Intensity-based 3-D Reconstruction with Non-linear Optimization

Christian Hopfgartner, Ingo Scholz, Martin Gugat, Günter Leugering and Joachim Hornegger
ISO Software Systeme GmbH, Nuremberg, Germany
Giesecke & Devrient GmbH, Munich, Germany
Chair of Applied Mathematics II, University Erlangen-Nuremberg, Erlangen, Germany
Chair of Pattern Recognition, University Erlangen-Nuremberg, Erlangen, Germany
christian.hopfgartner@isogmbh.de, ingo.scholz@gi-de.com,
[gugat,leugering]@am.uni-erlangen.de, joachim.hornegger@informatik.uni-erlangen.de,

Abstract

Image-based rendering methods such as light fields are used in computer graphics for the computation of new views of a scene out of a set of recorded images with known camera positions. The quality of the rendered images depends on the accuracy of information about the structure of the scene, consisting of a geometric 3-D model. In order to accurately reconstruct and optimize such a 3-D model, we define a residual function which represents the difference between an image rendered with the light field from a known viewpoint, and the original image at the same position. In order to get optimal results, we minimize this particular function by defining a non-linear least-squares problem which is solved by an appropriate optimization method. Here, we use a largely unknown, non-monotone variant of the Levenberg-Marquardt method. Thus, starting from a flat 3-D model in our experiments the non-monotone variant decreases the error by a factor of up to **3** in comparison to the monotone variant.

Keywords: *Non-linear optimization, image-based rendering, 3-D reconstruction, light field*

1 Introduction

Modeling of virtual environments has long been the sole domain of computer graphics when explicit geometric modeling was the prevalent technique. With the emergence of so-called *image-based* rendering techniques, reconstructing a virtual environment from images and video sequences of real scenes using computer vision methods became feasible. The light field, which is one of the most common representations of image-based models, stores scene appearance in a set of images, and generates novel views by combining these images based on the known circumstances of recording. These are the pose of the camera during recording and its internal parameters. Computer vision provides techniques such as structure-from-motion and camera self-calibration to obtain this information.

Quality of images rendered from a light field thus depends on one hand on the accuracy of camera parameter estimation. On the other hand, quality is further improved if some information about scene geometry is available, which consists at least of the distance of objects from the camera for each image. This applies the more the further the originally recorded images are apart, corresponding to a sparser sampling of the scene. While reconstructing scene geometry from the input images is possible, it usually lacks dense information as well as accuracy and thus leads to blurring, distortions and superpositions in rendered images. We therefore propose an improved scheme based on errors in output image intensity using non-linear optimization to generate and refine scene geometry.

Finding a way to measure the quality of images rendered from a light field is an essential part of our approach. The light field generates an output image by combining intensity values of the closest neighboring input images. Leaving out one input image, rendering a new image at exactly the same position and computing the difference between the two therefore yields the difference between the real appearance of the scene and an artificial view. The smaller this difference, the better the underlying geometric model. The comparison of the two images is performed pixel by pixel. We define the difference between two pixels as the absolute mean difference between their intensities. The residual vector used in the following optimization step is composed of all these differences.

Scene geometry itself is represented by a global triangle mesh, the so-called *global proxy*. Our objective is to reconstruct this triangle-mesh in such a way that the differences between the rendered and the original image are minimized, which corresponds to a minimization of the result of the residual function. As we deal here with an overdetermined system of equations, there may be no unique solution. This objective is formulated as a non-linear least-squares problem, where the sum of the squares of the residual function result is minimized, in order to optimize scene geometry as a whole. We solve this problem by applying a non-monotone extension of the Levenberg-Marquardt method in order to improve the results. This extension allows for a partial increase of the residual values, often resulting in a more global solution as it does not necessarily converge to the nearest local minimum. To our knowledge, this particular extension has not been introduced in the literature as a solution to similar problems so far.

This article is structured as follows. In Section 2, an overview over the literature regarding light field rendering and reconstruction as well as similar approaches to geometry improvement is given. The computation of residual vector and objective function is presented in Section 3, while in Section 4, the optimization process using the extended Levenberg-Marquardt algorithm is described. Section 5 covers implementation issues such as the exploitation of sparse matrices, and in Section 6, we present experimental results. The contribution is completed by a summary and an outlook to possible improvements.

2 Related Work

The underlying motivation of this contribution is an improvement of rendering quality in the context of image-based modeling. Light fields, as the representative of image-based models used here, were first introduced by Levoy et al. [10] using a two-plane parameterization, thereby restricting camera positions and focal planes to these planes. Gortler et al. [4] added scene depth per input image in order to improve rendering quality. Global scene geometry, however, was introduced some years later by Buehler et al. [1] in the form of a global, geometric proxy as part of the *Unstructured Lumigraph.* It is therefore the rendering method of choice here. A quite comprehensive and still up-to-date summary of model variants and rendering methods can be found there as well, the light field renderer actually used here is described in [17]. An even broader view of image-based rendering is given in [14].

Light field models may be reconstructed from input images of real scenes using structure-from-motion approaches. These are usually based on the availability of point feature correspondences, which can be acquired automatically using feature tracking methods, such as the Kanade-Lucas-Tomasi tracker [15]. Camera poses and other parameters may be estimated from these by means of factorization methods [16] and self-calibration [5]. As a by-product, simple geometry meshes per image can be generated by triangulating the tracked features [8]. Detailed descriptions of the reconstruction method used for the examples in the experiments section are given in [6] and [13].

An early approach to generating depth information from multiple images using image appearance is the space-sweeping approach [2]. Given that the camera positions for a set of images showing the same scene from different view points are known, a depth plane is moved from back to fore through the scene. Projecting all images onto this plane, the correct depth is found when all color values projected to a pixel coincide. Yang et al. [18] apply this technique to light field rendering, generating depth for each rendered image on the fly and in real-time using graphics hardware.

Improving geometry meshes using appearance-based methods has been proposed before likewise. Considering that a 2-D mesh generated from a point cloud has many solutions, Morris and Kanade [12] implement a search algorithm for improving a given triangulation using edge swapping. Consistency with recorded images is verified by applying a texture map onto the 3-D object. The task is formulated as a maximum likelihood problem and solved by a greedy search algorithm.

An approach similar to the one we propose in the following was published by Eckert et al. [3]. A 3-D model of an object is first reconstructed using the shape-from-silhouette reconstruction technique, and the resulting, fine-grained mesh is subsequently refined using an appearance-based method as well. However, the shape silhouette serves here as an additional constraint which is not available in our examples, and only a conventional, monotone gradient descent technique is applied.

3 The objective function

3.1 Scene geometry and input images

The global scene geometry for light field generation is represented by a triangle mesh of V vertices $v_i = (v_i^x, v_i^y, v_i^z)^\top$ $(1 \leq i \leq V)$. These vertices can be shifted in all three dimensions in space. Therefore, the geometry may be determined by $3V$ parameters

$$\mathbf{x} := (v_1^x, v_1^y, v_1^z, \ldots, v_V^x, v_V^y, v_V^z)^\top \tag{1}$$

If, as in our experiments, an 8×8 triangle mesh is used, the total number of parameters is 192.

The color model used for the images is RGB with three channels per pixel. Each image has a resolution of $w \times h$ pixels. In our experiments, we use image resolutions of 256×256 and 512×512 pixels.

3.2 Image rendering

In order to generate a new, virtual image from its neighboring images we use the Unstructured Lumigraph renderer by Buehler et al. [1] as implemented in

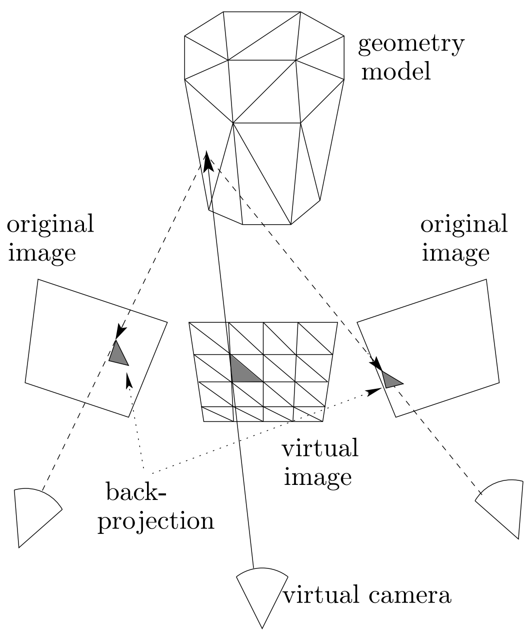


Figure 1: Rendering of virtual views using the Unstructured Lumigraph. The virtual image is subdivided and corresponding texture triangles are selected by intersecting the viewing rays with the geometric model.

the lgf3 library [17]. The data required for rendering are the neighboring images along with their camera parameters, and the triangle mesh which represents the scene geometry.

Image formation is performed as shown in figure 1. The image to be rendered is subdivided by a regular grid of triangles and for each vertex, depth is calculated by intersecting a viewing ray with the geometry mesh. By back-projecting the intersection into the previously selected neighboring images, the corresponding pixel positions are determined from each of them. Thus, for each triangle of the regular grid, a contributing texture triangle is selected from each source image and superimposed according to previously computed weights.

Here, we define the rendering as a mapping

$$\begin{aligned} f : \mathbb{R}^{3V} &\rightarrow \mathbb{R}^{3wh} \\ \mathbf{x} &\mapsto f(\mathbf{x}) =: \mathbf{y} \end{aligned}$$

where $\mathbf{x}$ is the parameter vector defined in equation (1) determining scene geometry. The vector $\mathbf{y}$ resulting from f represents the color values of the pixels of the rendered image in the form

$$\mathbf{y} = (\mathbf{y}_{1,1}, \dots, \mathbf{y}_{1,w}, \dots, \mathbf{y}_{h,1}, \dots, \mathbf{y}_{h,w})^\top, \tag{2}$$

where $\mathbf{y}_{i,j}$ denotes the color values of the pixel in the i-th row and j-th column. It is defined in RGB color space as

$$\mathbf{y}_{i,j} = (y^R_{i,j}, y^G_{i,j}, y^B_{i,j})^\top. \tag{3}$$

The three components denote the red, green and blue color channels of the pixel. Each of the channels may be an integral value between 0 and 255. Thus, each pixel may have one of 256^3 different colors.

3.3 Image comparison

In order to determine the quality of a rendered, virtual image, the virtual camera pose is chosen to be one of the camera poses of the original images. The original image is not used for image formation, i. e., it is removed from the light field for the time being. Thus, if the mapping from neighboring to new image were flawless, original and rendered image would be equal.

For the comparison of rendered and original image we compute the L_1 distance (Manhattan distance) between the intensities of corresponding pixels and divide it by three, averaging over the color channels. This reduces the relative weight of outliers and large differences in a single color channel.

$$\begin{aligned} g : \mathbb{R}^{3wh} &\rightarrow \mathbb{R}^{wh} \\ \mathbf{y} &\mapsto g(\mathbf{y}) \end{aligned}$$

with

$$g(\mathbf{y}) = \begin{pmatrix} g_1(y^R_1, y^G_1, y^B_1) \\ \vdots \\ g_{wh}(y^R_{wh}, y^G_{wh}, y^B_{wh}) \end{pmatrix} \tag{4}$$

and

$$g_i(\mathbf{y}_i) = \frac{1}{3}\Big(|y^R_i - \hat{y}^R_i| + |y^G_i - \hat{y}^G_i| + |y^B_i - \hat{y}^B_i|\Big), \quad \forall i = 1, \dots, wh. \tag{5}$$

The pixel intensities of the original image are stored in $\hat{\mathbf{y}}$ and the rendered ones in $\mathbf{y}$.

The smaller the difference between the channels of $\mathbf{y}_i$ and $\hat{\mathbf{y}}_i$, the smaller the value of $g_i(y_i)$. If the difference of all channels is equal to zero, $g_i(\mathbf{y}_i)$ equals to zero as well.

Note that since the color of each pixel is described by three integral values, the range of g_i consists of 766 possible elements $\{0, \frac{1}{3}, \frac{2}{3}, 1, \dots, 255\}$. This discontinuity has an impact on the computability of the Jacobian matrix of Section 3.6. Its treatment will therefore be described more specifically in 5.

3.4 The residual function

We define the residual function as a composition of image rendering and subsequent image comparison

$$\begin{aligned} R : \mathbb{R}^{3V} &\rightarrow \mathbb{R}^{wh} \\ \mathbf{x} &\mapsto R(\mathbf{x}) := (g \circ f)(\mathbf{x}). \end{aligned} \tag{6}$$

In order to find the optimal global scene geometry $\mathbf{x}^*$, we have to solve the overdetermined system

$$R(\mathbf{x}) = 0 \tag{7}$$

which has fewer degrees of freedom than equations ($3V < wh$) so that an exact solution usually does not exist.

3.5 The objective function

A solution $\mathbf{x}^*$ has to be found for which the value $\|R(\mathbf{x})\|$ becomes minimal. Here as well as for the remainder of this article, $\|\cdot\|$ denotes the Euclidean norm. We define the objective function as

$$\begin{aligned} F : \mathbb{R}^{3V} &\rightarrow \mathbb{R} \\ \mathbf{x} &\mapsto F(\mathbf{x}) := \frac{1}{2}\|R(\mathbf{x})\|^2 \end{aligned} \qquad (8)$$

with the goal to determine the vector $\mathbf{x}^*$ which minimizes F. This leads to a non-linear optimization problem without constraints, i. e., the non-linear least-squares problem

$$\min_{\mathbf{x}\in\mathbb{R}^{3V}} F(\mathbf{x}), \quad F(\mathbf{x}) = \frac{1}{2}\|R(\mathbf{x})\|^2 \quad . \qquad (9)$$

3.6 Derivatives of the objective function

In order to solve the function of equation (9) we need information about the derivatives of first and second order of F. The first derivative is given as

$$F'(\mathbf{x})^\top = J(\mathbf{x})^\top R(\mathbf{x}) \in \mathbb{R}^{3V} \qquad (10)$$

where $J(\mathbf{x})$ is the $wh \times 3V$ Jacobian matrix of $R(\mathbf{x})$. The second derivative is given as

$$F''(\mathbf{x}) = J(\mathbf{x})^\top J(\mathbf{x}) + B(\mathbf{x}) \in \mathbb{R}^{3V\times 3V} \qquad (11)$$

where $B(\mathbf{x})$ contains derivatives of second order of $R(\mathbf{x})$. Computing all of these is very expensive. Since $\|B(\mathbf{x})\| \rightarrow 0$, if $\|R(\mathbf{x})\| \rightarrow 0$, which happens in the vicinity of an optimal solution $\mathbf{x}^*$, we omit $B(\mathbf{x})$ in equation (11).

4 Optimization

For the optimization we use the Levenberg-Marquardt method [9, 11]. This is an iterative method, whose search direction is interpolated between the direction of the gradient method and the Gauss-Newton method. Thus, it combines the positive properties of both methods. First, it ensures the convergence far away from a minimizer point, and second, it exhibits fast local convergence near a minimizer point. In the Levenberg-Marquardt method, both, the search direction and the step length are computed simultaneously. In order to compute one step $\mathbf{d}_k$, we solve the system of equations

$$(\mathbf{J}_k^\top \mathbf{J}_k + \lambda\mathbf{I})\mathbf{d}_k = -\mathbf{J}_k^\top \mathbf{R}_k \qquad (12)$$

based on the second order Taylor approximation of the objective function in $\mathbf{x}_k$. In our case, $\mathbf{J}_k = J(\mathbf{x}_k)$ and $\mathbf{R}_k = R(\mathbf{x}_k)$. From now on, this shorter notation will be used. $\mathbf{J}_k$ denotes the Jacobian matrix of $\mathbf{R}_k$, and $\mathbf{I}$ the identity matrix. The step length and search direction of $\mathbf{d}_k$ depend on the choice of the Levenberg-Marquardt parameter λ. If $\lambda = 0$, the direction $\mathbf{d}_k$ is the same as if we used the Gauss-Newton method. The step length is maximal. If we let $\lambda \rightarrow \infty$, the search direction is the direction of the steepest descent with a very small step length, since $\lim_{\lambda\rightarrow\infty}(\mathbf{J}_k^T\mathbf{J}_k + \lambda\mathbf{I})^{-1} \rightarrow 0$.

In contrast to the original Levenberg-Marquardt algorithm we employ a scheme proposed by Kelley [7] to determine the parameter λ. Its choice depends on the ratio of the actual reduction to the predicted reduction of the objective function. The actual reduction Δ_a is defined as

$$\Delta_a = F_k - F_{k+1} \quad , \qquad (13)$$

where $F_k = F(\mathbf{x}_k)$ from equation (8). The predicted reduction Δ_p, based on the quadratic approximation of F in the k-th iteration point $\mathbf{x}_k$,

$$\begin{aligned} \widetilde{F}(\mathbf{x}_k + \mathbf{d}_k) &= \widetilde{F}_k(\mathbf{d}_k) = \\ &= F_k + \mathbf{d}_k^\top \mathbf{J}_k^\top \mathbf{R}_k + \frac{1}{2}\mathbf{d}_k^\top(\mathbf{J}_k^\top \mathbf{J}_k)\mathbf{d}_k, \end{aligned}$$

is defined as

$$\Delta_p = -\frac{1}{2}\mathbf{d}_k^\top \mathbf{J}_k^\top \mathbf{R}_k + \frac{1}{2}\lambda\|\mathbf{d}_k\|^2 \quad . \qquad (14)$$

The ratio Δ_a/Δ_p specifies the quality of the approximation $\widetilde{F}_k$. The bigger the ratio, the better F is approximated by $\widetilde{F}$ in a neighborhood of $\mathbf{x}_k$.

In order to minimize the objective function of equation (9) given the problem of optimizing scene geometry we considered three different variants of the Levenberg-Marquardt algorithm. The first one is the standard monotone variant (*MLM*) which reduces the residual error F_k in each iteration. In contrast to that, the non-monotone approach Zhang and Chen [19] allows for an increase of the error for a limited number of iterations, and may thus find a better minimizer point. The non-monotone approach is described in two variants *NMLM1* and *NMLM2*, pointing out the differences to the monotone approach.

Algorithm 4.1 (MLM) *Given are $\mu > 0$, $\nu > 1$, $\lambda > 0$, $F(\mathbf{x}_0)$ and an initial state vector $\mathbf{x}_0$. Set $k = 0$.*

1. *Compute $\mathbf{R}_k$, $\mathbf{J}_k$ and $F'_k = \mathbf{J}_k^\top\mathbf{R}_k$. If the stop criterion is fulfilled, STOP!*

2. *Solve $(\mathbf{J}_k^\top\mathbf{J}_k + \lambda\mathbf{I})\mathbf{d}_k = -\mathbf{J}_k^\top\mathbf{R}_k$ to obtain $\mathbf{d}_k$.*

3. *Compute $F_{k+1} = F(\mathbf{x}_k + \mathbf{d}_k)$.*

4. *Compute $\Delta_a = F_k - F_{k+1}$, $\Delta_p = (\lambda\|\mathbf{d}_k\|^2 - F'_k\mathbf{d}_k)/2$.*

5. *If $\Delta_a/\Delta_p < \mu$, set $\lambda = \lambda \cdot \nu$, GOTO 2.*

6. *If $\mu \leq \Delta_a/\Delta_p$, set $\lambda = \lambda/\nu$ and $k = k+1$, GOTO 1.*

In Section 5.3 we provide more information on the stop criterion used for this algorithm.

We now relax the criterion for accepting $\mathbf{d}_k$. The actual reduction is defined as

$$\Delta_a = F_{\max} - F_{k+1} \tag{15}$$

where $F_{\max}$ is defined by

$$F_{\max} = \max\left(F_k, F_{k-1}, \cdots, F_{k-M}\right) \tag{16}$$

and F_i does not exist for negative i. Let $M \geq 0$ be given, a parameter which defines the maximum number of consecutive iterations for which an increase of the objective function value is allowed. The following partial algorithms only describe the differences to algorithm *MLM*.

Algorithm 4.2 (NMLM1) *Additionally to MLM, $\eta > 0$ and $M \geq 0$ are given. Set $k = 0$.*

4. *Compute $\Delta_a = F_{\max} - F_{k+1}$, $\Delta_p = (\lambda\|\mathbf{d}_k\|^2 + F_k'\mathbf{d}_k)/2$.*

5. *Compute* $\widetilde{\mu}_k = \begin{cases} \mu, \text{if } M = 0 \\ \min\left(\mu, \eta\frac{\|F_k'\|^2\,\|\mathbf{d}_k\|^2}{\Delta_p}\right), \text{if } M > 0 \end{cases}$

6. *If $\Delta_a/\Delta_p < \widetilde{\mu}_k$, set $\lambda = \lambda \cdot \nu$, GOTO 2.*

7. *If $\widetilde{\mu}_k \leq \Delta_a/\Delta_p$, set $\lambda = \lambda/\nu$ and $k = k+1$, GOTO 1.*

For a further relaxation, Δ_p is set to

$$\Delta_p = -\frac{1}{2}\mathbf{d}_k^\top \mathbf{J}_k^\top \mathbf{R}_k \tag{17}$$

which is based on the quadratic Levenberg-Marquardt model of F in $\mathbf{x}_k$,

$$\widehat{F}(\mathbf{x}_k + \mathbf{d}_k) = F_k + \mathbf{d}_k^\top \mathbf{J}_k^\top \mathbf{R}_k + \frac{1}{2}\mathbf{d}_k^\top(\mathbf{J}_k^\top \mathbf{J}_k + \lambda \mathbf{I})\mathbf{d}_k.$$

This leads to the following second non-linear algorithm.

Algorithm 4.3 (NMLM2) *In addition to NMLM1, $\lambda_{\min} > 0$ is given. Set $k = 0$.*

4. *Compute $\Delta_a = F_{\max} - F_{k+1}$, $\Delta_p = -F_k'\mathbf{d}_k/2$.*

5. *Compute $\widehat{\mu}_k$ with* $\widehat{\mu}_k = \begin{cases} \mu, \text{ if } M = 0 \\ \min\{\mu, \eta\frac{\|F_k'\|^2\,\|\mathbf{d}_k\|^2}{\Delta_p}\|\mathbf{J}_k^\top \mathbf{J}_k + \lambda\mathbf{I}\|_\infty\}, \\ \quad \text{if } M > 0 \end{cases}$

6. *If $\Delta_a/\Delta_p < \widehat{\mu}_k$, set $\lambda = \lambda \cdot \nu$, GOTO 2.*

7. *If $\widehat{\mu}_k \leq \Delta_a/\Delta_p$, set $\lambda = \max(\lambda/\nu, \lambda_{\min})$ and $k = k+1$, GOTO 1.*

5 Implementation Issues

5.1 Computing the Jacobian matrix

We compute the Jacobian matrix $\mathbf{J}$ of the residual function numerically since we have no information about analytical derivatives. For this, we favor the use of central differences over the use of forward differences. Although the computation of central differences is twice as expensive, we get a smaller truncation error so that the iteration points will be computed more exactly.

The adaption of the step length is important during the computation of the Jacobian matrix. If the changes in vector $\mathbf{x}$ are sufficiently small, they do not result in changes in the rendered image, because an image consists of finite color values. Expressed more mathematically, f maps to discrete values.

Thus, if we compute all columns $\mathbf{J}_j$, $1 \leq j \leq 3V$, of $\mathbf{J}$ with a small change δ, determined by machine precision, such that $f(\mathbf{x} + \delta\mathbf{e}_j) = f(\mathbf{x}) = f(\mathbf{x} - \delta\mathbf{e}_j)$, where $\mathbf{e}_j$ denotes the j-th unit vector, this results in

$$R(\mathbf{x} + \delta\mathbf{e}_j) = R(\mathbf{x} - \delta\mathbf{e}_j) \tag{18}$$

and therefore, $\mathbf{J}$ contains only zeros.

The optimal step length $\delta = 0.01$, which is used in our experiments, was determined experimentally.

5.2 Efficient storage of the Jacobian matrix

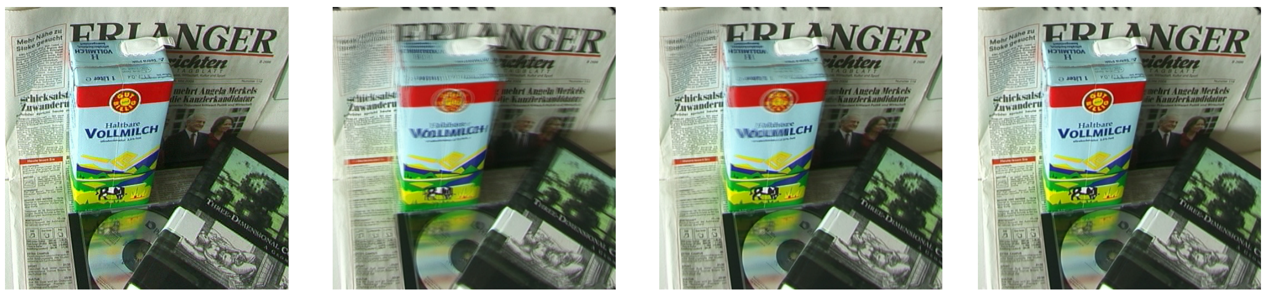
In our experiments, the percentage of elements of $\mathbf{J}$ equal to zero was about 99%. Therefore, we store it as a sparse matrix. This has two advantages.

To begin with, the matrix-matrix multiplication is distinctly accelerated. In each iteration we multiply the transposed Jacobian matrix of $\mathbf{R}$ with the Jacobian matrix itself. In our experiments, the Jacobian matrix has $262\,144 \times 192$ entries. Stored as a full matrix, the time to multiply the two matrices is about 300 seconds, which is not an acceptable time, considering that we compute up to 100 iterations. Stored as a column oriented sparse matrix, one multiplication takes only about 0.5 seconds.

Moreover, the amount of memory required is reduced considerably. Each entry of the Jacobian matrix is represented by a `double` value of 8 bytes. Stored as a full matrix, it requires about 384MB, but stored as a column oriented sparse matrix it requires only 8MB. Note that more refining of the mesh or a higher resolution of the images yields a Jacobian matrix, whose size quickly exceeds the main memory available, if we were to save it as a full matrix.

5.3 Choice of the stop criterion

The most widely used stop criterion $\|F'(\mathbf{x}_k)\| < \epsilon$, where $\epsilon > 0$ is a small number, cannot be used for our



(a) original image

(b) rendered image before optimization

(c) rendered image after optimization with the monotone algorithm

(d) rendered image after optimization with the non-monotone algorithm

Figure 2: Example sequence *Milk* and optimization using the two different optimization algorithms. Rendered images are generated from five neighboring images.

problem. The reason is the finite difference approximation of $\|F'(\mathbf{x})\|$, which can only obtain a finite number of values.

We let the algorithm stop if the Levenberg-Marquardt parameter λ exceeds the threshold $\lambda_{\max} = 10^{14}$. At this point, the computed length of $\mathbf{d}_k$ is so small that there is no difference between F_{k+1} and F_k because of the discrete state of the residual function.

6 Experimental Results

In the following we demonstrate the reconstruction of scene geometry on three example light fields. For the preceding camera parameter estimation to work well the surfaces of objects in the scenes are mostly lambertian. However, minor parts of the scene, such as the CD case in example 1, may be reflective. The lighting conditions are mostly constant in all light fields. The experiments are conducted on an Intel Pentium 4 processor with 3GHz, 2GB main memory, and an nVidia GeForce MX440SE graphics card.

For the optimization algorithm we choose the following set of parameters:

$$\mu = 0.55,\ \nu = 2.0,\ \eta = 10^{-3},\ \lambda_{\min} = 1.0,\ \lambda_{\max} = 10^{14}$$

For the numerical computation of the Jacobian matrix we set $\delta = 0.01$. As initial value we set the Levenberg-Marquardt parameter $\lambda = 1.0$. We represent the scene geometry with a regular 8×8 triangle mesh. As initial data for the optimization we choose a flat mesh parallel to the virtual image plane.

6.1 Example 1: Milk

The first example sequence, named *Milk*, originally contains 190 images recorded with a hand-held digital video (DV) camera. Only one of the images, shown in figure 2(a), is used to optimize a global proxy, but each of the two following experiments is performed with a different number of neighboring images. All images have a resolution of 512×512 pixels.

Experiment	**Algorithm**	**Error**	
		initial	**final**
Milk 1	*MLM*	$17.4 \cdot 10^7$	$10.9 \cdot 10^7$
	NMLM1	$17.4 \cdot 10^7$	$3.81 \cdot 10^7$
	NMLM2	$17.4 \cdot 10^7$	$3.47 \cdot 10^7$
Milk 2	*MLM*	$21.1 \cdot 10^7$	$11.9 \cdot 10^7$
	NMLM1	$21.1 \cdot 10^7$	$5.75 \cdot 10^7$
	NMLM2	$21.1 \cdot 10^7$	$5.58 \cdot 10^7$
Perrier	*MLM*	$9.93 \cdot 10^7$	$1.26 \cdot 10^7$
	NMLM1	$9.93 \cdot 10^7$	$1.25 \cdot 10^7$
	NMLM2	$9.93 \cdot 10^7$	$1.27 \cdot 10^7$
Santa	*MLM*	$1.97 \cdot 10^7$	$0.273 \cdot 10^7$
	NMLM1	$1.97 \cdot 10^7$	$0.246 \cdot 10^7$
	NMLM2	$1.97 \cdot 10^7$	$0.262 \cdot 10^7$

Table 1: Objective function values of all experiments before and after optimization given as the squared sum of mean differences per color channel.

As outlined in Section 3, a new image is generated in the same position and with the same camera parameters as the original image 2(a). In the first experiment, 5 source images are used. Figure 2(b) shows the rendered image before optimization of the scene geometry. The blurring and superpositions can be seen clearly, which arise due to the wrong scene geometry. Figure 2(c) was generated using the optimized scene geometry after applying the monotone optimization algorithm. In order to generate image 2(d) the scene geometry which results from optimization with the non-monotone algorithm *NMLM2* with $M = 4$ was utilized.

It can be seen that after applying the monotone algorithm blurring is reduced significantly, although some superpositions remain, like, e. g., the text on the milk carton. Even better results are achieved after applying the non-monotone algorithm. The obvious blurring is removed entirely in the optimized image. This impression is confirmed by the numbers of table 1. While the monotone variant only achieves a moderate reduction, the error is reduced consider-

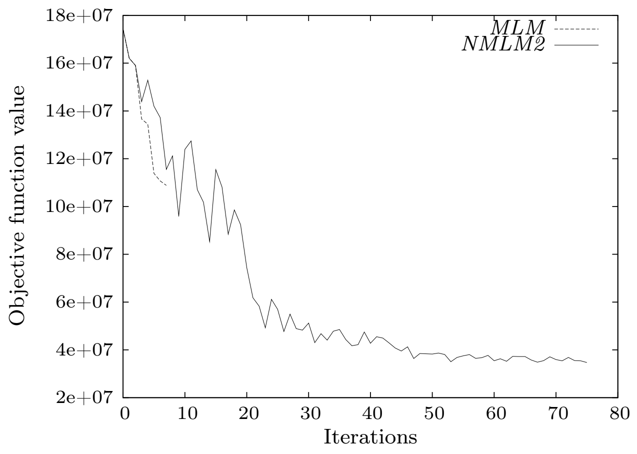


Figure 3: Progression of the objective function values while applying algorithms *MLM* and *NMLM2* to the *Milk1* image sequence. Allowing for a partial increase of the values results in a more global solution.

ably by the non-monotone algorithms. Figure 3 shows the progression of the objective function values during the optimization, comparing algorithms *MLM* and *NMLM2*. As can be seen, the monotone algorithm runs into the nearest local minimum, whereas the non-monotone algorithm has the ability to reach a more global minimum because of allowing for an increase of the objective function values during the optimization.

As a second experiment on this data, the results of figure 4 were generated from the same camera position as the images in figure 2, but this time, 10 source images were used. In figure 4(a) this results in even more superpositions and blurring than in figure 2(b) due to the wrong scene geometry. Again, the non-monotone Levenberg-Marquardt algorithm NMLM2 yields very good results, this time with $M = 8$. Images 4(c) and 4(d) show the difference between the original image 2(a) and the rendered images 4(a) and 4(b), respectively.

However, the figures in table 1 show that an increased number of sample images does not reduce the final error. There are two possible causes for this effect. Firstly, the greater the difference of the viewing angle of the contributing images, the greater the impact of wrong scene depth. The low resolution, 8×8 vertices mesh constitutes the main limiting factor here. Secondly, the limited accuracy of the estimated camera parameters additionally increases the residual error the farther the views are apart. Further experiments with even larger numbers of sample images and on different light fields confirmed these observations.

Figure 5 shows the optimized scene geometry for a side view of the scene. Image 5(a) depicts the geometry as used by the light-field renderer for rendering this single view, along with the corresponding texture mapped onto it. Image 5(b) constitutes the triangle mesh of the complete proxy after optimization.

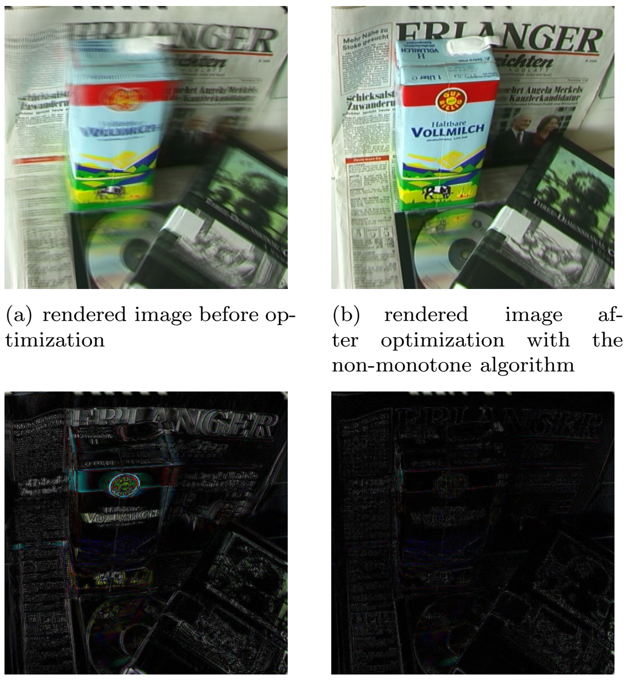

(a) rendered image before optimization
(b) rendered image after optimization with the non-monotone algorithm
(c) difference image before optimization
(d) difference image after optimization with the non-monotone algorithm

Figure 4: Geometry optimization of the *Milk* sequence rendered with 10 contributing, neighboring images. Rendered images and difference images before and after optimization are shown.

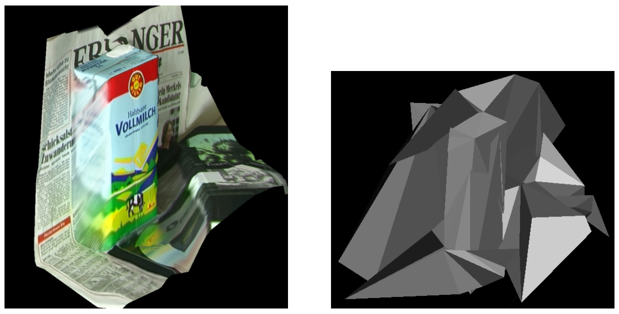

(a) textured geometry
(b) optimized proxy

Figure 5: Optimized geometry after applying the non-monotone Levenberg-Marquardt algorithm to the *Milk* scene.

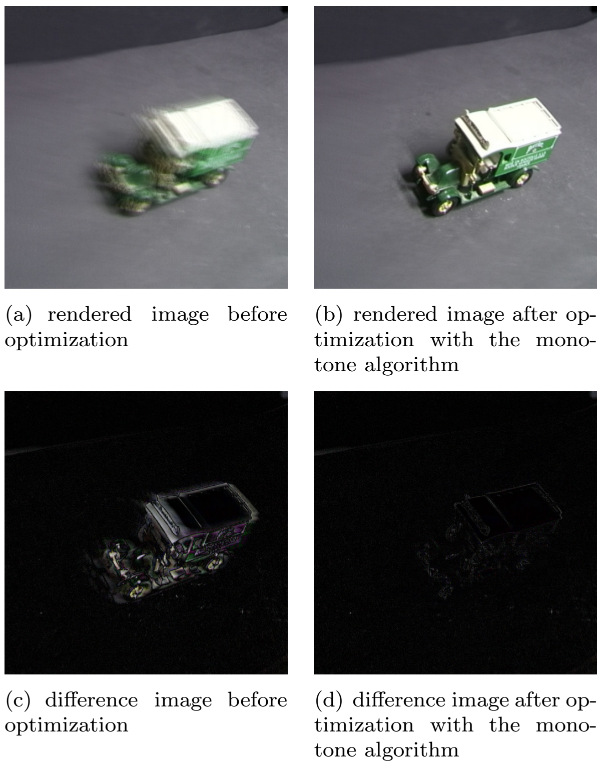

(a) rendered image before optimization
(b) rendered image after optimization with the monotone algorithm
(c) difference image before optimization
(d) difference image after optimization with the monotone algorithm

Figure 6: Geometry optimization of the *Perrier* sequence rendered with 10 contributing, neighboring images. Rendered images and difference images before and after optimization are shown.

6.2 Example 2: Perrier

Similarly to the second *Milk* sequence example, we use 10 source images to generate the results for the *Perrier* sequence in figure 6. Again, the first two images show the rendered images before and after optimization, respectively, both with a resolution of 512×512 pixels. Likewise, the bottom row images of figure 6 show the difference between the original image and the rendered images.

In this example light field, there were hardly any differences between the results of the non-monotone and the monotone algorithm. As can be seen in table 1, both of them yield good results, with the non-monotone algorithm yielding slightly smaller objective function values.

In figure 7 the optimized scene geometry is presented in the same way as in figure 5. The left hand image shows a textured per-image depth map as generated by the Unstructured Lumigraph, while the complete proxy is given in the right hand image. In this case, the object is not as easily recognizable in the geometry mesh as in the former example. This is due to a lack of further constraints, such as the consistency with more than one input image. A number of possible improvements of the approach will be discussed in Section 7.

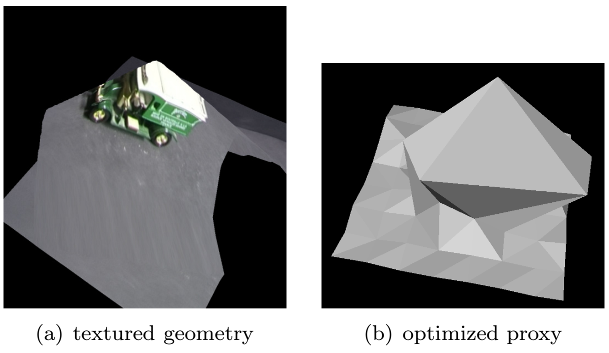

(a) textured geometry
(b) optimized proxy

Figure 7: Optimized geometry after applying the monotone Levenberg-Marquardt algorithm to the *Perrier* image sequence.

6.3 Example 3: Santa

The third example sequence consists of only 8 images of a small Santa Claus figure which were recorded with a turn-table setup. Therefore, we used all 7 remaining source images to generate the geometric proxy base on one of the input images. All images have a smaller resolution of 256×256 pixels. The example images of figure 8 again demonstrate the improvement of rendering quality using our optimization scheme. Both the rendered and the difference images illustrate the reduction of superposition artifacts.

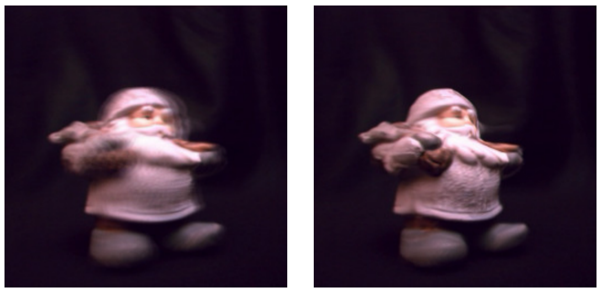
Here, the differences between the results of the non-monotone and the monotone algorithms are again more distinct compared to the *Perrier* example light field. But even though the objective function's value after optimization with the non-monotone algorithm is about 10 percent smaller than that of the monotone algorithm, the visible difference between the two is minor.

Finally, figure 9 again gives an impression of the geometry resulting from the optimization process.

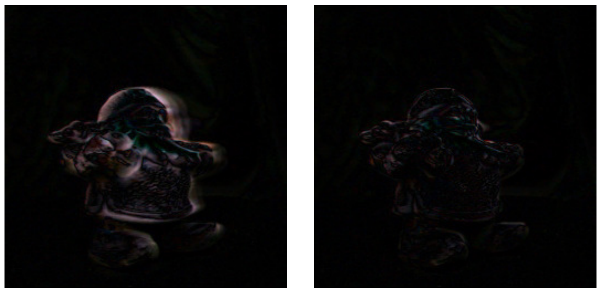
7 Summary and Future Work

In this contribution we have shown an approach to image-based 3-D geometry reconstruction based on the light field model. Starting with a flat 3-D mesh we improve the geometry of a scene by comparing rendered and original images and defining an objective function on the difference image. The geometry mesh is modified in order to minimize the difference using a non-monotone variant of the Levenberg-Marquardt algorithm. To our knowledge, this variant has not been applied to any similar problems so far.

In order to demonstrate the success of our method, we applied it to three light fields generated from different image sequences using structure-from-motion. In all three cases, the appearance of rendered images was improved considerably. In two out of three cases, the non-monotone variant was superior to the conventional, monotone Levenberg-Marquardt algorithm, as

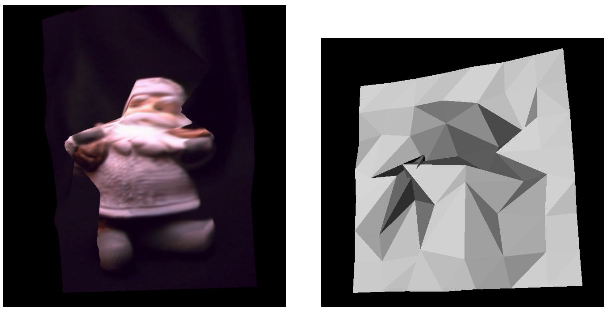


(a) rendered image before optimization (b) rendered image after optimization with the monotone algorithm



(c) difference image before optimization (d) difference image after optimization with the monotone algorithm

Figure 8: Geometry optimization of the *Santa* sequence rendered with 7 contributing, neighboring images. Rendered images and difference images before and after optimization are shown.



(a) textured geometry (b) optimized proxy

Figure 9: Optimized geometry after applying the monotone Levenberg-Marquardt algorithm to the *Santa* image sequence.

it does not converge to the nearest local minimum, but is able to reach a more global minimum.

So far, we did not impose any further constraints on the optimization but applied it to the geometry mesh as a whole. Mesh inconsistencies such as overlapping faces are not prevented and occluded faces are not excluded from optimization. These effects occur more frequently if the number of vertices in the geometry model is increased. The 8×8 grid selected for the experiments proved to be a good compromise. In the literature, appropriate constraints have been successfully applied and therefore, we expect further improvements from similar measures. Significant performance gains can be achieved by exclusively rendering image patches which are affected during numeric computation of the Jacobian matrix.

Acknowledgements

This work was supported in part by the German Research Foundation (DFG) under grant SFB 603/TP C2. Only the authors are responsible for the content.

References

[1] Chris Buehler, Michael Bosse, Leonard McMillan, Steven J. Gortler, and Michael F. Cohen. Unstructured lumigraph rendering. In *Proceedings of SIGGRAPH 2001*, pages 425–432, Los Angeles, CA, USA, August 2001. ACM Press, New York.

[2] Robert T. Collins. A space-sweep approach to true multi-image matching. In *Proceedings of the 1996 Conference on Computer Vision and Pattern Recognition (CVPR '96)*, pages 358–363. IEEE Computer Society Press, Washington, DC, USA, June 1996.

[3] Gerald Eckert, Jochen Wingbermühle, and Wolfgang Niem. Mesh based shape refinement for reconstructing 3d-objects from multiple images. In *Proceedings of the 1st European Conference on Visual Media Production*. The IEE, London, UK, March 2004.

[4] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The lumigraph. In *Proceedings of SIGGRAPH '96*, pages 43–54, New Orleans, LA, USA, August 1996. ACM Press, New York.

[5] Richard I. Hartley. Euclidean reconstruction from uncalibrated views. In *Proceedings of the Second Joint European - US Workshop on Applications of Invariance in Computer Vision*, pages 237–256. Springer-Verlag, London, 1993.

[6] Benno Heigl. *Plenoptic Scene Modeling from Uncalibrated Image Sequences.* ibidem-Verlag Stuttgart, January 2004.

[7] C. T. Kelley. *Iterative Methods for Optimization.* Frontiers in Applied Mathematics. Siam, Philadelphia, 1999.

[8] Reinhard Koch, Benno Heigl, and Marc Pollefeys. Image-based rendering from uncalibrated lightfields with scaleable geometry. In *10th International Workshop on Theoretical Foundations of Computer Vision*, pages 51–66, Dagstuhl Castle, Germany, March 2000. Springer-Verlag, Berlin.

[9] Kenneth Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics*, 2:164–168, 1944.

[10] Marc Levoy and Pat Hanrahan. Light field rendering. In *Proceedings of SIGGRAPH '96*, pages 31–42, New Orleans, LA, USA, August 1996. ACM Press, New York.

[11] Donald W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, 11(2):431–441, 1963.

[12] Daniel D. Morris and Takeo Kanade. Image-consistent surface triangulation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 332–338, Hilton Head, SC, USA, June 2000. IEEE Computer Society Press, Washington.

[13] Ingo Scholz. *Reconstruction and Modeling of Static and Dynamic Light Fields*. Logos Verlag, Berlin, August 2008.

[14] Heung-Yeung Shum, Shing-Chow Chan, and Sing Bing Kang. *Image-Based Rendering*. Springer-Verlag, Berlin, 2006.

[15] Carlo Tomasi and Takeo Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University, April 1991.

[16] Carlo Tomasi and Takeo Kanade. Shape and motion from image streams under orthography: A factorization method. *International Journal of Computer Vision*, 9(2):137–154, November 1992.

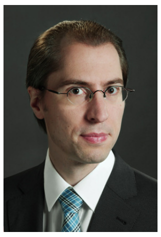
[17] Christian Vogelgsang. *The lgf3 Project: A Versatile Implementation Framework for Image-Based Modeling and Rendering*, volume 38 of *Arbeitsberichte des Instituts für Informatik*. Universität Erlangen-Nürnberg, Institut für Informatik, Erlangen, May 2005.

[18] Jason C. Yang, Matthew Everett, Chris Buehler, and Leonard McMillan. A real-time distributed light field camera. In *Proceedings of the 13th Eurographics workshop on Rendering*, pages 77–86, Pisa, Italy, 2002. Eurographics Association, Aire-la-Ville, Switzerland.

[19] J. Z. Zhang and L. H. Chen. Nonmonotone Levenberg-Marquardt algorithms and their convergence analysis. *Journal of Optimization Theory and Applications*, 92(2):393–418, 1997.

Biographies

Christian Hopfgartner studied mathematics at the University Erlangen-Nuremberg, Germany, between 2000 and 2007, with an emphasis on applied mathematics, nonlinear optimization and image processing. He graduated with the degree 'Diplom-Mathematiker'. Since 2007 he is working as a software developer at ISO Software Systeme GmbH, Nuremberg, Germany. His current responsibilities focus on predevelopment of postprocessing algorithms for CT-images.



Ingo Scholz studied computer science at the University Erlangen-Nuremberg, Germany, between 1994 and 2000, with an emphasis on pattern recognition and image processing. He graduated with the degree 'Diplom-Informatiker'. In 2001 he joined the research staff at the Institute for Pattern Recognition of the University Erlangen-Nuremberg, focusing his research on the reconstruction of light field models, camera calibration techniques and structure from motion. He received his doctorate degree in 2008. Since 2007 he is working for Giesecke and Devrient GmbH in Munich, Germany, as a software developer and project manager in automated banknote processing. He is a member of the german 'Gesellschaft für Informatik' (GI).

Martin Gugat has studied mathematics at the RWTH Aachen, Germany. He graduated in 1990 with the degree 'Diplom-Mathematiker'. Between 1990 and 2000 he worked at the Department of mathematics at the university of Trier, where he received his doctoral degree summa cum laude in 1994 and finished his habilitation in 1998. Between 2000 and 2003 he worked at the department of mathematics at the Technische Universität Darmstadt. Since 2003 he works at the department of mathematics of the Friedrich-Alexander University of Erlangen-Nuremberg. His research focuses on optimization. He is a member of the 'Society for Industrial and Applied Mathematics' (SIAM).

Günter Leugering obtained the diploma in mathematics from the University of Frankfurt (Main), Germany and his doctoral degree from the Technische Universität Darmstadt, Germany, in 1984. In 1989 he received a Heisenberg Professorship and was offered a tenure-track position at Georgetown University in Washington DC, USA. In 1992 he accepted an offer as an Associate Professor at the University of Bayreuth, Germany, and in 2000 he became full Professor at the Technische Universität Darmstadt. Since 2003 he is full professor at the University of Erlangen-Nuremberg, Germany. His main research area is optimization with PDEs, optimal control and homogenization.

Joachim Hornegger graduated in computer science and received his Ph.D. degree in Applied Computer Science (1996) at the University of Erlangen-Nuremberg (Germany). His Ph.D. thesis was on statistical learning, recognition and pose estimation of 3D objects. Joachim was a visiting scholar and lecturer at Stanford University (Stanford, CA, USA) in the academic year 1997/98. In 1998 he joined Siemens Medical Solutions Inc. where he was working on 3D angiography. In November 2001 Joachim was promoted to the director of medical image processing, and in March 2003 director of imaging systems. In parallel to his responsibilities in industry he was a lecturer at the Universities of Erlangen-Nuremberg (1998-1999), EichstÃďtt-Ingolstadt (2000), and Mannheim (2000-2003). Joachim is the author and coauthor of more than 150 scientific publications including a monography on applied pattern recognition and a book on statistical object recognition. Besides his education in computer science, in 2003 Joachim also achieved a diploma in Advanced Management (Cross Functional and General Management, Entrepreneurship, Accounting and Controlling) from the Fuqua School of Business (Duke University, NC, USA) and Siemens. In October 2003 Joachim became professor of Medical Image Processing at the University of Erlangen-Nuremberg and since October 2005 he is a chaired professor heading the Institute of Pattern Recognition. Joachim Hornegger is also professor of the Medical Faculty of the University of Erlangen-Nuremberg. His main research topics are currently medical image processing, medical vision, and pattern recognition. He is a member of IEEE Computer Society and GI.