A 3D Data Intensive Tele-immersive Grid

Benjamin Petit
INRIA - LIG

Thomas Dupeux
INRIA - LIG

Benoit Bossavit
INRIA - LABRI

Joeffrey Legaux
Université d'Orléans

Bruno Raffin
INRIA - LIG

Emmanuel Melin
Université d'Orléans

Jean-Sébastien Franco
INRIA - LABRI

Ingo Assenmacher
INRIA - LIG

Edmond Boyer
INRIA - LJK

ABSTRACT

Networked virtual environments like Second Life enable distant people to meet for leisure as well as work. But users are represented through avatars controlled by keyboards and mouses, leading to a low sense of presence especially regarding body language. Multi-camera real-time 3D modeling offers a way to ensure a significantly higher sense of presence. But producing quality geometries, well textured, and to enable distant user tele-presence in non trivial virtual environments is still a challenge today.

In this paper we present a tele-immersive system based on multi-camera 3D modeling. Users from distant sites are immersed in a rich virtual environment served by a parallel terrain rendering engine. Distant users, present through their 3D model, can perform some local interactions while having a strong visual presence. We experimented our system between three large cities a few hundreds kilometers apart from each other. This work demonstrate the feasibility of a rich 3D multimedia environment ensuring users a strong sense of presence.

Categories and Subject Descriptors

H.5.1 [**Multimedia Information Systems**]: Artificial, augmented, and virtual realities

General Terms

Experimentation

1. INTRODUCTION

3D networked virtual environments first targeted the gamer community, but applications like Second Life show they can also effectively support social networks and distributed workplaces. Such environments settle the foundations of what may become a new 3D interactive media. However they still lack many features to enable rich human interactions, amongst them the ability to ensure a strong presence of each user, and the possibility to support data intensive environments.

User presence is today limited to an avatar controlled by simple devices, mainly a keyboard and a mouse. Though this avatar can show some likeness with its user, its abilities for body language are almost nonexistent, significantly impairing the user virtual presence. Camera-based 3D modeling is a promising way of computing in real-time a 3D clone of the user. The 3D geometry enables full-body mechanical interactions, triggered for instance when detecting collisions with virtual objects. Once textured with the photometric data extracted from the cameras, the 3D model captures the user appearance and conveys its body language. Difficulties include computing a good quality model, transmitting it at a frame rate and a latency ensuring a good level of tele-presence. An adapted display environment is also required to experience a convincing 3D immersion in the virtual environment.

Materials for discussions may involve large and complex data sets. For instance coordinating actions to face a natural disaster requires to have access to multiple data, some pre-existing terrain data complemented by simulation data, information from on-line sensors or observations from people on the field. It raises issues related to data access, sharing and aggregation.

Existing tele-immersive systems achieve various levels of presence, depending on the number and layout of cameras and the 3D modeling algorithm used. Some systems offer a free viewpoint on the observed user, enabling only the visual presence of the user [9]. Other systems compute a partial 3D model based on a depth map [10, 6, 3, 13]. It provides a partial visual and geometrical presence (thickness of modeled objects is unknown for instance). Multi-camera systems can achieve a full-body visual and geometrical presence, some point based algorithms rather focusing on visual presence [4], while mesh based algorithms are more versatile [11, 12].

Most existing tele-immersive experiments associate distant users in a simple 3D environment with limited interaction capabilities. To our knowledge, the collaborative geoscience experiment between UC Berkeley and UC Davis is one of the few tele-immersive experiments involving complex data [1].Two persons are interacting around a scientific data set, one being present through its 3D model computed from a multi-camera system, while the other is immersed in a CAVE.

This paper and the attached video show that associating multi-camera 3D modeling, data grid and interactions

MM'10, October 25–29, 2010, Firenze, Italy.

[1] See http://tele-immersion.citris-uc.org

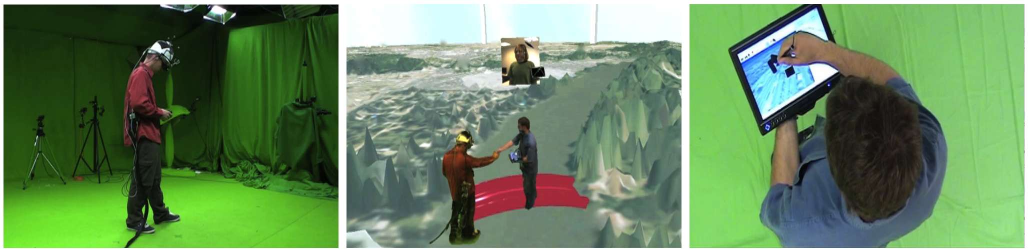

Figure 1: Left: user at Grenoble wearing an HMD in the 3D modeling studio. Middle: all users meet in the virtual environment. Right: user at Bordeaux using a touch pad.

can lead to a rich 3D multimedia experience. We set-up a distributed application involving 3 cities a few hundreds of kilometers apart from each other. At Orléans, a user has simple tele-presence and interaction capabilities (one camera, one large display). The second site at Grenoble hosts a multi-camera system. The user, recorded by the cameras, is modeled in 3D at the camera frame rate. This provides a textured 3D mesh ensuring the user 3D tele-presence on distant sites. On this site the user is equipped with a tracked head-mounted display, ensuring an intense feel of immersion in the shared 3D scene. This site hosts a terrain server providing data to be rendered by each of the three sites. The last location, Bordeaux, also has a multi-camera system for real-time 3D modeling, while visualization takes place on a large display. The user 3D interactions are performed through a touch pad. In the 3D scene, each user can freely move on the terrain and add some simple objects. Users are present in the scene through their 2D or 3D model. Though the networks between the three sites are heterogeneous, the three users were able to interact with a strong feel of presence.

This application is built from various components assembled with FlowVR, a middleware dedicated to large interactive applications [7]. FlowVR associates a hierarchical component approach with a data flow model. It leads to very modular distributed applications, enabling to keep the application complexity under control. A protocol for transferring 3D primitives for rendering enables to forward and combine various 3D data to the rendering hosts [1]. The 3D modeling on Grenoble and Bordeaux is a parallel application running on a cluster to produce meshes at the camera acquisition rate [11]. The terrain application serves data from a large data set relying on a parallel level-of-details algorithm to ensure an interactive service of the terrain [2].

We first present the application and its architecture (Sec. 2) before discussing some experimental results (Sec. 3) and concluding.

2. ARCHITECTURE

We focused our efforts to build an application that is highly modular, yet efficient. Given the number of resources, i.e. machines, cameras, displays, local and long distant networks, and the heterogeneity of the various computing tasks involved, software engineering is crucial to keep debugging, tuning, deployment and execution reasonably feasible. For that purpose, we rely on a hierarchical component model supported by FlowVR, a middleware dedicated to large applications. FlowVR enables to define the application as an assembly of components. The application is specialized and mapped on the computing resources in a second step, once the target grid is known. The deployment of computing tasks and the data transfers are managed by the middleware run-time, keeping away the developers from such a burden. The application we built for this experiment involves 270 processes distributed on 16 hosts totaling up to 72 processing cores distributed amongst 3 sites.

FlowVR favors the design of small components, with a simple and clear behavior. Our application combines 3 main types of components, which are usually defined from sub components.A rendering component is in charge of producing images for a display. It has two input ports, one for receiving streams of 3D primitives and the other one for externally controlling the viewport matrix. The 3D primitives define the 3D objects to be included in the scene. A 3D primitive is self-contained, i.e. it contains all data (meshes, textures, etc.) and codes (shaders) necessary for rendering, including its position in the scene [1]. Thus, a rendering component just has the responsibility of rendering the primitives it receives. Being self-contained, primitives can be emitted from different distributed sources, a key flexibility characteristic for keeping the application modular.

The 3D primitives are produced by another type of components, the 3D data producers. These components receive various input events and produce 3D primitives. Usually these 3D primitives are simply broadcasted to all rendering components. But in some cases, like for the terrain server, the 3D stream produced depends on the viewport of the rendering component. The 3D data producer receives the various viewport matrices, and can produce view-dependent 3D primitives using level-of-detail algorithms for instance.

A third class of components is related to input devices, i.e. sensors. These components output events that are usually forwarded to 3D data producers. We use different types of devices. For instance we use joypads on 2 sites to move in the scene. The associated component produces displacement matrices that are forwarded to the local data producer in charge of the user model to move its position. At Bordeaux, we use a touch pad and the advanced 3D interaction paradigm called Navidget [5]. The pad runs a rendering component (at a reduced level of detail), giving a view on the scene, as well as an input component. By a combination of intuitive 2D gestures the user can specify a displacement in the scene or add and remove some simple geometric objects. The geometric objects are not defined in this component. They are managed by a specific 3D data producer that receives orders from the pad input component. The data producer generates 3D primitives according to orders and send them to renderers.

Table 1: Average network traffic measured between the 3 sites (in Mbit/s) with one user per site. Diagonal terms represent the total traffic measured on each cluster.

	Bordeaux	Grenoble	Orléans
Bordeaux	120	175	49
Grenoble	185	600	47
Orléans	2	2	2

At Grenoble we use the more complex setup where the user is modeled in real time from cameras. He also wears and head mounted display (HMD). The HMD is tracked so it can display a view aligned with the users's position. This user can thus experience a first person visualization. His 3D model is also aligned with his real body, i.e. the virtual model of its body he sees in the HMD is aligned with its real body. Pointing at a virtual object with a hand becomes natural. The user can make short range moves just by walking while staying in the acquisition space. He also has a joypad for more important displacements. We thus have at Grenoble input components for the tracker, the cameras and the joypad, a 3D data producer for 3D modeling, and a rendering component powering the HMD.

3. EXPERIMENTS

Our application involves 2 multi-camera platforms, one at Bordeaux and one at Grenoble, respectively using 8 0.5M pixel and 8 1M pixel cameras. Video acquisition and computation are parallelized on a local cluster. The 3D modeling algorithm produces a mesh associated with a stream of textures. On average for one user standing in the acquisition space, the mesh produced is about 150 Kbit (about 1100 vertex) in Bordeaux and 300 Kbit (about 2600 vertex) in Grenoble. We also send silhouette masks to decode the textures. To reduce network traffic we reduce texture resolution by half, yielding 5 Mbit of texture data per iteration for 8 cameras for Bordeaux and 10 Mbit for Grenoble. When the cameras are running at 16Hz at Bordeaux and 24Hz at Grenoble, this produces streams of 80 Mbit/s and 240 Mbit/s.

The terrain server manages real data from a LIDAR airborne scan with an average resolution of one meter. The data set consists of hundreds of tiles of about 2 millions triangles each. A given point of view usually requires 9 tiles or about 140 MB, which cannot be transmitted over the network at interactive frame rates. Thus, the server relies on a level of detail parallel algorithm that drastically reduces the volume of data to be transfered. The data are reconstructed on the fly according to their distance from the point of view, their relevance and level of details desired. For example, the level of details for the touch pad is lower than for an image displayed on a wall to match each device capabilities. Data are reduced to 80 KB for standard displays and 3.5 KB for the touch pad. For a frame rate of 50 fps, the bandwidth requirement is about 96 Mbit/s for 3 classical points of views and one supplementary point of view for the touch pad. This of course is a peak value attained when all viewpoints are moving simultaneously. The average is lower as the server does not send already cached data when the viewpoints are standing still.

Orléans has a internal gigabit Ethernet network for its cluster and a 100 Mbits/s connection to the Internet. The Bordeaux cluster also uses a gigabit Ethernet network and a 1 Gbits/s connection to the Internet. The cluster at Grenoble has a DDR Infiniband network (20 Gbits/s) and a 10 Gbits/s Ethernet connection to the Internet. Traffic between Bordeaux and Grenoble moves on the dedicated 10 Gbits/s network of the Grid'5000 experimental grid, while Orléans has only access to the regular Internet network. Table 1 shows the average network traffic measured during execution. Grenoble, benefiting from both a high performance network and powerful processing nodes, handles a higher amount of data than the other sites. The other sites suffer network congestions and CPU overloads, and cannot receive the full data streams, triggering adaptive mechanisms integrated in the application to sub-sample data streams.

We also gathered latency data by measuring the time it takes from video acquisition to the rendering of the corresponding 3D model. The latency measures do not include the actual camera acquisition time, the rendering on GPU and the display. Figure 2 shows that latencies vary significantly between Grenoble and Bordeaux, due to the differences in clusters capabilities and network bandwidth. At Grenoble, the latency remains below 100 ms for the local 3D model, and in the order of 100 ms for the one received from Bordeaux. These values are very acceptable for interactions. At Bordeaux, the latency for the local 3D model is about 140ms due to its lower networking and computing capabilities. The 3D model transmitted from Grenoble to Bordeaux leads to a very high latency of about 500ms. We still do not have a clear understanding of this behavior. As expected, latencies at Orléans were high (above 500 ms) due to its very limited network capabilities.

Globally the users were very satisfied with the feel of presence and discussion capabilities offered by this setup (a conferencing audio system complemented the visual feedback). The best conditions were experienced at Grenoble, where frame rates and latencies where the smallest, but also because the HMD provided the user a strong feel of immersion. He really felt like the user from Bordeaux was right in front of him. 3D modeled users were instinctively relying on their body to transmit information (hand pointing or more unconscious postures).

4. CONCLUSION AND DISCUSSION

The application developed is a proof of concept that a tele-immersive grid could be a very effective environment for virtual distant collaboration. The user wearing the HMD experiences the best feel of immersion and presence. Being well integrated in the 3D environment, he naturally uses corporal expressions for communication.

The main challenge to design this application was to keep it modular so that it could be managed by several developers from different sites. The approach we used, relying on hierarchical components, proved efficient.

These experiments also revealed some limitations. The application should evolve towards a service-oriented architecture, where the application builds itself at run-time as new services register or unregister. We relied on simple strategies to control the amount of data transfers, but several other optimizations could be implemented. Data transfers are a critical issue for scaling to many participants, and other research teams have been investigating these issues like in [8]. The HMD appeared to pair well with multi-camera based 3D modeling. But the user is modeled with the HMD

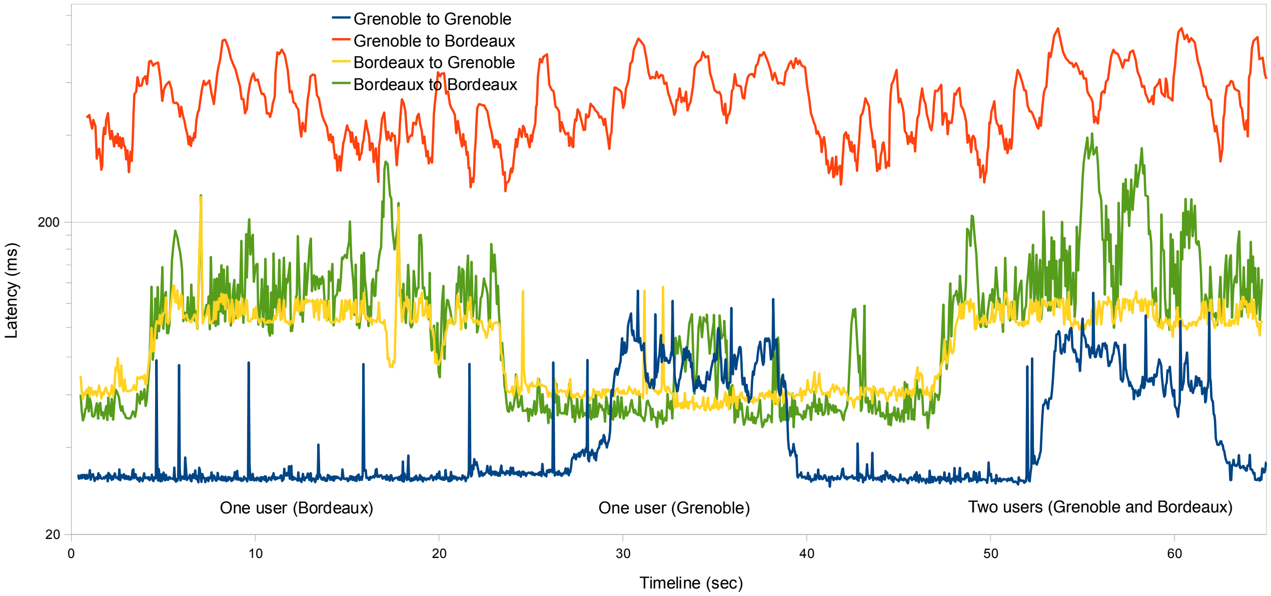

Figure 2: 3D modeling related latencies at Bordeaux (8 cameras at 16 fps) and Grenoble (8 cameras at 24 fps) with one or 2 users (log scale on the Y-axis). Orléans numbers are omitted for sake of clarity.

occluding part of his face, making eye contact with others impossible.

5. ACKNOWLEDGMENTS

This work was mainly funded by Agence Nationale de la Recherche, contract ANR-06-MDCA-003. Experiments used the Grid'5000 testbed supported by INRIA, CNRS, RENATER, several Universities and funding bodies [2]. Thanks to the Geo-Hyd company who granted access to accurate real land data.

6. REFERENCES

[1] J. Allard and B. Raffin. A Shader-Based Parallel Rendering Framework. In *IEEE Visualization Conference*, pages 127–134, Minneapolis, USA, Oct. 2005.

[2] S. Arvaux, J. Legaux, S. Limet, E. Melin, and S. Robert. Parallel lod for static and dynamic generic geo-referenced data. In *ACM VRST'08*, pages 301–302, New York, 2008. ACM.

[3] H. H. Baker, N. Bhatti, D. Tanguay, I. Sobel, D. Gelb, M. E. Goss, W. B. Culbertson, and T. Malzbender. Understanding performance in coliseum, an immersive videoconferencing system. *ACM Trans. Multimedia Comput. Commun. Appl.*, 1(2):190–210, 2005.

[4] M. Gross, S. Würmlin, M. Naef, E. Lamboray, C. Spagno, A. Kunz, E. Koller-Meier, T. Svoboda, L. Gool, S. Lang, K. Strehlke, A. V. Moere, and O. Staadt. Blue-C: a Spatially Immersive Display and 3D Video Portal for Telepresence. *ACM Transactions on Graphics*, 22(3):819–827, 2003.

[5] M. Hachet, F. Decle, S. Knödel, and P. Guitton. Navidget for 3D interaction: Camera positioning and further uses. *International Journal of Human-Computer Studies*, 67(3):225 – 236, 2009.

[6] P. Kauff and O. Schreer. An Immersive 3D Video-Conferencing System Using Shared Virtual Team User Environments. In *International Conference on Collaborative Virtual Environments*, pages 105–112, 2002.

[7] J.-D. Lesage and B. Raffin. A Hierarchical Component Model for Large Parallel Interactive Applications. *Journal of Supercomputing*, 7(1):1–20, July 2008. Extended version of NPC 2007 article.

[8] J.-M. Lien, G. Kurillo, and R. Bajcsy. Multi-camera tele-immersion system with real-time model driven data compression. *The Visual Computer*, 26:3–15, 2010.

[9] W. Matusik and H. Pfister. 3D TV: a scalable system for real-time acquisition, transmission, and autostereoscopic display of dynamic scenes. *ACM Trans. Graph.*, 23(3):814–824, 2004.

[10] J. Mulligan and K. Daniilidis. Real time trinocular stereo for tele-immersion. In *International Conference on Image Processing*, volume 3, pages 959–962, 2001.

[11] B. Petit, J.-D. Lesage, C. Ménier, J. Allard, J.-S. Franco, B. Raffin, E. Boyer, and F. Faure. Multi-Camera Real-Time 3D Modeling for Telepresence and Remote Collaboration. *International Journal of Digital Multimedia Broadcasting*, 2010:12 pages, 2010.

[12] Z. Shujun, W. Cong, S. Xuqiang, and W. Wei. DreamWorld: CUDA-Accelerated Real-time 3D Modeling System. In *IEEE VECIMS*, Hong Kong, China, May 2009.

[13] W. Wu, R. Rivas, A. Arefin, S. Shi, R. M. Sheppard, B. D. Bui, and K. Nahrstedt. MobileTI: a portable tele-immersive system. In *ACMM'09*, pages 877–880, New York, NY, USA, 2009. ACM.

[2] https://www.grid5000.fr