EUROGRAPHICS 2001 / A. Chalmers and T.-M. Rhyne
(Guest Editors)

Volume 20 (2001), Number 3

On-the-Fly Processing of Generalized Lumigraphs

Hartmut Schirmacher, Li Ming, Hans-Peter Seidel †

Max-Planck-Institut für Informatik
Stuhlsatzenhausweg 85
66123 Saarbrücken, Germany

Abstract

We introduce a flexible and powerful concept for reconstructing arbitrary views from multiple source images on the fly. Our approach is based on a Lumigraph structure with per-pixel depth values, and generalizes the classical two-plane parameterized light fields and Lumigraphs. With our technique, it is possible to render arbitrary views of time-varying, non-diffuse scenes at interactive frame rates, and it allows using any kind of sensor that yields images with dense depth information. We demonstrate the flexibility and efficiency of our approach through various examples.

1. Introduction

Efficient and flexible image-based scene representation is still one of the most important challenges in computer graphics. Although recently a lot of ground-breaking work has been done in this field, there are still many situations for which none of the currently available techniques seems suitable. This especially holds for the representation of *dynamic* scenes and *immediate* processing of multi-image data for redisplay, as it is important for applications such as interactive 3-D movies, augmented reality, and tele-immersion.

Since *purely* image-based techniques such as light field rendering rely on a large number of very densely spaced cameras, they require custom light field camera hardware. Other techniques need only a few input images, but some additional geometric information for reconstructing views from a scene. For non-synthetic scenes, this geometric information has to be inferred from the input images, or must be acquired by additional sensors.

Thanks to years of computer vision research, bi- and trifocal camera sensor systems are currently emerging on the market and will soon provide us with hybrid image and geometry data (per-pixel depth) at video frame rates. Several of these sensors have to be combined in order to fully represent a scene, and the data from these different sensors must be fused in order to reconstruct arbitrary views from this representation.

One important problem is that so far efficient multi-image rendering algorithms require computationally expensive intermediate data representations that cannot be computed on the fly. The most prominent examples for this kind of scene representations are traditional Lumigraphs, layered depth images, and surface light fields. They all share the idea of resampling and processing the image and geometry input data to allow for very efficient and high-quality rendering of the represented scene, and this resampling and processing is usually very expensive.

In this article, we present a method for reconstructing arbitrary views from multiple images with depth using a generalized Lumigraph data structure and a warping-based rendering algorithm. In contrast to previous work, our approach allows rendering time-varying, non-diffuse scenes from an arbitrary number of images with depth. We present an distributed image-based rendering architecture that can handle all the processing stages from image capture to compositing and rendering on the fly, at interactive frame rates. We also demonstrate an early version of a complete tele-immersion system called the *Lumi-Shelf*, which combines some consumer quality video cameras and standard PCs with a hierarchical correspondence-based depth-from-stereo software and the proposed on-the-fly Lumigraph back-end.

In the next Section, we will first review previous work on view reconstruction from multiple images, and then go

† {schirmacher,ming,hpseidel}@mpi-sb.mpg.de

into the necessary details of light field and Lumigraph rendering. In Section 4 we present our generalized Lumigraph concept and show how existing techniques can be applied and extended in order to allow the complete acquisition-to-rendering pipeline to be processed on the fly. In Section 5 we show experimental results for synthetic and real world data, as well as some early results of our Lumi-Shelf experiments. We conclude this article in Section 6, and point out some interesting directions of future research.

2. Related Work

Our work aims at rendering novel views from a collection of time-varying images with depth. We briefly review the most important concepts in this area, before we go into some more details of light field and Lumigraph rendering that we will build upon later.

Warping. Dense range data has lots of important applications in computer graphics [19]. The depth value defines a correspondence between the pixel and a point in 3-D space. This is exploited by the so-called *image reprojection* [4, 18, 17], also referred to as *3-D image warping*. One basic problem of reprojection is the change in sampling density between the reference image and the desired view. This can be accounted for by using more sophisticated (and computationally expensive) reconstruction filters, performing an *inverse mapping* [24], or by first *pre-warping* the image within the reference image plane (taking into account the novel view point), and then using texture mapping hardware to perform the filtered reprojection onto the desired viewing plane [20, 21, 25].

Multi-image warping. One problem that cannot be avoided with a single input image is missing information about surfaces hidden in the reference view, resulting in reconstruction *holes* in the novel view. The only valid way of filling in this missing information is the use of additional information from other input images. However, this introduces two additional problems: the *selection* of the relevant samples from every image, and the *compositing* of the final color from the chosen samples.

Techniques like that of Mark et al. [13] aim at improving the reconstruction from image pairs by taking into account *confidence* and *connectedness* measures. Laveau and Faugeras [11] describe how to predict novel views from an arbitrary number of images, but their approach does not seem suitable for interactive-rate applications.

LDI. Max [15] as well as Shade et al. [28] proposed to resample the image information into a so-called *layered depth image* (LDI). The LDI contains multiple samples per pixel, accounting for all surfaces intersected by the ray through that particular pixel. This deals very well with occlusions, but unfortunately the LDI technique is only suited for diffuse or artificially shaded surfaces, and the input resampling seems to be computationally expensive.

Tele-immersion systems. Several application prototypes combine a certain number of cameras and/or depth sensors to record or transmit and redisplay a scene at interactive rates. See Raskar et al. [23] for more insight into this field. Geometry acquisition methods include passive techniques as well as active lighting (visible or invisible), and the sensor types range from binocular and trinocular systems to complete camera matrices [5, 10], or camera domes [22]. Recently, Matusik et al. have presented the *image-based visual hull* technique [14]. It differs from the other methods in that it uses silhouette information for robust estimation of an approximate geometry.

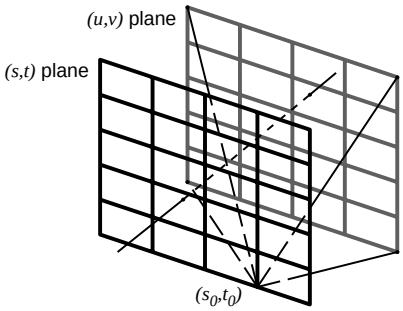

Figure 1: *Sketch of a two-plane-parameterized light field slab. A rays leaving the space between the two planes is characterized by its intersection with the two planes. All rays through one point (eye point) on the* (s,t) *plane form a sheared perspective image on the* (u,v) *plane.*

Scene representation and rendering is done in many different ways, but is usually restricted to a certain application or sensor type. Furthermore, no system supports rendering of non-diffuse surfaces. Our rendering approach can be used with combinations of all kinds of image-with-depth sensors, and it also supports directionally varying object appearance (if the sensors support this).

Light field techniques. For exploiting the directionally dependent color information in the original images, a *light field*-like technique is required, where each resulting pixel is computed by interpolation from multiple reference samples that are closest to the corresponding viewing ray. This is also called a "ray database" approach [16]. Since previous light field and Lumigraph techniques are the basis for our work, we will explain them more deeply in the next section, before going into the details of our new method in Section 4.

3. Light Fields and Lumigraphs

The basic idea of light fields [12] and Lumigraphs [6] is to capture the light distribution within a bounded region of 3-D space. Either the scene or the viewer are restricted to that region (inward- vs. outward-looking light fields). Fig. 1 shows an example of a two-plane parameterized light field, where a ray leaving (or penetrating) the region between the two

planes is parameterized by its intersections with the two planes. Six pairs of such planes (called *light slabs*) define the complete light field.

The resulting structure resembles an array of sheared perspective images, with the centers-of-projection on one plane (*eye point plane*), and the images on the other plane (*image plane*). One important property of this structure is that these light field images *share* their image plane, so we can assume that we have an input sample from each image for every point on this plane. This makes it much easier to select the "best" samples for reconstruction, which is done by quadrilinear interpolation from the 16 samples that are given by the nearest four points on each of the two planes (cf. Fig. 1).

Several parameterizations have been proposed and discussed as alternatives to the classical two-plane parameterization [8, 2, 30, 1]. However, the two-plane parameterization has been used and studied quite intensively, maybe because of its simplicity and suitability for hardware-accelerated rendering. Recently, Chai et al. [3] have presented a thorough analysis of the sampling issues for two-plane parameterized light fields.

The problem with all the classical light field parameterizations is that they cannot be mimicked very well with real-world cameras, except with custom hardware [32, 31]. Isaksen et al. [9] have weakened these restrictions a bit and allow arbitrary camera rotations around the optical center. In the next section, we will further generalize the Lumigraph in that respect, so that very simple and cost-effective setups like our *Lumi-Shelf* (see Sec. 4.3) can be used directly for Lumigraph capture and viewing.

Adaptive Sampling. Sloan et al. [29] introduced a generalization of the regularly sampled light field by allowing an arbitrary set of sample points on the eye point plane. For reconstruction, these eye points are triangulated, and within such a viewing triangle the color values are interpolated from the three images corresponding to the vertices of that triangle. This interpolation step can be approximated efficiently using texture mapping and alpha blending.

Geometry Information. The Lumigraph also makes use of geometric information in the form of a coarse triangle mesh. During rendering, rays are cast through the vertices of the viewing triangles, and the resulting intersection is used for depth-correcting the interpolation process. This can be thought of as shifting the source image triangle according to its optical flow (see also Fig. 5). If the depth varies strongly within the triangle (and similarly the optical flow), it will be subdivided and treated recursively.

Heidrich and Schirmacher [7, 25] proposed a different depth correction approach by using per-pixel depth information and forward-warping the source pixels. This is in some sense very similar to the polygon-based depth correction, since both methods simply shift the samples within the Lumigraph image plane, according to the optical flow for the desired view point. One important difference between the two techniques is the sample selection. In order to determine the required source region in the input images, Gortler et al. perform a sparse sampling of the scene's depth, whereas Schirmacher et al. use a conservative estimation of the optical flow, exploiting the complete depth information in each image [25]. In Section 4.2 we will generalize this techniques to the case of Lumigraphs that are not two-plane parameterized, and to account for time-varying depth information.

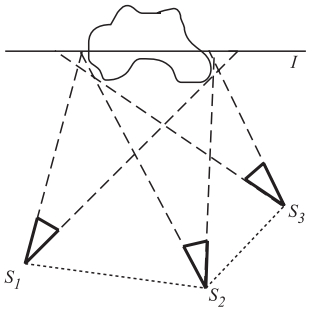

Figure 2: *Generalized Lumigraph slab. Each sensor with optical center S_k must completely see the object's silhouette, the cameras should be at roughly the same distance to the virtual image plane I, and the surface mesh defined by the optical centers must be convex.*

4. On-the-Fly Lumigraph Processing

In this section we introduce the generalized Lumigraph and show how to apply and extend the different techniques presented in the previous section for on-the-fly processing.

4.1. The Generalized Lumigraph

As already mentioned, the basis for our work is the two-plane parameterized Lumigraph with per-pixel depth information as presented by Heidrich and Schirmacher (cf. Sec. 3). The per-pixel depth information is a very general representation that can be used with all kind of synthetic geometry (polygons, curves, procedural) as well as real-world geometry data (depth from stereo, passive and active depth sensors).

In order to match real camera setups more closely, we weaken the assumption of an eye point plane and use a more general camera surface. For inward-looking light fields, the goal usually is to generate arbitrary views of an object or scene. In practice, this implies that every sensor S_k should see the complete object silhouette. For one slab of the Lumigraph, we define a common *virtual image plane I* through the center of the object (see Fig. 2). All cameras should keep roughly the same distance to the image plane in order to restrict the sampling rate to a manageable range.

For this set of camera points, we can easily construct a

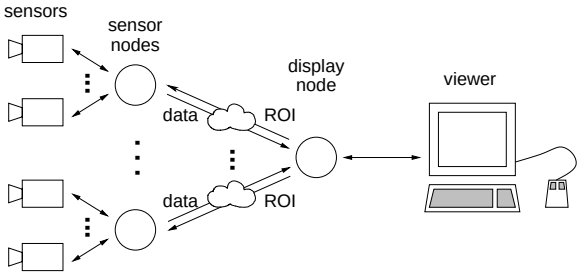

Figure 3: *Distributed Lumigraph processing. The display host is connected to a number of sensor hosts. Each sensor host processes an arbitrary number of sensors. The display host updates the region-of-interest information for each sensor, and the sensors send the data from the corresponding image regions.*

camera surface by projecting the points into the image plane, triangulating the projected points, and lifting that triangulation back into 3-space. This camera mesh is used to define the neighborhood relationships for the region-of-interest determination (see Sec. 4.2) and for rendering (Sec. 4.4). Recently in a work parallel to this [26], it has been found out that the camera mesh must be *convex* in order to prevent problems in the rendering step. This will be briefly explained in the partitioning paragraph in Sec. 4.4.

Now we have a complete generalized Lumigraph slab, defined by the image plane and the convex mesh of cameras with their intrinsic and extrinsic parameters.

4.2. Processing Phases and Distribution

The Lumigraph processing pipeline consists of two main parts, the *sensor* part and the *rendering* or *display* part. The sensor part includes the image capture as well as depth reconstruction (Sec. 4.3), whereas the rendering part performs the sample selection, sample reprojection, and compositing (see Sec. 4.4).

Depending on the computational demands or physical location of the sensors and the power of the display host, the Lumigraph processing can be parallelized and distributed on several inter-networked computation nodes. The most natural way to do that is sketched in Fig. 3. The data of several sensors is processed by a *sensor node*, which transmits the desired data to a *display node*.

The display node is informed about the desired view, and exploits this information for determining a so-called *region of interest* (ROI) for every sensor. This region of interest is an estimate of the subset of image data needed from the sensor, and helps reducing the transmission load significantly. Additionally, the sensor might use the ROI information to reduce the internal bus load or computation cost (e.g. perform stereo matching only on the desired scan line segments).

Figure 4: *Our Lumi-Shelf, consisting of a bookshelf and six consumer-quality Fire-Wire video cameras connected to three Linux PCs.*

Moreover, the sum of data in all regions of interest increases sub-linearly with the number of sensors [25], so that this scheme scales very well to a large number of sensors, with or without network connection. Section 4.4 will give insight into the ROI computation.

4.3. Acquisition and the Lumi-Shelf

Every Lumigraph input camera is treated as a black-box sensor that delivers a 2-D image containing dense color and depth information. This data may or may not be varying over time.

Still images with depth can originate from many sources. From real-world images, depth values can be reconstructed by techniques like voxel coloring [27], and for synthetic images depth is usually available directly through the rendering engine (e.g. ray tracer, OpenGL, etc.).

Practical examples for time-varying sensors are stereo camera pairs, cameras with range finders, and many more. Even a whole *camera matrix* [10] can be treated in the same way by using its output images separately.

As a low-cost example for a complete real-world video sensor setup, we built a device we called the *Lumi-Shelf*, since it is basically a bookshelf with two rows of cameras (see Fig. 4). We use six consumer quality Fire-Wire video cameras aligned in two rows. We partition the cameras into stereo pairs, and every stereo pair is connected to one PC for stereo processing. The image data is transferred to the PC via the Fire-Wire interface. Then the two images are rectified, and a block matching-based hierarchical depth-from-stereo algorithm determines the disparity (and thus depth) between pixels in the two images.

In our experimental setup, the preliminary implementation of the hierarchical block matching tries to trade quality for speed. Currently the system computes a dense range map at near-interactive rates (0.5-3 frames/second) for a resolution of 320×240 pixels, but the reconstructed depth values contain a lot of false matches and tend to smear objects near their silhouette. We are currently working on improving this algorithm, and also comparing it to commercial products such as the *small vision systems* software by SRI International.

4.4. Rendering

The rendering of arbitrary views from the generalized Lumigraph extends the work of Schirmacher et al. [25]. The generalized camera setup and the time-varying nature of the scene require some changes that will be explained in more detail.

Overview. The principle of the rendering algorithm is very simple. First, we project the camera mesh triangles onto the virtual image plane in order to *partition* the image into triangular regions. Each of these regions will be reconstructed from the three camera points that build the triangle. Then, using the partitioning, we determine the *region of interest* for each sensor, which is the set of all pixels that can somehow contribute to reconstruct the image in all regions that this sensor is associated with. Then we *reproject* all pixels in each sensor's region of interest into the virtual image plane, taking into account the user's viewing position. We do a *compositing* step including a Z-buffer test, and thus create a single texture for the virtual image plane. Finally, we reproject the virtual image plane into the user's view using OpenGL by drawing a single textured polygon.

Reprojection. The basic operation for generating a novel view is to *reproject* the input samples according to their depth and to the new viewpoint (cf. Fig. 5). As mentioned in Sec. 2, the main problem here is to choose an output image resolution that matches that of the input. In the Lumigraph case this is relatively easy, since we have chosen the common Lumigraph image plane I and the cameras S_k in such way that if the sensors all have similar optics and resolution, the size of a sensor pixel projected into the common image plane does not vary too strongly. So we reproject the sensor samples into I to account for the new viewing position, and then draw the resulting image as a texture map on a single polygon representing I. (cf. Fig. 5). As explained in Sec. 2 this approach is also known as *pre-factored warping*. Note that since the geometric relations between the sensor and I do not change, the projection coefficients can be precomputed for every sensor.

In order to determine a suitable resolution for the Lumigraph image plane, we measure the distance of two neighboring pixels, projected into I. We do this for one centered pixel pair per sensor, and thus estimate the average pixel distance. The Lumigraph's image plane sampling density can then be chosen quite conservatively, e.g. corresponding to twice this average projected pixel size.

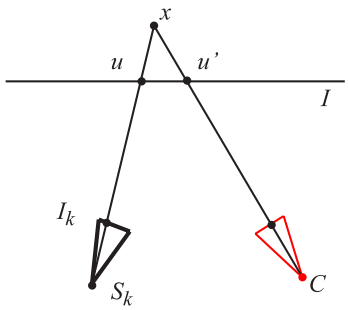

Figure 5: *Pixel reprojection from the sensor's image plane I_k to the Lumigraph image plane I. By knowing the pixel's depth, we can reconstruct the 3-D point X, and then project from there into I. The difference $(u' - u)$ on I is the optical flow of that pixel for the view C.*

Partitioning. For reconstructing an arbitrary view C from the Lumigraph, it is first necessary to determine which sensors should contribute to which part of the reconstructed image. Therefore the camera mesh with vertices S_k is projected into the image plane as seen from the desired viewpoint C (cf. Fig. 2). The resulting image triangle (S'_a, S'_b, S'_c) on I will be interpolated from the image data of the three corresponding sensors S_a, S_b, S_c, since these sensors see the scene from similar directions as the viewing rays passing through the triangle. This means that every sensor S_k should contribute to all samples that project into the *triangle fan* F_k around S'_k (cf. Fig. 6). So this fan is the *target region* that should be filled by warped samples from the corresponding sensor S_k.

This approach does not work properly if the projection of two (or more) camera mesh triangles overlaps on the image plane. That situation can be avoided by restricting the Lumigraph to convex camera meshes [26]. In this case no ray can pass through multiple camera triangles before intersecting the Lumigraph image plane.

Optical flow and ROI detection. Now that we know what the *target* region for every sensor is, we must take into account the optical flow (cf. Fig. 5) and find all those samples that will be "shifted" into the target region due to their optical flow. Since we can only *forward-project* the sensor samples, it is not so easy to determine the sensor pixel that corresponds to a certain output pixel. Therefore, we perform a kind of *inverse mapping* for each triangle fan F_k. Given conservative bounds on the optical flow into the desired target fan F_k, we extend F_k (or its bounding rectangle) by the maximal flow and obtain the *region of interest* (ROI). By projecting all samples from the ROI, we make sure to get all the data that is available for the target region F_k.

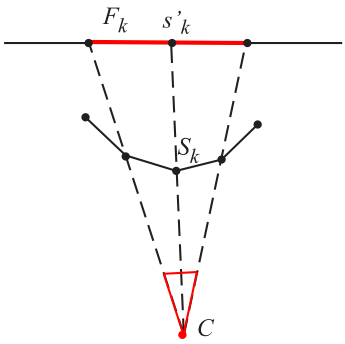

Figure 6: *Partitioning and region of interest. Every sensor S_k must deliver samples for the projected triangle fan around S'_k. C denotes the desired view to be reconstructed.*

In previous work on static two-plane-parameterized Lumigraphs [25], the optical flow and ROI was conservatively estimated using global bounds on the scene's depth. However, since the depth range varies with time, we choose a very simple and convenient approach. Initially, for a sensor S_k we extend the target region F_k by some small offset R_0 (e.g. 10 pixels to each side), and use this as the ROI. During the reconstruction of the first view, we warp all the pixels from the ROI into our desired view. Now we determine the maximal optical flow R_k that occurred during this warping phase, and use $R_k + R_0$ for the reconstruction of the next view. If we assume the changes in the scene to be smooth, this method is sufficient for detecting all necessary sensor samples. If some application requires a truely conservative estimate for the ROI, the maximal pixel flow on I must be derived from the depth value bounds of the current sensor data, in a way similar to that presented in the previous work [25].

Compositing. When a sample is reprojected into the image plane I, it must be composited with other samples that project to the same output pixel. Furthermore, occlusions must be handled correctly. In the case of multiple source images and non-diffuse scenes it is not possible to use an occlusion-compatible reprojection order, so we need to perform a Z-buffer test prior to interpolating the remaining samples. To guarantee a smooth blending, the weight of a pixel should decrease with increasing distance from its sensor's projected vertex S'_k. So we choose the barycentric weight of the pixel position, which is 1 at S'_k, and falls off linearly to 0 on any boundary edge of the triangle fan F_k, cf. Fig. 6. (It should be noted that the barycentric coordinate computation for every sample is relatively expensive, since we must first determine the triangle into which the sample has been projected.) Since the number of samples that map to the same pixel can be arbitrary, we also need to normalize the final sum of weighted samples. This is the main reason why compositing cannot be performed using standard OpenGL blending.

Figure 7: *Preliminary results from our Lumi-Shelf experiments. The moving people in front of the shelf (cf. Fig. 4) can be rendered at near-interactive rates, although the faulty depth reconstruction reduces the image quality considerably.*

5. Results

We have implemented the proposed Lumigraph techniques and tested them on off-the shelf dual-Pentium III Linux PCs running at 800 MHz and equipped with nVIDIA GeForce graphics cards. We used the system to render all combinations of static vs. time-varying and synthetic vs. real-world data.

5.1. Static Lumigraphs / Rendering Performance

The single-processor version of the Lumigraph renderer (no sensor processing and network transfer) typically processes and displays images of 256×256 pixels at 9 frames/sec. The overhead for barycentric coordinate computation (compared to constant blending weights) adds roughly 8–10 percent to the rendering time. The upper left image in Fig. 8* shows the Lumigraph reconstruction of a synthetic fish that has been rendered from nine viewpoints using 3D StudioMax in a resolution of 512×486 pixels. The 8-bit depth values for the fish have been provided directly by the rendering software through a custom plug-in.

The upper right image in Fig. 8* shows a reconstruction of a real-world wooden elk. The depth values for the object have been reconstructed using a voxel coloring technique

in a preprocessing step. The elk's surface is very glossy, and some parts on the head are too thin to be reconstructed with the chosen voxel resolution. Despite these general reconstruction problems, and although the reconstructed depth values are not extremely accurate, still the resulting images are of very high quality.

5.2. Lumigraphs from Animations

For demonstrating the performance of the technique on time-varying imagery with high-quality depth values, we rendered an animation sequence of a talking beetle using the 3D StudioMax software. The 70 frames of the animation were rendered from nine different viewpoints with an image resolution of 512×486 pixels.

We use a playback program that reads the image and depth files from disk one by one and sends them over a network connection to a destination machine. We distributed the nine players plus the display process on three PCs connected via a 100MBit Ethernet LAN. The result is an "immersive movie" playing at roughly 2–4 frames/second, where the user can move freely within the scene while the movie is playing. The lower image row in Fig. 8* shows snapshots from the same movie, but reconstructed from only four of the movie streams at 6–8 frames per second.

Of course for real movie applications one would have to consider a much more efficient and compact encoding and decoding of the image data. If this encoding also allows to extract only the ROI from the movie stream, it might well be possible to run all playback processes and the display process on the same machine while maintaining interactive rates.

5.3. Experiments with the Lumi-Shelf

Finally, we experimented with our Lumi-Shelf prototype (cf. Fig. 4) for demonstrating the processing of real-world video data. As shown in the snapshots in Fig. 7, it is possible to render arbitrary views of the real-world scene at frame rates that are limited mostly by the software depth-from-stereo implementation (currently 1–2 frames/sec). The quality of the resulting views is much worse than that of the rendered animations, mainly due to large number of wrong results in the depth reconstruction. However, the system can render the dynamic scene as observed from the cameras in the shelf, and the user can change the view point freely within the region of space lying in front of the virtual camera surface.

With a better depth sensor (e.g. a better depth-from-stereo software), the image quality should be considerably better, since the basic principle of recombining the sensor data has been demonstrated successfully on synthetic and static real-world scenes in Sections 5.1 and 5.2.

6. Conclusions and Future Work

We introduced the generalized Lumigraph, a versatile concept for rendering from multiple static or time-varying source images. The proposed rendering architecture can be used in conjunction with any type of image and depth sensor, and makes no restricting assumptions on the scene. We applied and extended various techniques and algorithms to enable a complete on-the-fly processing of all data, and showed that we yield interactive or near-interactive frame rates, depending mainly on the Lumigraph image resolution and on the computational cost for reconstructing depth values on the fly. We demonstrated the quality and flexibility of our concept by applying it to various synthetic and real-world scenarios.

In the future, it seems imperative to extend this approach to light field parameterizations that allow seamless all-around viewing. One step in this direction is to use arbitrary convex camera meshes [26] and warp directly into a plane parallel to the user's image plane. It is also important to use multi-resolution techniques, e.g. when the sampling density on the output image plane varies strongly between the different sensors. Furthermore, it seems very promising to combine our technique with some kind of efficient multi-view video encoding in order to reduce memory and network transfer costs. One could further think of extending the technique to time-varying camera parameters, e.g. to moving or zooming cameras.

But still one of the most important challenges for the future remains the research for robust, fast, and non-invasive sensors that can deal with arbitrary non-diffuse scenes.

Acknowledgments

The authors wish to thank Prof. Philipp Slusallek for making his video lab available for the Lumi-Shelf experiments, as well as for the numerous fruitful discussions. Thanks to Michael Repplinger for his Linux Fire-Wire library support, to Markus Weber for generating the animation sequences using 3D StudioMax, and to Pascal Schüler for reconstructing the elk. Special thanks to Hendrik Lensch and Michael Gösele for help with the manuscript.

References

1. Emilio Camahort and Don Fussell. A geometric study of light field representations. Technical Report TR99-35, Department of Computer Sciences, University of Texas, 1999.
2. Emilio Camahort, Apostolos Lerios, and Donald Fussell. Uniformly sampled light fields. *Eurographics Rendering Workshop 1998*, pages 117–130, 1998.
3. Jin-Xiang Chai, Xin Tong, Shing-Chow Chan, and Heung-Yeung Shum. Plenoptic sampling. *Proc. SIGGRAPH 2000*, pages 307–318, 2000.
4. S. Chen and L. Williams. View interpolation for image synthesis. In *Proc. SIGGRAPH '93*, pages 279–288. ACM, 1993.

5. H. Fuchs, G. Bishop, K. Arthur, L. McMillan, R. Bajcsy, S. Lee, H. Farid, and Takeo Kanade. Virtual space teleconferencing using a sea of cameras. In *Proc. First International Conference on Medical Robotics and Computer Assisted Surgery*, pages 161 – 167, June 1994.

6. S.J. Gortler, R. Grzeszczuk, R. Szeliski, and M.F. Cohen. The Lumigraph. In *Proc. SIGGRAPH '96*, pages 43–54, 1996.

7. W. Heidrich, H. Schirmacher, H. Kück, and H.-P. Seidel. A warping-based refinement of Lumigraphs. *Seventh International Conference in Central Europe on Computer Graphics and Visualization (Winter School on Computer Graphics)*, 1999.

8. Insung Ihm, Sanghoon Park, and Rae Kyoung Lee. Rendering of spherical light fields. *Pacific Graphics '97*, 1997. Held in Seoul, Korea.

9. Aaron Isaksen, Leonard McMillan, and Steven J. Gortler. Dynamically reparameterized light fields. *Proc. SIGGRAPH 2000*, pages 297–306, 2000.

10. Yuichi Ohta Kiyohide Satoh, Itaru Kitahara. 3D image display with motion parallax by camera matrix stereo. In *Proc. 3rd Intl. Conf. on Multimedia Computing and Systems (ICMCS'96)*, 1996.

11. S. Laveau and O.D. Faugeras. 3-D scene representation as a collection of images. *ICPR-A*, 94:689–691, 1994.

12. M. Levoy and P. Hanrahan. Light field rendering. In *Proc. SIGGRAPH '96*, pages 31–42, 1996.

13. W. Mark, L. McMillan, and G. Bishop. Post-rendering 3d warping. In *Proc. 1997 Symposium on Interactive 3D Graphics*, pages 7–16, 1997.

14. Wojciech Matusik, Chris Buehler, Ramesh Raskar, Steven J. Gortler, and Leonard McMillan. Image-based visual hulls. *Proc. SIGGRAPH 2000*, pages 369–374, 2000.

15. N. Max and K. Ohsaki. Rendering trees from precomputed z-buffer views. In *Rendering Techniques '95 (Proc. Eurographics Workshop on Rendering)*, pages 165–174, 1995.

16. L. McMillan and S. Gortler. Image-based rendering: A new interface between computer vision and computer graphics. *Computer Graphics*, 33(4), November 1999.

17. Leonard McMillan. *An Image-based Approach to Three-Dimensional Computer Graphics*. PhD thesis, University of North Carolina at Chapel Hill, 1997.

18. Leonard McMillan and Gary Bishop. Plenoptic modeling: An image-based rendering system. *Proc. SIGGRAPH 95*, pages 39–46, 1995.

19. Lars Nyland, David McAllister, Voicu Popescu, Chris McCue, Anselmo Lastra, Paul Rademacher, Manuel Oliveira, Gary Bishop, Gopi Meenakshisundaram, Matt Cutts, and Henry Fuchs. The impact of dense range data on computer graphics. In *Proc. Multi-View Modeling and Analysis Workshop (MVIEW99), Part of CVPR99*, June 23-26, 1999.

20. Manuel Oliveira and Gary Bishop. Factoring 3-D image warping equations into a pre-warp followed by conventional texture mapping. Technical Report TR99-002, Department of Computer Science, University of North Carolina at Chapel Hill, 1999.

21. Manuel M. Oliveira, Gary Bishop, and David McAllister. Relief texture mapping. *Proc. SIGGRAPH 2000*, pages 359–368, 2000.

22. Peter Rander, P.J. Narayanan, and Takeo Kanade. Virtualized reality: Constructing time-varying virtual worlds from real events. In *Proc. IEEE Visualization '97*, pages 277–283, 1997.

23. Ramesh Raskar, Greg Welch, Matt Cutts, Adam Lake, Lev Stesin, and Henry Fuchs. The office of the future: A unified approach to image-based modeling and spatially immersive displays. *Proc. SIGGRAPH 98*, pages 179–188, 1998.

24. G. Schaufler and M. Priglinger. Efficient displacement mapping by image warping. In *Rendering Techniques '99 (Proc. 10th Eurographics Workshop on Rendering)*, 1999.

25. Hartmut Schirmacher, Wolfgang Heidrich, and Hans-Peter Seidel. High-quality interactive Lumigraph rendering through warping. In Sidney Fels and Pierre Poulin, editors, *Proc. Graphics Interface 2000*, Montreal, Canada, 2000. CHCCS, CHCCS.

26. Hartmut Schirmacher, Christian Vogelgsang, Hans-Peter Seidel, and Günther Greiner. Free form light fields. Technical Report 3/2001, IMMD 9, Universität Erlangen-Nürnberg, Erlangen, Germany, 2001.

27. S. Seitz and C. Dyer. Photorealistic scene reconstruction by voxel coloring. In *Proc. CVPR*, pages 1067– 1073, 1997.

28. Jonathan Shade, Steven J. Gortler, Li wei He, and Richard Szeliski. Layered depth images. *Proc. SIGGRAPH 98*, pages 231–242, 1998.

29. Peter-Pike Sloan, Michael F. Cohen, and Steven J. Gortler. Time critical Lumigraph rendering. *1997 Symposium on Interactive 3D Graphics*, pages 17–24, 1997.

30. G. Tsang, S. Ghali, E.L. Fiume, and A.N. Venetsanopoulos. A novel parameterization of the light field. In H. Niemann, H.-P. Seidel, and B. Girod, editors, *Image and Multidimensional Digital Signal Processing '98 (Proc. 10th IMDSP Workshop)*. infix, 1998.

31. Stanford University. Light field camera project. `http://www-graphics.stanford.edu/projects/lightfield/`.

32. Jason C. Yang and Leonard McMillan. Light fields on the cheap. SIGGRAPH 2000 technical sketch.

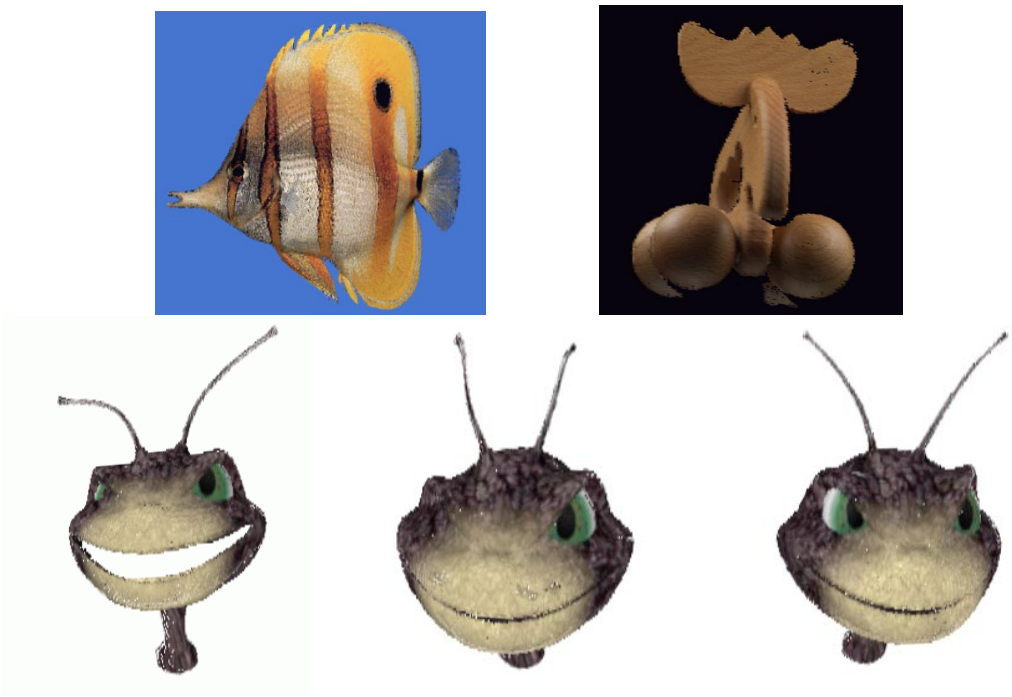

Figure 8: *[Schirmacher et al.] Upper row: static scenes. The fish is reconstructed from nine images with* 512×486 *pixels each, at 1.5–2.5 fps. The real elk's geometry has been reconstructed using the voxel coloring technique in a preprocess, and rendering (from six* 760×510*-pixel images) runs at 1–2 fps. Lower row: Some images from an "immersive movie". A talking beetle animation has been rendered and recorded from four to nine different viewpoints in a resolution of* 512×486*. The viewer can move freely while the Lumigraph movie is running over the network.*