Grimage: Markerless 3D Interactions

Jérémie Allard [*] Sim Group - MGH/CIMIT
Clément Menier [†] INPG
Bruno Raffin [‡] INRIA
Edmond Boyer [§] UJF
François Faure [¶] UJF

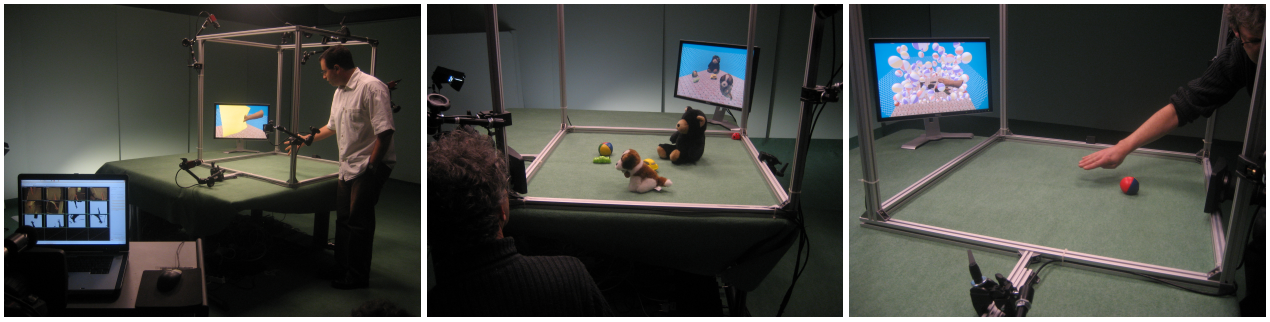

Figure 1: *The Grimage experience.*

Abstract

Grimage glues multi-camera 3D modeling, physical simulation and parallel execution for a new immersive experience. Put your hands or any object into the interaction space. It is instantaneously modeled in 3D and injected into a virtual world populated with solid and soft objects. Push them, catch them and squeeze them.

CR Categories: I.3.2 [Computer Graphics]: Graphics Systems—Distributed/network graphics I.4.5 [Image Processing and Computer Vision]: Reconstruction

Keywords: Markerless 3D Modeling; 3D Interactions; Soft Objects Simulation; Multi-cameras; PC Cluster

1 Introduction

The goal of the project is to associate computer vision, physical simulation and parallelism to move one step towards the next generation of virtual reality applications. We aim at extracting the maximum information from a set of calibrated cameras, avoiding markers not to be intrusive. On the simulation side the goal is to enable building large simulations involving complex various objects. Having access to parallelism, we can reach interactive executions for I/O and computation intensive applications.

Markerless multi-camera based 3D modeling is seldom used in virtual reality. However, getting a 3D surface or volume of the objects or users present in the interaction space can be a very useful data to improve interaction. In particular, it enables to compute collisions not only with the few positions that classical marker based 3D trackers give, but with the full object.

When rendering the model, texturing makes a significant difference. The raw 3D model of an object or person is difficult to identify. Once textured, recognizing it becomes almost immediate. Textured 3D modeling can be used in mixed reality environments or for multi-site collaborations. For instance several persons, physically located at different sites, could meet in a virtual room and bring objects with them. Having lifelike 3D models (shape, color and dynamics behavior) significantly improves the sense of presence.

3D models can also be used for computing interactions, a fundamental aspect of virtual environments. Once the real world objects have been modeled, they can be inserted into a simulation populated with pure virtual objects. The only particularity of real objects is that their state cannot be modified by the simulation (as long as no haptic system is used).

Developing applications that include numerous animated objects of different natures, like rigid, mass-spring, deformable or fluid objects is a challenging problem. It requires advanced algorithms and a software infrastructure for coupling these algorithms with the right balance between integration and modularity. Our project relies on such a simulation framework, called SOFA. Associating 3D modeling and physically-based simulations leads to intuitive and rich interactions.

Getting such an application parallelized and running on multiple CPUs is a challenging and essential work, as interactions can only be effective in a real-time context. This parallelization effort is a way to get access today to the I/O and computing power that will be available tomorrow into commodity computers. To tackle with the complexity of the resulting application, we developed a middleware library dedicated to interactive applications called FlowVR. It enforces code modularity, leveraging software engineering issues while enabling high performance executions on distributed and parallel architectures. We also take advantage of the parallelism offer by last generation GPUs, running part of SOFA computations with the NVIDIA Cuda library.

Various applications could benefit from the technology we present. Telepresence through a textured 3D model would ease interaction between people from different locations that meet in a common virtual space (a social community, a workplace, a game, a learning and

[*]e-mail: jeremie.allard@codrt.fr
[†]e-mail: clement.menier@imag.fr
[‡]e-mail: bruno.raffin@imag.fr
[§]e-mail: edmond.boyer@inrialpes.fr
[¶]e-mail: francois.faure@imag.fr

training environment, etc.).

Complex real-time physical simulations are a major challenge for applications like surgery planing, training on assembly tasks, or training for critical situation management (earthquake, fire, airplane/boat/car accidents, etc.). Injecting the 3D model into the simulation could for instance enable to detect collisions with the full body when testing the ergonomic of a new airplane (for maintenance teams, flying crew or passengers) on a virtual model.

2 Context

Camera based markerless interactions were pioneered by Krueger & al. [1985] that used one camera. Multiple cameras were used latter by Kanade & al. [1997]. Their system, called *Virtualized Reality*, uses 49 cameras where video streams are first stored on disks before being post-processed to compute a 3D model for each frame. Real time interaction were therefore not possible. Other off-line systems were also developed [Hilton and Starck 2004], with the goal of improving the quality of the 3D models.

Other initiatives focused on real-time 3D modeling but with a limited number of cameras (less than 5) [Cheung et al. 2000]. Following such approach approach Hasenfratz & al. [2004] proposed some simple 3D interactions.

For a larger number of cameras, Borovikov & al. [2003] present a distributed solution for post-processing the images stored on a distributed data base. Wu & al. [2006] propose a full parallel system, from acquisition to 3D modeling. But they rely on an algorithm with a cubic complexity in the precision of the 3D model, limiting the scalability.

Other initiatives focus on free viewpoint video, where the goal is to give a distant viewer the ability to chose an arbitrary viewpoint on the scene. Applications first target telepresence and 3D TV rather than interaction with virtual objects [Goldlücke and Magnor 2003; Matusik and Pfister 2004; Gross et al. 2003; Carranza et al. 2003]. Different 3D modeling algorithms and parallelization approaches are proposed, some of them relying on GPUs [Li et al. 2003; Li et al. 2004].

To our knowledge, our project is the first to associate multi-camera 3D modeling and physical simulations for markerless 3D interactions.

3 User Experience

The platform we present is a small scale of the Grimage platform located at INRIA Rhne-Alpes. It gathers a set of cameras surrounding an acquisition space, a PC cluster for computations, and a multi-display rendering. Instead of having a large acquisition space enabling the 3D modeling of the user full body, it is limited to a 1 m^3 space at a table height (Fig. 1). To interact, the user just has to put is hand or any other part of its body in the interaction space. He can also put in this interaction space other objects, like his favorite Siggraph goody. He sees all these elements of the real world modeled in 3D and textured on a display located behind the interaction space. He also sees the virtual objects populating the virtual world. Once 3D modeled, the real elements just become, from the simulation point of view, virtual solid objects that are not sensible to external forces. All these objects, the 3D modeled ones as well as the virtual ones (solid or soft) can interact with each other according to the constraints they are submitted to.

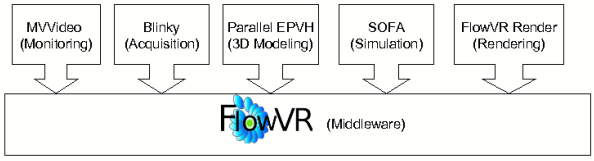

Figure 2: *The main software components the Grimage platform relies on.*

Figure 3: *The processing pipe-line from camera acquisition to rendering. Components can be distributed on different computing nodes. 3D Modeling and simulation are internally parallelized.*

4 Technical Innovations

The project is built upon a software suite we developed over the last 4 years (Fig. 2). Its components are assembled into a processing pipe-line (Fig. 3). Each one incorporates various scientific and technological innovations detailed bellow.

4.1 Video Acquisition and Camera Control

Getting quality data from a set of cameras highly depends on the parameter settings. For assisting the user into this process we developed several tools. These tools are technologically innovative in the sense that they target sets of cameras driven by a PC cluster.

The Blinky software library enables real-time acquisition of images for multiple cameras spread over a PC cluster. It also enables to change the camera parameters (shutter speed, aperture, zoom...).

MVVideo is a graphical user interface to remotely control the acquisition, synchronization and display of video streams from multiple cameras distributed on a PC cluster. The software displays the video streams from all cameras simultaneously in reduced resolution for monitoring, and it enables to control all camera parameters (e.g. gain, shutter speed, focus, white balance).

For 3D modeling each image is processed by a background subtraction algorithm. The background does not have to be uniform as it is

learned during an initialization phase.

4.2 3D Modeling

From the video streams, we compute a 3D model of the objects being present into the acquisition space. We compute the *visual hull* of the objects by reconstructing their shape from the silhouettes extracted from the video streams. Geometrically, the visual hull is the intersection of the *viewing cones*, the generalized cones whose apex are the cameras' projective centers and whose cross-sections coincide with the scene silhouettes. When considering piecewise-linear image contours for silhouettes, the visual hull becomes a regular polyhedron. The algorithm we use, called Exact Polyhedral Visual Hull algorithm [Franco and Boyer 2003], computes the complete and exact visual hull polyhedron with regard to silhouette inputs.

To achieve real time 3D modeling, we developed a parallel version of the exact polyhedral visual hull algorithm. It relies on a 3 step pipe-line, each step being itself parallelized [Franco et al. 2004]. This approach enables to reach high frequencies with a low latency.

The 3D model is then sent to 2 components: simulation and rendering. When sent to rendering, it is textured, from the same set of images that was used to compute the silhouettes. The exact polyhedral visual hull algorithm guarantees the 3D model can be projected back to the original silhouette with no error, a property that leads to a better quality texture mapping.

4.3 Simulation

The virtual objects as well as the real ones, once modeled, are animated by the SOFA simulation software. SOFA is a new open source framework primarily targeted at medical simulation research [Allard et al. 2007].

Its architecture relies on several innovative concepts, in particular the notion of multi-model representation. In SOFA, most simulation components (deformable models, collision models, instruments, etc.) can have several representations, connected together through a mechanism called mapping. Each representation is optimized for a particular task such as mechanical computations, collision detection or visualization. This clear separation between the functional aspects of the simulation components allows flexible modeling based on a reduced set of models. At a finer level of granularity, the physical models (i.e. any model that behaves according to the laws of physics) are decomposed into a set of basic components such as topology, degrees of freedom, internal force fields and constraints acting on the degrees of freedom.

Another key aspect of SOFA is the use of a scene-graph to organize and process the models and components while clearly separating the computation tasks for their possibly parallel scheduling. This data structure, inspired by well-known rendering scene-graphs, is new in physically-based animation. Beside classical traversal actions such as rendering or bounding-box computations, physical actions such as force accumulation or state vector operations are triggered by components dedicated to physical animation. This allows us to easily design differential equation solvers suitable for single objects as well as complex systems made of different kinds of interacting physical bodies (rigid bodies, deformable solids, fluids). The demo presented with this paper applies implicit time integration with iterative solution. The maximum number of iterations is tuned to limit the computation time. This creates a trade-off between accuracy and computation time that allows us to meet the real-time constraint without sacrifying stability.

A parallel version of SOFA on GPU using the NVIDIA Cuda library has been developed. It enables to significantly speed-up some computations.

4.4 Rendering

Data to be rendered, either provided by SOFA or from the 3D modeling algorithm are distributed to dedicated rendering nodes, enabling multiple display rendering.

For that purpose, we use FlowVR Render [Allard and Raffin 2005]. Existing parallel or remote rendering solutions rely on communicating pixels, OpenGL commands, scene-graph changes or application-specific data. We rely on an intermediate solution based on a set of independent graphics primitives that use hardware shaders to specify their visual appearance. Compared to an OpenGL based approach, it reduces the complexity of the model by eliminating most fixed function parameters while giving access to the latest functionalities of graphics cards. It also suppresses the OpenGL state machine that creates data dependencies making primitive re-scheduling difficult.

Using a retained-mode communication protocol transmitting changes between each frame, combined with the possibility to use shaders to implement interactive data processing operations instead of sending final colors and geometry, we are able to optimize the network load. High level information such as bounding volumes is used to setup advanced schemes where primitives are issued in parallel, routed according to their visibility, merged and re-ordered when received for rendering. Different optimization algorithms can be efficiently implemented, saving network bandwidth or reducing texture switches for instance.

4.5 Coupling and Distribution

Coupling the different software components involved into this project and distributing them on the nodes of a PC cluster for reaching real-time executions is performed through the FlowVR middleware [Allard et al. 2004; Allard and Raffin 2006].

FlowVR enforces a modular programming that leverages software engineering issues while enabling high performance executions on distributed and parallel architectures. FlowVR relies on a data-flow and component oriented programming approach that has been successfully used for other scientific visualization tools. Developing a FlowVR application is a two step process. First, modules are developed. Modules encapsulate a piece of code, imported from an existing application or developed from scratch. The code can be multithreaded or parallel, as FlowVR enables parallel code coupling. In a second step, modules are mapped on the target architecture and assembled into a network to define how data are exchanged. This network can make use of advanced features, from bounding-box based routing operations to complex message filtering or synchronization operations.

The FlowVR run-time engine runs a daemon on each node of the cluster. This daemon is in charge of synchronization and data exchange between modules. They hide all networking aspects to modules, making module programming easier. Each daemon manages a shared memory segment. Messages handled by modules are directly written and read from this memory segment. If data exchange is local to a node, it only consists in a pointer exchange, while the daemon takes care of transferring data through the network for inter-node communications.

5 Conclusion

Grimage associates 3 main components:

- A multi-camera 3D modeling environment based on the Exact Polyhedral Visual Hull algorithm. It enables markerless full body interactions.
- The SOFA framework for real-time simulations. It allows to build complex but modular models using a scene-graph description. SOFA is used to compute the interactions between real and virtual objects.
- The FlowVR middleware dedicated to distributed interactive applications. Its data flow and component oriented model enforces the application modularity while enabling efficient executions on PC clusters. FlowVR is used to assemble all components of the application, distribute an run them on a PC cluster to reach a real-time performance.

The result is a platform where users can experiment intuitive and rich 3D interactions with various solid and soft virtual objects.

Future work will focus on exploring the different forms of interactions Grimage enables, on integrating new and improved algorithms, and on consolidating and extending the software architecture, a critical aspect to overcome the complexity of the application.

6 Credits

Thanks to all people that participated to the development of the various components of this project: Nicolas Turro, INRIA, Florian Geffray, INRIA, David Knossow, INRIA, Everton Hermann, INRIA, Jean-François Cuniberto, INRIA, Frederic Devernay, INRIA, Remi Ronfard, INRIA, Herv Mathieu, INRIA, Stéphane Cotin, CIMIT, Pierre-Jean Bensoussan, INRIA, François Poyer, INRIA, Christian Duriez, INRIA, Hervé Delingette, INRIA, Laurent Grisoni, INRIA.

This work was supported by our institutions: INRIA, U-Grenoble, the Laboratoire d'Informatique de Grenoble (LIG), the Laboratoire Jean Kuntzmann (LJK), and MGH/CIMIT.

These works have been partly funded by the RNTL project Geobench, the ACI Cyber-II, the ACI Ocetre, the ARA DALIA, the European project Holonics and the European project Odyseus

.

References

ALLARD, J., AND RAFFIN, B. 2005. A Shader-Based Parallel Rendering Framework. In *IEEE Visualization Conference*.

ALLARD, J., AND RAFFIN, B. 2006. Distributed Physical Based Simulations for Large VR Applications. In *IEEE Virtual Reality Conference*.

ALLARD, J., GOURANTON, V., LECOINTRE, L., LIMET, S., MELIN, E., RAFFIN, B., AND ROBERT, S. 2004. FlowVR: a Middleware for Large Scale Virtual Reality Applications. In *Euro-Par 2004 Parallel Processing: 10th International Euro-Par Conference*, 497–505. http://flowvr.sf.net.

ALLARD, J., COTIN, S., FAURE, F., BENSOUSSAN, P.-J., POYER, F., DURIEZ, C., DELINGETTE, H., AND GRISONI, L. 2007. SOFA: an Open Source Framework for Medical Simulation. In *Medicine Meets Virtual Reality (MMVR)*. http://www.sofa-framework.org.

BOROVIKOV, E., SUSSMAN, A., AND DAVIS, L. 2003. A High Performance Multi-Perspective Vision Studio. In *17th Annual ACM International Conference on Supercomputing, San Francisco (USA)*.

CARRANZA, J., THEOBALT, C., MAGNOR, M., AND SEIDEL, H. 2003. Freeviewpoint video of human actors. In *Proceedings of ACM SIGGRAPH 03*, 569–577.

CHEUNG, G., KANADE, T., BOUGUET, J.-Y., AND HOLLER, M. 2000. A real time system for robust 3d voxel reconstruction of human motions. In *Computer Vision and Pattern Recognition 00*, vol. II, 714 – 720.

FRANCO, J., AND BOYER, E. 2003. Exact Polyhedral Visual Hulls. In *Proceedings of BMVC2003*.

FRANCO, J.-S., MÉNIER, C., BOYER, E., AND RAFFIN, B. 2004. A Distributed Approach for Real Time 3D Modeling. In *Proceedings of the IEEE Workshop on Real Time 3D Sensors and Their Use*.

GOLDLÜCKE, B., AND MAGNOR, M. 2003. Real-Time Microfacet Billboarding for Free-Viewpoint Video Rendering. *Proc. IEEE International Conference on Image Processing (ICIP'03),* Barcelona, Spain (September), 713–716.

GROSS, M., WUERMLIN, S., NAEF, M., LAMBORAY, E., SPAGNO, C., KUNZ, A., KOLLER-MEIER, E., SVOBODA, T., GOOL, L. V., S. LANG, K. S., MOERE, A. V., AND STAADT, O. 2003. Blue-C: A Spatially Immersive Display and 3D Video Portal for Telepresence. In *Proceedings of ACM SIGGRAPH 03*.

HASENFRATZ, J.-M., LAPIERRE, M., AND SILLION, F. 2004. A real-time system for full body interaction with virtual worlds. 147–156.

HILTON, A., AND STARCK, J. 2004. Multiple view reconstruction of people. In *3DPVT '04: Proceedings of the 3D Data Processing, Visualization, and Transmission, 2nd International Symposium on (3DPVT'04)*, IEEE Computer Society, Washington, DC, USA, 357–364.

KANADE, T., RANDER, P., AND NARAYANAN, P. 1997. Virtualized Reality: Constructing Virtual Worlds from Real Scenes. *IEEE Multimedia, Immersive Telepresence 4*, 1 (January), 34–47.

KRUEGER, M. W., GIONFRIDDO, T., AND HINRICHSEN, K. 1985. Videoplace – an artificial reality. In *CHI '85: Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM Press, 35–40.

LI, M., MAGNOR, M., AND SEIDEL, H.-P. 2003. Hardware-Accelerated Visual Hull Reconstruction and Rendering. In *Proceedings of Graphics Interface'2003*.

LI, M., MAGNOR, M., AND SEIDEL, H.-P. 2004. A hybrid hardware-accelerated algorithm for high quality rendering of visual hulls. In *GI '04: Proceedings of the 2004 conference on Graphics interface*, Canadian Human-Computer Communications Society, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 41–48.

MATUSIK, W., AND PFISTER, H. 2004. 3D TV: A Scalable System for Real-Time Acquisition, Transmission, and Autostereoscopic Display of Dynamic Scenes. In *Proceedings of ACM SIGGRAPH 04*.

WU, X., TAKIZAWA, O., AND MATSUYAMA, T. 2006. Parallel Pipeline Volume Intersection for Real-Time 3D Shape Reconstruction on a PC Cluster. In *Proceedings of the Fourth IEEE International Conference on Computer Vision Systems (ICVS'06)*, IEEE Computer Society, Washington, DC, USA.