Remote Rendering Strategies for Large Biological Datasets

Wolfram Schoor[1,3] and Marc Hofmann[1] and Simon Adler[1,3] and Werner Benger[2] and Bernhard Preim[3] and Rüdiger Mecke[1]

[1]Fraunhofer Institute for Factory Operation and Automation, Germany

wolfram.schoor,marc.hofmann,simon.adler,ruediger.mecke@iff.fraunhofer.de

[2]Center for Computation and Technology, Louisiana State University, USA

werner@cct.lsu.edu

[3]Otto von Guericke University Magdeburg, Germany

preim@isg.cs.uni-magdeburg.de

Abstract

This paper presents remote rendering technologies to support the analysis and exploration of large biological datasets evaluated for a medium sized biology research institute. Reconstructed three-dimensional models and other data domains stored in a database are the foundation for the visualization. Remote Rendering via a web-based client enables a widespread access to the data with interactive frame rates and high-quality rendering results without the demand of new client hardware. Moreover, the rendering server has the capability to manage multiple independent or collaborative sessions and creates only a very small network traffic over time. Two worst case scenarios were tested i) a network connection with very small bandwidth, and ii) a thin graphical server solution for rendering.

1 Introduction

Remote Rendering commonly means interactive visualization of three-dimensional datasets via a client/server architecture over a network. Visualization of complex 3D models is still a task hard to accomplish without specialized hardware. Furthermore, ad-hoc visualizations over distance are playing a key role in future visualization trends. One problem of visualization is that huge amounts of data must be handled, which requires not only a fast CPU and high memory capacities (RAM), but also a high degree of graphics output manageable by modern graphics cards.

These resources cannot be covered and the data amount is increasing faster than processors [Moore, 1998]. Another aspect is that the data must be

stored somewhere, and a heterogeneous data storage on different clients is not acceptable.
These drawbacks can be overcome by the concept of remote rendering, also called remote visualization. The datasets are computed on a rendering machine with high graphics capabilities - the so-called *server*, and only rendered images are sent to the working station - called *clients*. For remote visualization purposes the following key issues denote indicators of quality:

- Network latency
- Network throughput
- Error recovery
- Security issues
- Hardware/software incompatibility
- Graphics performance (fps)
- Scalability (user number)
- Network robustness
- Balance (local/remote resources)

The work presented here is motivated by a given shortcoming regarding biological model visualization, namely limited bandwidth, slow clients with poor graphics capabilities, large biological 3D models and the demand to real-time interaction. The latter cannot be covered by existing conventional remote rendering techniques. Therefore, a remote rendering architecture with reduced network traffic generating correct high-quality renderings for low level end user PCs with interactive frame rates is given in this paper.

2 Related Work

This section briefly shows the stages of remote rendering techniques from the past to today, presents a possible classification of remote rendering applications dealing with polygonal models, and introduces established applications. The treatment of other model representation forms will not be part of this paper. Nevertheless, the principle way of converting these representations into a polygonal representation as pre-step can be performed.

2.1 Historical Outline

The idea of remote rendering/visualization is not new. Various remote rendering applications have been developed in the past.
One of the most popular remote rendering applications for 2D systems was the X Window protocol [Scheifler & Gettys, 1986] for X systems. The extension to transfer 3D primitives is the OpenGL Stream Codec (GLS) [Dunwoody, 1996] introduced by SGI.
A standard technique for remote display of 3D applications is GLX [Womack & Leech, 1998], the "OpenGL Extension to the X Window System"[1]. Using GLX for remote rendering, GL commands are encoded by the client side's API and sent to the X server within GLX packages. These commands are decoded by the X server and submitted to the OpenGL driver for rendering on graphics hardware at the server.

[1] The pendant for Windows operating systems is *WGL* and *CGL* for Mac OS X respectively.

2.2 Classification

Adapted from [Schmalstieg & Gervautz, 1996], remote rendering techniques of 3D geometry models can be classified as follows:

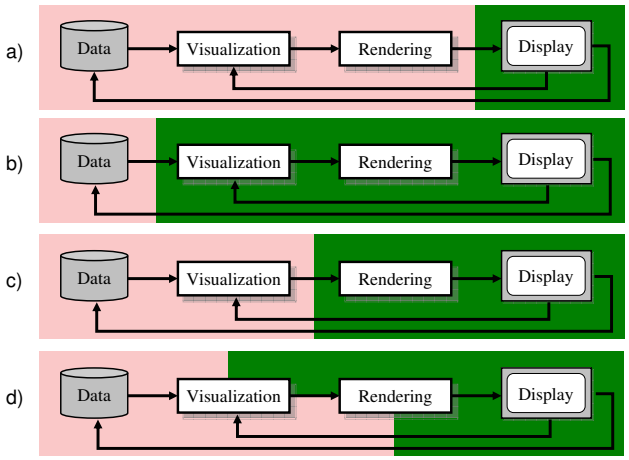


Figure 1: Remote rendering types a) image-based, b) geometry replication, c) immediate-mode, and d) hybrid rendering

1. *Image-based:*
   Rendering is performed by the sender, and the resulting stream of pixels is sent over the network as provided in [Scheifler & Gettys, 1986] (see Figure 1 (a)).

2. *Geometry-based:*
   A copy of the geometric database is stored locally to be accessed by the rendering process. The database can either be available before application start (kept on local hard disk), or downloaded just before usage, such as VRML / X3D browsers can do [Gueziec et al., 1999], [Web3D Consortium, Inc., 1998] and [Web3D Consortium, Inc., 2004] (see Figure 1 (b)).

3. *Immediate-mode drawing:*
   The low-level drawing commands used by drawing APIs are issued by the application performing the rendering. They are not immediately executed, but sent over the network as a kind of remote procedure call. The actual rendering is then performed by the remote machine (e.g. distributed GL [Shreiner et al., 2005]), see Figure 1 (c).

4. *Hybrid rendering:*
   In this approach a 3D geometry as low resolution representation is rendered on the client's side and the server transmits images to the client

as well. The used mode depends on the performed user action (computational vs. network load) [Koller et al., 2004] or is dynamically adapted to the available network capacities (see Figure 1 (d)).
Another quite different hybrid rendering approach is presented in [Lin et al., 2007]. JPEG images are used to stream embedded 3D model information decoded as RGB information.

2.3 Existing Remote Visualization Tools

A variety of remote visualization applications exists in the IT universe. Many of them are well established and cover a wide range of different visualization capabilities or even have their focus on cluster applications.
In the following, a brief overview of existing remote visualization tools will be given. This list is not intended to be exhaustive.

Application	Image-based	Geometry-based	Hybrid	Collaboration
HP Remote Rendering Service [Hewlett-Packard, 2007]	+	-	-	+
SGI VizServer [SGI Inc., 1999]	+	-	-	+
Sun Shared Visualization [Sun Microsystems, Inc., 2008]	+	-	-	+
Chromium [Humphreys et al., 2002]	+	+	o	o
VR-Juggler [Bierbaum et al., 2001]	o	+	-	+
Scan View [Koller et al., 2004]	+	+	+	-

Table 1: Comparison of remote rendering tools,
(+) available, (-) not available, (o) unknown

2.4 Findings

Summarized, existing remote rendering techniques cover a wide range of problem statements including image-based remote renderings (suitable for fast network connections) or geometry-based remote renderings (e.g. cluster rendering with sufficient client's graphics hardware). Some of these approaches support collaborative remote rendering as well.
Hybrid rendering methods for their capabilities could be identified as adequate rendering method for low and medium level clients using networks with medium network speed.

3 System Demands

In this section the requirements of the involved underlying system architecture and its components as well as special user interests are discussed with respect to the chosen use case; a medium sized biology research institute.
Main issues are the individual clients' hardware configurations in the network, the data transfer capabilities and the state of scientific data in the investigated institution.

3.1 Clients Configuration

A medium-sized scientific biology research institute with approximately 300 potential clients was evaluated. These clients are completely heterogeneous, they cover a big inventory of out-of-date machines[2] as well as a few state-of-the-art hardware machines. The utilized systems are as diverse as the clients' hardware configurations. Whereas the processor speed is widely feasible, the graphics hardware is not. An nVidia GeForce2 MX is for example capable of rendering 20 million triangles per second (see `www.nvidia.com`). This means that at most $800,000$ triangles can be rendered per frame in real-time during interaction. One biological dataset consists of more than 80% more triangles than this hardware can manage. These preconditions have to be taken into account in the design of a remote visualization application for large biological models.

3.2 Server Configuration

Two different server machines could be used for testing the remote visualization performance.
The first one is a high performance computer with two Quad-Core Opteron 2.6 GHz, 16 GB RAM, a Quadro FX 5600 graphics card and furthermore a Tesla S870 processor with 500 GFlop and a shared 1 GBit connection (bottleneck) to the "Deutsches Forschungsnetz" (DFN) and to the biology research institute. An average server upload B_{up} of 2 MBit with average ping rates t_{avg} lesser than 23 ms could be identified ($\sigma_{B_{up}} \approx 0.2$ MBit, $\sigma_{t_{avg}} 3$ ms).
The second server is a minimal graphics server configuration with a 2.8 GHz Pentium IV with 4 GB RAM and *WindowsXP* as operating system. The rendering job is done by one *Nvidia 7800 GTX* graphics card. This server is directly located at the biology research institute. The minimum network connection for this setup is bound to the local switches and achieves 100 MBit.

3.3 Data Transfer

The data transfer capabilities are one major aspect in the remote data visualization and can be formulated regarding to Equation 1.

$$B \geq \frac{(R_w \; R_h) \; C_{depth} \; f \; n}{c_{factor}} \tag{1}$$

[2] e.g. *Win98* Pentium III's with less or equal 500 MB RAM

Without any compression c_{factor} and for a screen resolution R_wxR_h (XGA), a color depth C_{depth} of 3 Bytes/pixel and a frame rate f of 26 frames/sec for only one client n, this would require at least a bandwidth B of:

$$B \geq \frac{(1024 * 768)\ pixel * 3\ Bytes/pixel * 26\ sec^{-1} * 1}{1}$$

$$B \geq 60\ MByte/s \tag{2}$$

which is still high and only possible with at least 1 GBit/s ethernet[3] (see example calculation in Equation 2).
Using JPEG compression this can be reduced by a factor c_{factor} of $10 - 300$ (depending on the resulting image quality) and will also work with slower connections. A compression factor of 50 will result in a data transfer rate of approximately 1200 kByte/s which might fit in a 10 MBit data connection, if the other users produce only very low amounts of traffic. This means a permanent data stream would load the network traffic and also prevent highly detailed images on the clients' side.
The local network speed at the biology research institute depends on the used switches and includes 100 MBit as well as 1 GBit connections. This fact prevents a standard remote rendering strategy due to the reason that high-quality renderings would not fit into a 100 MBit connection if a couple of other users share the same server connection (big n).

3.4 Data Resources

The biological data pool consists of different data domains from biological material. The internal knowledge of the data's specific characteristics can be advantageous to find solutions for a well suited remote visualization approach.
For the visual analysis, specifically microtome serial section data of different stages of barley seed development (*days after flowering* or DAF) are used, similar to the method described in [Gubatz et al., 2007].
Each dataset consists of approximately $2,000$ slice images of barley caryopsis, which were obtained from a microtome at 3 μm slice thickness. Digitized with a color CCD camera, images originally measured 1600×1200 pixels at a spatial resolution of 1.83×1.83 μm per pixel. Since color-space analysis revealed an almost linear correlation and a limited range, it was possible to reduce the optical resolution of 12 bits for each RGB channel to 8 bit grayscale images without significant losses [Schoor et al., 2008a], thus reducing the data volume for each dataset to approximately 5 GB. Depending on the developmental stage of the specimen, sections can be too large to be captured in a single frame at the given spatial resolution.
Available data include gene expression assays utilizing mRNA *in-situ hybridization* (ISH) probing of histological cross-sections. Whereas *in-situ* data allows

[3] An $SXGA$ screen resolution requires respectively around 100 MByte/s.

visualization of spatial gene expression patterns, staining with gene-specific samples requires complex chemical protocols, for that ISH image data origins from different individual caryopsis.
Magnetic resonance spectroscopy measurements (^{1}H-NMR) of caryopses at different points in time were sampled in a specific device. While being nondestructive, the detected proton distribution has a lower spatial resolution of approximately $10\mu m$ (isotropic) per voxel, and does not resolve histology or structural features.
Other potential datasets, such as macroarray gene expression data or metabolomic assays by laser microdissection (LMD), could be integrated into the database to enhance synergy and facilitate the inference of analysis results. Figure 2 presents the database with different data domains and an example image, and Figure 3 a pie chart breaking down data distribution per model. The organization is derived from the *entity-attribute-value* (EAV) design as in [Anhøj, 2003]. This approach has the advantage of high flexibility when extending the database with other data domains.

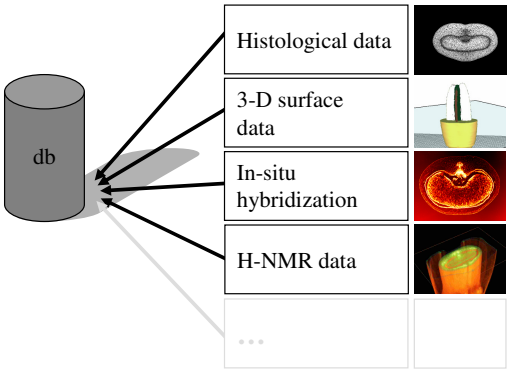


Figure 2: Schematic view: Integration of different data domains by using an *EAV* model for the visualization database.

3.5 Visualization Aspects

Another very important precondition is the range of functions of the remote rendering application to develop. At the moment the bio researchers test a software application which is especially tailored to them, namely the *BioVizIt* application. This software is initially a single-user stand alone application, which is not network compatible.
Among others, the scope of this solution covers the following features and actions:

- scene graph,
- transparency and color settings to objects,

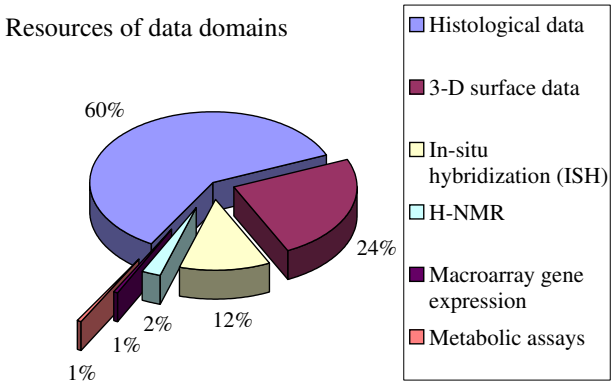


Figure 3: Data distribution for one model, the absolute data totaling around 9 GB

- hide, copy, delete, create, rename, group and ungroup objects,
- affine transformations, model deformations and model measurements,
- variable and textured clipping planes per object,
- image segmentation techniques,
- annotations

Another point the application must meet is the production of high-quality renderings on demand. All these features and actions have to be considered as part of the resulting remote rendering application to best possibly support the biologist's work.

3.6 Findings

As conclusion, there are different possibilities to perform remote visualization tasks as proposed in Section 2. As possible candidates only those solutions are appropriate which meet the requirements of the Section 3.1-Section 3.5 .
The clients' hardware denotes that the datasets to be worked with are too large and too complex to be viewed with local resources. The standard approach for remote visualization (pure image transfer)is not sufficient for this application due to the fact that rendered images with complex color patterns and differing inter-frame correlations typically lack in being processed and transmitted with interactive frame rates.
As solely remote rendering approach which fits the system requirements a hybrid rendering approach is applicable. Due to the specific visualization requirements depicted in this section a conception with an explicit control to data exchange mechanisms and adaption possibilities of the visualization is necessary. A solution to achieve this is to extend the existing *BioVizIt* application with specific hybrid remote rendering capabilities.

4 The Concept of a Hybrid Remote Rendering Approach

In the following, the remote visualization concept is described, where a hybrid rendering approach is the basis. Furthermore, the architecture's adaptions of the BioVizIt platform, referring to network compatibility and approaches addressing limited bandwidth, are described.

4.1 Hybrid Rendering

Figure 4 illustrates the idea of the hybrid approach. In the beginning of the

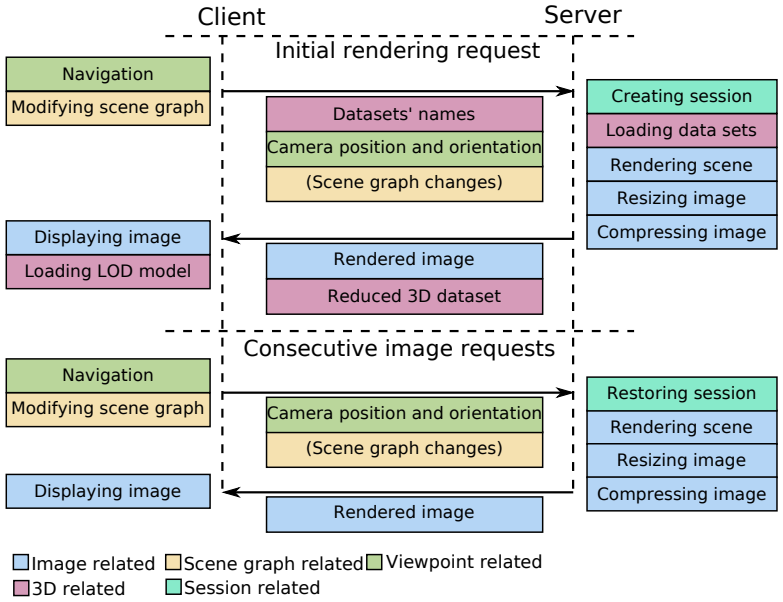


Figure 4: Sequential diagram of the proposed hybrid rendering approach

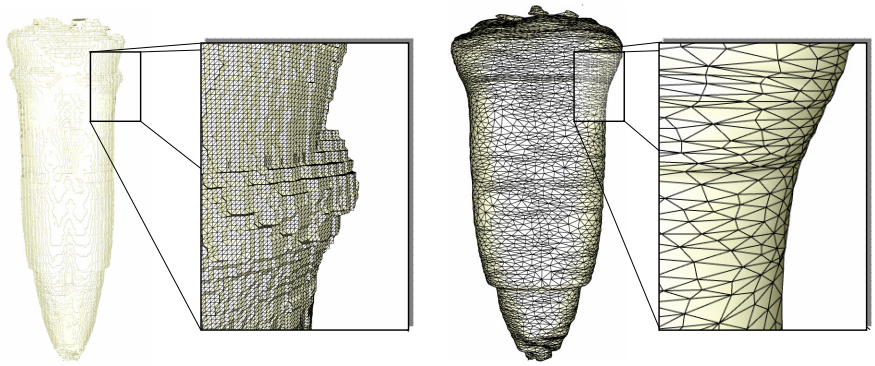
rendering process (initial rendering request) a request with datasets' names and initial settings is sent to the server. As a result, the server application is loading the requested file from the database, and in addition a low resolution 3D geometry model (*level of detail* also *LOD*) is transferred to the client. Moreover, the graphics server submits the first remote-rendered image to the client.

If in the following, user navigation is performed (consecutive rendering requests), the local 3D model is rendered by the client itself. Fast movements can be achieved without having performance problems due to network limitations. The local 3D model can be reduced (detail loss) up to one per mill (approximately 4-5 MB) of the original data amount of the 3D model. This is equivalent to the data amount necessary performing real-time server-sided

image rendering in XGA mode for two frames. Even if the bandwidth is less than 10 MBit an initial loading delay for an unoptimized data transfer of 4 seconds for the complete low resolution model would be acceptable to the user, if afterwards a stable real-time interaction can be performed.
In case of stopping the navigation or changing the scene graph for example, a command is immediately sent to the server (containing at least the new camera position and orientation) to request a new rendered image. This approach ensures real-time model interactions with a high-quality rendering result during the model exploration.
A comparable approach is used by a variety of programs which display only a subset of the whole dataset or even a faster rendering mode (point rendering) to ensure real-time model interaction, for example *Polyworks*[4].

Geometry Submission

The server stores the information of the original 3D models and also in coarser resolutions (*LOD*). This storage overhead is uncritical because data storage is not the limiting size for this application. Datasets are generated according to the proposed technique in [Schoor et al., 2008b] (see Figure 5). If a dataset



a) b)

Figure 5: Surface models of different grains' outer hull at different resolutions a) grain with a mesh size of 560,000 triangles and detail with highlighted faces b) reduced mesh with approximately 10,000 highlighted faces and detail with highlighted faces

is requested, the original dataset is loaded into the server application. At the same time a low resolution model is transferred from the database to the client.

Image Submission

The image submission by demand is a concept to overcome the server network load. Additionally, image compression algorithms [Taubman & Marcellin, 2001] and residual images (see [Yoon & Neumann, 2000]) are used to reduce the network traffic.

[4] `http://www.innovmetric.com`

4.2 Network Interconnect

Another important aspect besides the underlying data transfer per se is how the data will be transported. To achieve high resolution rendering results on the client, only a reliable data transfer method is an option. This excludes for example *UDP* or *UTP* [Thibault et al., 2002] data transfer, because in such data transfer mode the correct rendering result is not warranted due to unordered or lost packages. The network protocol layer *TCP* is sufficient for the required purposes, albeit *UDP* is faster, for example.
By contrast the TCP connection ensures the correct receiving of either geometry models or rendered images.

4.3 Architecture Adaption of the BioVizIt Platform

For an extension of the existing solution as server variant the range of functions must be appended to network and session routines. Multiple network connections through clients must be promptly accepted and processed, using multithreading techniques. User events are registered in a message queue. This queue is held in the operating system and members of the queue are passed to the application (for example panning the window). Received messages by the application are then handled in a procedure of the primary thread.
To avoid blockages in the processing of this thread, a new thread – the "server thread" – is generated during the program start to wait for new client requests via Transmission Control Protocol (TCP).
If a client connects to the server over a predefined port, a new "client thread" is generated immediately for this client. This thread is responsible for further communication with the client.
The client thread reads the request from the server's socket thread and interprets it through the HyperText Transfer Protocol (HTTP). By accessing a virtual document, additionally the document area of the HTTP request is used, which includes the remote rendering control commands of the client for the server. These commands are processed sequentially.
At the end of this processing the client is enqueued to a list, which contains clients awaiting an image synthesis step.

5 Implementation and Results

In this section, the essential aspects of the implementation are presented. The identified key issues for remote visualization are examined regarding the presented approach.
Network latency The latency at the beginning of a session depends on how the geometry transmission is handled. The coarse 3D model can first be accessed after full model upload. This can take a few seconds for the geometry transmission ($3 - 4$ s for server case 2). This is an initial step which is only performed

once a model is loaded. After that, the interaction is performed with latencies < 30 ms due to local rendering.
The rendering result in high resolution takes less than 1 second after the mouse button is released (indicating the end of user navigation). The time needed to display the result on the client is the sum of the steps which are shown in Figure 6.

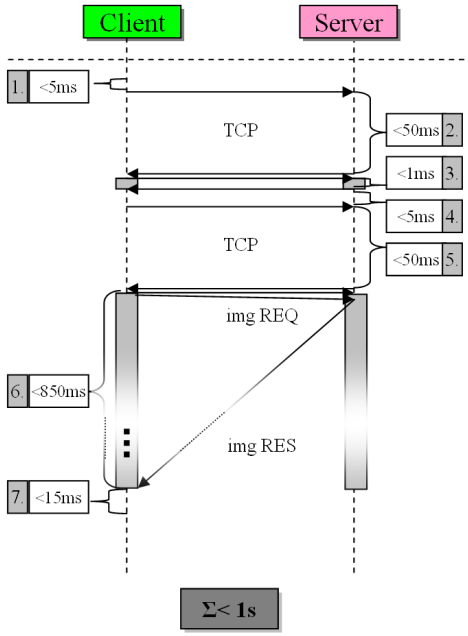


Figure 6: Sequential diagram of server and client transmissions with approximate time spans for one remote rendering step for server case 2

The sequences in Figure 6 are:

1. the control signal from interaction device (mouse) to the client
2. establishing TCP connection (3-way handshake)
3. sending the rendering information via html request (position, orientation, fov, etc.)
4. doing the rendering job on the server
5. establishing TCP connection again
6. sending an image request to the server and getting the respective data
7. using these data to display the scene in high resolution

Graphics performance The graphics performance of the client rendering stage depends on the clients hardware and on the level of detail in which the 3D model was submitted. For the tested application even on a Pentium III client a coarse 3D model with $20,000$ points could be interactively visualized.
During the observation of the model (no model interaction) no rendering step is required. That means that a real-time application can be guaranteed for the model interaction.
Network throughput The network throughput per high resolution image request is equivalent to the physical size of this image and corresponds to approximately 300 kB. With an initial geometry transfer of 4-5 MB for the low resolution 3D model and an average image request of 3-5 s, a data volume per client of 10 MB is transmitted within the first minute.
Scalability The approach provided here has a good scalability. The rendering of scenarios, such as described in Section 3, can be accomplished on graphics servers without problems with interactive frame rates (Case 2: minimum graphics server solution). That means, in the worst case[5] the remote rendering image of the last handled client will be displayed with an additional delay of approximately 1 s if the data transfer is ignored. This is the theoretical maximum number of users which can be handled by the server under the assumption that a delay of more than 2 seconds is not acceptable to users.
Taking the 100 MBit connection as lower bound for the data channel (see Section 3), more than 40 clients can request a full resolution image at the same time without causing additional delay at a client station due to the data transfer. Assumed that a 2 MBit connection is the average connection configuration for the server case 1, and a full image resolution is requested, only one client request per second is possible without causing an additional delay on the client.
Error recovery The error recovery is automatically performed by the network protocol layer (TCP) which provides explicit error corrections. Furthermore, a transmission control and data ordering is also part of this protocol. This is paid with a higher latency of the remote rendering.
Network robustness In general, the failure of a network connection in the beginning of a work session leads to an insufficient remote visualization. During the session a temporary failure of the network connection in the time scale of seconds can be bridged without problems due to the local downsampled version of the 3D model. A loss of connectivity within a larger time range disallows a practical use of the here provided technique due to the missing detail information the remote-rendered image provides.
Security issues The data pool of plant biological data consists of confidential information. Even an internal use must contain at least a simple protection mechanism due to project internal information. Therefore, a simple login query was integrated to authenticate the users with respective project rights. Furthermore, each session on the remote visualization server is initially hidden to other users.
This approach does not cover any wiretap operation or similar attacks via net-

[5] e.g. if the server gets a synchronous request of 25 clients

work. The image data as well as the 3D low resolution models are submitted plain without protection mechanism up to now. Possible are techniques like in [Koller et al., 2004] which provides small shifts in the submitted image data (jitter) to prevent a reconstruction of the high resolution 3D model data or use additionally a secure transfer protocol (e.g. HTTPS).
Balance between local/remote resources The architecture of the hardware at the scientific institution is naturally very heterogeneous. A uniform strategy to extensively use clients' hardware is therefore not applicable. As the least common denominator, a low resolution 3D model, which is tailored to the actual hardware configuration, is used at the clients' side to achieve real-time interaction.
Hardware/software incompatibility The visualization software is written in *C++*, uses only libraries like *wxWidgets* and *OpenGL*, and is therefore platform-independent for supported systems. The visualization service tests for hardware premises, if specific features are not supported by actual server hardware, alternatively slower CPU-based implementations exist (e.g. deformation model).
The client's visualization front-end uses a platform-independent web interface, based on *Java® Version* 1.5, *Java 3D™*. This front-end was tested with *Firefox 3.0*, *Internet Explorer 6.0 / 7.0*, *Opera*, and *Chrome Beta*. The application even runs on a mobile device with *Windows CE*. Incompatibilities are not known so far. The tests covered only the general operating.

Figure 7 shows a screenshot of the provided web interface and the visualization options.

6 Conclusion

In this paper a remote rendering approach was presented, which is applicable to medium-sized enterprises who have to deal with a large amount of scientific data. With a minimum of effort and the use of existing hardware this approach allows to visualize, explore and modify data. It could be shown that a simultaneous use of 5 high-level sessions could be managed with no recognizable loss of performance even with the server case 1.
The clients hardware constitution must not be state-of-the-art which makes this approach of visualization furthermore attractive in a financial manner. Instead of the high end graphics server (case 1), the local visualization server with the minimal graphics configuration (case 2) is to prefer. The network bandwidth could be identified as limiting constraint respective to the number of clients contemporaneously used.

7 Future Work

If the remote rendering application will be used with many users, the aid of more than one rendering machine can be usefull. In the next term a larger focus will

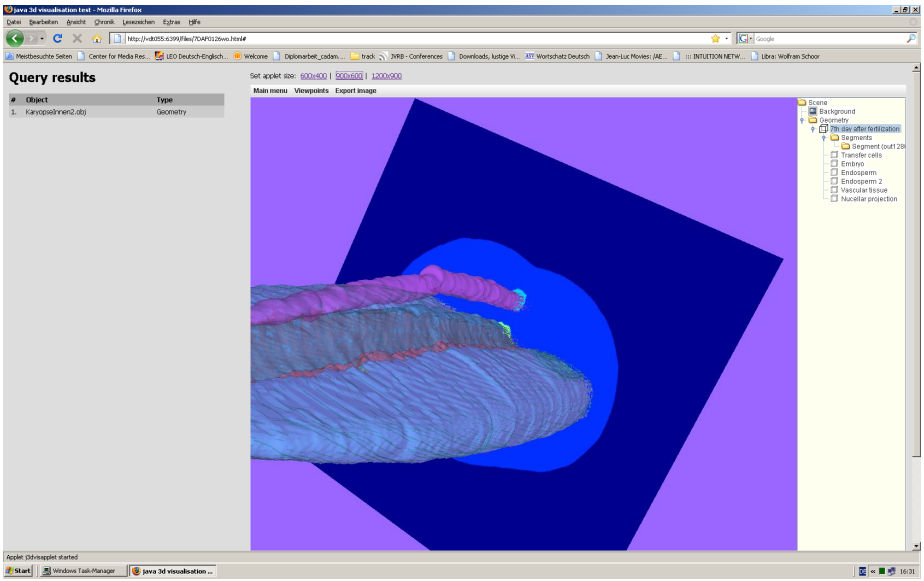


Figure 7: Screenshot of the clients web interface showing a reconstructed model of a caryopsis with clipping plane

be placed on data modification possibilities and supporting full functionality to biologists in comparison to the stand-alone BioVizIt application.
A linking of the VISH platform [Benger et al., 2007] with the provided Java web-interface is planned.
Applying non photorealistic rendering (NPR) aspects like the line drawings based on multi-resolution meshes [Ni et al., 2006] or feature-line-based city visualizations [Quillet et al., 2006] to remote rendering applications, this can improve the performance if using a purely image-based remote rendering approach.
It is planned to expand the intranet-based solution to the internet for selected public datasets so that interested users can interactively explore information within the provided 3D models from the institutes website.

8 Acknowledgments

This research work is being supported by the BMBF grants 0313821B and 0313821A. The authors would also thank the DFG for supporting the cooperation preparative between the Fraunhofer Institute and the Louisiana State University by the grant SCHO 1346/1-1.

References

[Anhøj, 2003] Anhøj, J. (2003). Generic design of web-based clinical databases. *Journal of Medical Internet Research*, 5(4), e27. URL: `http://www.jmir.org/2003/4/e27/`.

[Benger et al., 2007] Benger, W., Ritter, G., & Heinzl, R. (2007). The Concepts of VISH. In *Proceedings of the 4th High-End Visualization Workshop: Open issues in visualization with special concentration on applications in astrophysics, numerical relativity, computational fluid dynamics and high-performance computing*.

[Bierbaum et al., 2001] Bierbaum, A., Just, C., Hartling, P., Meinert, K., Baker, A., & Cruz-Neira, C. (2001). VR Juggler: A Virtual Platform for Virtual Reality Application Development. In *VR '01: Proceedings of the Virtual Reality 2001 Conference (VR'01)* (pp. 89–96). Washington, DC, USA: IEEE Computer Society.

[Dunwoody, 1996] Dunwoody, C. (1996). *The OpenGL stream codec : A specification.* Technical report, Silicon Graphics, Inc.

[Gubatz et al., 2007] Gubatz, S., Dercksen, V. J., Brüß, C., Weschke, W., & Wobus, U. (2007). Analysis of barley (Hordeum vulgare) grain development using three-dimensional digital models. *The Plant Journal*, 52(4), 779–790.

[Gueziec et al., 1999] Gueziec, A., Taubin, G., Horn, B., & Lazarus, F. (1999). A Framework for Streaming Geometry in VRML. *IEEE Computer Graphics and Applications*, 19(2), 68–78.

[Hewlett-Packard, 2007] Hewlett-Packard (2007). Advantages and Implementation of HP Remote Graphics Software HP Remote Graphics Software enables 2D and 3D real-time interactive graphics and collaboration from a distance. White Paper.

[Humphreys et al., 2002] Humphreys, G., Houston, M., Ng, R., Frank, R., Ahern, S., Kirchner, P. D., & Klosowski, J. T. (2002). Chromium: a stream-processing framework for interactive rendering on clusters. *ACM Trans. Graph.*, 21(3), 693–702.

[Koller et al., 2004] Koller, D., Turitzin, M., Levoy, M., Tarini, M., Croccia, G., Cignoni, P., & Scopigno, R. (2004). Protected interactive 3d graphics via remote rendering. *ACM Trans. Graph.*, 23(3), 695–703.

[Lin et al., 2007] Lin, N.-S., Huang, T.-H., & Chen, B.-Y. (2007). 3D Model Streaming based on JPEG 2000. *IEEE Transactions on Consumer Electronics*, 53(1), 182–190.

[Moore, 1998] Moore, G. E. (1998). Cramming more components onto integrated circuits. *Proceedings of the IEEE*, 86(1), 82–85.

[Ni et al., 2006] Ni, A., Jeong, K., Lee, S., & Markosian, L. (2006). Multi-scale line drawings from 3d meshes. In *I3D '06: Proceedings of the 2006 symposium on Interactive 3D graphics and games* (pp. 133–137). New York, NY, USA: ACM.

[Quillet et al., 2006] Quillet, J. C., Thomas, G., Granier, X., Guitton, P., & Marvie, J. E. (2006). Using expressive rendering for remote visualization of large city models. In *Web3D '06: Proceedings of the eleventh international conference on 3D web technology* (pp. 27–35). New York, NY, USA: ACM.

[Scheifler & Gettys, 1986] Scheifler, R. W. & Gettys, J. (1986). The X window system. *ACM Trans. Graph.*, 5(2), 79–109.

[Schmalstieg & Gervautz, 1996] Schmalstieg, D. & Gervautz, M. (1996). Demand-driven geometry transmission for distributed virtual environments. *Computer Graphics Forum*, 15(3), 421–431.

[Schoor et al., 2008a] Schoor, W., Bollenbeck, F., Hofmann, M., Mecke, R., Seiffert, U., & Preim, B. (2008a). Automatic Zoom and Pseudo Haptics to Support Semiautomatic Segmentation Tasks. In V. Skala (Ed.), *16th WSCG 2008*, WSCG'2008 Full Papers Proceedings (pp. 81–88).: WSCG University of West Bohemia. Full Paper.

[Schoor et al., 2008b] Schoor, W., Bollenbeck, F., Weier, D., Weschke, W., Preim, B., Seiffert, U., & Mecke, R. (2008b). VR-Based Visualization and Exploration of Plant Biological Data. *Journal of Virtual Reality and Broadcasting*. submitted.

[SGI Inc., 1999] SGI Inc. (1999). SGI® OpenGL Vizserver™ 3.5 Visualization and Collaboration (Application-Transparent, Remote, Interactive). White Paper.

[Shreiner et al., 2005] Shreiner, D., Woo, M., Neider, J., & Davis, T. (2005). *OpenGL(R) Programming Guide: The Official Guide to Learning OpenGL(R), Version 2 (5th Edition) (OpenGL)*. Addison-Wesley Professional.

[Sun Microsystems, Inc., 2008] Sun Microsystems, Inc. (2008). Seeing the Future with the Sun™ visualization system: Scaling, sharing, and scheduling visualization resources. White Paper.

[Taubman & Marcellin, 2001] Taubman, D. S. & Marcellin, M. W. (2001). *JPEG 2000: Image Compression Fundamentals, Standards and Practice*. Norwell, MA, USA: Kluwer Academic Publishers.

[Thibault et al., 2002] Thibault, S., Cavin, X., Festor, O., & Fleury, E. (2002). Unreliable transport protocol for commodity-based opengl distributed visualization. In *Workshop on Commodity-Based Visualization Clusters, Boston, MA*.

[Web3D Consortium, Inc., 1998] Web3D Consortium, Inc. (1998). *ISO/IEC 14772-1:1998: Information technology — Computer graphics and image processing — The Virtual Reality Modeling Language — Part 1: Functional specification and UTF-8 encoding*. International Organization for Standardization.

[Web3D Consortium, Inc., 2004] Web3D Consortium, Inc. (2004). *ISO/IEC 19775-1:2004 Information technology — Computer graphics and image processing: Extensible 3D (X3D) — Part 1: Architecture and base components*. International Organization for Standardization. URL: `www.web3d.org/x3d/specifications`.

[Womack & Leech, 1998] Womack, P. & Leech, J. (1998). *OpenGL graphics with the X Window System, version 1.3*. Technical report, Silicon Graphics, Inc.

[Yoon & Neumann, 2000] Yoon, I. & Neumann, U. (2000). Ibrac: Image-based rendering acceleration and compression. *Eurographics 2000, Vol*, 19, 321–330.