lgf3 - A Versatile Framework for Image-based Modeling and Rendering

Christian Vogelgsang
University of Erlangen-Nuremberg

Günther Greiner
University of Erlangen-Nuremberg

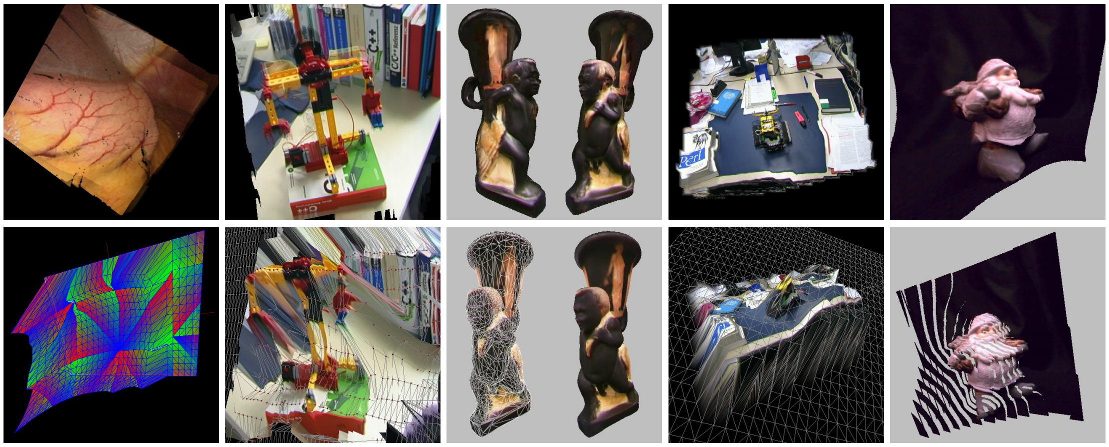

Figure 1: Various advanced image-based rendering techniques can be realized with the *lgf3* framework. Top row shows final renderings and the bottom one illustrates the alternative educational views of a renderer. Techniques (from left to right): Two-plane parameterized lumigraph, free-form light field, light field mapping[Chen et al. 2002], unstructured lumigraph[Buehler et al. 2001], and texture slicing[Vogelgsang and Greiner 2003]

Image-based modeling and rendering are investigated in both the computer graphics and the computer vision community with different focus. Although both share many ideas, cooperative work is hard due to subtle differences in notion or application. To remedy this problem, the *lgf3* framework was devised to provide a common foundation for application development combining both fields. State of the art rendering techniques can now be used in direct combination with advanced computer vision techniques.

For efficient geometric modeling and rendering high level scene graph APIs are commonly used. In image-based rendering there are no comparable abstractions available and developers can only use the less powerful low-level APIs like OpenGL for rendering or OpenCV for vision tasks. Currently there is no toolkit available that provides a generic and versatile higher level abstraction of image-based modeling approaches. The goal of the *lgf3* project is to provide this rich implementation environment for rapid development and efficient research in image-based techniques.

The focus of the framework lies on implementing a processing pipeline for modeling and rendering of real world scenes. A typical application first records one or more image sequences. Then a geometric context for the images is established with the help of camera calibration techniques. The focus lies on uncalibrated sequences from handheld-cameras, but the framework also supports previously calibrated streams. The scene model can then be enriched with geometric object information, e. g. by adding scanned meshes. This source data of the scene is transformed into a representation and parameterization suitable for image-based rendering techniques. E. g. a two-plane parameterized light field needs a rewarping step to generate camera views layed out on a grid. Finally, novel views of the scene can be generated for interactive visualization or for a feedback loop that uses the interpolated views to improve the calibration result.

The framework is structured into modules with a base module managing a scene database with all information recorded or derived from a real scene. The other modules access this database to fetch data for processing and feed back their results. By analyzing a wide range of existing algorithms, we designed generic components suitable for implementing current and novel techniques. The first module handles camera *calibration* and includes a wide range of computer vision algorithms. It fetches image sequences from the scene and adds camera parameters. The second module copes with parameter *conversion* of the scene model and transforms the view sequence into a model suitable for rendering. Finally, the *render* module incorporates interactive visualization techniques (see Figure 1) for the generated models. Additionally, a *GUI* module provides components and helps setting up full applications easily.

For image-based rendering the *lgf3* framework provides a powerful interface that supports pixel- and OpenGL-based rendering. It allows an efficient implementation of IBR techniques and it automatically handles cameras and the rendering setup for offscreen and interactive scenarios. The scene database can be filled by the calibration module or by persistent database connectors. The framework takes care of all resource management and especially handles long image sequences well. It implements demand loading, copy-on-write and caching strategies to guarantee access performance. This is implemented in a completely transparent way, so the developer can entirely focus on the render algorithm.

BUEHLER, C., BOSSE, M., MCMILLAN, L., GORTLER, S. J., AND COHEN, M. F. 2001. Unstructured lumigraph rendering. In *SIGGRAPH 2001, Computer Graphics Proceedings*, ACM Press / ACM SIGGRAPH, E. Fiume, Ed., 425–432.

CHEN, W.-C., BOUGUET, J.-Y., CHU, M. H., AND GRZESZCZUK, R. 2002. Light field mapping: Efficient representation and hardware rendering of surface light fields. In *SIGGRAPH 2002, Computer Graphics Proceedings*, ACM Press / ACM SIGGRAPH, J. Hughes, Ed., 447–456.

VOGELGSANG, C., AND GREINER, G. 2003. Interactive range map rendering with depth interval texture slicing. In *Workshop Vision, Modeling and Visualization*, Infix, Munich, Germany, T. Ertl, B. Girod, G. Greiner, H. Niemann, and H.-P. Seidel, Eds., 477–484, 622.