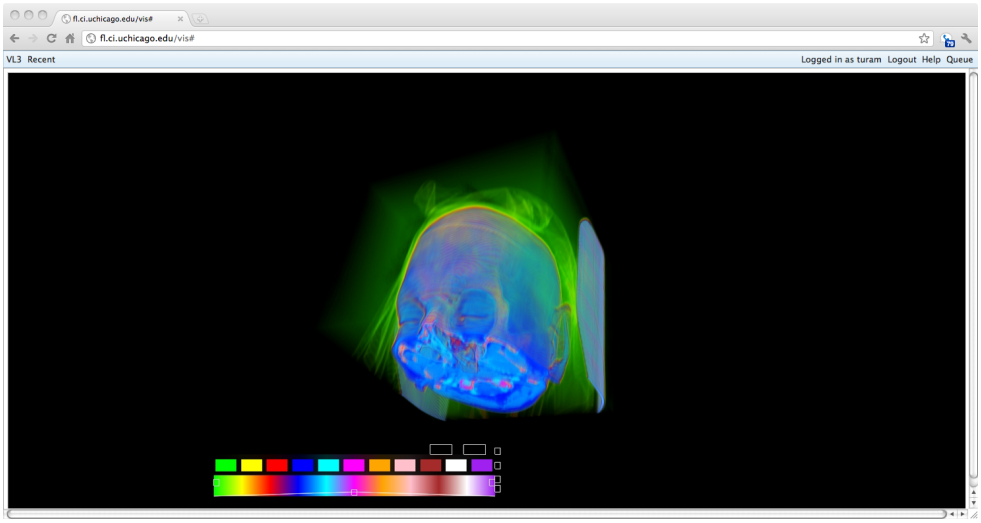


Figure 6.5. This screenshot shows the integration of CoWebViz as web-based front-end of a high-performing visualization cluster setup at the Argonne National Laboratories. This screenshot was provided by Tom Uram.

6.3.1 Viewing visualization in TPM

CoWebViz's monoscopic visualization was integrated into the TelePresence Microscopy Collaboratoration (TPM) [238] for testing purpose without its control functionality[4] (see Section 3.3.3.2). The visualization was rendered and stored as mJPEG file using MedVolViz and CoWebViz on a visualization cluster at the University of Chicago's Computation Institute. CoWebViz was then started without base application to play the recorded visualization session in loop on a defined URL. Since the TPM is a system that already provides access to various mJPEG web camera streams, the integration of CoWebViz's mJPEG stream was straight forward. This setup can be used as described above to play a recorded visualization session. But a pure real-time rendered visualization stream can also be integrated into TPM or any other external web application, with the user modifying the visualization directly via CoWebViz's control client instance.
TPM is used to stream real-time video from a hospital's operating room to a hospital's conference room. This integration test might be of interest for the audience, since it shows that a monoscopic or stereoscopic visualization can be integrated additionally to a surgical video stream.

6.3.2 Integration into a high-performance computing environment

CoWebViz was used to provide high-performing interactive visualization to researchers who render visualization on a visualization cluster at the Argonne National Laboratory's Future Laboratories (e.g. for astrophysics).

[4]The integration into TPM was tested together with Nestor J. Zaluzec, Electron Microscopy Center, Argonne National Laboratory, Argonne, IL, USA.

CoWebViz was integrated with full functionality into a web application as part of a Grid computing setup to be seamlessly usable from a pure web browser. The external web application handles the user authentication (via MyProxy [274]), the data selection, the initialization, and the management of the visualization session[5]. The integrated CoWebViz provides the interactive access to the visualization. Figure 6.5 shows a screenshot of CoWebViz as it is integrated into this Grid web application. The top menu bar shows the user information, a help button for additional information, and all necessary functionality to manage the visualization. The large part of the application screen is reserved for the visualization presentation, which is equally usable as a standalone CoWebViz.

[5]The integration of CoWebViz, including information and screenshot are provided by Tom Uram (Argonne National Laboratories Future Laboratories).

7 Performance test results and comparison

This section provides detailed performance test results of CoWebViz's usage under peak performance in defined test environments and its comparison to a selection of the related work tools. These measured results stay in contrast to the pure observations based on the practical utilization described in the previous chapter.
The performance comparison in Section 7.1 compares different methods of transferring the visualization by starting with the class configuration and ending with a configuration that uses all optimizations. This comparison provides a quantified description of CoWebViz's usability, but also provides a hint of the performance of other applications that use these methods. This comparison of the principles, based on CoWebViz's implementation, is of interest, because most applications are either not freely available or bound to specific visualization applications, which prevents a direct comparison.
Performance test results of the supported stereoscopic visualization types show each type's usability (see Section 7.1.2). The scalability test results show the possible usability with up to 6 simultaneous clients (see Section 7.1.2).
All of these tests are based on CoWebViz internal measurements, which provides detailed information, but can not be compared with related work. Thus, a self-developed frame rate counter (see Section 2.3.2.3) was used on the client side to provide a performance comparison of CoWebViz, VNC, and ScreenLeap (see Section 7.2).

7.1 System performance tests

The monoscopic, stereoscopic, and scalability performance test results of the following sections were conducted as described in the Section 2.3.

7.1.1 Comparison of different visualization transfer optimization levels

The baseline configuration (A) is the transfer of any processed image with a fixed quality, as used in the virtual anatomy class (CoWebViz 0.1). The second configuration (B) is the transfer of only those images that have been modified with a fixed quality, which is also used by other web-based visualization applications. The third test configuration (C) is the transfer of only those images that have been modified with an automatic quality/network adaption, which is the optimized version of CoWebViz 0.2. The quantified results of these tested configurations are presented in Table 7.1.

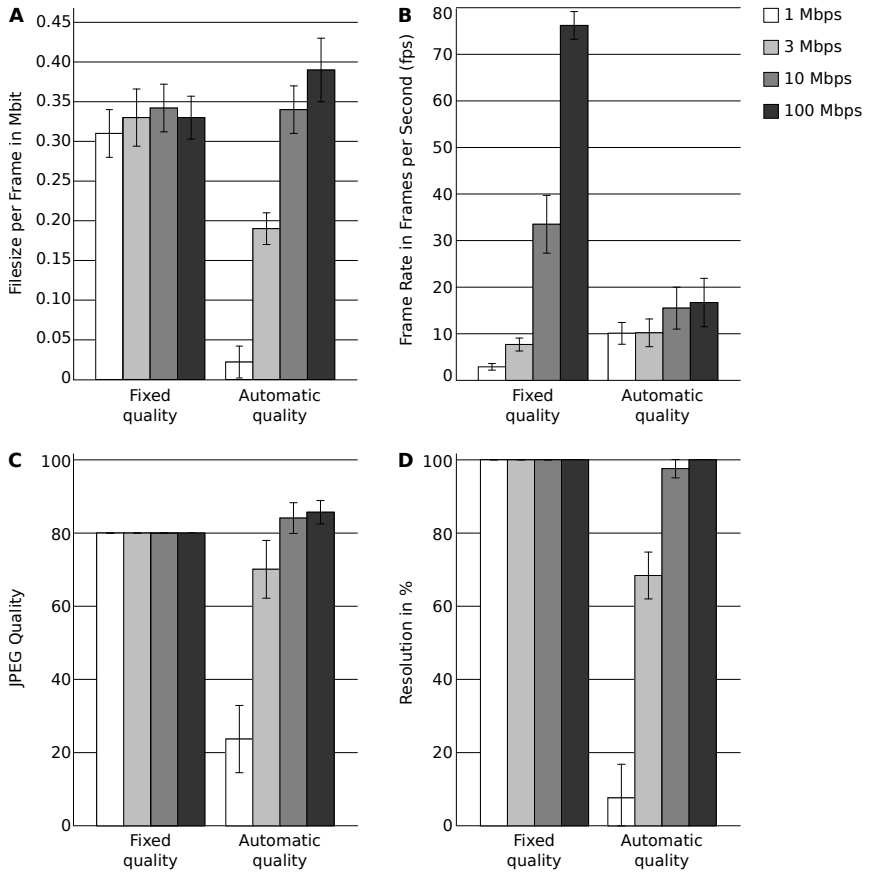


Figure 7.1. Comparison of different variables of the most simple (all processed images with fixed quality settings, see configuration A in Table 7.1) and most advanced image transfer type (modified images with automatic quality settings, see configuration C in Table 7.1): *(A)* The file size per image, *(B)* the frame rate, *(C)* the JPEG quality, and *(D)* the resolution (partially based on [51]).

A. Fixed quality, send always: The transmission of all processed images (regardless of whether the image has modifications or not) with a fixed JPEG quality of 80 and a resolution of 1024 x 768 results in a very high maximum frame rate of 76.2 fps on a fast network (see Table 7.1, A). Associated with this high frame rate is a very high network throughput of 25 Mbps, but also a high server (51.7%), and client CPU usage (86.7%). On the other side, if using a slow connection of 1 Mbps, the frame rate is very low (2.9 fps). Both, the high and low image transfer rate is a direct consequence of the available network bandwidth. Since only few of the transmitted images are actually modified this method has a huge redundancy. [51]

B. Fixed quality, send modified: The sole transmission of the modified images with a fixed quality and resolution eliminates the previously observed redundancy (see Table 7.1, B). The maximal possible frame rate therefore is the base application's frame rate. This method

is often used by other web-based remote visualization systems, with the difference of pulling single images instead of a persistent motion JPEG connection. On the 11 and 90 Mbps networks, the frame rate is reduced by 59.5 fps (78%) and 18.0 fps (54%) and the network throughput by 19.6 Mbps (78%) and 6.2 Mbps (54%), respectively. On the 3 and 1 Mbps networks, the frame rate and throughput reduces only insignificantly by less than 1 fps and 1 Mbps. This almost non-existing decrease on the slow networks shows that the network in configuration A was already on its full capacity. If there is no modified visualization at the base application, no images are transmitted. Consequently the frame rate and network throughput decreases to 0 and the client and sever CPU usage go into an idle state. [51]

C. Auto quality, send modified: The automatic quality adjustment towards the available network conditions and the sole transfer of modified images is a necessary step to provide interactive usability on all network types (see Table 7.1, C). On the higher bandwidth connections of 11 and 90 Mbps, the frame rate slightly changes by less than 1 fps towards configuration B. But due to the JPEG quality increase of 5.7 and 4.1, the network throughput also increases by 1.2 and 0.2 Mbps, on both networks. On the lower bandwidth connections of 3 and 1 Mbps, however, the frame rate increases by 2.0 fps (24%) and 7 fps (374%), respectively. Due to the associated JPEG quality decrease of 9.9 and 56.5 and a image resolution decrease of 31.6% and 92.4%, the network throughput is almost similar to configuration B. Accompanied with the frame rate increase is the client/server CPU usage increase of 11%/13.3% and 6.8%/13.2%, on the 3 and 1 Mbps network, respectively. This is caused by the higher image procession need. As previously, if there is no modified visualization, there is no active image transfer with a frame rate of 0. But in contrast to the previous system, each user is provided with an image of maximal quality, which results in the best possible image quality on all network types during a viewing phase. This behavior is illustrated in in Figure 7.2, which shows the recording of several parameters during a whole visualization session. [51]

Figure 7.1 illustrates the differences caused by the automatic quality adjustment (configuration C) compared to the fixed quality setting (configuration A). The fixed quality setup results in a low file size variation (Fig. 7.1A) but in a large frame rate variation (Fig. 7.1B) on different networks. Vice versa, the automatic quality setup results in a large file size variation, but a low frame rate variation. This difference between the two setups is caused by the adapted JPEG quality and image resolution towards the available bandwidth as shown in Fig. 7.1C/D. These subfigures show equal JPEG quality and image resolution settings on the fixed setup, but varying values on the automatic quality setup. The latter leads to reduced file sizes and consequently results in the higher frame rate. [51]
Figure 7.2 illustrates the auto quality adjustment's behavior of the JPEG quality (Fig. 7.2b), frame rate (Fig. 7.2c), bandwidth (Fig. 7.2d) and the mouse event rate (Fig. 7.2e) over the course of a 140 second visualization session. Example image cutouts of selected phases are presented

in Figure 7.2a. A modified image is caused by mouse events, given by the user (Fig. 7.2e). Different mouse event behaviors cause a different visualization transfer behavior between 20-40, 60-80, and 100-120 seconds. The events are raised slow and constant, fast and constant, and slow and inconstant in the first, second, and third phase, respectively. These event variations cause a slightly different visualization streaming behavior with a JPEG quality that slowly changes over time, changes abruptly, and constantly varies with interruptions of maximal quality in the first, second, and third phase, respectively. The overall enhancement of the automatic quality adjustment is highlighted in Figure 7.2b, which are the different quality levels of 100 (1), 85 (2), 60 (3) and 35 (4) for each of the different network types. Between these modification phases, the quality is maximized on all networks. These quality levels result in the different image quality, shown in Figure 7.2a.

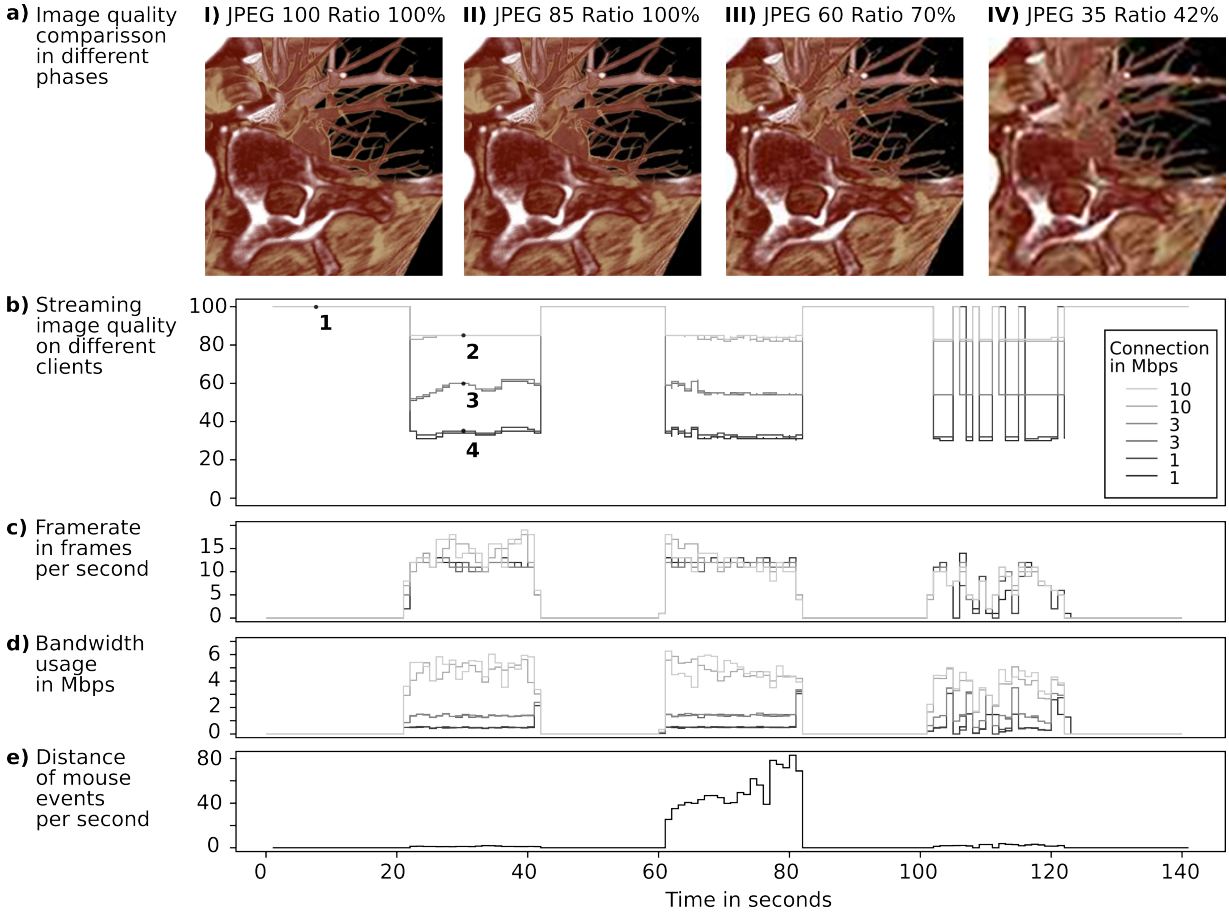


Figure 7.2. CoWebViz's behavior during three visualization modification phases of 20 sec (at 20, 60, and 100 sec). As shown by the distance between the mouse pointer events in Subfigure **e**, the mouse movement is slow & steady at 20, fast & steady at 60, and slow & unsteady at 100 sec. This results in a specific image quality (**a, b**), frame rate (**c**), and bandwidth usage (**d**). (Subfigure (**a**) from [51])

Table 7.1. Performance measurement results of different algorithm settings (**I**) and stereoscopic transfer types (**II**). The values are presented as mean ± standard deviation. (fixed resolution = 1024 x 768, fixed JPEG quality = 80; partially based on [51])

Configuration	Network Type	Frame rate in fps	Through-put in Mbps	Client CPU usage in %	Server CPU usage in %	JPEG quality	Image resolu-tion in %
I. Method comparison on the basis of monoscopic visualization							
A. Send *all* images, fixed resolution/ JPEG quality	90 Mbps	76.2±3.0	25.2±0.1	86.7±1.5	51.7±0.2	80±0.0	100±0.0
	11 Mbps	33.5±6.2	11.4±0.1	85.6±1.7	23.6±0.4	80±0.0	100±0.0
	3 Mbps	7.7±1.4	2.6±0.0	46.1±0.5	13.0±0.3	80±0.0	100±0.0
	1 Mbps	2.9±0.7	0.9±0.2	15.9±20.1	7.7±3.3	80±0.0	100±0.0
B. Send *modified* images, fixed resolution/ JPEG quality	90 Mbps	16.7±0.2	5.6±0.1	54.1±1.2	18.5±0.4	80±0.0	100±0.0
	11 Mbps	15.5±4.0	5.2±0.1	61.8±1.9	16.5±0.3	80±0.0	100±0.0
	3 Mbps	8.2±0.1	2.6±0.0	44.6±1.2	13.6±0.3	80±0.0	100±0.0
	1 Mbps	2.7±1.2	0.9±0.4	7.9±6.8	8.0±3.5	80±0.0	100±0.0
C. Send modified images: *automatic* setting of resolution/ JPEG quality	90 Mbps	17.3±0.3	6.8±0.1	52.5±1.5	17.6±0.5	85.7±0.2	100±0.0
	11 Mbps	15.5±0.3	5.4±0.1	63.7±1.9	21.7±0.8	84.1±0.3	97.6±0.2
	3 Mbps	10.2±0.2	2.1±0.0	55.6±2.0	26.9±0.9	70.1±0.6	68.4±0.5
	1 Mbps	10.1±2.3	0.3±0.5	14.7±3.5	21.2±4.6	23.5±9.2	7.6±9.2
II. Comparison of stereoscopic visualization transfer techniques based on Configuration 3							
D. Anaglyph stereoscopy	90 Mbps	14.9±6.1	5.3±2.5	44.6±20.3	121.3±8.6	85.7±3.1	100±0.1
	11 Mbps	14.9±5.1	5.6±2.1	63.0±23.5	130.0±17.7	82.7±5.9	94.3±4.5
E. 3D TV	90 Mbps	16.2±5.1	6.2±1.8	64.5±27.2	40.7±17.1	85.8±3.3	100±0.0
	11 Mbps	15.1±4.9	5.8±1.9	70.3±28.2	37.6±14.8	85.5±3.5	99.6±1.3
F. Send two eye perspective views	90 Mbps	16.4±4.6	5.1±1.8	77.3±28.0	73.9±37.5	79.3±6.3	89.1±7.1
		15.8±5.2	5.0±1.9			80.1±7.0	89.7±6.8
	11 Mbps	13.9±4.1	2.9±1.1	79.7±27.7	59.2±25.6	66.9±11.5	63.9±13.8
		13.8±4.5	2.8±1.1			66.4±13.1	62.1±14.7

7.1.2 Stereoscopic modes

All stereoscopic visualization tests are based on configuration C of Section 7.1.1.

The utilization of anaglyphic visualization results in a similar client side observable performance as it does for monoscopic visualization. The one big difference lies in the server CPU usage increase of 103% and 108% on the 90 Mbps and 11 Mbps networks, respectively. This high increase is caused by the constant merging of two input images into one anaglyph image. This procession overhead causes a slight frame rate decrease on the fast networks (-2.4 fps, -0.6 fps), which in turn causes a slight decrease of the network throughput (-1.4 Mbps) and, subsequently, in a slightly lower client side CPU usage. The JPEG quality and image resolution is slightly lower on the 11 Mbps network, which is likely caused by a different bandwidth availability during the algorithm's initial bandwidth test.

Half-wide side-by-side stereoscopic content also results in a similar performance as the anaglyph content, although with a lower extra CPU usage, when using the single view capturer. The frame rate and throughput are almost equal. The slight JPEG quality and image resolution difference on the 11 Mbps network is likely again a result off the bandwidth availability during the algorithm's initial bandwidth test.

The two-view stereoscopy requires to transfer two streams from the server to the client for a single stereoscopic presentation. On each tested network, both related streams have very similar values of ±1. This is an expected behavior, since both are streamed in a single session, conducted with equal network conditions and initial bandwidth test. However, these minimal differences between the values (including the one slightly larger difference of the image resolution at the 11 Mbps network) point out that both streams are transferred independently from each other. Whereas the difference between two related streams are very small, there is a larger difference when comparing a single stream to the standard monoscopic visualization (configuration C). On the 90 and 11 Mbps network, all but the CPU usage values are decreased. The automatic quality adjustment algorithm results in a reduced JPEG quality (about -6.5 and -17.5) and image resolution (about -10.5% and -34.0%). Despite this decrease of the quality values, also the frame rate is slightly decreased by 0.6 and 1.6 fps on the 90 and 11 Mbps networks, respectively. These changes are a direct consequence of the automatic quality algorithm, which accounts for multiple views. This fact can be seen in the increased client (24.8% and 16.0%) and server side CPU usage (56.3% and 37.5%), which increases despite the fact that all other values are lower. [51]

7.1.3 Scalability tests

In order to show CoWebViz's scalability, it was tested with 1 to 6 simultaneous accessing clients as described in the Section 2.3.2. The results are presented in Figure 7.3. They show different performance behaviors on the client (Fig. 7.3A-D) and server side (Fig. 7.3E-F).
CoWebViz retains a steady performance on each network type, independently of the amount of the tested 1 to 6 simultaneous accessing clients. This can be seen in a similar average frame rate of 10.5 fps, 11.4 fps, and 10.9 fps (Fig. 7.3C) per network type. This similar frame rate is caused at the expense of the image file size (Fig. 7.3B) and, consequently, the JPEG quality (Fig. 7.3A), both are showing separate curves on each network type. Averaged over all 6 tests, it results in a specific JPEG quality of 26.3, 56.9, and 69.4 (Fig. 7.3A) and consequently in a specific image file size of 3165 KByte, 15238 KByte, and 25519 KByte (Fig. 7.3B) on a 1 Mbps, 3 Mbps and 10 Mbps network, respectively. The small variations of each graph are caused by network variations. The combination of the frame rate and the file size leads to an average client side bandwidth usage of 0.3 Mbps, 1.4 Mbps, and 2.2 Mbps (Fig. 7.3D).
Whereas the client side performance is almost constant between 1 and 6 clients, the server side is marked by constantly increasing values with an increasing number of connected clients. The

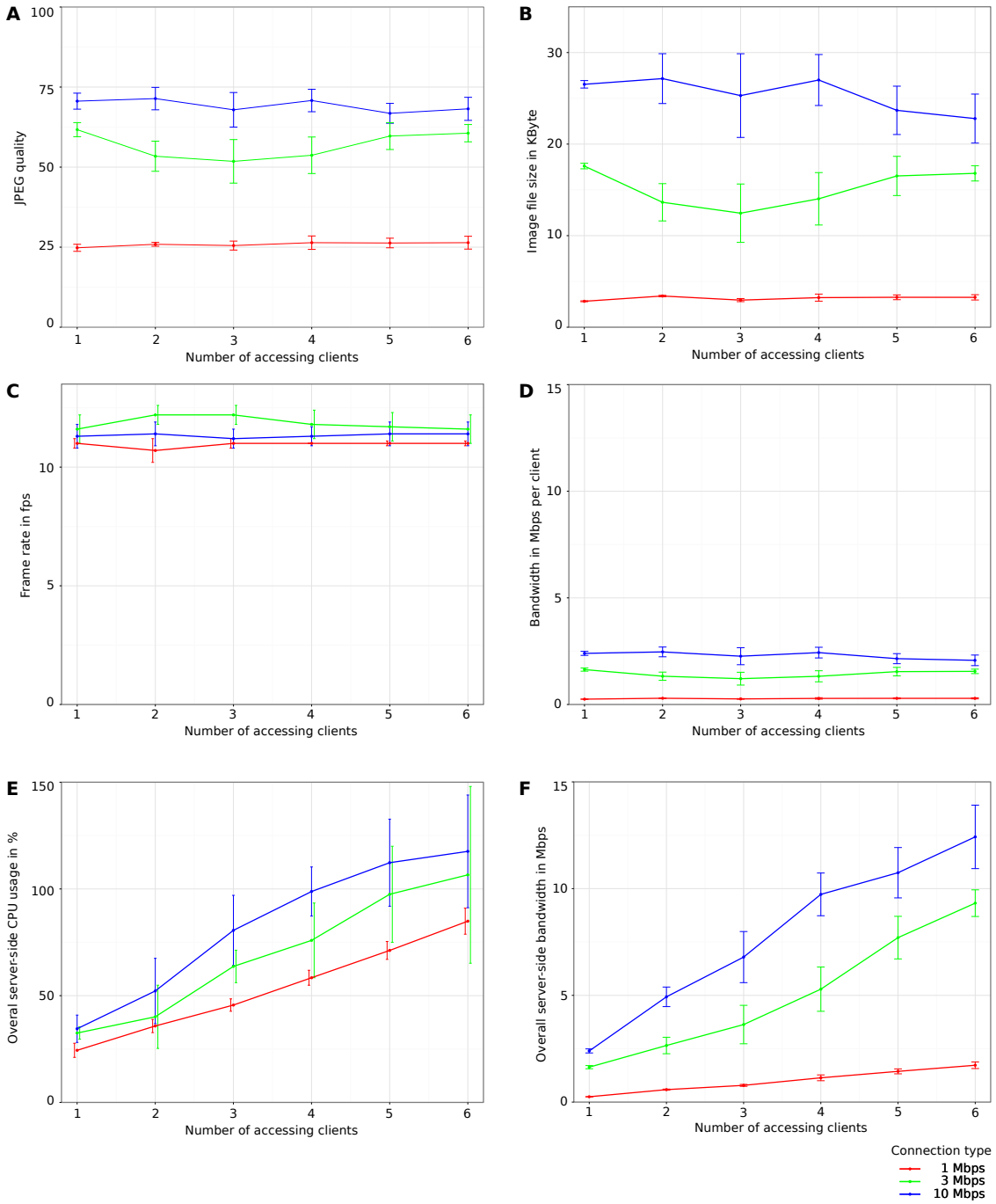


Figure 7.3. Scalability of CoWebViz using monoscopic visualization. The figures show the mean and standard deviation of following variables while simultaneously accessing with different numbers of clients. The client side behavior is described by the JPEG quality (A), file size per image (B), frame rate (C), and bandwidth usage (D). Notable parameters on the server side are the server side CPU usage (E) and overall bandwidth usage (F).

constantly low client side throughput (Fig. 7.3D) leads to an increasing bandwidth usage on the server side (Fig. 7.3F). Also the bandwidth usage on the 1 Mbps, 3 Mbps, and 10 Mbps networks increases: 0.6-3.6 Mbps, 3.2-18 Mbps, and 4.6-24 Mbps (respectively with 1 to 6 connecting clients). Due to the higher image processing need also the server CPU usage (Fig. 7.3E) is increasing with more simultaneously accessing clients.

7.2 Comparison to similar tools

This section presents the results of the blackbox frame rate tests of CoWebViz and two selected tools as described in Section 2.3.2. The measured frame rates are illustrated in Figure 7.4 as total mean frame rate per session (white bars) and interquartile mean frame rate per session (gray bars). Quartiles divide the measured data in four equally large parts to show the mean frame rate of the slowest (<25%), the fastest (>75%) and two middle time periods of the test session.

The bargraph shows a total frame rate of 1.45±0.64 fps, 4.01±1.14 fps, and 8.92±2.35 fps for VNC, Screenleap.com, and CoWebViz, respectively. CoWebViz has the highest frame rate, which is more than twice the frame rate of the other tools. The low frame rate of the VNC client might be explainable by its optimization for the remote usage of full desktop metaphors. Such requires much more often the transmission of little modified parts than the re-transmission of the whole desktop metaphor (necessary in the case of volume visualization). Screenleap.com results in a higher frame rate than VNC, but its maximum frame rate (above quartile 75%) is still lower than

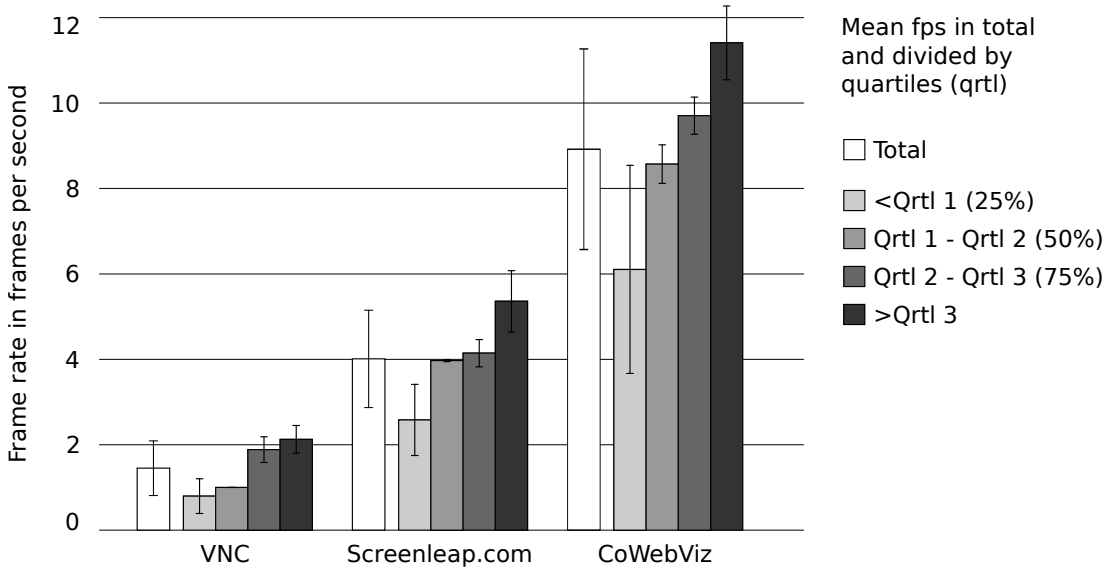


Figure 7.4. Comparison of CoWebViz's frame rate with alternative remote visualization applications.

CoWebViz's minimum frame rate (below quartile 25%). Both, Screenleap.com and CoWebViz are transmitting JPEG images of the whole visualization. The higher frame rate of CoWebViz might be explained by the utilization of a highly parallelized server application and/or by the mJPEG streaming, instead of single image pulling (see Section 4.1.2).
CoWebViz's maximum frame rate is 11.41±0.9 fps (>75% of the data). Despite the fact that this value is acquired via client side measurement, it still lies within the range of the data shown in the performance test results in Section 7.1. This basically shows the consistency of CoWebViz's performance across different measurement methodologies.

8 Discussion

An educational scenario that requires the most dynamic and life-like volume visualization of the human body to present visual answers to all emerging questions from the students is its usage in an interactive and remotely shared gross anatomical class. It requires interactive and stereoscopic volume visualization, rendered on full resolution volume data of modern imaging modalities (e.g. Computed Tomography (CT) or Magnetic Resonance Imaging (MRI)). Because such visualization can not be rendered on any computer (e.g. available in a typical classroom), remote visualization needs to be provided using client software that is as simply deployable and usable as possible. This is particularly important if remote participants need to be involed in the scenario.

The approach proposed by this work provides volume visualization with all desired aspects (see Section 1.4) and lowest software requirements on the client side by using a pure standard web browser. The importance of pure web browser solutions is for example being shown by the literature analysis, which shows a shift from the usage of web browsers with added software (mean publication date: 2005, N=21) towards the usage of pure web browsers in recent years (mean publication date: 2010, N=15). In contrast to each single visualization aspect (interactive, stereoscopic, collaborative), which was previously possible, this approach combines all concepts allowing for various extensive use cases with reduced user involvement.

The feasibility of this approach was analyzed via a proof of concept conduction in an educational class scenario. A proof of concept conduction is an adequate methodology, because it allows to analyze new concepts in a real-world application with direct observations and experience feedback. The virtual anatomy class was a use case with exceptional high exploitation of different techniques and media and, thus, very suitable to test a complex scenario. However, due to the class design, the developed prototype *Co*llaborative *Web*-based *Vi*sualiz*z*ation (CoWebViz) was only being used by the lecturer. Its application in other uses cases, e.g. presentations on a conference room's 3DTV (Section 6.2.2), high-performance computing environments (Section 6.3.2), and clinical surgery (Section 6.2.3), show their basic feasibility and benefit. CoWebViz's usage in defined test environments shows the technical feasibility via performance comparisons with different transfer methods and related work tools.

The proof of concept conduction and any subsequent performance test proofs the initially defined hypothesis valid. The four research requirements that need to be fulfilled are discussed in the following sections.

8.1 Minimum user knowledge and involvement during setup and usage of interactive remote visualization in real-time

The setup phase includes all steps that need to be done before the system can be used. The approach of this work was to reduce these steps by reducing client requirements. In case of the proof of concept conduction, the stereoscopic presentation system was already available at a designated location on the university campus. The specific steps that were necessary to use this setup with CoWebViz are described in Section 6.1 and compared with the previous class system in Section 6.1.3. From these descriptions follows that CoWebViz requires less steps for the setup then the previous system. When the server application is started, a participant is only required to open a web browser and to load the specified Uniform Resource Locator (URL), which is comparable to the usage of most other web applications. The host of the visualization session, however, still needs to initialize the server session on the server command line, which is not straight forward for a naïve user. But the remote session initiation and data upload via pure web browser front-ends is straight forward and was already shown by other projects [69, 251].

The approach of using pure web browsers is especially interesting, if it is to be used on a corporate or hospital computer, which is mostly associated with limited user rights. Such an environment would aggravate the instantaneous deployment and usage of native applications, including web-based applications with added software (e.g. Java or Flash [169, 275]). In turn, no further deployment is required if only a standard web browser is required. The capability of being ad-hoc usable on a remote computer without deployments was shown by a daily-life incident where the system needed to be demonstrated to remote collaborators without prior scheduling (see Section 6.2.4). Due to their wide availability, usage, and ease of use, web-based applications are familiar to most people and, thus, have a positive effect in people using it [4].

CoWebViz's approach to ease the system's usage phase is to provide an experience that is as close to the local usage of the base application as possible with the aim of being most transparent to the user. CoWebViz was developed as web-based front-end for possibly any visualization application and is specialized for remote and collaborative stereoscopic visualization on different stereoscopic systems. In case of the class usage, the base application was already known to the lecturer and, thus, did not require any new learning. But the second class scenario was lectured by student teaching assistants (see Section 6.2.1), who had never used the system before. A client side Graphical User Interface (GUI) with domain specific vocabulary was implemented specifically for them on top of the base application's control mechanism (see Section 5.6). However, they used this GUI only in the beginning. With our direct support available in these sessions, they quickly switched to the faster but non-graphical control mechanism of the base visualization application. The first class scenario shows that the system reduces the need to acquire new knowledge by extending existing base applications (see generic integration discussion in Section 8.3). The second scenario shows that the added GUI might be of some value for people who

have no direct personal support, e.g. because they are remote. Because the teaching assistants only needed to learn the specific commands of the base application, it also shows that CoWebViz allows for a transparent usage of the base application.

There are various other software systems available that could potentially be used to provide remote and collaborative visualization, including web-based approaches. Most of them differ to CoWebViz's approach by requiring special software and/or hardware deployments and especially by not having the necessary stereoscopic functionality. The work that is partially and directly related to this work as defined by the requirements is shown in Table 3.2 on page 41 and discussed in following paragraphs.

Video/audio conferencing systems can most often be used to share a desktop metaphor with a visualization application. Group-to-group conferencing systems (e.g. Access Grid [110], EVO [194]) provide an environment that brings multiple participants from multiple remote locations virtually close together (tele-presense) by using multicast, multiple cameras and microphones at each location. They are used for large cooperative projects or for special events, e.g. for the SARS outbreak in 2004 [198]. Such systems provide extensive functionality and require a relatively low bandwidth due to the frequent usage of multicast, but are complex in deployment and usage [276]. Standard person-to-person video conferencing systems (e.g. H.323-based systems [193]) provide a similar but usually inferior functionality [277, 278]. They are widely available by being deployed at conference rooms of almost every department. Both system types utilize high-performing video codecs that provide a decent quality with a low network load [279]. However, they are not primarily designed to be used for visualization and, thus, may require specific configurations at the server and client side, as for example to provide stereoscopic visualization, a desired image quality/resolution, and interactive access [108]. They usually require a specific software and/or hardware deployment. Even if some web-based conference systems with added software exist (e.g. EVO [194] or Portable Access Grid [280]), pure web browsers cannot fulfill all requirements of a video conferencing system yet and, thus, rely on added software [85].

General remote screen sharing applications were developed by many companies [95, 281–283] and are widely applied to access remote desktop metaphors [69], which might include a visualization application [284]. Such systems utilize various kinds of algorithms to reduce the data transfer [234, 285], as discussed in Section 8.4. Whereas early systems required a special software deployment at the client side, some have been developed that run within web browsers, but still rely on added software. Envision for example provides a pure web browser interface for most tasks, but requires a Virtual Network Computing (VNC) Java plugin for the interactive visualization itself [69]. In recent years, many long existing systems were extended by a browser-based access method [281, 282]. Other systems were specifically developed for the pure web-based access purpose [57, 58]. Guacamole [58] for example provides a VNC server access via a pure web browser for a single participant. Ischimura et al. [57] provides a highly scalable web-based

approach via a standard web server, which however is due to its architecture not highly interactive. ScreenLeap.com [96] provides a pure web-based remote viewing access to a desktop metaphor via a Java web-start application as server, but is not controllable from the client side. However, these applications are usually specialized for the remote access to the whole desktop metaphor. In order to be used with stereoscopic visualization applications (see Section 8.2) and a multitude of stereoscopic display devices, they would require special configurations and/or developments.

Special visualization streaming applications provide all the required functionality for specific use cases of projects and communities, which are not or not without special configurations possible by general screen sharing applications. They are used to access visualization applications in many scientific disciplines [75, 82, 286] and are sometimes optimized to transfer very high pixel amounts for tiled displays [208–210]. But similar to the general screen sharing applications, they mostly require specific software deployments at the client side. Only very few and the most recent projects provide a remote visualization via pure web browsers [56, 59, 61–65]. ParaViewWeb [59] is the remote visualization system of the ParaView server. Its client is either based on pure web browsers, Java, or Flash. Its web-based part is similar to CoWebViz's technique regarding the web-based access, but without the specific stereoscopic functionality (see Section 8.2 and 8.4). Another very recent development is Vitrall [65], which provides a visualization access via pure web browsers but also native software. Vitrall mainly provides remote and collaborative access to surface visualization via OpenSceneGraph [243], but is extendable. Vitrall was already tested with remote access to a one-view stereoscopic system. The other systems are less advanced and, due to their system design, provide a lower frame rate [56, 61–64]. None of the related work provides an automatic quality setting specific for each user's network condition as required to serve participants with low bandwidth connections (see Section 8.4) nor do they provide functionality for different stereoscopic display systems (see Section 8.2).

The technology that CoWebViz applies to transfer visualization to a web browser is the pushing of binary Joint Photographic Experts Group (JPEG) images via a persistent multipart Hypertext Transfer Protocol (HTTP) message, which is called motion JPEG (mJPEG). mJPEG was chosen, because: *1.* it is potentially the best performing choice (see Chapter 4), *2.* many web browsers support it for the playback of webcam video streams, and *3.* because of the positive experiences in the external project TelePresence Microscopy Collaboratoration (TPM) (see Section 6.3.1). TPM was used to stream video from an operating room to the audience at a conference room of the University of Chicago hospital[1]. This scenario is important, because of its technical similarity to remote visualization and its usage in the same environment as CoWebViz was planned for.

CoWebviz's feasibility to provide remote access to interactive volume visualization in real-time was shown by its usage in the virtual anatomy class (see Section 6.1.1) and associated comparative tests with a parallel running Access Grid deployment (Section 6.1.2). Real-time

[1]Personal communication with Nestor J. Zaluzec and John C. Alverdy, University of Chicago, IL, USA.

interactivity requires a timely presentation of a modified visualization after it was requested by the user (e.g. using a mouse and/or keyboard). Real-time interactivity highly depends on the specific network connection, the transfer compression, the specific application's implementation, and the utilized hardware. Such applications therefore have high optimization potentials. CoWebViz's approach to retain interactivity is an automatic quality adjustment algorithm, which is discussed in Section 8.4. This system cannot be compared directly to most web-based systems of the related work, because they are not freely available or utilize other visualization types. However, their implemented methods were compared analytically (see Section 4.1.2) and by black box frame rate measurements on the client side (see Section 7.2). The superiority of CoWebViz's approach was shown in both cases.

Hypertext Markup Language (HTML) 5 video streaming enables the utilization of high-end video codecs, as e.g. H.264, on pure web browsers. Using such codecs with efficient inter frame compression would seem to be a better choice than mJPEG. But the evaluation in Section 4.1.1 shows that HTML 5 currently results in a delayed visualization presentation. This is mainly caused by the buffering algorithm suggested by the HTML 5 specification [85], which is optimized for a continuous non-interrupting instead of an instantaneous presentation [254]. Thus, HTML 5 does not provide a reliable low presentation delay on the web browser, which currently makes this choice not usable for real-time visualization.

Streaming inter frame compressed video results in a continuous low bit-rate stream with a high frame rate (in movies about 30 frames per second (fps)). Sending an equivalent frame rate via single images would result in a very high bandwidth usage. But in order to reach such a high frame rate with an interactive visualization system, the user would need to modify the visualization continuously at a great pace without any time for viewing. In practice, visualization is modified gradually (modification phases) and then viewed for a period of time (viewing phase). This was also the case in the virtual anatomy class, where modification phases were usually shorter than viewing phases. There were even periods of up to 30 minutes without any visualization usage at all. But despite sending only modified images, a peak performance usage might lead to a high bandwidth load (see Chapter 7). Nevertheless, due to frequent viewing breaks without any bandwidth usage at all, the overall summarized network usage of sending single images is only a fraction of the short peaks of high network usage.

Another alternative to mJPEG is the continuous transfer of single Base64 compressed images [287]. These have the advantage of being trouble-free transferable on web-based systems [60]. They can be pushed to (e.g. via WebSockets [60]) and pulled from a web browser (HTTP requests or a Comet-style application as used by Guacamole [58]). The big disadvantage is that such images are about 30% larger than its binary counterpart (see Section 4.1.3), which makes this choice less efficient.

The best other alternative to mJPEG is the continuous pulling of single binary images from a web browser (see Section 4.1.2). This is the default technique used to load images of standard web pages, which makes this technique to be best supported on all web browsers (see Section

4.1). However, every image needs to be requested by a separate connection [89], which results in a higher transfer overhead and thus in a slower possible frame rate (see Section 4.1.2). In contrast, mJPEG just pushes a new image on the initially opened connection and thus has a lower network overhead. Many of the existing web-based visualization applications use this transfer technique, as e.g. ParaViewWeb [59].

The technique analysis in Section 4.1.1 shows that mJPEG is a favorable and well performing choice. However, it is currently not usable on all web browsers, because Internet Explorer and Opera do not support mJPEG by default. Providing interactive visualization with best performance and quality on all web browsers would therefore require to provide different methods. The need of providing fallback mechanisms is a big disadvantage of several web technologies, e.g. video streaming [221]. Regardless, a specific fallback mechanism without added software as substitute to mJPEG (Safari, Chrome and Firefox) could be pulling single JPEG images (Internet Explorer, Opera) as described by ParaViewWeb or, alternatively, the transfer via WebSockets [60]. The specific image transfer technique is independent of the stereoscopic presentation discussed in Section 8.2 and the automatic quality adjustment discussed in Section 8.4.

The requirement of using pure web browsers reduces the choice of image codecs to JPEG [257] and Portable Network Graphics (PNG) [256] (see Section 4.1.3). PNG is ideal to compress text or line drawings with hard edges and lossy JPEG to compress images with smooth transitions, to get low file sizes [288]. The typical volume visualization of the human body shows muscles, organs, and bones, each usually with smooth transitions. JPEG is therefore a good choice to compress the visualization. Another image format is JPEG 2000 [259], which utilizes wavelet compression instead of the discrete cosine transformation of JPEG, and results in a higher quality/file size ratio. Since this is ideal for remote visualization, it is used in many projects that utilize native applications. But since JPEG 2000 and other newer image compression formats, e.g. WebP [261] and H.264 intra frame compression [262], are not supported by default on any web browser, they can not be used.

8.2 Support of multiple participants at different locations with different stereoscopic systems simultaneously

The previous discussion in Section 8.1 highlights the importance of the easy accessibility of interactive visualization and discusses available remote visualization approaches and implementations using web browsers. The described pure web-based systems were mostly developed for a monoscopic usage. Stereoscopic remote visualization, in contrast, is less frequently used and mainly based on native applications, e.g. the special Access Grid-based deployment of the previous virtual class setup [108]. The general screen sharing applications (e.g. Guacamole [58]) might only

be able to provide a well performing interactive remote stereoscopic visualization with special configurations or additional developments. The specific web-based applications (e.g. ParaViewWeb [59]) are used together with a base application and, thus, are bound to the stereoscopic capabilities and visualization algorithms of their base application. Only Vitrall [65] provides a specific single stereoscopic content type. The simultaneous provision of multiple stereoscopic content types to serve different stereoscopic systems simultaneously would require additional development in each of the described cases. No literature was found that describes such an easy to use system for stereoscopic volume visualization.

CoWebViz fills this gap and provides the same ease of use for stereoscopic remote visualization on different stereoscopic display systems via a pure web browser as it does for monoscopic visualization. Its feasibility is shown by its usage in different practical scenarios for stereoscopic but also monoscopic visualization (see Section 6.1 to 6.3), which is underpinned by performance tests in Section 7.1.2 (see also discussion in Section 8.4). As described in Section 3.2, various stereoscopic display systems exist, which partially utilize differing content types. Thus, in order to provide ad-hoc usability in a remotely scattered group with different stereoscopic systems, multiple content types need to be provided simultaneously. CoWebViz provides three different stereoscopic content types (two-view, side-by-side and anaglyph), which can be used on different classes of stereoscopic display devices (projector-based setups, three-dimensional (3D) displays and standard two-dimensional (2D) displays). This is a range from standard desktop and tablet computers to large-scale projection setups.

The projector-based setup was used successfully during the anatomy classes. Due to its large presentation, this display technique provides a high degree of visual immersion.

The side-by-side stereoscopic visualization on a 3DTV was successfully used in multiple scenarios. At first it was used for several visualization demonstration sessions in a hospital conference room. But to be highlighted is the stereoscopic usage within an operating room to inform a plastic surgeon during reconstructive surgery, which shows the feasibility of CoWebViz's usage in a real practical medical scenario. The usage on 3DTVs is an important use case, because of the low cost of such TVs and their simple setup.

A content type that requires no stereoscopic hardware deployment at all, is anaglyph stereoscopic visualization, which only requires low-cost glasses. Its big disadvantage is the color shift caused by the colorized lenses, which, however, is still under research and can be considerable minimized [289, 290]. But its ubiquitous usability makes this technique still interesting for simple use cases [291]. Because of this simple accessibility, it was implemented to be used for self-directed learning on-campus, but was only tested in test sessions.

Monoscopic visualization was also used for different scenarios, e.g. an ad-hoc demonstration of CoWebViz to collaborators with a parallel teleconference and to provide researchers with interactive visualization rendered on a remote visualization cluster.

Auto stereoscopic systems are increasingly often used, because of their advantage of not requiring any personal gear (e.g. polarized glasses). Such systems, however, are currently not supported by CoWebViz, because they require a specific content type that has all stereoscopic views merged into a single image. This means that any two neighboring pixels or sub-pixels of an image presented on most auto stereoscopic 3DTVs represent another stereoscopic view. Such a merged image cannot be transferred with image compression and would require client side procession, which would entail further research.

Stereoscopic displays are not yet available as widely as standard monoscopic displays, which are provided with every computer workstation. But stereoscopic display devices are increasingly often deployed at special visualization centers [161, 292], conference rooms/lecture halls [293], or directly at the demanding departments [236]. An important factor that influences the availability of stereoscopic devices is the movie industries' initiative of producing more stereoscopic movies in the past years, which entailed an increased production and wider availability of stereoscopic devices. Whereas the hardware is more often available, the specific software for interactive visualization still needs to be deployed, which is eased by CoWebViz.

CoWebViz can be used simultaneously by multiple participants from different remote locations. The scalability analysis (see Section 7.1.3) shows that the system has a good performance with up to the maximal tested six simultaneously accessing participants. It is further shown that the server's Central Processing Unit (CPU) usage and network throughput increases with every additional user. Thus, the maximal possible number of simultaneously accessing users highly depends on the environment in which the system is used. But it is expected that most use cases are not requiring more than about four simultaneously accessing groups, which is discussed by the following examples.

The proof of concept conduction in the class involved two participating groups, each with a two-view stereoscopic visualization. While this could be slightly increased to provide visualization to more than two groups, a real contrast would be the usage for self-directed learning by students. This could potentially require the synchronous access by 10 to 100 students, depending on the class size. This scenario, however, would not be viable, since only one student would be able to modify the visualization at any time. Providing each student with a freely controllable visualization would require a separate server instance for every student. This, however, would be a whole another use case with only one participant per instance. It is therefore estimated that a small group of up to four remote participants is realistic, at least for education use cases. Four participants with a two-view stereoscopic visualization (each view 1024x768 and a JPEG quality of 85 and a peak frame rate of 15 fps) would result in a peak network usage of about 25 Mbits. This is high, but currently viable with a high-performing network on the server side. A possible solution to provide visualization to more participants would be the usage of multicast networks. Multicast however requires special network arrangements between the server and each client, which would reduce the ad-hoc usability. An alternative that does not require any specific client adjustments

but more effort on the server side would be the addition of load-balancing servers. This issue, however, was not a central topic of this work, because the proof of concept conduction did not require large amounts of simultaneously accessing users. The central issue rather was the network throughput optimization to each single client as discussed in Section 8.4.

The sharing of monoscopic visualization is not new compared to the other web-based and non web-based systems. However, while developing such a system, the pattern of how remote users interact with each other needs to be considered. Interactive systems with multiple users have been used with different interaction methods. Prominent methods are to give only one person the right to modify and the others only the right to view [270], to give all participants the right to modify after a request [271], and to let every participant freely modify after oral coordination [160]. The idea behind the proof of concept was to provide a most simple system with least required user knowledge. Thus, CoWebViz's method is to inform the participants about the current state and to give everyone the instantaneous right to just modify, if nobody else is modifying. Because of the few direct users in the proof of concept conduction, these methods could not be evaluated against each other. But this method seems to be as natural as possible without any system restrains. It requires mutual oral coordination, which is always necessary for synchronous collaborative working and can be done via video and/or teleconferencing. No barriers regarding this method were observed during the proof of concept conduction.

CoWebViz's stereoscopic and collaborative functionality can be seen as a stereoscopic remote visualization service, which is a mix of a stereoscopic player [129] (taking stereoscopic input formats and providing desired output formats) and a video-streaming server (taking input and streaming to multiple participants). By requiring no special software deployment, CoWebViz allows for various advanced use cases with the utilization of interactive 3D and stereoscopic visualization on remote computers and multiple participants.

8.3 Generic support of any existing and future visualization application

The provision of different stereoscopic content types at the client side as discussed in the previous section is necessary to include various display devices and, thus, participants. In contrast, the generic integration of different visualization applications and stereoscopic content types at the server side is necessary to be independent of a single visualization type and application. In the first place, this is important for keeping the development overhead low via functional separation and to provide a quick access to existing visualization applications. In the second place, it has the advantage of providing remote access to existing applications, which are already familiar to the user. The familiarity of an application and/or application logic potentially reduces the time that needs to be spent for learning because it keeps the knowledge that a user requires low.

The generic integration of whole desktop metaphors and specific monoscopic visualization applications is already widely used on native and pure web-based remote visualization applications [58, 69, 95]. It is also already possible to integrate some single stereoscopic content types via existing applications (e.g. half-wide side-by-side for 3DTVs) handled as monoscopic visualization. However, the simultaneous usage of different display devices and more complex stereoscopic visualization formats (e.g. two-view stereoscopy) requires a specific handling and processing of the stereoscopic input content. Existing remote visualization applications (see Table 3.2 on page 41) would require substantial development in order to achieve this, while keeping the client side requirements low.

CoWebViz's architecture shows the feasibility of integrating an existing visualization application with a two-view side-by-side stereoscopy (see Section 6.1.1 and 5.5). The effort, required to integrate another stereoscopic format highly depends on the specific format (e.g. very simple is top-to-bottom and interlaced), but is basically enabled by the architecture. While CoWebViz currently integrates a stereoscopic application with two views, some stereoscopic display devices require more than two views to provide a higher visual immersion, e.g. a partial CAVE with three or four walls, a floor, and/or a ceiling (introduced in Section 3.2.2.4). The integration of more than two views is possible, but would require few straightforward modifications. The integration of monoscopic visualization is a subset of the stereoscopic streaming with only one instead of two views.

The generic integration is solved by using screen scraping, which is already used by other remote applications [58, 95, 96]. It uses an operating system's window system (e.g. the Linux X-Server [103]) as an intermediate layer and is therefore generic towards any existing and future application that runs on this window system. The proof of concept conduction shows that this approach is simple to use and is capable of providing an adequate performance for an interactive usage (see test in Chapter 7). A disadvantage of this approach is that the server always requires a window system. This causes technical complexity in case of servers without a connected display, which is common on visualization clusters. Other methods were described that allow the integration of visualization by utilizing knowledge about the visualization application's interior design. TechViz for example integrates visualization generically for OpenGL applications, by accessing the OpenGL data (e.g. vertices) on a low operating system level to modify the visualization application's rendering to custom needs [294]. This however is only possible with a specific application's insight (e.g. the application is OpenGL based). In contrast, volume visualization is often developed specifically for each application with CPU or Graphics Processing Unit (GPU)-based rendering and, thus, can not be integrated generically. The most generic access to different kinds of visualization applications therefore is to screen scrape, where the word "generic" relates to the window system.

Direct connections to specific visualization applications are not generic anymore, but might enable a more efficient connection as well as more technical possibilities. CoWebViz generally

supports such cases by providing an architecture that eases the development of specific modules to integrate visualization libraries (e.g. VTK [295]) or external visualization applications via shared memory (see Section 5.1.1), as it was used by the previous classroom's setup [188].

*Med*ical *Vol*ume *Vi*sual*iz*ation (MedVolViz) is the stereoscopic visualization application that was used for the proof of concept conduction. It has no GUI in order to emphasize on the visualization itself. A new user only requires to learn few key commands to access all necessary functionality for a quick and full usability. But, due to the need of remembering these key commands, which might complicate a usage, we provided a GUI on top of these commands. This was simply done on the layer on which CoWebViz works: the event transfer. Buttons were created on the client side that release the specified events on button press (see Section 5.6). This functionality was further extended by the session recording/playback functionality (see Section 5.6). As already discussed in Section 8.1, both were not used besides tests, but could potentially be of value in the described class environment. In contrast, an often used extension is the pure control view that only allows to modify the visualization. Such is necessary to reduce the bandwidth usage in environments that already have a large projection of the visualization, e.g. the class. This view was only used by the lecturer and the teaching assistants but could be used by any participant of an audience, which allows for new scenarios by including all participants instantaneously and directly in the visualization usage via pure web-browser.

An obstacle with the generic integration of many visualization applications is the GUI. While the quality of the application window's visualization part can be reduced without a direct usage limitation (apart from not being able to see all details), the GUI needs to be available in high quality for readability. CoWebViz is developed to screen scrape the whole or a part of a window in order to allow the remote access of pure visualization. While the first method is ad-hoc usable without specific additions, the second method requires to add specific client side additions (one time only) in order to make the hidden functionality available (see Section 5.1.1). Such a solution is the addition of GUI controls or key commands at the client side that provide control commands, necessary to access the hidden GUI (as discussed in the previous paragraph). CoWebViz could easily be configured specifically for a new application with hidden GUI. But an automated and much broader solution is already proposed by Lamberti et al. [207], who provide a framework that analyses the GUI as image and automatically identifies control elements (e.g. buttons), which are then linkable to a new remote client GUI.

In contrast to the other web-based remote visualization applications, CoWebViz provides additional functionality specifically for educational scenarios (see Section 5.6). The session scripting (record, organization in lectures, playback) might ease the usage for the lecturer and people who have never used the system before [296]. Other visualization input types (e.g. a movie without sound or a webcam stream) can also be streamed, as for instance to play a recorded visualiza-

tion session or an animation from remote. A screenshot taking functionality was added to allow students or any other participant to take a stereoscopic (anaglyph) or monoscopic image for personal documentation. These additions, however, have only been developed and tested, but not yet evaluated in the educational scenario.

Hospital Information Systems (HISs) are being more often developed as web-based systems [42]. But their imaging functionality is mostly being integrated via added software [297–299]. This development choice is presumably chosen because of its simple way to get comprehensive functionalities. However, the native integration of pure web-based visualization into a pure web-based application seems natural and the technical possibilities provided by pure web browsers are continuously increasing [85]. It would provide for a seamless usability by not requiring differing techniques and therefore again reduce the user involvement. The feasibility of such an integration on the client side was already successfully shown with CoWebViz (see Section 6.3.1). In the first scenario, this was done via the direct integration of the pure monoscopic visualization (without interactivity) into the TPM [203] and in the second scenario by integrating the whole CoWebViz functionality into a large visualization cluster environment with a web-based Grid-computing scheduler[2].

8.4 Automatic quality adjustment during the runtime to optimize the balance between performance and quality on a given network condition

CoWebViz's performance optimization is a direct result of the additional development following the first practical usage in the virtual anatomy class. These developments are the parallel architecture, the automatic quality adjustment algorithm and the event transfer optimization. CoWebViz's automatic quality algorithm continuously adjusts the visualization quality specifically for every user's bandwidth connection (see Section 7.1). It retains a minimum interactivity on different very low to high bandwidth conditions (see Figure 5.4 on page 62). Such an adaption is especially necessary to allow a real-time visualization usage in collaborative settings with multiple client systems connected via standard unconstrained networks (e.g. the Internet).

Movie and video streaming services cover such varying environments via client side buffering and by providing multiple versions of a video stream, each for a specific class of bandwidth environments. They retain the frame rate by utilizing state of the art video codecs and compression towards defined bitrates, each allowing for a fluent playback with according best quality on the correlating network. Frame rate is a common descriptive metric in computer graphics and video techniques to describe performance of video-like techniques (continuous images), which gives a

[2]Personal communication with Nestor J. Zaluzec (2011) and Tom Uram (2012), Argonne National Laboratories, IL, USA.

hint to the fluency perceived by a user. In contrast to videos with consecutively changing images, visualization is only changed after requested and only needs to be transferred in these cases. A descriptive metric for interactive visualization could therefore also be the round-trip time from giving an event to seeing the resulting visualization. This event-to-image-time should be as short as possible (usually few milliseconds [7]). Frame rate, however, is a common and well known metric and multiple event-to-image-times, considered over time, result again in the frame rate. But a maximum frame rate is only reached with a continuous modification and, thus, is only a metric of maximum performance, as presented in the performance tests in Chapter 7. Because of these reasons, the modification and viewing phases (see Section 8.1) need to be considered separately while discussing algorithms to improve the access to highly interactive remote visualization.

CoWebViz's automatic quality adjustment algorithm calculates an ideal (maximal) image file size for each specific network connection in order to provide a desired frame rate during a visualization modification phase. This calculation is basically the division of the available bandwidth by the amount of the desired frame rate. The ideal image file size provides a reference value towards which the image quality of each image is continuously adapted. Maximizing the image quality after a specified time of no modification results in an optimal visualization presentation for the viewing phase.

While this adaption includes a tradeoff between a decreased JPEG quality/resolution and performance, the theoretical best-case scenario is to continuously send every modified image with high frame rate and a constant high quality. However, this case is only possible in real-time on high-performing networks (16.7 fps with a quality of 80 on 90 Megabit per second (Mbps) and 15.5 fps with a quality of 80 on 11 Mbps). On a low bandwidth network of 1 Mbps it results in only 2.7 fps. Thus, it is necessary to provide specific quality settings for any participant instead of a static quality (see Section 7.1.1 B vs. C).

Most related web-based systems transfer each single modified image with a static predefined quality setting. Single images can be reduced by manipulating many specific parameters, as e.g. in case of JPEG the JPEG quality and image resolution, but also the JPEG quantization matrix [269]. But the usage of a statically defined quality results in decreasing frame rates with decreasing bandwidth conditions. Of the web-based related work, only ParaViewWeb also modified the quality according to the two phases (e.g. maximizing the quality for viewing). ParaViewWeb [59] provides the most advanced approach of the related work by using a predefined JPEG quality of 50 for every connection type during the modification phase and 100 during the viewing phase. But as shown in Section 7.1.1 and 7.2, a static quality setting does not account for different bandwidth conditions for any client.

A theoretical maximum performance of CoWebViz is shown by transferring every image that CoWebViz processes. This setup resulted in the transfer of as much as 76 frames per second on a very high bandwidth connection of 90 Mbps and a static JPEG quality of 80 (see Section 7.1.1 A).

While the base application is not rendering on such a high pace, these frame rates include lots of duplicate frames and, subsequently, a high network overhead (see Section 7.1.1 A vs. B).

Compared to the related work, CoWebViz's approach is a very performant method (see Section 7.1.1 and 7.2). General non-web-based remote screen sharing systems utilize all kinds of algorithms in order to prevent and/or reduce the transfer of image data, e.g. moving image parts on the client side while transferring only the missing data in VNC. Such methods are very beneficial to access standard desktop metaphors, including multiple windows with standard GUI elements and little fast changing visualization. But as shown in Section 7.2 this approach seems not to be as sufficient for highly interactive visualization with content where almost any pixel changes between any two consecutive modified images.

CoWebViz, in contrast to the related web-based work, also includes the stereoscopic visualization with multiple views into the quality adjustment (see Section 5.2.2). This is done by reducing the ideal image file size by the number of opened views for each participant's network connection. As described in the Section 5.2.2, this results again in a quality reduction for each separate view of a two-view stereoscopic visualization, but is the straightforward continuance of the algorithm. The initiation of multiple views occurs while using e.g. a two-view stereoscopic visualization as well as the parallel usage of two identical views. An anaglyph stereoscopic visualization, in contrast, is equal to the monoscopic visualization with a higher server load for the creation of the anaglyph visualization. A side-by-side stereoscopic visualization for state of the art 3DTVs is comparable to monoscopic visualization with a high resolution.

CoWebViz's parallel executing architecture is a consequence of the requirement to provide a specific image quality for every connected client. Serving each client with a specific quality is necessary to optimally balance the visualization quality with the available bandwidth that is preset and specific between the server and each connected client. The architecture allows to send multiple streams, each with a specific configuration to one or multiple other participants simultaneously. During a viewing phase the system requires very few resources, including no data transfer and a low server CPU usage of 9.6% to 40.5% for 1 to 6 participants, respectively, but at the same time having a maximum quality. During a modification phase, these values change. As described in Section 7.1.3, the automatic quality algorithm retains the frame rate and the quality of a single user session, independently of the amount of participants (from 1 to 6). However, on the server side the CPU and bandwidth usage increases linear from 21.8% to 82.5% on 1 to 6 clients, respectively. This causes the existence of an upper limit of simultaneously accessing participants, depending on the available server side CPU and network conditions. However, high numbers of simultaneous users are not feasible, as already discussed in Section 8.3. But utilizing the system with about 6 participants is also no performance issue for most state of the art servers, because it only requires about 1-2 cores of a standard computer and a network of 15 Mbps, which is usually available at universities. The procession of image scaling and compression could further be sped up by using General-purpose Computing on a Graphics Processing Unit (e.g. CUDA [300]).

Whereas the image transfer is the largest data transfer of the event-to-image time, the other crucial transfer is the control event transfer from the client to the server. Web browsers were initially not developed for interactive real-time systems. The common technique to transfer data from the client to the server is therefore a Representational State Transfer (REST)-style interface, which is still used by most applications. It requires the establishment of a new connection for any event, which results in unnecessary network overhead. Since the visualization on a remote server will only be modified after the corresponding events reached the server, the event transfer needs to be as fast as possible. WebSockets is still a new but very promising development allowing for faster data transfers and thus is a more appropriate mechanism to transfer data in real-time. This is shown by CoWebViz's WebSockets implementation which is almost 50% faster than the REST-style interface (see Section 4.2). In the related work, the event transfer technique is not always as clearly described as the visualization transfer. It is very likely that most of the systems still use a REST-style interface and only some already use WebSockets (e.g. Vitrall [65]). CoWebViz mainly utilizes WebSockets. But, since WebSockets are still in development and REST-style interfaces seemed to be more robust against network issues and available on every web browser, they are very appropriate as fallback mechanism.

9 Conclusion

This dissertation's contribution is a proof of concept conduction that shows the feasibility of conjointly using interactive, stereoscopic, and remote collaborative visualization for anatomical education via pure web browser based clients.

The system (CoWebViz) described in Chapter 5 is one of the first systems that provides fluid interactive visualization in real-time via pure web browsers. Only recent developments made such extensive applications on web browsers without added software possible. To the best of our knowledge, the combination of supporting all requirements stated in the introduction Section 1.4 is currently unique, which allows for extensive use cases.

Stereoscopic and real-time interactive medical volume visualization for collaborative usage is not bound to special software deployments. Due to the system's stereoscopic emphasize described in Section 5.5, it is simultaneously usable with different stereoscopic content types to serve different stereoscopic setups at different locations. This is made possible by the separate treatment of each stereoscopic input view on the server side, as discussed for the generic integration in Section 8.3. Chapter 6 shows that not only setups with a single stream (half-wide side-by-side, anaglyph), but also setups with multiple streams can be served (two-projector solutions). To the best of our knowledge, this support of various stereoscopic systems via such a lightweight client is currently unique.

The quality and performance optimization described in Section 5.2.2 is a newly combined approach to provide fluid interactivity for each simultaneously accessing user by adjusting towards each user's specific bandwidth connection. As shown in Chapter 7, this approach is superior to the tested existing applications, including a native application.

Web browsers are known for their familiarity and web-based applications for their simple deployability. CoWebViz's practical usage described in Chapter 6 shows that the usage of web browsers supports the reduction of time-consuming steps, necessary before and during a system usage. To the best of our knowledge, this is the first time that such a pure web-based system has been used for such an usually resource intensive stereoscopic and collaborative setup in medical education.

The interest of its usage across the boundaries of anatomical education is shown by other applications described in Section 6.2 and 6.3. To emphasize is CoWebViz's current usage to inform surgeons during reconstructive surgery in the operating room, which shows the interest and feasibility of its usage in practical medicine. Its usage as a visualization service in a data center for remote scientists shows its relevance for the field of high-performance scientific visualization.

List of Abbreviations

2D two-dimensional.

3D three-dimensional.

4CIF 4x Common Intermediate Format.

ACM DL ACM digital library.

CAD Computer-aided Design.

CAVE Cave Automatic Virtual Environment.

CIF Common Intermediate Format.

CoWebViz *Co*llaborative *Web*-based *Vi*sualiz*z*ation.

CPU Central Processing Unit.

CSS Cascading Style Sheets.

CT Computed Tomography.

DICOM Digital Imaging and Communications in Medicine.

DTI Diffusion Tensor Imaging.

eps events per second.

fMRI Functional Magnetic Resonance Imaging.

fps frames per second.

GPGPU General-purpose Computing on a Graphics Processing Unit.

GPU Graphics Processing Unit.

GUI Graphical User Interface.

HIS Hospital Information System.

HL7 Health Level Seven.

HTML Hypertext Markup Language.

HTTP Hypertext Transfer Protocol.

IEEE DL IEEE Xplore digital library.

IHE Integrating the Healthcare Enterprise.

JIF JPEG file interchange format.

JPEG Joint Photographic Experts Group.

Mbps Megabit per second.

MedVolViz *Med*ical *Vol*ume *Vi*sualiz*z*ation.

MIME Multipurpose Internet Mail Extension.

mJPEG motion JPEG.

MRI Magnetic Resonance Imaging.

OSI Open Systems Interconnection Model.

PACS Picture Archiving and Communication System.

pJPEG pulling JPEG.

PNG Portable Network Graphics.

POI Point of Interest.

REST Representational State Transfer.

SVG Scalable Vector Graphics.

TPM TelePresence Microscopy Collaboratoration.

URL Uniform Resource Locator.

VNC Virtual Network Computing.

VRML Virtual Reality Modeling Language.

WADO Web Access to DICOM Persistent Objects.

X3D Extensible 3D.

XDS Cross-Enterprise Document Sharing.

List of Figures

List of Tables

List of Listings

Bibliography

[1] Munzner T, Johnson C, Moorhead R, Pfister H, Rheingans P, and Yoo TS. NIH-NSF visualization research challenges report summary. *IEEE Comput Graph Appl*, 26(2):20–4, 2006.

[2] Nußbeck G. Taxonomy-based assessment of personal health monitoring in ambient assisted living. In *Ambient Assisted Living*, pages 199–211. Springer, 2012.

[3] Huang HK. *PACS and Imaging Informatics: Basic Principles and Applications*. Wiley-Blackwell, 2 edition, 1 2010. ISBN 9780470373729.

[4] Yi MY and Hwang Y. Predicting the use of web-based information systems: self-efficacy, enjoyment, learning goal orientation, and the technology acceptance model. *International Journal of Human-Computer Studies*, 59(4):431–449, 2003.

[5] Yi JS, ah Kang Y, Stasko JT, and Jacko JA. Toward a deeper understanding of the role of interaction in information visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1224–1231, 2007.

[6] Silverstein JC, Parsad NM, and Tsirline V. Automatic perceptual color map generation for realistic volume visualization. *J Biomed Inform*, 41(6):927–35, Dec 2008.

[7] Schulzrinne H, Casner S, Frederick R, and Jacobson V. RTP: A Transport Protocol for Real-Time Applications. In *Request for Comments*, number 3550 in Request for Comments. Internet Engineering Task Force, IETF, June 2003.

[8] Price A, Subbarao M, and Wyatt R. Two Eyes, 3D: Stereoscopic Design Principles. In *American Astronomical Society Meeting Abstracts*, volume 221 of *American Astronomical Society Meeting Abstracts*, page 302.01. January 2013.

[9] Laha B, Sensharma K, Schiffbauer JD, and Bowman DA. Effects of immersion on visual analysis of volume data. *IEEE Trans Vis Comput Graph*, 18(4):597–606, Apr 2012. doi: 10.1109/TVCG.2012.42.

[10] Reichelt S, Häussler R, Fütterer G, and Leister N. Depth cues in human visual perception and their realization in 3d displays. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 7690, page 10. 2010. ISSN 0277-786X.

[11] Ishikawa N, Watanabe G, Iino K, Tomita S, Yamaguchi S, Higashidani K, Kawachi K, and Inaki N. Robotic internal thoracic artery harvesting. *Surg Today*, 37(11):944–6, 2007. doi:10.1007/s00595-007-3542-4.

[12] Wang J, Wang X, Xie M, He L, Lv Q, and Wang L. Clinical value of stereoscopic three-dimensional echocardiography in assessment of atrial septal defects: feasibility and efficiency. *J Huazhong Univ Sci Technolog Med Sci*, 29(6):791–4, Dec 2009. doi: 10.1007/s11596-009-0624-0.

[13] Brown PM, Hamilton NM, and Denison AR. A novel 3d stereoscopic anatomy tutorial. *The Clinical Teacher*, 9(1):50–53, 2012. ISSN 1743-498X. doi:10.1111/j.1743-498X.2011.00488.x.

[14] Sielhorst T, Bichlmeier C, Heining S, and Navab N. Depth perception–a major issue in medical ar: evaluation study by twenty surgeons. *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2006*, pages 364–372, 2006.

[15] Fraser JF, Allen B, Anand VK, and Schwartz TH. Three-dimensional neurostereoendoscopy: subjective and objective comparison to 2d. *Minim Invasive Neurosurg*, 52(1):25–31, Feb 2009. doi:10.1055/s-0028-1104567.

[16] Isenberg P, Elmqvist N, Scholtz J, Cernea D, Ma KL, and Hagen H. Collaborative visualization: definition, challenges, and research agenda. *Information Visualization*, 10(4):310–326, 2011.

[17] Lievens F and Jordanova M. Is there a contradiction between telemedicine and business? *Journal of telemedicine and telecare*, 10(suppl 1):71–74, 2004.

[18] Helck A, Matzko M, Trumm C, Grosse C, Piltz S, Reiser M, and Ertl-Wagner B. Interdisciplinary expert consultation via a teleradiology platform-influence on therapeutic decision-making and patient referral rates to an academic tertiary care center. In *RöFo-Fortschritte auf dem Gebiet der Röntgenstrahlen und der bildgebenden Verfahren*, volume 181, pages 1180–1184. Georg Thieme Verlag, 2009.

[19] Cardoen B, Demeulemeester E, and Beliën J. Operating room planning and scheduling: A literature review. *European Journal of Operational Research*, 201(3):921–932, 2010.

[20] Gering DT, Nabavi A, Kikinis R, Hata N, O'Donnell LJ, Grimson WE, Jolesz FA, Black PM, and Wells WM 3rd. An integrated visualization system for surgical planning and guidance using image fusion and an open mr. *J Magn Reson Imaging*, 13(6):967–75, Jun 2001.

[21] Mlyniec P, Jerald J, Yoganandan A, Seagull FJ, Toledo F, and Schultheis U. imedic: a two-handed immersive medical environment for distributed interactive consultation. *Medicine Meets Virtual Reality 18: NextMed*, 163:372, 2011.

[22] Enzenhofer M, Bludau HB, Komm N, Wild B, Mueller K, Herzog W, and Hochlehnert A. Improvement of the educational process by computer-based visualization of procedures: Randomized controlled trial. *Journal of Medical Internet Research*, 6(2), 2004.

[23] Craig P, Wozniak H, Hyde S, and Burn D. Student use of web based lecture technologies in blended learning: Do these reflect study patterns. In *ASCILITE. Same places, different spaces*. 2009.

[24] Wiecha JM, Gramling R, Joachim P, and Vanderschmidt H. Collaborative e-learning using streaming video and asynchronous discussion boards to teach the cognitive foundation of medical interviewing: a case study. *Journal of medical Internet research*, 5(2), 2003.

[25] Cyber-Anatomy Corp. Cyber-anatomy med. URL `http://www.cyber-anatomy.com`, Last Accessed: 14. June 2013.

[26] O'Byrne PJ, Patry A, and Carnegie JA. The development of interactive online learning tools for the study of anatomy. *Med Teach*, 30(8):e260–71, 2008. doi:10.1080/01421590802232818.

[27] Han J, Kim J, Lee D, Lee J, Kim J, Park K, and Kang H. Three-dimensional visualization of human vocal organ on the web. *J Digit Imaging*, 15 Suppl 1:264–6, 2002. doi:10.1007/s10278-002-5050-9.

[28] Silverstein JC, Amine M, Dech F, Jurek P, Pires I, Tsirline V, and Walsh C. Web-based viewer for systematic combination of anatomy and nomenclature. *Stud Health Technol Inform*, 119:518–22, 2006.

[29] Silén C, Wirell S, Kvist J, Nylander E, and Smedby O. Advanced 3d visualization in student-centred medical education. *Med Teach*, 30(5):e115–24, Jun 2008. doi:10.1080/01421590801932228.

[30] Petersson H, Sinkvist D, Wang C, and Smedby O. Web-based interactive 3d visualization as a tool for improved anatomy learning. *Anat Sci Educ*, 2(2):61–8, Mar 2009. doi:10.1002/ase.76.

[31] Crossingham JL, Jenkinson J, Woolridge N, Gallinger S, Tait GA, and Moulton CAE. Interpreting three-dimensional structures from two-dimensional images: a web-based interactive 3d teaching model of surgical liver anatomy. *HPB*, 11(6):523–528, 2009.

[32] Temkin B, Acosta E, Hatfield P, Onal E, and Tong A. Web-based three-dimensional virtual body structures: W3d-vbs. *J Am Med Inform Assoc*, 9(5):425–36, 2002.

[33] Birr S, Mönch J, Sommerfeld D, and Preim B. A novel Real-Time Web3D Surgical Teaching Tool based on WebGL. In *Bildverarbeitung für die Medizin (BVM)*, page to appear. 2012.

[34] Qualter J, Sculli F, Oliker A, Napier Z, Lee S, Garcia J, Frenkel S, Harnik V, and Triola M. The biodigital human: a web-based 3d platform for medical visualization and education. *Stud Health Technol Inform*, 173:359–61, 2012.

[35] Dev P, Srivastava S, and Senger S. Collaborative learning using internet2 and remote collections of stereo dissection images. *Clin Anat*, 19(3):275–83, Apr 2006. doi:10.1002/ca.20313.

[36] Silverstein JC, Walsh C, Dech F, Olson E, Papka ME, Parsad N, and Stevens R. Immersive virtual anatomy course using a cluster of volume visualization machines and passive stereo. *Stud Health Technol Inform*, 125:439–44, 2007.

[37] John NW. The impact of web3d technologies on medical education and training. *Computers & Education*, 49(1):19–31, August 2007.

[38] Montagnat J, Jouvenot D, Pera C, Frohner A, Kunszt P, Koblitz B, Santos N, and Loomis C. Bridging clinical information systems and grid middleware: a medical data manager. *Stud Health Technol Inform*, 120:14–24, 2006.

[39] Germain C, Breton V, Clarysse P, Gaudeau Y, Glatard T, Jeannot E, Legré Y, Loomis C, Magnin I, Montagnat J, Moureaux JM, Osorio A, Pennec X, and Texier R. Grid-enabling medical image analysis. *J Clin Monit Comput*, 19(4-5):339–49, Oct 2005. doi:10.1007/s10877-005-0679-9.

[40] Wu J, Siewert R, Specovius S, Löhnhardt B, Grütz R, Dickmann F, Brandt A, Scheel M, Oswald D, and Krefting D. Web-based interactive visualization of grid-enabled neuroimaging applications. *Stud Health Technol Inform*, 175:107, 2012.

[41] Kunihiko N, Toru H, Kenji O, Bisser R, Toru T, and Kazufumi K. Volume rendering using grid computing for large-scale volume data. In *Computer-Aided Design and Computer Graphics, 2009. CAD/Graphics' 09. 11th IEEE International Conference on*, pages 470–475. IEEE, 2009.

[42] Huang HK. From PACS to web-based ePR system with image distribution for enterprise-level filmless healthcare delivery. *Radiol Phys Technol*, 4(2):91–108, Jul 2011. doi:10.1007/s12194-011-0122-5.

[43] Liu D, Hua KA, and Yu N. Enable web-based interactive applications of high-resolution 3d medical image data. In *IEEE International Conference on Multimedia and Expo*, pages 679–682. IEEE, 2007.

[44] Helgeson JW. *The Software Audit Guide*. ASQ Quality Press, 2009. ISBN 9780873897730.

[45] National Center for Biotechnology Information (NCBI). PubMed.gov – US National Library of Medicine National Institutes of Health. URL `http://www.ncbi.nlm.nih.gov/pubmed/`, Last Accessed: 14. June 2013.

[46] Google. Google Scholar. URL `http://scholar.google.com`, Last Accessed: 14. June 2013.

[47] National Center for Biotechnology Information (NCBI). Mesh (medical subject headings). URL `http://www.nlm.nih.gov/mesh`, Last Accessed: 14. June 2013.

[48] IEEE. IEEE Xplore Digital Library. URL `http://ieeexplore.ieee.org/Xplore`, Last Accessed: 14. June 2013.

[49] ACM. ACM Digital Library. URL `http://dl.acm.org/`, Last Accessed: 14. June 2013.

[50] Rohrer RM and Swing E. Web-based information visualization. *Computer Graphics and Applications, IEEE*, 17(4):52–59, 1997.

[51] Kaspar M, Parsad NM, and Silverstein JC. An optimized web-based approach for collaborative stereoscopic medical visualization. *J Am Med Inform Assoc*, 0:1–9, 2012.

[52] Kaspar M, Parsad NM, and Silverstein JC. CoWebViz: interactive collaborative sharing of 3D stereoscopic visualization among browsers with no added software. In *Proceedings of the 1st ACM International Health Informatics Symposium*, IHI '10, pages 809–816. ACM, New York, NY, USA, 2010. ISBN 978-1-4503-0030-8. doi:10.1145/1882992.1883113.

[53] Kaspar M, Dech F, Parsad NM, and Silverstein JC. Web-based stereoscopic visualization for the global anatomy classroom. *Studies in Health Technology and Informatics*, 163:264–270, 2011. ISSN 0926-9630.

[54] Zhu M and Yadav NK. A distributed system for parallel simulations. In *Global Telecommunications Conference*, pages 1–5. IEEE, 2008. ISBN 978-1-4244-2324-8. doi: 10.1109/GLOCOM.2008.ECP.335.

[55] Mouton C and Grimstead I. Collaborative visualization current systems and future trends. In *Proceedings of the 16th International Conference on 3D Web Technology*, pages 101–110. ACM, 2011.

[56] Zhao Y, Hu C, Huang Y, and Ma D. Collaborative visualization of large scale datasets using web services. In *Proceedings of the Second International Conference on Internet and Web Applications and Services*, pages 62–. IEEE Computer Society, Washington, DC, USA, 2007. ISBN 0-7695-2844-9. doi:10.1109/ICIW.2007.22.

[57] Ichimura S and Matsushita Y. Lightweight desktop-sharing system for web browsers. In *Information Technology and Applications, 2005. ICITA 2005. Third International Conference on*, volume 2, pages 136–141. IEEE, 2005.

[58] Gedda R. Need desktop access over the web? try some guacamole, 2010. URL `http://www.techworld.com.au/article/345260/need_desktop_access_over_web_try_some_guacamole/`, Last Accessed: 14. June 2013.

[59] Jourdain S, Ayachit U, and Geveci B. Paraviewweb, a web framework for 3d visualization and data processing. *International Journal of Computer Information Systems and Industrial Management Applications*, 3:870–877, 2011.

[60] Wessels A, Purvis M, Jackson J, and Rahman SS. Remote data visualization through websockets. In *Information Technology: New Generations (ITNG), 2011 Eighth International Conference on*, pages 1050–1051. IEEE, 2011.

[61] Mc Lane JC, Czech WW, Yuen DA, Knox MR, Greensky JB, Kameyama MC, Wheeler VM, Panday R, and Senshu H. Ubiquitous interactive visualization of 3-D mantle convection through web applications using java. In *Proceedings of the 4th International Symposium on Advances in Visual Computing, Part II*, ISVC '08, pages 1011–1021. Springer-Verlag, Berlin, Heidelberg, 2008. ISBN 978-3-540-89645-6. doi:10.1007/978-3-540-89646-3_101.

[62] Greensky JBSG, Czech WW, Yuen DA, Knox MR, Damon MR, Chen SS, and Kameyama MC. Ubiquitous interactive visualization of 3d mantle convection using a web-portal with java and ajax framework. *Visual Geosciences*, 2008.

[63] McLane JC, Czech WW, Yuen DA, Knox MR, Wang S, Greensky JBS, and Sevre EOD. Ubiquitous interactive visualization of large-scale simulations in geosciences over a java-based web-portal. *Concurr Comput : Pract Exper*, 22(12):1750–1773, August 2010. ISSN 1532-0626. doi:10.1002/cpe.v22:12.

[64] Zhou Y, Weiss RM, McArthur E, Sanchez D, Yao X, Yuen D, Knox MR, and Czech WW. Webviz: A web-based collaborative interactive visualization system for large-scale data sets. In *AGU Fall Meeting Abstracts*, volume 1, page 1354. 2010.

[65] Śniegowski P, Błażewicz M, Grzelachowski G, Kuczyński T, Kurowski K, and Ludwiczak B. Vitrall: web-based distributed visualization system for creation of collaborative working environments. In *Proceedings of the 9th international conference on Parallel Processing and Applied Mathematics - Volume Part I*, PPAM'11, pages 337–346. Springer-Verlag, Berlin, Heidelberg, 2012. ISBN 978-3-642-31463-6. doi:10.1007/978-3-642-31464-3_34.

[66] Suwelack S, Maier S, Unterhinninghofen R, and Dillmann R. Web-based interactive volume rendering. *Stud Health Technol Inform*, 163:635–7, 2011.

[67] Vazhkudai SS, Kohl JA, and Schwidder J. A java-based science portal for neutron scattering experiments. In *Proceedings of the 5th international symposium on Principles and practice*

of programming in Java, PPPJ '07, pages 21–30. ACM, New York, NY, USA, 2007. ISBN 978-1-59593-672-1. doi:10.1145/1294325.1294329.

[68] Grimstead IJ, Avis NJ, and Walker DW. Automatic distribution of rendering workloads in a grid enabled collaborative visualization environment. In *Proceedings of the 2004 ACM/IEEE conference on Supercomputing*, page 1. IEEE Computer Society, 2004.

[69] Johnson GP, Mock SA, Westing BM, and Johnson GS. EnVision: a Web-Based tool for scientific visualization. In *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, CCGRID '09, pages 603–608. IEEE Computer Society, Washington, DC, USA, 2009. ISBN 978-0-7695-3622-4. doi:10.1109/CCGRID.2009.80.

[70] Holmes VP, Linebarger JM, Miller DJ, Vandewart RL, and Crowley CP. Evolving the web-based distributed SI/PDO architecture for High-Performance visualization. In *Proceedings of the 34th Annual Simulation Symposium (SS01)*, pages 151–158. IEEE Computer Society, Washington, DC, USA, 2001.

[71] Qiao W, McLennan M, Kennell R, Ebert DS, and Klimeck G. Hub-based simulation and graphics hardware accelerated visualization for nanotechnology applications. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1061–1068, October 2006. ISSN 1077-2626. doi:10.1109/TVCG.2006.150.

[72] Xiao M, Liang Q, and Xiao Y. Implementation of campus Grid-Based visualization techniques. In *Wireless Communications, Networking and Mobile Computing*, pages 1–4. IEEE, October 2008. ISBN 978-1-4244-2107-7. doi:10.1109/WiCom.2008.2993.

[73] Eichberger G, Perry C, Walker H, Hastings P, Linsen L, and Frank L. Interactive 3D graphics for web-based data analysis and visualization for the digital fish library (DFL). In *ACM SIGGRAPH 2006 Research posters*, SIGGRAPH '06. ACM, New York, NY, USA, 2006. ISBN 1-59593-364-6. doi:10.1145/1179622.1179818.

[74] Chen J, Yoon I, and Bethel W. Interactive, internet delivery of visualization via structured prerendered multiresolution imagery. *IEEE Trans Vis Comput Graph*, 14(2):302–12, 2008. doi:10.1109/TVCG.2007.70428.

[75] Poliakov AV, Albright E, Corina D, Ojemann G, Martin RF, and Brinkley JF. Server-based approach to web visualization of integrated 3-D medical image data. *Proceedings / AMIA Annual Symposium AMIA Symposium*, pages 533–537, 2001. ISSN 1531-605X.

[76] Xiao Y, Yi M, Li Y, and Xiao M. Study on application of long-distance education resource grid. In *E-Business and E-Government (ICEE)*, pages 62–65. IEEE, May 2010. ISBN 978-1-4244-6647-4. doi:10.1109/ICEE.2010.23.

[77] Doyle MD, Ang CS, Martin DC, and Noe A. The visible embryo project: embedded program objects for knowledge access, creation and management through the world wide web. *Comput Med Imaging Graph*, 20(6):423–31, 1996.

[78] Bethel W, Siegerist C, Shalf J, Shetty P, Jankun-Kelly T, Kreylos O, and Ma KL. Visportal: Deploying grid-enabled visualization tools through a web-portal interface. In *Workshop on Advanced Collaborative Environments*. Citeseer, Seattle, WA (US), 06/22/2003, 2003.

[79] Jankun-Kelly TJ and Ma K. VisSheet redux: redesigning a visualization exploration spreadsheet for the web. In *ACM SIGGRAPH 2002 conference abstracts and applications*, SIGGRAPH '02, pages 329–329. ACM, New York, NY, USA, 2002. ISBN 1-58113-525-4. doi: 10.1145/1242073.1242337.

[80] Lefer W and Pierson JM. Visualization services on the web. In *SNPD'00 (International Conference on Software Engineering Applied to Networking and Parallel/Distributed Computing)*. Citeseer, 2000.

[81] Kačeniauskas A and Pacevič R. VizLitG: grid visualization e-Service enabling partial dataset transfer from storage elements of gLite-based grid infrastructure. *J Grid Comput*, 9(4):573–589, December 2011. ISSN 1570-7873. doi:10.1007/s10723-011-9193-0.

[82] Lamberti F, Sanna A, and Henao Ramirez EA. Web-based 3D visualization for intelligent street lighting. In *Proceedings of the 16th International Conference on 3D Web Technology*, Web3D '11, pages 151–154. ACM, New York, NY, USA, 2011. ISBN 978-1-4503-0774-1. doi:10.1145/2010425.2010452.

[83] Vickery R, Martin J, Fowler J, Moorehead R, Dandass Y, Atkinson T, Cedilnik A, Adams P, and Clarke J. Web-Based high performance remote visualization. In *Proceedings of the 2007 DoD High Performance Computing Modernization Program Users Group Conference*, HPCMP-UGC '07, pages 364–369. IEEE Computer Society, Washington, DC, USA, 2007. ISBN 0-7695-3088-5. doi:10.1109/HPCMP-UGC.2007.82.

[84] Elmaghraby AS, Elfayoumy SA, Karachiwala IS, Graham JH, Emam AZ, and Sleem A. Web-based performance visualization of distributed discrete event simulation. In *Proceedings of the 31st conference on Winter simulation: Simulation—a bridge to the future - Volume 2*, WSC '99, pages 1618–1623. ACM, New York, NY, USA, 1999. ISBN 0-7803-5780-9. doi:10.1145/324898.325346.

[85] Berjon R, Leithead T, Navara ED, O'Connor E, and Pfeiffer S. HTML 5.1. a vocabulary and associated APIs for HTML and XHTML. W3c working draft, W3C, Dec 2012. Http://www.w3.org/TR/2012/WD-html51-20121217/, Last Accessed: 17. March 2013.

[86] Kordon F et al. An introduction to rapid system prototyping. *Software Engineering, IEEE Transactions on*, 28(9):817–821, 2002.

[87] VideoLAN. Videolan project (vlc). URL `http://www.videolan.org`, Last Accessed: 14. June 2013.

[88] The xiph open source community. icecast server software for streaming multimedia. URL `http://www.icecast.org/`, Last Accessed: 14. June 2013.

[89] Fielding R, Gettys J, Mogul J, Frystyk H, Masinter L, Leach P, and Berners-Lee T. Hypertext Transfer Protocol – HTTP/1.1. In *Request for Comments*, number 2616 in Request for Comments. Internet Engineering Task Force, IETF, June 1999.

[90] DWR Community. Direct Web Remoting (DWR), easy AJAX for JAVA. URL `http://directwebremoting.org/dwr/index.html`, Last Accessed: 14. June 2013.

[91] Jboss Community. Jboss application server. URL `http://www.jboss.org`, Last Accessed: 14. June 2013.

[92] Levinson E. The mime multipart/related content-type. In *Request for Comments*, number 2387 in Request for Comments. Internet Engineering Task Force, IETF, 1998.

[93] Agarwal B, Tayal S, and Gupta M. *Software engineering & testing: an introduction*. Infinity Science Press, har/cdr edition, 2009. ISBN 1934015555.

[94] King D and Wendell J. Vinagre – a VNC client for the GNOME desktop. URL `http://projects.gnome.org/vinagre/`, Last Accessed: 14. June 2013.

[95] Richardson T, Stafford-Fraser Q, Wood KR, and Hopper A. Virtual network computing. *IEEE Internet Computing*, 2(1):33–38, 1998.

[96] Truong T, Huynh A, Gentilello L, and Liu S. Screenleap, 2011. URL `http://www.screenleap.com`, Last Accessed: 14. June 2013.

[97] Moolenaar B. Vim – the editor. URL `http://www.vim.org/`, Last Accessed: 14. June 2013.

[98] The Eclipse Foundation. The Eclipse C/C++ Development Tooling. URL `http://www.eclipse.org/cdt/`, Last Accessed: 14. June 2013.

[99] Free Software Foundation, Inc. The GNU compiler collection. URL `http://gcc.gnu.org/`, Last Accessed: 14. June 2013.

[100] boost community. boost c++ library. URL `http://www.boost.org`, Last Accessed: 14. June 2013.

[101] Cloudmeter, Inc. Pion network library. URL `https://github.com/cloudmeter/pion`, Last Accessed: 14. June 2013.

[102] Kohlhoff CM. Boost.asio. URL `http://www.boost.org/libs/asio`, Last Accessed: 14. June 2013.

[103] XOrg foundation. X window system. URL `http://www.x.org`, Last Accessed: 14. June 2013.

[104] VirtualGL community. libjpeg-turbo. URL `http://libjpeg-turbo.virtualgl.org`, Last Accessed: 14. June 2013.

[105] Independent JPEG Group. JPEG standard library. URL `http://www.ijg.org`, Last Accessed: 14. June 2013.

[106] FFmpeg. Ffmpeg complete, cross-platform solution to record, convert and stream audio and video. URL `http://www.ffmpeg.org`, Last Accessed: 14. June 2013.

[107] jQuery Foundation. jquery. URL `http://jquery.com`, Last Accessed: 14. June 2013.

[108] Silverstein JC, Walsh C, Dech F, Olson E, Papka M, Parsad N, and Stevens R. Multi-parallel open technology to enable collaborative volume visualization: how to create global immersive virtual anatomy classrooms. *Stud Health Technol Inform*, 132:463–8, 2008.

[109] Hubert B. The wonder shaper. URL `http://lartc.org/wondershaper/`, Last Accessed: 14. June 2013.

[110] Access Grid Community. Access Grid. URL `http://www.accessgrid.org`, Last Accessed: 14. June 2013.

[111] Röntgen WC. *Eine neue Art von Strahlen.* Verlag und Druck der Stahel'schen K. Hof-und Universitäts-Buch-und Kunsthandlung, 1896.

[112] Hounsfield GN. Computed medical imaging. *Journal of Computer Assisted Tomography*, 4(5):665, 1980. ISSN 0363-8715.

[113] Preim B and Bartz D. *Visualization in Medicine: Theory, Algorithms, and Applications (The Morgan Kaufmann Series in Computer Graphics).* Morgan Kaufmann, 1 edition, 7 2007. ISBN 9780123705969.

[114] Jin M, Wernick M, Yang Y, Brankov J, Gravier E, Feng B, and King M. 5d image reconstruction for tomographic image sequences. In *Signals, Systems and Computers, 2006. ACSSC'06. Fortieth Asilomar Conference on*, pages 1973–1977. IEEE, 2006.

[115] NEMA. *Digital Imaging and Communications in Medicine (DICOM).* National Electrical Manufacturers Association, Virginia, USA, 2009.

[116] NEMA. *DICOM - Part 5: Data Structures and Encoding*. National Electrical Manufacturers Association, Virginia, USA, 2009.

[117] Tzannes A. Compression of 3-dimensional medical image data using part 2 of jpeg 2000. *Aware Inc, Nov*, 2003.

[118] Purchasing N and Agency S. Comparative specifications: 64 slice ct scanners. Technical Report CEP08027, NHS, 2009. URL `http://www.impactscan.org`, Last Accessed: 14. June 2013.

[119] Lacroute P and Levoy M. Fast volume rendering using a shear-warp factorization of the viewing transformation. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 451–458. ACM, 1994. ISBN 0897916670.

[120] Neophytou N and Mueller K. Gpu accelerated image aligned splatting. In *Fourth International Workshop on Volume Graphics, 2005.*, pages 197–242. IEEE, 2005. ISSN 1727-8376.

[121] Muller MA, Marincek B, and Frauenfelder T. State of the art 3d imaging of abdominal organs. *JBR-BTR*, 90(6):467–74, 2007.

[122] Rosset A, Spadola L, and Ratib O. Osirix: An open-source software for navigating in multidimensional dicom images. *Journal of Digital Imaging*, 17(3):205–216, 2004. ISSN 0897-1889. doi:10.1007/s10278-004-1014-6.

[123] Tovée MJ. *An introduction to the visual system*. Cambridge University Press, Cambridge, UK, 2nd ed edition, 2008. ISBN 9780521883191.

[124] Dodgson NA. Autostereoscopic 3d displays. *Computer*, 38(8):31–36, 2005.

[125] Jones GR, Lee D, Holliman NS, and Ezra D. Controlling perceived depth in stereoscopic images. In *Stereoscopic displays and applications VIII. Proceedings of SPIE*, pages 42–53. SPIE, 2001.

[126] McAllister DF. Stereo and 3-d display technologies. *Encyclopedia of imaging science and technology*, 2002.

[127] Meschede D and Vogel H. *Gerthsen Physik, 23. Auflage*. Springer Verlag, 2006. ISBN 9783540254218.

[128] Minoli D. *3D Television (3DTV) Technology, Systems, and Deployment: Rolling Out the Infrastructure for Next-Generation Entertainment*. CRC Press, 2010. ISBN 1439840660.

[129] Wimmer P. Anaglyph methods comparison, 2010. URL `http://3dtv.at/Knowhow/AnaglyphComparison_en.aspx`, Last Accessed: 14. June 2013.

[130] Jorke H and Fritz M. Infitec - a new stereoscopic visualisation tool by wavelength multiplex imaging. *Journal of Three Dimensional Images*, 19:50–56, 2005.

[131] Moore JR, Dodgson NA, Travis ARL, and Lang SR. Time-multiplexed color autostereoscopic display. In *Proceedings of SPIE (the international society for optical engineering)*, pages 10–19. Citeseer, 1996.

[132] Sandin DJ, Sandor E, Cunnally W, Resch M, DeFanti TA, and Brown MD. Computer-generated barrier-strip autostereography. In *Proc SPIE, Non-Holographic True 3D Display Technologies*, volume 1083, pages 65–75. 1989.

[133] van Berkel C and Clarke JA. Characterisation and optimisation of 3d-lcd module design. In *Proc. SPIE*, volume 3012, pages 179–186. 1997.

[134] Yeom S, Stern A, and Javidi B. Compression of 3d color integral images. *Optics Express*, 12(8):1632–1642, April 2004.

[135] Min SW, Kim J, and Lee B. Three-dimensional electro-floating display system based on integral imaging technique. *Stereoscopic Displays and Applications XVI, Electronics Imaging, paper 5664A-37, San Jose, CA*, 2005.

[136] Halle M. Autostereoscopic displays and computer graphics. In *ACM SIGGRAPH 2005 Courses*, page 104. ACM, 2005.

[137] Perlin K, Poultney C, Kollin JS, Kristjansson DT, and Paxia S. Recent advances in the nyu autostereoscopic display. In *Proceedings of SPIE*, volume 4297, page 196. Citeseer, 2001.

[138] Matusik W and Pfister H. 3D TV: a scalable system for real-time acquisition, transmission, and autostereoscopic display of dynamic scenes. In *ACM SIGGRAPH 2004 Papers*, pages 814–824. ACM, 2004.

[139] Levoy M and Hanrahan P. Light field rendering. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 31–42. ACM, 1996.

[140] Maeda H, Hirose K, Yamashita J, Hirota K, and Hirose M. All-around display for video avatar in real world. In *ISMAR*, pages 288–289. IEEE Computer Society, 2003. ISBN 0-7695-2006-5.

[141] Jones A, McDowall I, Yamada H, Bolas M, and Debevec P. An interactive 360 light field display. In *ACM SIGGRAPH 2007 emerging technologies*, page 13. ACM, 2007.

[142] Favalora G, Dorval RK, Hall DM, Giovinco M, and Napoli J. Volumetric three-dimensional display system with rasterization hardware. In *Proc SPIE*, volume 4297, pages 227–235. 2001.

[143] Sullivan A. 3-deep: New displays render images you can almost reach out and touch. *Spectrum, IEEE*, 42(4):30–35, 2005. ISSN 0018-9235.

[144] Saito H, Kimura H, Shimada S, Naemura T, Kayahara J, Jarusirisawad S, Nozick V, Ishikawa H, Murakami T, Aoki J, et al. Laser-plasma scanning 3d display for putting digital contents in free space. In *Proceedings of International Symposium on Electronic Imaging, Stereoscopic Displays and Applications*, volume 19, pages 6803–07. 2008.

[145] Hoshi T, Takahashi M, Nakatsuma K, and Shinoda H. Touchable holography. In *ACM SIGGRAPH 2009 Emerging Technologies*, page 23. ACM, 2009.

[146] Francone J and Nigay L. Using the user s point of view for interaction on mobile devices. In *IHM '11, the 23th ACM International Conference of the Association Francophone d'Interaction Homme-Machine*. 2011.

[147] NVIDIA. Nvidia 3d vision. URL `http://www.nvidia.com/object/3d-vision-about.html`, Last Accessed: 14. June 2013.

[148] Krishnaprasad NK, Vishwanath V, Venkataraman S, Rao AG, Renambot L, Leigh J, Johnson AE, and Davis B. JuxtaView-a tool for interactive visualization of large imagery on scalable tiled displays. In *cluster*, pages 411–420. IEEE, 2004.

[149] Chae S, Majumder A, and Gopi M. Hd-graphviz: highly distributed graph visualization on tiled displays. In *Proceedings of the Eighth Indian Conference on Computer Vision, Graphics and Image Processing*, page 43. ACM, 2012.

[150] Febretti A, Nishimoto A, Thigpen T, Talandis J, Long L, Pirtle J, Peterka T, Verlo A, Brown M, Plepys D, et al. Cave2: a hybrid reality environment for immersive simulation and information analysis. In *IS&T/SPIE Electronic Imaging*, pages 864903–864903. International Society for Optics and Photonics, 2013.

[151] Renambot L, Rao A, Singh R, Jeong B, Krishnaprasad N, Vishwanath V, Chandrasekhar V, Schwarz N, Spale A, Zhang C, et al. Sage: the scalable adaptive graphics environment. In *Proceedings of WACE*, pages 2004–09. Citeseer, 2004.

[152] Johnson G, Abram G, Westing B, Navr'til P, and Gaither K. Displaycluster: An interactive visualization environment for tiled displays. In *IEEE International Conference on Cluster Computing*, pages 239–247. 2012. doi:10.1109/CLUSTER.2012.78.

[153] Sandin DJ, Margolis T, Ge J, Girado J, Peterka T, and DeFanti TA. The varrier autostereoscopic virtual reality display. *ACM Transactions on Graphics*, 24(3):894–903, 2005. ISSN 0730-0301.

[154] Shibano N, Hareesh PV, Hoshino H, Kawamura R, Yamamoto A, Kahiwagi M, and Sawada K. Cyberdome: Pc clustered hemi spherical immersive projection display. In *ICAT*. 2003.

[155] Czernuszenko M, Pape D, Sandin D, DeFanti T, Dawe GL, and Brown MD. The immersadesk and infinity wall projection-based virtual reality displays. *ACM SIGGRAPH Computer Graphics*, 31(2):46–49, 1997. ISSN 0097-8930.

[156] Johnson AE, Leigh J, Morin P, and Keken PV. Geowall: Stereoscopic visualization for geoscience research and education. *IEEE Computer Graphics and Applications*, 26(6):10–14, 2006.

[157] RealD. RealD – the new 3D. URL `http://www.reald.com`, Last Accessed: 14. June 2013.

[158] Krah CH. Three dimensional display system. *United States patent US 7843449*, November 2010.

[159] DeFanti T, Cruz-Neira C, and Sandin D. Surround screen projection-based virtual reality: The design and implementation of the cave. In *Conference Proceedings, ACM SIGGRAPH*, volume 93. 1993.

[160] Leigh J, Johnson AE, DeFanti TA, and Brown MD. A review of tele-immersive applications in the cave research network. In *Virtual Reality*, pages 180–187. 1999.

[161] Höllerer T, Kuchera-Morin J, and Amatriain X. The allosphere: a large-scale immersive surround-view instrument. In C Cruz-Neira and D Reiners, editors, *EDT*, volume 252 of *ACM International Conference Proceeding Series*, page 3. ACM, 2007. ISBN 978-1-59593-669-1.

[162] Funkhouser TA. Network topologies for scalable multi-user virtual environments. In *Proceedings of the 1996 Virtual Reality Annual International Symposium (VRAIS 96)*, page 222. Citeseer, 1996. ISBN 0818672951.

[163] ITU-T. Information technology – open systems interconnection – basic reference model: The basic model. In *ITU-T Recommendation*, ITU-T Recommendation. Telecommunication standardization sector of ITU, ITU-T, 1994.

[164] Oppenheimer P. *Top-Down Network Design, Third Edition*. Cisco Press, 2011.

[165] Deolasee P, Katkar A, Panchbudhe A, Ramamritham K, and Shenoy P. Adaptive push-pull: disseminating dynamic web data. In *Proceedings of the 10th international conference on World Wide Web*, pages 265–274. ACM, 2001.

[166] Bozdag E, Mesbah A, and Van Deursen A. A comparison of push and pull techniques for ajax. In *Web Site Evolution, 2007. WSE 2007. 9th IEEE International Workshop on*, pages 15–22. IEEE, 2007.

[167] Tongleamnak S, Covavisaruch N, and Vatanawood W. Programmable web-based volume visualization. In *The 1st ECTI Annual Conference*. 2004.

[168] Hunter J, Henderson M, and Khan I. Collaborative annotation of 3d crystallographic models. *J Chem Inf Model*, 47(6):2475–84, 2007.

[169] Sun Microsystems, Oracle. Java. URL `http://www.oracle.com/technetwork/java/index.html`, Last Accessed: 14. June 2013.

[170] Gerth VE and Vize PD. A java tool for dynamic web-based 3D visualization of anatomy and overlapping gene or protein expression patterns. *Bioinformatics (Oxford, England)*, 21(7):1278–1279, April 2005. ISSN 1367-4803. doi:10.1093/bioinformatics/bti120.

[171] Oracle. Java 3d. URL `https://java3d.java.net/`, Last Accessed: 14. June 2013.

[172] Ding R, Gao J, Chen B, Siepmann JI, and Liu Y. Web-Based visualization of atmospheric nucleation processes using Java3D. In *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, CCGRID '09, pages 597–602. IEEE Computer Society, Washington, DC, USA, 2009. ISBN 978-0-7695-3622-4. doi:10.1109/CCGRID.2009.56.

[173] ISO/IEC 19775/19776/19777. Extensible 3d (X3D). In *International Organization for Standardization*. ISO, Geneva, Switzerland, 2010.

[174] Kelle O. Remote brain image segmentation. In *Information Technology Applications in Biomedicine*. IEEE, 1999. doi:10.1109/ITAB.1999.842313.

[175] Villasenor M, Flores F, and Algorri M. Anatomical models for virtual reality and web-based applications. In *Engineering in Medicine and Biology Society, 2001. Proceedings of the 23rd Annual International Conference of the IEEE*, volume 4, pages 3769–3772. IEEE, 2001.

[176] Raposo A, Magalhaes L, and Ricarte I. Working with remote VRML scenes through low-bandwidth connections. In *Computer Graphics and Image Processing*, pages 34–41. IEEE, 1997. ISBN 0-8186-8102-0. doi:10.1109/SIGRA.1997.625145.

[177] Moore C, McClurg D, Soreide N, Hermann A, Lascara C, and Wheless G. Exploring 3-dimensional oceanographic data sets on the web using virtual reality modeling language. In *OCEANS '99 MTS/IEEE. Riding the Crest into the 21st Century*, volume 3, pages 1501–1503. IEEE, 1999. ISBN 0-7803-5628-4. doi:10.1109/OCEANS.1999.800217.

[178] Lee KW, Truong N, Rhodes B, Mclaren J, and Wang L. Development of a remote access control laboratory using xPC target and virtual reality modeling language. In *Intelligent and Advanced Systems (ICIAS)*, pages 1–6. IEEE, June 2010. ISBN 978-1-4244-6623-8. doi:10.1109/ICIAS.2010.5716182.

[179] Kostaridis A, Biniaris C, Foukarakis I, Kaklamani D, and Venieris I. A web-based distributed computing framework for antenna array modeling. *IEEE Communications Magazine*, 42(10):81–87, October 2004. ISSN 0163-6804. doi:10.1109/MCOM.2004.1341265.

[180] Jacinto H, Kéchichian R, Desvignes M, Prost R, and Valette S. A web interface for 3d visualization and interactive segmentation of medical images. In *Proceedings of the 17th International Conference on 3D Web Technology*, pages 51–58. ACM, 2012.

[181] Kim N, Lee DH, Kim JH, Kim Y, and Cho HJ. Web based 3-d medical image visualization on the pc. *Stud Health Technol Inform*, 52 Pt 2:1105–10, 1998.

[182] Melzer K, Lipinski HG, and Grönemeyer DHW. X3d-technologies for medical image visualization. In *ACM SIGGRAPH 2004 Posters*, page 112. ACM, 2004. ISBN 1581138962.

[183] Huang J and Cheng B. Interactive visualization for 3D pipelines using Ajax3D. In *Proceedings of the 2009 International Conference on Networking and Digital Society - Volume 01*, pages 21–24. IEEE Computer Society, Washington, DC, USA, 2009. ISBN 978-0-7695-3635-4-01. doi:10.1109/ICNDS.2009.12.

[184] Marrin C. *WebGL Specification, Version 1.0*. Khronos Group, Beaverton, Oregon, USA, February 2011.

[185] Congote J, Segura A, Kabongo L, Moreno A, Posada J, and Ruiz O. Interactive visualization of volumetric data with webgl in real-time. In *Proceedings of the 16th International Conference on 3D Web Technology*, pages 137–146. ACM, 2011.

[186] Behr J, Eschler P, Jung Y, and Zöllner M. X3DOM: a DOM-based HTML5/X3D integration model. In *Proceedings of the 14th International Conference on 3D Web Technology*, pages 127–135. ACM, 2009.

[187] Noumeir R and Pambrun JF. Using jpeg 2000 interactive protocol to stream a large image or a large image set. *J Digit Imaging*, 24(5):833–43, Oct 2011. doi:10.1007/s10278-010-9343-0.

[188] Hereld M, Papka ME, Insley JA, Norman ML, Olson EC, and Wagner R. Interactive large data exploration over the wide area. In *Proceedings of the 2011 TeraGrid Conference: Extreme Digital Discovery*, page 19. ACM, 2011.

[189] Lamberti F and Sanna A. A streaming-based solution for remote visualization of 3d graphics on mobile devices. *IEEE Trans Vis Comput Graph*, 13(2):247–60, 2007. doi:10.1109/TVCG.2007.29.

[190] Halbach T and Wien M. Concepts and performance of next-generation video compression standardization. In *5th Nordic Signal Processing Symposium (NORSIG-2002)*. 2002.

[191] JTC 1/SC 29. Text of ISO/IEC FDIS 23008-1/2 information technology – high efficiency coding and media delivery in heterogeneous environments. Technical report, ISO/IEC, 2013.

[192] Hewage C, Karim H, Worrall S, Dogan S, and Kondoz A. Comparison of stereo video coding support in mpeg-4 mac, h. 264/avc and h. 264/svc. *Proc of IET Visual Information Engineering-VIE07*, 2007.

[193] ITU-T. H.323 (12/09). packet-based multimedia communications systems. In *ITU-T Recommendation*, ITU-T Recommendation. Telecommunication standardization sector of ITU, ITU-T, Nov 2000.

[194] Caltech. Evo – the collaborative network, 2009. URL `http://evo.caltech.edu/evoGate/`, Last Accessed: 14. June 2013.

[195] Foster I and Kesselman C. *The grid: blueprint for a new computing infrastructure*. Morgan Kaufmann, Amsterdam, 2nd ed edition, 2004. ISBN 1558609334.

[196] Nets Group, Computer Science, University College London. vic - video conferencing tool. URL `http://mediatools.cs.ucl.ac.uk/nets/mmedia/wiki/VicWiki#VideoconferencingToolVIC`, Last Accessed: 14. June 2013.

[197] Nets Group, Computer Science, University College London. Robust audio tool (rat), 2008. URL `http://mediatools.cs.ucl.ac.uk/nets/mmedia/wiki/RatWiki#RobustAudioToolRAT`, Last Accessed: 14. June 2013.

[198] San Diego Supercomputer Center. High-tech collaboration helps taiwan fight sars, 2003. URL `http://www.sdsc.edu/News%20Items/PR052903.html`, Last Accessed: 14. June 2013.

[199] JTC 1/SC 29/WG 11. Text of ISO/IEC 14496-10:200X/FDAM 1 multiview video coding (doc. n9978). Technical report, ISO/IEC, Hannover, Germany, 2008.

[200] Zhentang J and Xiaotai N. Stereo video communication and key techniques. In *Second International Symposium on Electronic Commerce and Security*. 2009.

[201] Reveiu A, Dardala M, and Smeureanu I. A MPEG-21 based architecture for data visualization in multimedia web applications. In *Proceedings of the 2008 International Conference Visualisation*, pages 84–89. IEEE Computer Society, Washington, DC, USA, 2008. ISBN 978-0-7695-3271-4. doi:10.1109/VIS.2008.13.

[202] Merkle P, Brust H, Dix K, Müller K, and Wiegand T. Stereo video compression for mobile 3d services. In *3DTV Conference: The True Vision-Capture, Transmission and Display of 3D Video, 2009*, pages 1–4. 2009.

[203] Zaluzec NJ. Telepresence system overview - white paper - (draft). Technical report, Electron Microscopy Center, Materials Science Division Argonne National Laboratory, 2002. URL `http://tpm.amc.anl.gov/TPMDocs/TPMOverview2002.pdf`, Last Accessed: 14. June 2013.

[204] Lamberti F, Zunino C, Sanna A, Fiume A, and Maniezzo M. An accelerated remote graphics architecture for pdas. In *Proc. the 8th International Conference on 3D Web Technology*. Citeseer, 2003.

[205] Sanna A, Paravati G, Godio E, and Fiorella D. Victory project workpackage delivery: Visualization of 3d complex scenes on mobile devices. Technical report, Information Society Technologies (IST) Programme, 2007.

[206] He L, Ming X, Ding W, and Liu Q. A novel approach to remote access picture archiving and communication system on mobile devices over wireless networks. In *Biomedical and Health Informatics (BHI), 2012 IEEE-EMBS International Conference on*, pages 581–583. IEEE, 2012.

[207] Lamberti F and Sanna A. Extensible guis for remote application control on mobile devices. *Ieee Computer Graphics and Applications*, 28(4):50–57, Jul-Aug 2008.

[208] Giordani A. Feature - ultra-fast networks: The final frontier, 2010. URL `http://www.isgtw.org/?pid=1002828`, Last Accessed: 14. June 2013.

[209] Taylor E. Argonne streaming visualization sends images across the world, January 2010. URL `http://www.ci.uchicago.edu/news/detail.php?id=257`, Last Accessed: 14. June 2013.

[210] Karonis N, Papka M, Binns J, Bresnahan J, Insley J, Jones D, and Link J. High-resolution remote rendering of large datasets in a collaborative environment. *Future Generation Computer Systems*, 19(6):909–917, August 2003. doi:10.1016/S0167-739X(03)00070-0.

[211] Christodoulou L, Mayron LM, Kalva H, Marques O, and Furht B. 3d tv using mpeg-2 and h.264 view coding and autostereoscopic displays. In *ACM Multimedia*, pages 505–506. 2006.

[212] Kurillo G, Bajcsy R, Kreylos O, and Rodriguez R. Teleimmersive environment for remote medical collaboration. *Stud Health Technol Inform*, 142:148–50, 2009.

[213] Bandwidth A. Wireless live streaming video of laparoscopic surgery: A bandwidth analysis for handheld computers. *Medicine meets virtual reality 02/10: digital upgrades, applying Moore's law to health*, 85:150, 2002.

[214] Eolas Technologies Inc. Eolas. URL `http://www.eolas.com/`, Last Accessed: 14. June 2013.

[215] Image Stream Medical, Inc. Livestream network. URL `http://www.imagestreammedical.com`, Last Accessed: 14. June 2013.

[216] Hahm JS, Lee HL, Kim SI, Shimizh S, Choi HS, Ko Y, Lee KG, Kim TE, Yun JW, Park YJ, et al. A remote educational system in medicine using digital video. *Hepato-gastroenterology*, 54(74):373–376, 2007.

[217] Shimizu S, HAN HOS, Okamura K, Yamaguchi K, and Tanaka M. Live demonstration of surgery across international borders with uncompressed high-definition quality. *HPB*, 9(5):398–399, 2007.

[218] Ilgner JFR, Kawai T, Shibata T, Yamazoe T, and Westhofen M. Evaluation of stereoscopic medical video content on an autostereoscopic display for undergraduate medical education. In *Proceedings of SPIE*, volume 6055, page 605506. 2006.

[219] Justus Ilgner M, Park JJH, Daniel Labbé M, and Westhofen M. Using a high-definition stereoscopic video system to teach microscopic surgery. *Stereoscopic Displays and Virtual Reality Systems XIV*, 6490, 2007.

[220] Sinton FC. Html5 video available on the web, Feb 2011. URL `http://blog.mefeedia.com/html5-feb-2011`, Last Accessed: 14. June 2013.

[221] Pilgrim M. Video on the web. URL `http://diveintohtml5.org/video.html`, Last Accessed: 14. June 2013.

[222] Pantos R and May W. HTTP Live Streaming. In *Informational Internet-Draft*. Internet Engineering Task Force, IETF, Oct 2012.

[223] IHE International, Inc. It infrastructure technical framework volume ii revision 8 - transactions. Technical report, IHE International, Inc., 2011. URL `http://www.ihe.net/Technical_Framework/index.cfm#IT`, Last Accessed: 14. June 2013.

[224] NEMA. Digital imaging and communications in medicine (dicom); part 18: Web access to dicom persistent objects (wado). Technical report, National Electrical Manufacturers Association, 2006. URL `ftp://medical.nema.org/medical/dicom/2006/06_18pu.pdf`, Last Accessed: 14. June 2013.

[225] Noumeir R and Pambrun JF. Images within the electronic health record. In *Image Processing (ICIP), 2009 16th IEEE International Conference on*, pages 1761–1764. IEEE, 2009. ISSN 1522-4880.

[226] Engelmann U. Teleradiologie: Der weg von der forschung in die regelversorgung (german). *MDI*, 2011.

[227] Arguiñ andarena EJC, Macchi JE, Escobar PP, del Fresno M, Massa JM, and Santiago MA. Dcm-ar: A fast flash-based web-pacs viewer for displaying large dicom images. In *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, pages 3463 –3466. 31 2010-sept. 4 2010. ISSN 1557-170X. doi: 10.1109/IEMBS.2010.5627827.

[228] Open source clinical image and object management, 2010. URL `http://www.dcm4che.org`, Last Accessed: 14. June 2013.

[229] Linder E, Lundin M, Thors C, Lebbad M, Winiecka-Krusnell J, Helin H, Leiva B, Isola J, and Lundin J. Web-based virtual microscopy for parasitology: a novel tool for education and quality assurance. *PLoS Negl Trop Dis*, 2(10):e315, 2008. doi:10.1371/journal.pntd.0000315.

[230] Lien CY, Teng HC, Chen DJ, Chu WC, and Hsiao CH. A web-based solution for viewing large-sized microscopic images. *Journal of Digital Imaging*, 22(3):275–285, 2009.

[231] Tous R, Delgado J, Zinkl T, Toran P, Alcalde G, Goetz M, and Ferrer Roca O. The anatomy of an optical biopsy semantic retrieval system. *Multimedia, IEEE*, 19(2):16–27, 2012.

[232] Stegmaier S, Diepstraten J, Weiler M, and Ertl T. Widening the remote visualization bottleneck. In *Proceedings of the 3rd International Symposium on Image and Signal Processing and Analysis*, volume 1, pages 174–179. IEEE, 2003. ISBN 953184061X. ISSN 1330-1012.

[233] Commander D. The VirtualGL Project, 2009. URL `http://www.virtualgl.org`, Last Accessed: 14. June 2013.

[234] Stegmaier S, Magallón M, and Ertl T. A generic solution for hardware-accelerated remote visualization. In *Proceedings of the symposium on Data Visualisation 2002*. Eurographics Association Aire-la-Ville, Switzerland, 2002.

[235] Löhnhardt B, Dickmann F, Quade M, Skrowny D, Heppner S, Kaspar M, Kepper N, Krefting D, Steinke T, and Sax U. Workshop: Evaluation of visualization approaches in a biomedical grid environment. In *2010 Sixth IEEE International Conference on e-Science Workshops*. 2010.

[236] Kaspar M, Löhnhardt B, Kepper N, Knoch TA, Krefting D, Steinke T, Viezens F, Dickmann F, and Sax U. Poster: Interaktive 3d/4d-visualisierung in biomedizinischen grid-infrastrukturen. In *54. Jahrestagung der Deutschen Gesellschaft für Medizinische Informatik, Biometrie und Epidemiologie e.V. (GMDS)*. 2009.

[237] Branzan Albu A, Laurendeau D, Gurtner M, and Martel C. A web-based remote collaborative system for visualization and assessment of semi-automatic diagnosis of liver cancer from ct images. *Stud Health Technol Inform*, 111:75–8, 2005.

[238] Zaluzec NJ. TelePresence Microscopy Collaboration. URL `http://tpm.amc.anl.gov`, Last Accessed: 14. June 2013.

[239] Postek MT, Bennett MH, and Zaluzec NJ. Telepresence: A new paradigm for solving contamination problems. In *Analytical and diagnostic techniques for semiconductor materials, devices, and processes: joint proceedings of the symposia on: ALTECH 99: satellite symposium to ESSDERC 99: Leuven, Belgium: The Electrochemical Society Symposium on Diagnostic Techniques for Semiconductor materials and Devices*, page 331. The Electrochemical Society, 1999. ISBN 1566772397.

[240] Stokoe KH, Rathje E, Wilson CR, Rosenblad BL, and Menq FY. Development of the nees large-scale mobile shakers and associated instrumentation for in situ evaluation of nonlinear characteristics and liquefaction resistance of soils. In *13th World Conference on Earthquake Engineering*, pages 1–6. 2004.

[241] Bohne-Lang A, Groch WD, and Ranzinger R. Aismig–an interactive server-side molecule image generator. *Nucleic Acids Res*, 33(Web Server issue):W705–9, Jul 2005.

[242] University of Chicago. Anatomically correct, April 2009. URL `http://press.mcs.anl.gov/futureslab/2009/04/10/anatomically-correct`, Last Accessed: 14. June 2013.

[243] OSG Community. Openscenegraph. URL `http://www.openscenegraph.org`, Last Accessed: 14. June 2013.

[244] Freitag LA and Loy RM. Comparison of remote visualization strategies for interactive exploration of large data sets. In *Proc. of the 15th Int. Parallel & Distributed Processing Symposium (IPDPS, Los Alamitos, USA), pp.* 1/281/2-1/283, *Apr*, pages 23–27. Citeseer, 2001.

[245] Engel K and Ertl T. Texture-based volume visualization for multiple users on the world wide web. In *5th Eurographics Workshop on Virtual Environments*, pages 115–124. Citeseer, 1999.

[246] Turner D, Papageorgiou P, Chin C, Yates C, and Kimpe T. Server-client architecture in medical imaging. *United States patent US 7,890,573*, 2011.

[247] Liu D, Hua KA, and Sugaya K. A framework for Web-Based interactive applications of High-Resolution 3D medical image data. In *Proceedings of the 19th IEEE Symposium on Computer-Based Medical Systems*, pages 119–124. IEEE Computer Society, Washington, DC, USA, 2006. ISBN 0-7695-2517-1. doi:10.1109/CBMS.2006.12.

[248] Baker M and Ong H. A grid enabled liquid crystal structure modelling and visualization web portal. Technical report, DSG Technical Report 2002.02, 2002.

[249] Foster I, Insley J, von Laszewski G, Kesselman C, and Thiebaux M. Distance visualization: Data exploration on the grid. *Computer*, 32(12):36, December 1999.

[250] Pan Z, Yang B, Zhang M, Yu Q, and Lin H. Remote visualization based on grid computing. In *ICCSA (2)*, pages 236–245. 2004.

[251] Löhnhardt B, Kaspar M, Grütz R, Viezens F, and Dickmann F. Workshop: A prototype system for advance reservations in a biomedical grid computing visualization infrastructure. In *7th IEEE International Conference on e-Science, Computing Advances in Life Science workshop*. University of Göttingen, 2011.

[252] Kitware. Paraview - open source scientific visualization. URL `http://www.paraview.org/`, Last Accessed: 14. June 2013.

[253] Binns J, DiCarlo J, Insley JA, Leggett T, Lueninghoener C, Navarro JP, and Papka ME. Enabling community access to teragrid visualization resources. *Concurrency and Computation-Practice & Experience*, 19(6):783–794, April 2007. doi:DOI10.1002/cpe.1080.

[254] Metzger F, Rafetseder A, and Tutschku K. A performance evaluation framework for video streaming. In *Packet Video Workshop (PV), 2012 19th International*, pages 19–24. IEEE, 2012.

[255] Ng A, Greenfield P, and Chen S. A study of the impact of compression and binary encoding on soap performance. In *Proceedings of the Sixth Australasian Workshop on Software and System Architectures (AWSA2005)*, pages 46–56. Citeseer, 2005.

[256] ISO/IEC 15948. Information technology – computer graphics and image processing – portable network graphics (png): Functional specification. In *International Organization for Standardization*. ISO, Geneva, Switzerland, 2004.

[257] Wallace GK. The jpeg still picture compression standard. *Communications of the ACM*, 34(4):30–44, 1991.

[258] Miano J. *Compressed image file formats: Jpeg, png, gif, xbm, bmp*. Addison-Wesley Professional, 1999.

[259] ISO/IEC 15444. Information technology – jpeg 2000 image coding system. In *International Organization for Standardization*. ISO, Geneva, Switzerland, 2004.

[260] Skodras A, Christopoulos C, and Ebrahimi T. The jpeg 2000 still image compression standard. *Signal Processing Magazine, IEEE*, 18(5):36–58, 2001.

[261] Google. WebP – A new image format for the Web. URL `https://developers.google.com/speed/webp/`, Last Accessed: 14. June 2013.

[262] ITU-T. H.264 advanced video coding for generic audiovisual services. Technical report, ITU-T, 2007.

[263] Hickson I. The web sockets API. W3C working draft, W3C, December 2009. Http://www.w3.org/TR/2009/WD-websockets-20091222/, Last Accessed: 17. March 2013.

[264] Wu W, Uram T, and Papka ME. Web 2.0-based social informatics data grid. In *Proceedings of the 5th Grid Computing Environments Workshop*, pages 1–7. ACM, 2009.

[265] Kaspar M, Parsad NM, and Silverstein JC. Cowebviz – interactive collaborative sharing of 3d stereoscopic visualization among browsers with no added software. In *1st ACM International Health Informatics Symposium*. 11 2010.

[266] Kaspar M, Dech F, Parsad NM, and Silverstein JC. Web-based stereoscopic visualization for the global anatomy classroom. *Stud Health Technol Inform*, 163:264–70, 2011.

[267] Meyer-Spradow J, Ropinski T, Mensmann J, and Hinrichs K. Voreen: A rapid-prototyping environment for ray-casting-based volume visualizations. *Computer Graphics and Applications, IEEE*, 29(6):6–13, 2009.

[268] Axis Communications. URL `http://www.axis.com/`, Last Accessed: 14. June 2013.

[269] Weinstock N, Baumann M, Anderson S, and Fiedler R. Architecture and method for remote platform control management. *United States patent US 7970859 B2*, 2011.

[270] Hewlett-Packard. Remote graphics software. URL `http://www.hp.com/united-states/campaigns/workstations/remote-graphics-software.html`, Last Accessed: 14. June 2013.

[271] Cisco. Webex meetings. URL `http://www.webex.de/`, Last Accessed: 14. June 2013.

[272] Google. chrome. URL `http://www.google.com/chrome`, Last Accessed: 14. June 2013.

[273] Python Software Foundation. Python Programming Language. URL `http://python.org/`, Last Accessed: 14. June 2013.

[274] Novotny J, Tuecke S, and Welch V. An online credential repository for the grid: Myproxy. In *High Performance Distributed Computing, 2001. Proceedings. 10th IEEE International Symposium on*, pages 104–111. IEEE, 2001.

[275] Adobe Systems Software Ireland Ltd. Adobe flash professional. URL `http://www.adobe.com/de/products/flash.html`, Last Accessed: 14. June 2013.

[276] Lewis GJ, Hasan SM, Alexandrov VN, Dove MT, and Calleja M. Multicast application sharing tool–facilitating the eminerals virtual organisation. In *Computational Science–ICCS 2005*, pages 359–366. Springer, 2005.

[277] Polycom. URL `http://www.polycom.de/`, Last Accessed: 14. June 2013.

[278] Cisco. URL `http://www.cisco.com/`, Last Accessed: 14. June 2013.

[279] Kamaci N and Altunbasak Y. Performance comparison of the emerging h. 264 video coding standard with the existing standards. In *Multimedia and Expo, 2003. ICME'03. Proceedings. 2003 International Conference on*, volume 1, pages I–345. IEEE, 2003.

[280] Portlet Access Grid (PAG), 2009. URL `http://www.rcs.manchester.ac.uk/research/PAG`, Last Accessed: 14. June 2013.

[281] Microsoft. Windows Server 2012 – Remote Desktop Services Overview. URL `http://technet.microsoft.com/en-us/library/hh831447.aspx`, Last Accessed: 14. June 2013.

[282] Citrix. Citrix XenApp. URL `http://www.citrix.de/products/xenapp/overview.html`, Last Accessed: 14. June 2013.

[283] TeamViewer. TeamViewer - the All-In-One Software for Remote Support and Online Meetings. URL `http://www.teamviewer.com`, Last Accessed: 14. June 2013.

[284] Ito K, Shimada J, Katoh D, Nishimura M, Yanada M, Okada S, Ishihara S, and Ichise K. Interactive multicentre teleconferences using open source software in a team of thoracic surgeons. *Journal of telemedicine and telecare*, 18(8):465–469, 2012.

[285] Zhang X and Takahashi H. A hybrid data compression scheme for improved vnc. *Systemics, Cybernetics and Informatics*, 5(2):1–4, 2007.

[286] Hobona G, James P, and Fairbairn D. Web-based visualization of 3D geospatial data using Java3D. *IEEE Computer Graphics and Applications*, 26(4):28–33, August 2006. ISSN 0272-1716.

[287] Josefsson S. The Base16, Base32, and Base64 Data Encodings. In *Request for Comments*, number 4648 in Request for Comments. Internet Engineering Task Force, IETF, October 2006.

[288] Wiggins RH, Davidson HC, Harnsberger HR, Lauman JR, and Goede PA. Image file formats: Past, present, and future1. *Radiographics*, 21(3):789–798, 2001.

[289] McAllister DF, Zhou Y, and Sullivan S. Methods for computing color anaglyphs. In *Proceedings of SPIE*, volume 7524, page 75240S. 2010.

[290] Didyk P, Ritschel T, Eisemann E, Myszkowski K, and Seidel HP. A perceptual model for disparity. *ACM Trans Graph*, 30(4):96, 2011.

[291] Dodgson NA. Optical devices: 3d without the glasses. *Nature*, 495(7441):316–317, 2013.

[292] McGinity M, Shaw J, Kuchelmeister V, Hardjono A, and Favero DD. Avie: a versatile multi-user stereo 360 interactive vr theatre. In *Proceedings of the 2007 workshop on Emerging displays technologies: images and beyond: the future of displays and interacton*, page 2. ACM, 2007.

[293] Kaspar M, Dech F, Parsad NM, and Silverstein JC. *Web-based Stereoscopic Visualization for the Global Anatomy Classroom*, volume Studies in Health Technology and Informatics, pages 264–270. IOS Press, 2011.

[294] Techviz. URL `http://www.techviz.net/`, Last Accessed: 14. June 2013.

[295] The medical imaging interaction toolkit (mitk), 2009. URL `http://www.mitk.org/`, Last Accessed: 14. June 2013.

[296] Hughes CE, Stapleton CB, Hughes DE, and Smith EM. Mixed reality in education, entertainment, and training. *Computer Graphics and Applications, IEEE*, 25(6):24–30, 2005.

[297] Hasselberg M, Wolf I, Nolden M, Seitel M, Meinzer H, and Engelmann U. Mitk als telekonferenzfähiges plugin in der chili-workstation. In *Bildverarbeitung für die Medizin*. CEUR Workshop Proceedings, 2007.

[298] Arguinarena EJC, Macchi JE, Escobar PP, Del Fresno M, Massa JM, and Santiago MA. Dcm-ar: a fast flash-based web-pacs viewer for displaying large dicom images. *Conf Proc IEEE Eng Med Biol Soc*, 2010:3463–6, 2010. doi:10.1109/IEMBS.2010.5627827.

[299] CHILI. Digital radiology. URL `http://www.chili-radiology.com/de`, Last Accessed: 14. June 2013.

[300] Lietsch S and Marquardt O. A cuda-supported approach to remote rendering. *Advances in Visual Computing*, pages 724–733, 2007.

Mathias Kaspar

Education

01/2009 - 08/2013	**Ph.D. student** *Georg-August-University, Göttingen, Germany*
10/2005 - 02/2008	**Master of Science - Medical Informatics** *Georg-August-University, Göttingen, Germany* Thesis: Import of genomic data into clinical and medical research systems
10/2002 - 09/2005	**Bachelor of Science - Medical Informatics** *Georg-August-University, Göttingen, Germany* Thesis: Analysis of tools to create CDA-documents

Professional experience

Since 05/2012	**Research assistant** *Comprehensive Heart Failure Center, University Medical Center Würzburg, Germany*
04/2011 - 10/2011	**Research assistant** *NorthShore University HealthSystem, Evanston IL, USA*
10/2009 - 10/2011	**Visiting student** *Computation Institute, University of Chicago, Chicago IL, USA*
11/2011 - 04/2012 & 01/2009 - 09/2009	**Research assistant** *Georg-August-Universität, Dept. Medical Informatics, Göttingen, Germany*

04/2008 - 12/2008	**Software Test Engineer / Software Developer** *Siemens AG, Industry-Sector, Erlangen, Germany*
10/2006 - 03/2007	**Internship as software developer in the US** *Siemens Medical Solutions, Soarian Enterprise, Malvern PA, USA*
10/2004 - 02/2008	**Student research assistant** *Georg-August-University, Dept. of General Medicine, Göttingen, Germany*

Grants

10/2009 - 02/2010	**DAAD stipend for Ph.D. students**

Professional Memberships

Since 01/2012	**Deutsche Gesellschaft für Medizinische Informatik, Biometrie und Epidemiologie (GMDS)**

Publications

Mathias Kaspar, Nigel M. Parsad, and Jonathan C. Silverstein. An optimized web-based approach for collaborative stereoscopic medical visualization. J Am Med Inform Assoc, 20(3):535-43, 2012.

Mathias Kaspar, Fred Dech, Nigel M. Parsad, and Jonathan C. Silverstein. Web-based stereoscopic visualization for the global anatomy classroom. *Stud Health Technol Inform*, 2011, 163:264-270. IOS Press.

Mathias Kaspar, Nigel M. Parsad, and Jonathan C. Silverstein. Cowebviz – interactive collaborative sharing of 3d stereoscopic visualization among browsers with no added software. In *1st ACM International Health Informatics Symposium*, November 2010.

Benjamin Löhnhardt, Frank Dickmann, Matthias Quade, Daniela Skrowny, Sabrina Heppner, **Mathias Kaspar**, Nick Kepper, Dagmar Krefting, Thomas Steinke, and Ulrich Sax. Evaluation of visualization approaches in a biomedical grid environment. In *Sixth IEEE International Conference*, 2010.

Frank Dickmann, **Mathias Kaspar**, Benjamin Löhnhardt, Nick Kepper, Fred Viezens, Frank Hertel, Michael Lesnussa, Yassene Mohammed, Andreas Thiel, Thomas Steinke, Johannes Bernarding, Dagmar Krefting, Tobias A. Knoch, and Ulrich Sax. Visualization in health grid environments: A novel service and business approach. In *Grid Economics and Business Models: 6th International Workshop, GECON 2009*, Delft, the Netherlands, August 24, 2009, Proceedings, page 150. Springer, 2009.

Frank Dickmann, **Mathias Kaspar**, Benjamin Löhnhardt, Tobias A. Knoch, and Ulrich Sax. Perspectives of medigrid. *Stud Health Technol Inform*, 147, 2009.

Conference posters

Mathias Kaspar, Benjamin Löhnhardt, Nikolaus Kepper, and Dagmar Krefting. Poster: Providing easy access to visualization for biomedical research. In *13th International Congress on Medical and Health Informatics*, 09 2010.

Mathias Kaspar, Benjamin Löhnhardt, Nikolaus Kepper, Tobias A. Knoch, Dagmar Krefting, Thomas Steinke, Fred Viezens, Frank Dickmann, and Ulrich Sax. Poster: Interaktive 3d/4d-visualisierung in biomedizinischen grid-infrastrukturen. In *54. Jahrestagung der Deutschen Gesellschaft für Medizinische Informatik, Biometrie und Epidemiologie e.V. (GMDS)*, 2009.

Würzburg, July 3, 2013