EUROGRAPHICS '99 / P. Brunet and R. Scopigno
(Guest Editors)

Volume 18 (*1999*), *Number 3*

Adaptive Acquisition of Lumigraphs from Synthetic Scenes

Hartmut Schirmacher, Wolfgang Heidrich, and Hans-Peter Seidel†

Computer Graphics Group (AG 4)
Max-Planck-Institut für Informatik
Im Stadtwald
66123 Saarbrücken, Germany

Abstract

Light fields and Lumigraphs are capable of rendering scenes of arbitrary geometrical or illumination complexity in real time. They are thus interesting ways of interacting with both recorded real-world and high-quality synthetic scenes.

Unfortunately, both light fields and Lumigraph rely on a dense sampling of the illumination to provide a good rendering quality. This induces high costs both in terms of storage requirements and computational resources for the image acquisition. Techniques for acquiring adaptive light field and Lumigraph representations are thus mandatory for practical applications.

In this paper we present a method for the adaptive acquisition of images for Lumigraphs from synthetic scenes. Using image warping to predict the potential improvement in image quality when adding a certain view, we decide which new views of the scene should be rendered and added to the light field. This a-priori error estimator accounts for both visibility problems and illumination effects such as specular highlights.

1. Introduction

Light fields[8] and Lumigraphs[5] are two image-based rendering concepts for reconstructing arbitrary views interactively from a database of dense samples. Since these methods can deal in real time with scenes of arbitrary visual and geometrical complexity (e.g. objects with very sophisticated reflection characteristics and inter-object occlusion), they are often the only way of exploring objects or whole scenes interactively. This is true both for synthetic scenes (used for lighting simulation etc.) and for reconstructing real-world scenes from video streams or sets of still images.

In the context of lighting simulation, for example, the current state-of-the-art is to demonstrate results through still images of a single view or a film showing a number of predetermined views. With techniques like light field rendering at hand, it seems desirable to capture the results of a sophisticated rendering algorithm or of some novel global illumination algorithm in a light field or Lumigraph representation. This gives the user or researcher a much better control over the viewing process (e.g. for visual quality control or even debugging) and leads to attractive ways of presenting rendering results.

The most apparent down side of both the light field and the Lumigraph approaches is the need for a very dense sampling of the light distribution or plenoptic function[1]. This imposes high costs for acquisition and storage. When rendering light field images from synthetic scenes, the acquisition of a single image (e.g. by ray tracing) can take many minutes or even hours.

When acquiring from video streams or still images, each individual image must be calibrated in order to reconstruct the viewing transformation, and then resampled to the internal light field parameterization, even if the resulting data structure only contains a small fraction of the original number of samples.

From this discussion it should be clear that one key interest in the context of light field and Lumigraph rendering lies in optimizing the set of viewpoints with respect to the over-

† email: *lastname*@mpi-sb.mpg.de

all reconstruction quality which can be achieved using the images associated with these views. Due to the high cost of image acquisition, it is imperative to have an *a priori* quality estimate for viewpoints. Using such a criterion, one can incrementally predict the optimal location for the next viewpoint to be inserted.

In contrast to the light field approach, the Lumigraph uses approximate geometric information in addition to the dense light samples. This information is used to improve the reconstruction quality for synthetic views. In this paper, we also employ geometric information, although in a different format, to decide which additional samples should be rendered and added to the light field or Lumigraph data structure in order to improve the quality.

More specifically, we propose an algorithm for the adaptive acquisition of images for Lumigraphs parameterized by two parallel planes from synthetic scenes. The resulting database can be used with both the light field and the Lumigraph techniques to generate new views from arbitrary positions.

To this end, we develop an a-priori error estimate predicting the gain in reconstruction quality when adding an arbitrary viewpoint to the image database. This error estimate is based on warping a subset of the already acquired images to the candidate viewpoint and estimating the resulting error due to inter-object visibility and due to change in color (e.g. for specular materials). In other words, the estimate predicts how well the image for a specific viewpoint could be reconstructed by means of warping. We demonstrate the effect of our algorithm by applying it to test scenes and showing the corresponding viewpoint meshes and reconstruction results. Finally, we discuss the advantages and drawbacks of our approach and point out possible directions of future research.

2. Previous Work

The work presented in this paper has its roots in light field and Lumigraph rendering as well as in image warping and morphing. Recently, much attention has been paid to both topics as image based rendering is becoming more and more important to the world of computer graphics. We will first review the work on light field representation and rendering, and then proceed with work on warping and Lumigraph refinement which relates to our error estimator. Finally we will shortly discuss related work in the field of ray tracing data interpolation.

2.1. Light Fields and Lumigraphs

Lumigraph and light field rendering have been introduced by Gortler et al.[5] and Levoy and Hanrahan[8]. Both approaches are based on storing samples of the so-called *plenoptic function*[1] which describes the directional radiance distribution for every point in space. The subset of that function in an occlusion-free space outside the scene can be represented in 4 dimensions. Both papers propose a light field parameterization by two parallel planes (like Camahort et al[3], we will in the following use the term 2PP). Figure 1 demonstrates the 2PP parameterization scheme. Every viewing ray passing through the scene is characterized by a pair of points (s,t) and (u,v) on the two planes. The set of all (u,v) samples through a single point (s_0,t_0) on the (s,t) plane is an image created by a sheared perspective projection from (s_0,t_0). In what follows, we refer to the (s,t) plane as the *viewpoint plane* and to the (u,v) plane as the *image plane*. The set of rays passing through the two planes is called a *light slab*. In order to view a scene from any point in the surrounding space, six light slabs are combined so that the six viewpoint planes cover some box surrounding the scene.

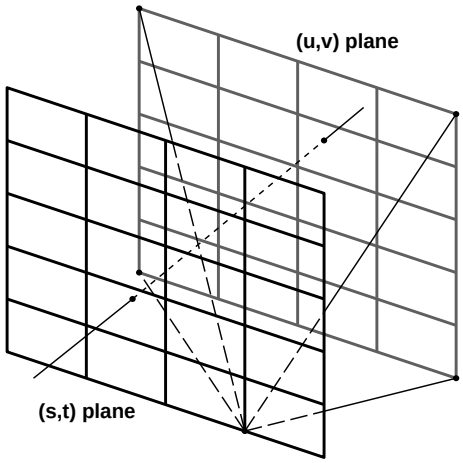


Figure 1: *Schematic view of a two-plane parameterized (2PP) light slab. Rays passing through a scene are characterized by a pair $[(s,t),(u,v)]$ of points. The set of all rays passing through one (s,t) point is a sheared perspective image in the (u,v) plane.*

Other light field parameterizations have also been proposed. Ihm et al.[7] presented the notion of *spherical light fields*, where rays are represented as a point on a sphere acting as a bounding volume for the object, plus a direction, which is also defined via a point on a sphere. Both spheres are subdivided adaptively. The rendering times are much slower than those using the 2PP, mostly due to the wavelet compression scheme the authors employed.

Camahort et al.[3] propose a two-sphere-parameterization (2SP) and a sphere-plane-parameterization (SPP). For the 2SP, rays are represented by their two intersection points $(u,v),(s,t)$ with the same sphere. For 2PP, the ray is defined by a normal direction (θ,ϕ) specifying the image plane through the center of the sphere, and by some (u,v) point on that plane. A regular (u,v) sampling thus results in an orthographic view of the scene. The performance of both param-

eterizations when used with a software reconstruction algorithm and nearest neighbor sampling is comparable to those of an irregular view point mesh using 2PP[9] and linear interpolation.

These alternative parameterizations have the advantage that they provide a more uniform sampling and avoid potential seams between slabs. On the down side, with the spherical and the 2SP parameterization, it is not possible to employ graphics hardware like Gortler et al.[5] for the rendering step, which is due to the non-regular sampling. Since hardware reconstruction is by far the fastest way of rendering light fields or Lumigraphs, this is a severe penalty. Also, the 2PP allows us to use a cheaper warping algorithm that produces less visibility problems[6]. These two facts are the reason why we apply the 2PP for our work.

Interactive View Reconstruction. The benefit of a dense sampling of the plenoptic function is that the "light database" can be queried very efficiently in order to create arbitrary views of the scene. With the 2PP and a regular sampling on both the image and the viewpoint plane, the radiance along an arbitrary ray can be computed through simple quadri-linear interpolation from the radiance stored for all rays through any combination of the the 2×2 nearest (s,t) neighbors and the 2×2 nearest (u,v) neighbors (see Figure 1). This process has constant time complexity. If an adaptive sampling on the viewpoint plane is used, as in the paper by Sloan et al.[9], then this process takes logarithmic time, since the triangle of viewpoints through which the ray passes has first to be determined. The same applies to multiresolution approaches, e.g.[7, 3]. Both regular and adaptive samplings can also be rendered very efficiently in hardware, using texture mapping and alpha blending[5, 9].

Depth Correction. The additional depth information stored in a Lumigraph in the form of a polygonal mesh allows for depth-correcting the viewing rays before the interpolation step. While linearly interpolated images become more and more blurry with increasing distance between the neighboring viewpoints, the depth correction can compensate that effect by some degree. The finer the mesh, the more accurate this correction can be. On the other hand, interactive displaying requires a very coarse mesh to be used (several hundred triangles at most). The idea of depth correction is that, with help of the geometric information, one can find the approximate intersection point of a viewing ray with the scene geometry. This allows for backward mapping that point into the source images, finding more appropriate (u,v) coordinates to interpolate from. Figure 2 illustrates this situation in two dimensions.

The depth correction can be performed for each viewing ray if the rendering is done in software, or it can be done for the vertices of each textured triangle which is drawn when using graphics hardware. Since the latter approximation is only valid if the depths of a triangle's vertices do not differ too much, Gortler et al. propose to subdivide the textured triangles until all depth values for one triangle are similar.

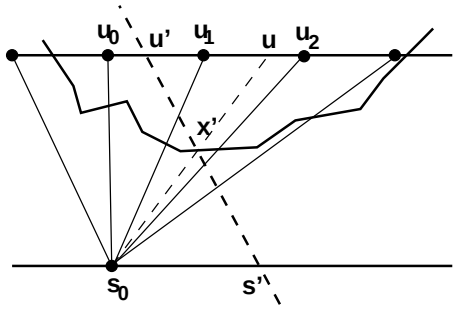


Figure 2: *Depth correction scheme sketched in 2D. The viewing ray (s',u') intersects the geometry at x'. For some nearby viewpoint s_0, instead of interpolating the color from u_0 and u_1 (which are the neighbors of u'), it is more appropriate to use u's neighbors u_1 and u_2 since they hold information about the color near x. u is found by intersecting (s_0,x') with the image plane (backward mapping).*

2.2. Coarsening and Refining Lumigraphs

Sloan et al.[9] organize viewpoints in an adaptive triangle mesh in order to control the number of viewpoints used for reconstruction. This is often desirable since the number of active viewpoints directly corresponds to the number of images that must be held in memory in order to reconstruct new views, and, for hardware implementation, also to the required amount of texture memory.

Starting with a fine, regular mesh of viewpoints, they coarsen the mesh by neglecting viewpoints in the Lumigraph that contribute little to the image quality. They predict the benefit and the cost of using a specific viewpoint (s,t) by using the distance of (s,t) from the viewing ray, from the predicted camera path, and from other currently valid viewpoints. This taxonomy is valid for the simple interpolation of images, because the quality of such an interpolated image only depends on the distance between the source images' centers of projection. However, this kind of prediction does not include the errors after a depth-corrected interpolation and, most importantly, only makes sense in the context of viewing, not during the light field acquisition.

Heidrich et al.[6] go the opposite way by adding new images into a sparse Lumigraph, thereby increasing the quality of images generated by quadri-linear interpolation. The new images are added by warping the closest images from the Lumigraph to a new viewpoint that also lies on the viewpoint plane in the 2PP. This warping of course also requires some geometric information, which they assume to be available in the form of a depth value per sample in the Lumigraph.

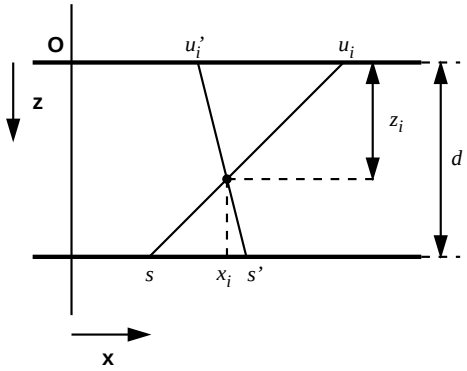


Figure 3: *Geometry for warping a pixel u_i with depth z_i from a view s to its new position u_i' in some novel view s'. d denotes the distance of the two Lumigraph planes.*

For synthetic images as in our application, these are easy to acquire.

This warping step directly corresponds to the depth correction in the Lumigraph approach, but it is more precise (due to the fact that per-pixel depth is used instead of a coarse polygonal mesh), and can be done as a preprocessing step, so that the actual on-screen rendering only requires linear interpolation, which is faster than depth corrected rendering. However, the size of the image database used for interactive rendering increases.

Due to the use of the 2PP, the warping equations are very simple. In fact they correspond to the depth correction formulae presented by Gortler et al.:

$$u_i' = \begin{cases} u_i - \frac{z_i}{d-z_i}\Delta s & z_i > -\infty \\ u_i + \Delta s & z_i = -\infty \end{cases}, \qquad (1)$$

where $\Delta s := s' - s$ is the translation of the view point in s direction, and $z_i = -\infty$ applies for background pixels. The pixel position u_i' in the novel view s' can be computed from the position u_i in the reference view s. The corresponding geometry is depicted in Figure 3.

Due to these simple formulae, the warping is very fast (more than 10 frames per second), and because of the special geometry of the 2PP, it produces few visibility artifacts[6].

2.3. Interpolating Ray-Tracing Data

While our work mainly concentrates on how to estimate a reconstruction error from pure image and range data (color and depth values only), there exist other approaches for avoiding the costly ray-tracing by interpolation. For example, in Teller et al.[10] and Bala et al.[2], the authors employ more sophisticated per-pixel data like surface normals and reflectance parameters. They warp pixels into their new location and interpolate the radiance value in 4D ray space, if possible. By exploiting surface and reflection data, they are able compute a priori error estimates for the radiance interpolation. This way, they can analytically predict small specularities, blockers, holes, and funnels for some kinds of reflectance models.

While this approachs seems quite convenient for speeding up the ray tracing of specific scenes, it requires modifying the kernel of the ray tracing code and does not seem suitable for being extended to real-world data like photographs. The image-based approach uses the ray tracer as a black box providing color and range data, and can very likely be extended to create Lumigraphs from sequences of camera and range sensor images. On the down side, the image based approach may miss small specular highlights if (and only if) they do not appear in any of the reference images (cf. discussion in Section 5).

3. Adaptive Lumigraph Acquisition

The idea of this paper is to perform an adaptive acquisition of views for Lumigraphs from synthetic scenes by incrementally refining an initially undersampled Lumigraph. Rendering is the only way to suppress artifacts resulting from illumination changes, as well as visibility problems in the sparse sampling. In order to do so, we use a modified Lumigraph refinement algorithm[6] for predicting the view point location which increases the reconstruction quality of the Lumigraph most. We construct an adaptive mesh containing the current view points as vertices, predict the benefit for inserting a new viewpoint at each of the edges, and choose to split the edge with the most benefit. The computation of the benefit is based on warping the nearest source images to the candidate point and estimating the morphing error including holes, pixels with only a single mapping source pixel, and the color blending error. Potential views with a small error will only have a small benefit since they could be reproduced accurately by the Lumigraph refinement technique presented in[6]. Since the insertion of a novel view point into the Lumigraph is usually very costly (e.g. ray tracing a complex scene can take many minutes or even several hours), the adaptive technique presents a way of minimizing the effort for acquiring a Lumigraph of a certain quality.

3.1. Estimating the Reconstruction Error

In order to estimate the error or potential benefit associated with any candidate viewpoint, we morph the new image from the nearest reference views. This warping step is performed along the lines of [6], which means that every pixel i from every selected reference frame is warped to its new location in the destination image by using Eq. 1. For every destination pixel j with coordinates (u_j, v_j), let the *source map* $M_j = \{i_0, i_1, i_2 \ldots\}$ be the set of pixels from different source images which map to (u_j, v_j). We determine the front most pixel $i_{\max}$ with the maximal Z value $z(i_{\max})$ from among M_j. Now we construct the *blend map* M_j' which is the subset M_j

containing all pixels at a depth within an ε-interval around z_{max}:

$$M'_j \;=\; \left\{ i \in M_j \,|\, |z(i) - z(i_{\max})| < \varepsilon \right\} \qquad (2)$$

The final color $Avg(j)$ of the pixel is determined by blending from all pixels in M' using a weighted sum:

$$Avg(j) \;=\; \frac{1}{\sum w_k} \sum_{k=0}^{|M'_j|} w_k Color(i_k). \qquad (3)$$

The weights account for the importance of each source image with respect to the final image and can, for example, be chosen as the inverse distance between the destination view point (s',t') and the view point (s_k,t_k) associated with the source pixel i_k:

$$w_k \;=\; \frac{1}{\left\| \begin{pmatrix} s' \\ t' \end{pmatrix} - \begin{pmatrix} s_k \\ t_k \end{pmatrix} \right\|}. \qquad (4)$$

In order to estimate the *blending error* $E_j^{(B)}$ associated with destination pixel j, we compute the weighted L_2 error which accounts for the deviations from that final color:

$$E_j^{(B)} \;=\; \frac{1}{\sum w_k} \sqrt{\sum_{k=0}^{|M'_j|} [\, w_k \| Color(i_k) - Avg(j) \| \,]^2}. \qquad (5)$$

Other sources of error in the reconstruction of a pixel j in the destination image are *holes* and *single maps*. In the case of holes, no source pixel maps to the destination pixel ($|M'_j| = 0$), and in the case of single maps only one source pixel is warped onto the destination location ($|M'_j| = 1$). Both situations will not induce any blending error, since at least two different color samples for the same pixel are needed to result in a non-zero error. However, in the case of holes and single maps, we cannot simply assume zero error for the target pixel, since we do not have enough information to estimate a plausible color range. In order to get a truely conservative error bound, it is necessary to assume that an arbitrary reconstruction error could result from such a pixel. To this end, we let holes and single maps contribute some constant error to the overall error estimate (see Eqs. 7, 8 below).

If Lumigraphs are used for generating views all around an object, it is reasonable to assume that the reference views contain the complete object. In other words, no part of the object lies outside of any reference view. This assumption can be exploited to tighten the error bound and eliminate an unneccessary part of the still conservative estimate. To this end, we check if holes map to the outside of any of the reference views. If the object lies completely within each view, the outside of a reference image is assumed to be background. So we only need to reversely apply the background case of Eq. 1 to the hole pixel in the destination image. A pixel j with destination coordinates (u'_j, v'_j) corresponds to a pixel outside the domain $dom(I)$ of a reference image I if

$$\left[\begin{pmatrix} u'_j \\ v'_j \end{pmatrix} - \begin{pmatrix} \Delta s(I) \\ \Delta t(I) \end{pmatrix} \right] \notin dom(I). \qquad (6)$$

Under the assumption of completely visible objects, we define the predicate $out(j)$ to be true if Eq. 6 holds for any of the reference images used for warping. For other light field types, e.g. outward-looking Lumigraphs, it is always false.

Now we can define the contribution of a hole pixel and of a single map pixel to the overall error. Holes induce a non-zero error contribution if they do not backward-map to the outside, and single mapped pixels contribute if they are not background. Using this, we define the *hole error* and the *single map error* for a pixel j as follows:

$$E_j^{(H)} \;=\; \begin{cases} w_H & |M'_j| = 0 \;\wedge\; \neg out(j) \\ 0 & else \end{cases}, \qquad (7)$$

$$E_j^{(S)} \;=\; \begin{cases} w_S & |M'_j| = 1 \;\wedge\; z(j) > -\infty \\ 0 & else \end{cases}. \qquad (8)$$

It is reasonable to choose the weights w_H and w_S from the same range as the norm of the color values, because a single pixel should not contribute more to the error than the maximal possible color deviation. Section 4 will give some hints about choosing appropriate error weights.

Now that we can determine the total per-pixel error by taking into account all three kinds of error (blending color difference, holes, and single maps), we can sum up this error using the L_2 norm in order to obtain some scalar error estimate for the whole view:

$$E \;=\; \frac{1}{N} \sqrt{\sum_{j=0}^{N-1} \left[E_j^{(B)} + E_j^{(H)} + E_j^{(S)} \right]^2}, \qquad (9)$$

where N is the number of pixels in the destination image. The resulting total error E lies in the same interval as the per-pixel error, with smaller values indicating better reconstruction quality.

Figure 12 (see color plates) visualizes the error distribution of some candidate viewpoint. It distinguishes between true holes (light red) and outside-mapping holes (darker red) as well as foreground and background single maps (light and darker green). All other colors show blending errors, with darker colors referring to higher error.

3.2. Adaptive Rendering Algorithm

In order to refine the set of viewpoints incrementally, we partition the viewpoint domain into a triangular mesh with the currently acquired views being assigned to the triangle vertices. For example, we start with four vertices/views representing the four corners of a rectangle in the viewpoint plane and create a mesh with two triangles. Figure 4 illustrates such an initial mesh. Each new vertex in the mesh implies the acquisition of a new image with the viewpoint as the center

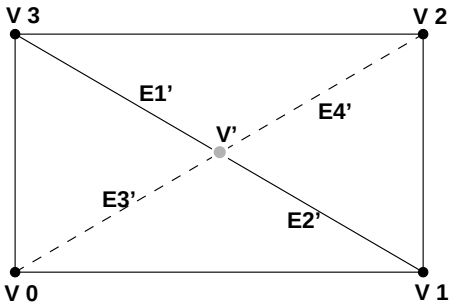


Figure 4: *A simple initial viewpoint mesh consisting of four corner vertices $V_0 \ldots V_3$ and five edges (solid lines). V' is a candidate viewpoint for the diagonal edge (V_1, V_3). When inserting V', the four edges $E'_1 \ldots E'_4$ would be created, so the new view would be warped from the images at $V_0 \ldots V_3$.*

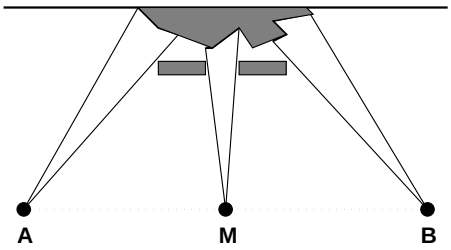


Figure 5: *If the edge (A, B) is split, the new viewpoint M observes parts of the scene which have been occluded when viewed from A and B. This can result in higher error values for the new half edges (A, M) and (M, B) as for the full edge (A, B).*

of a sheared perspective projection. Due to the cost of acquiring such an image, it seems reasonable to consider only refinement strategies which insert a single new vertex into the mesh. One such strategy is to split an edge by inserting a new vertex at the center of that edge. The vertex will be connected to the start and end vertex of the original edge, and with the third vertices of the triangles sharing that original edge. For edges at the border of the mesh, there is only one such triangle. For inner edges, there are exactly two. Figure 4 illustrates the topological changes when splitting an inner edge and inserting a new edge mid point.

Edge Weights. Each edge of the triangulation is assigned a weight corresponding to a *splitting benefit* value, which is given by the warping error (Eq. 9) of its midpoint. The weight indicates the benefit of inserting a new viewpoint at the edge's center. In order to compute that benefit, we need to determine all direct neighbours of the new potential viewpoint. The set of potential neighbors contains all vertices which would be directly connected to the new vertex if the edge were split. We can infer from Fig. 4 that these vertices include the start and end point of the split edge plus the remaining vertices of the one (in case of a border edge) or two triangles sharing the old edge.

After inserting the new vertex V', we must compute or update the weights of all edges in all triangles containing the new vertex, because the new vertex is a potential neighbor of the mid points of all those edges and thus contributes to all of the associated error values. In other words, the new vertex V' must be considered as one of the warping sources for computing any edge weight in the triangle fan around V'. For example, the insertion of V' in Fig. 4 would induce the weight recomputation for all remaining and new egdes (eight edges total).

Effect of splitting on the weights. When an edge (A, B) is split and two new edges (A, M) and (M, B) are created, one would like the weights of the new half edges to be smaller than that of the old split edge. Usually, this proves to be true, since the part of the error which is dominated by the differences between A and B decreases (e.g. holes and single map regions become smaller, and the blending error decreases since the angular difference from which points in the scene are observed gets halved). However, one can think of a few circumstances which can temporarily prevent the error from decreasing:

- New holes or single maps can occur when a viewpoint can see regions which have been completely occluded before (see Fig. 5). This can cause a temporary increase of the weight on the split edge, but after the new half edges are split, the occluded regions become smaller and the error decreases. Since the scene details that can be observed are limited to the size of a (u, v) pixel, the increase of error must terminate at some point.
- The error can be dominated by the other one or two neighbors orthogonal to the splitting direction (C and D in Fig. 6). If the error due to the difference between C and D does not get significantly smaller when inserting new points $M, M', \ldots$, this could generate an infinite sequence of splits of the same edge. But on the other hand, if the difference between C and D is big, the weights of (C, M) and (D, M) must also be significantly big. Note that this kind of error domination cannot be caused by blending error, since the blending colors are weighted with the inverse view point distance. So in the case sketched in Fig. 6 the shorter edges would become more and more important, while the C/D edges would lose their influence on the overall blending error.

Since the error function behaves the way described above, it seems reasonable to use additional geometrical methods for maintaining triangulation quality.

Triangulation quality. Like in most cases when employing an adaptive triangulation, we want the mesh to obey some geometric quality conditions. For example, there should be no very sharp angles which lead to degenerated triangles, and we want to prevent triangles from becoming too

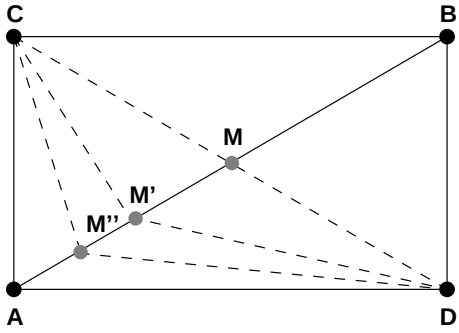


Figure 6: *Suppose the error of viewpoints on the edge (A,B) is dominated by C and D. If we insert M, the error on (A,M) and (M,B) can still be nearly as high as before. This could cause an infinite sequence of splits creating $M, M', M'', \ldots$*

small compared to the rectangle on the viewing plane. Both conditions are generally unwanted since they would have the effect that a view will get interpolated from several almost similar points. In order to prevent those degeneration, the triangle quality is incorporated into the weight computation. An edge *e* will be assigned a benefit of 0 if either the opposite angle or the area of one of the triangles sharing *e* are too small.

The resulting triangulation should also be "well conditioned" in the sense that every vertex is connected to its nearest neighbors via direct edges. The best way to achieve this is to update the triangulation incrementally after inserting a new vertex in order to assert the triangulation is locally Delaunay (for more information about Delaunay triangulations, see any book on computational geometry, e.g. [4]). However, the update of the triangulation involves the flipping of edges which can induce the computation of many new edge weights as well as an update of old weights. This increases the aquisition overhead (see results section).

Main Algorithm. The overall algorithm for adaptively creating Lumigraphs as sketched in Fig. 7 consists of creating an initial triangulation and a loop for inserting new images/vertices into the Lumigraph. Instead of terminating the program after a fixed number of splits, one could also end the main loop after the maximal weight has dropped below some maximal error bound ε. For many applications, however, one would like to specify the number of viewpoints used in a Lumigraph, e.g. due to the limited resources of the machine.

4. Results

We have built an experimental implementation of the adaptive Lumigraph acquisition algorithm as sketched in Sec. 3.2 and used it for computing adaptive Lumigraphs of several test scenes. In the experiments presented here, we have used the following parameters:

```
N ← number of desired viewpoints
create initial vertices V
for each vertex v∈V do
  acquire image I_v
create initial edges E
for each edge e∈E do
  compute benefit B_e
while |V| < N do {
  select e∈E with B_e = max{B_e' | e'∈E}
  split edge e
  v ← newly created vertex
  E_n ← {all edges of all triangles
        around v}
  acquire image I_v
  if [needed] then {
    flip non-Delaunay edges in V
    E_n ← E_n ∪ {all edges of all
              changed triangles}
  }
  for each edge n∈E_n do
    compute benefit B_n
}
```

Figure 7: *Simplified pseudo code for adaptive Lumigraph acquisition algorithm.*

- hole weight $w_H = 0.1 \times$ maximal color value
- single map weight $w_S = 0.1 \times w_H$
- do not allow splits if some angle would drop below 15 degrees
- do not allow a triangle to become smaller than $1/10000$th of the viewpoint domain

Fig. 9 (see color pages) shows the four extreme views of the "elephant" scene which is a ray-traced RenderMan object of wood material defined by a volumetric shader. These four views are used as the images associated with the initial vertices as sketched in Fig. 4.

The image resolution is 256×256 pixels, and ray tracing of a single view with an external rendering package takes approximately four minutes. In contrast, warping and error estimation for a view from four reference views consumes less than a second on a Silicon Graphics O2 195MHz R10k machine (without utilizing highly optimized code).

Fig. 12 (see color pages) displays the viewpoint plane meshes generated after 40, 100, and 160 iterations of the algorithm. Right from the beginning, the lower region of the viewpoint plane gets sampled more densely due to the greater changes in color (highlights) and visibility (leg occlusions) when viewing the elephant from below. After 160 iterations, the time consumed for the Lumigraph acquisition was approximately ten and a half hours for ray tracing and 17 minutes for error estimation after split operations. Instead of the proposed incremental Delaunay triangulation, we per-

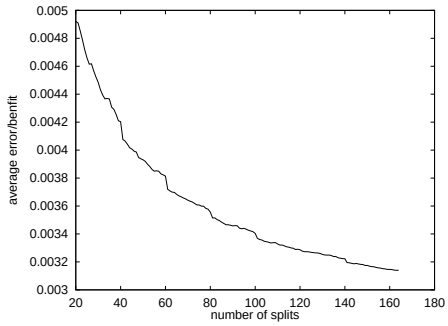


Figure 8: *Average error/benefit value after each iteration for the "elephant" scene. The error decreases quite smoothly during the whole acquisition phase.*

formed a global Delaunay retriangulation of the mesh every 20 iterations, which required eight times the recomputation of all weights. This took another 30 minutes, but this cost could be significantly reduced by applying the smarter incremental update of the Delaunay triangulation as proposed in Section 3. In total, the overhead for the adaptive acquisition lies somewhere between 2.5% – 7.5% of the rendering time even for this relatively simple scene. For more complex scenes, the overhead is negligible.

In order to analyze the contribution of holes, single maps, and blending error to the total error bound, we have visualized the different error contributions for each viewpoint before its insertion. Fig. 12 (see color pages) shows the color-coded error distributions for each worst potential viewpoint (which means the next viewpoint to be inserted) after 40, 100, and 160 iterations, respectively. Please note that the colors of the error images have been enhanced in order to render the blending error a little bit more visible. As one can see, the initially large hole and single map regions (red and green, respectively) become smaller and smaller, although new small holes may pop up like in the bottom-most error image. The blending error also decreases drastically as the viewpoints fall closer together.

In order to validate the convergence of our approach towards a good Lumigraph reconstruction quality, we have plotted the average error/benefit value against the iteration number in Fig. 8. The error decreases after almost every split operation, and the curve converges to zero. In accordance with these results, Fig. 10 shows an image which has been generated by the Lumigraph refinement morphing[6] for the next potential viewpoint after 160 iterations. The image quality is very high, and the image cannot be distinguished by eye from the ray traced image, although one can guess the differences from the bottom-most error image in Fig. 12.

Finally, we have tested the resulting Lumigraph in a viewer which is capable of displaying irregularly sampled light fields at interactive rates along the lines of Sloan et al. [9]. However, our viewer implementation does not exploit the Lumigraph's geometric information (e.g. for performing viewing-time depth correction). Since the images are blended together by using a simple interpolation scheme, the light field appears blurry (cf. Fig. 11). Depth correction as proposed by Gortler et al. would remove most of these artifacts and produce images similar in quality to the warped images in Fig. 10. However, the approximation error of the geometric model used for backward-mapping the viewing rays is a considerable limitation of the proposed depth-correction scheme.

5. Conclusions and Future Work

We have introduced a method for adaptively acquiring images for Lumigraphs and light fields from synthetic scenes. The acquisition is steered by a morphing-based a priori error estimator which conservatively accounts for all shading errors and visibility events that can be derived from the given set of viewpoints. Our method enables the Lumigraph creator to specify the desired number of view points as a constraint and then automatically construct a Lumigraph which is optimal with respect to the achievable reconstruction quality through morphing or depth-corrected blending. In order to further refine the Lumigraph, the adaptive acquisition method can be applied over and over in order to acquire new real samples, or new viewpoints may be morphed from the existing ones by means of the warping-based refinement algorithm presented in [6].

The implemented algorithms have proved to be successful in that they generate adaptive viewpoint meshes which correspond to the color and visibility structure of the scene. The overhead for the adaptive method compared to pure image acquisition is relatively small and can be disregarded for complex scenes which take a long time to render. The error bound converges well in our examples, and the reconstruction quality of the generated Lumigraphs is very high. From subjective observations at viewing time it can be said that the reconstruction quality seems to be evenly distributed in the viewpoint domain.

One limitation of our method is the fact that small highlights and small occluded areas cannot be predicted if they are not visible from any of the current viewpoints. These problematic sections will then never appear in any of the reconstructed views. This problem can only be solved by a posteriori error bounds (which are not affordable in the context of rendering synthetic scenes), or by exploiting more information from the ray tracer like surface and reflectance data. Another small drawback of our error predictor is that the error convergence can be troubled by very small areas of blending errors due to numerical discrepancies of geometric visibility. Fortunately, these errors only dominate when the total error is already very low. A method to avoid this prob-

lem could be some kind of edge detection on the error image combined with a down-weighting of those edges.

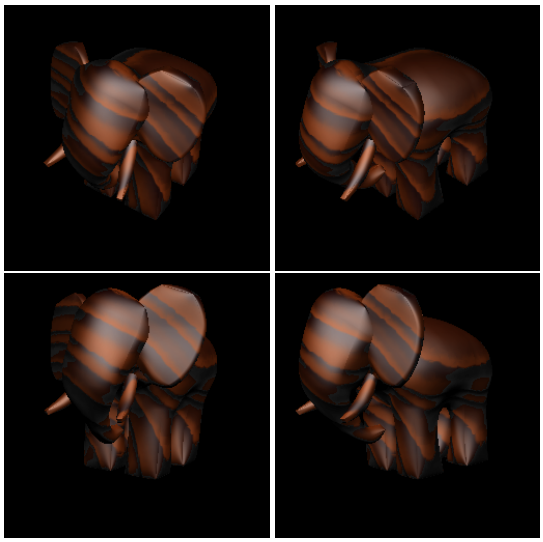
There is much room for future work on adaptive Lumigraphs. The first and most simple extension of our system would be to render only those pixels of a novel view which cannot be reconstructed correctly by morphing. This would save a large amount of rendering time. Second, the results of this work should be transferred to the acquisition of Lumigraphs from video streams. The a priori error estimator can be used to pick an optimal sequence of frames from a large stream of images. The morphing-based predictor can be extended to general 3D warping approaches in order to serve in the context of rendering intermediate frames for animation sequences. And last but not least, one should investigate the issue of compressing adaptively refined Lumigraphs, probably exploiting the information given by the morphing predictor. This way, one should be able to achieve highly competitive compression rates when using adaptive Lumigraphs.

Acknowledgments

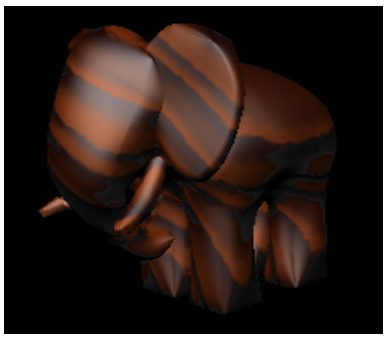
The authors wish to thank Hendrik Kück for teaching our light field viewer how to process adaptive Lumigraphs, and Katja Daubert for help with the Lumigraph test scenes. The code for Delaunay triangulation was taken from Jonathan Richard Shewchuk's `triangle` package (available at `www.cs.cmu.edu/~quake/triangle.html`). Part of the authors' work was performed within the computer graphics group, University of Erlangen, and was funded by the *Deutsche Forschungsgemeinschaft* (DFG) under grant SFB 603/C2 (*Analysis, Coding, and Processing of Light Fields for Acquiring Realistic Model Data*).

References

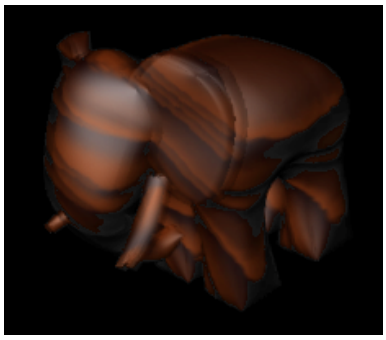
1. E.H. Adelson and J.R. Bergen. *Computational Models of Visual Processing*, chapter 1 (The Plenoptic Function and the Elements of Early Vision). MIT Press, Cambridge, MA, 1991.
2. K. Bala, J. Dorsey, and S. Teller. Bounded-error interactive ray tracing. In *ACM Transactions on Graphics (to appear)*. ACM Press.
3. E. Camahort, A. Lerios, and D. Fussell. Uniformly sampled light fields. In G. Drettakis and N. Max, editors, *Rendering Techniques '98 (Proc. 9th Eurographics Workshop on Rendering)*, pages 117–130. Springer Verlag, 1998.
4. H. Edelsbrunner. *Algorithms in Computational Geometry*. Springer Verlag, 1987.
5. S.J. Gortler, R. Grzeszczuk, R. Szeliski, and M.F. Cohen. The lumigraph. In *Computer Graphics Proceedings, Annual Conference Series (Proc. SIGGRAPH '96)*, pages 43–54, 1996.
6. W. Heidrich, H. Schirmacher, and H.-P. Seidel. A warping-based refinement of lumigraphs. In N.M. Thalman and V. Skala, editors, *Proc. WSCG '99*, 1999.
7. I. Ihm, S. Park, and R.K. Lee. Rendering of spherical light fields. In *Proceedings of Pacific Graphics '97*, 1997.
8. M. Levoy and P. Hanrahan. Light field rendering. In *Computer Graphics Proceedings, Annual Conference Series (Proc. SIGGRAPH '96)*, pages 31–42, 1996.
9. P.-P. Sloan, M.F. Cohen, and S.J. Gortler. Time critical lumigraph rendering. In *Proc. 1997 Symposium on Interactive 3D Graphics*, pages 17–24. ACM, 1997.
10. S. Teller, K. Bala, and J. Dorsey. Conservative radiance interpolants for ray tracing. In X. Pueyo and P. Schröder, editors, *Rendering Techniques '96 (Proc. 7th Eurographics Workshop on Rendering)*. Springer Verlag, 1996.



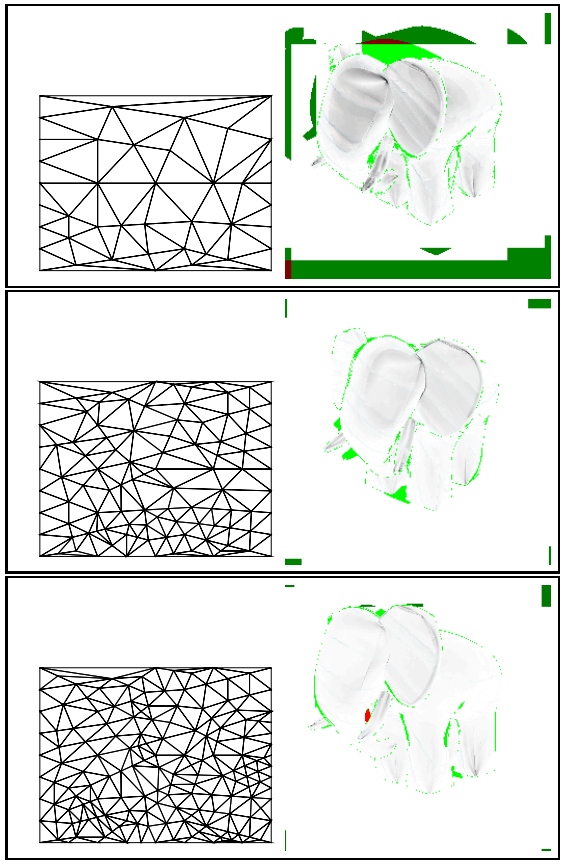
[Schirmacher et al.] Figure 9: *The four extreme views of the "elephant" scene. The differences between the views are the changing highlights on the ear and the hind legs, as well as several self occlusion effects.*



[Schirmacher et al.] Figure 10: *Morphed elephant corresponding to the next viewpoint to be inserted after 160 iterations (cf. last row in Fig. 12). By eye, this image cannot be distinguished from the ray-traced original.*



[Schirmacher et al.] Figure 11: *Novel view reconstructed at interactive frame rates by pure blending from the elephant light field with 160 view points. One can see the blurring artifacts because no depth correction is performed.*



[Schirmacher et al.] Figure 12: *Left column: viewpoint plane meshes for the "elephant" scene after 40, 100, and 160 iterations (top down). Right column: corresponding error images for the next potential viewpoint. Darker colors denote bigger error, holes are red, single maps green. Darker red/green visualizes holes/single maps which do not generate any error because they map to the outside of a reference view (cf. Sec. 3.1).*