# Reconstruction and Modeling of Static and Dynamic Light Fields

Der Technischen Fakultät der
Universität Erlangen–Nürnberg
zur Erlangung des Grades

## DOKTOR–INGENIEUR

vorgelegt von

Ingo Scholz

Erlangen — 2008

Als Dissertation genehmigt von der
Technischen Fakultät der
Universität Erlangen-Nürnberg

| | |
|---|---|
| Tag der Einreichung: | 26.11.2007 |
| Tag der Promotion: | |
| Dekan: | Prof. Dr. J. Huber |
| Berichterstatter: | Prof. em. Dr. H. Niemann |
| | Prof. Dr. G. Greiner |

*In memory of Dieter Scholz*

# Acknowledgements

Writing a thesis such as this one is a major venture in one's life, and most people, including me, would neither start nor be able to finish it without the encouragement, support, and contribution of many others. Since I would probably fail to consider all of them if I tried, I would nevertheless like to express my gratitude here to all involved, and name those personally who were most important.

Foremost, I would like to thank my advisor Prof. em. Dr. Heinrich Niemann for his trust in offering me a doctorate position as part of Sonderforschungsbereich (SFB) 603, funded by the German Research Foundation (DFG), and his fair comment and advice over the past years. Likewise, my thanks go to Prof. Dr. Joachim Denzler who first invited me to stay at the Chair for Pattern Recognition after graduation and his continuous support during my first years there, to Prof. Dr. Joachim Hornegger who kindly offered me asylum in the year after I had already left the institute, and to Prof. Dr. Günther Greiner for my warm and fruitful inclusion at the computer graphics department and for reviewing my work.

I had the great luck of working in an environment which was both pleasant and professionally stimulating, thanks to my colleagues at the Chair for Pattern Recognition and throughout the SFB. My long-time office mates Dr. Jochen Schmidt and Dr. Florian Vogt were always there to discuss my problems with me, offer advice or explanation, or just share a good jest, a thing which is rare to find and which I greatly appreciate. The same is true for Dr. Marcin Grzegorzek sharing an office with me later on, who always offered a helping hand whenever necessary. Just as important to me was the cooperation and support of many other colleagues, Stefan Wenhardt, Benjamin Deutsch, Christoph Gräßl, and Timo Zinßer, just to name a few, as well as the technical support provided by Fritz Popp and Walter Fentze.

From my predecessor in project C2 of SFB 603, Dr. Benno Heigl, I received an invaluable head start for my research and he thus made most of my results possible in the first place. My copartner in C2 at the Chair for Computer Graphics, Dr. Christian Vogelgsang, shares an equally large portion of the success of my work by providing the implementation of all rendering technology which I so relied on. The SFB offered a very fruitful environment for cooperation and interdisciplinary work, thanks to people like Marco Winter, Martin Meister, Ulrich Fecker, and many others I worked with who filled this idea with life. And finally, I would not want to forget my students whose work helped considerably to give my thesis its final shape, especially Mario Fritz, Christian Hopfgartner, Roger Müller, and Thomas Sünkel.

Especially over the last year after I had left the institute, my family, friends, and most important my girlfriend Isabella, were essential to me to keep me going on. I would not only want to express my gratitude to them for their support and encouragement, but also to apologize for neglecting them for such a long time.

Ingo Scholz

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Modeling of virtual environments has found its way into many applications, not only in entertainment, but also in engineering, medicine, economics, and education. In entertainment, these virtual environments are most commonly used in movie productions and video games, but while here the acceptance of non-realistic or artificial scenes is inherent, medical applications in particular require a very high amount of realism. Modern techniques in geometric modeling allow for a very realistic reconstruction of difficult features, but they also require some artistic abilities to obtain these results.

An alternative to traditional geometric modeling was introduced with the concept of image-based rendering (IBR), its most prevalent representations being the *lumigraph* [Gor96] and the *light field* [Lev96]. By using images of a real environment they allow to create models of real scenes and objects and to visualize them in photo-realistic quality. Nevertheless, reconstruction of such an image-based model from collections of images or image streams is a challenging task for static scenes, but even more so for dynamically changing environments. Approaches to its robust solution are the topic of this work.

## 1.1 Light Fields from Image Sequences

Simply speaking, an image-based model such as a light field requires only a set of images and the knowledge of the camera's *pose* (i. e., position and orientation) and internal parameters during recording of each image as input data. From this data, it is possible to generate an image of the recorded area, be it a single object or a large scene, from any viewpoint which is reasonably close to the original camera positions. As an example, imagine an Internet vendor who wishes to present a 3-D model of his merchandise in his online shop. Using a light field, he only needs to

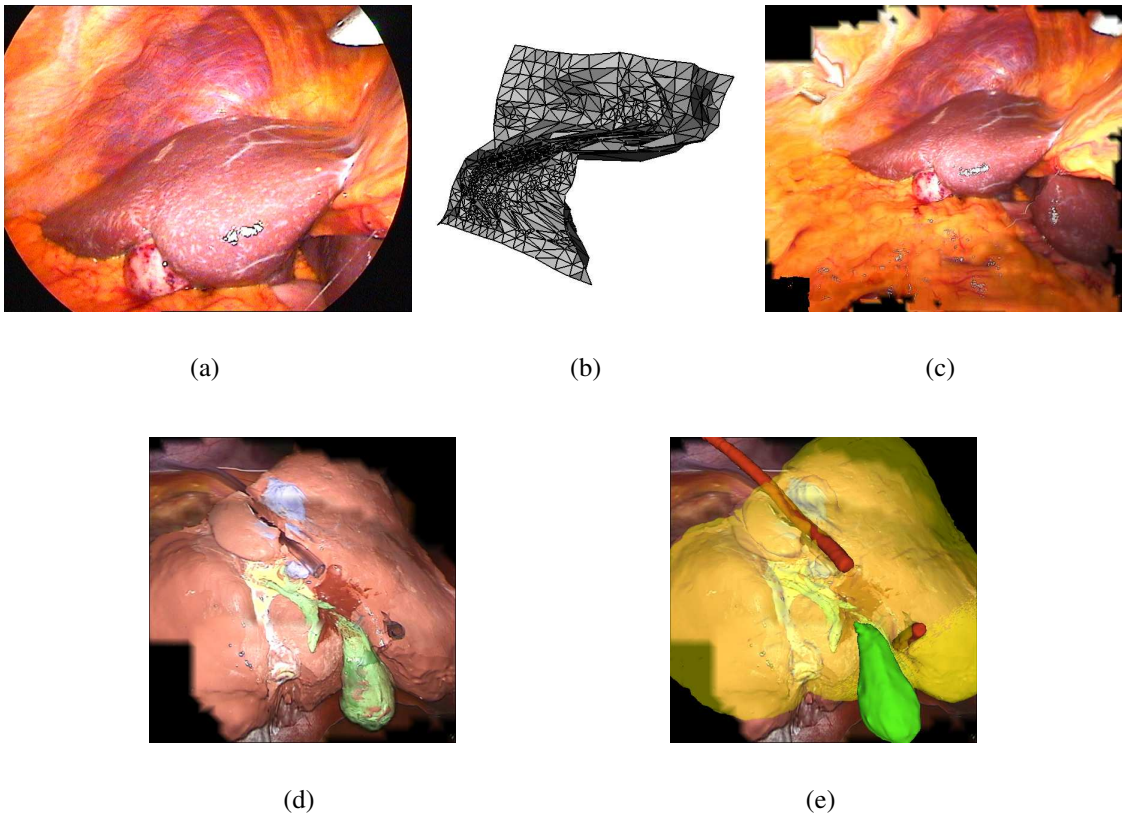


Figure 1.1: Light fields in endoscopic surgery: (a) view of the surgeon through an endoscope, (b) surface reconstruction and (c) image rendered from the resulting light field showing the scene from a greater distance. (d) Light field of a liver/gall bladder model with (e) a superimposed CT scan. Images by courtesy of F. Vogt [Vog06].

take a few dozen images of a product and thus enables his customers to inspect the product from any viewpoint they desire.

Another, more scientific example for the use of light fields is endoscopic surgery [Vog06]. Here, the images of an endoscope are combined to give the surgeon a three-dimensional impression of the operation site where he can navigate in without any restriction due to cumbersome medical instruments. This is illustrated by Figure 1.1 showing an example from a laparoscopic removal of the gall bladder (cholecystectomy). Through the endoscope only a close-up view with limited viewing angle is available to the surgeon, such as in Figure 1.1(a). By reconstructing the geometry of the operation area from several endoscopic images, as depicted in Figure 1.1(b), the scene may be visualized from an overview perspective as in Figure 1.1(c) by combining the texture information of several images. Additionally, if 3-D reconstruction and image-based modeling were successful, different imaging modalities, such as computed tomography (CT),

(a) (b) (c)

Figure 1.2: Three different ways of image acquisition for light fields: (a) Using a robot arm, which also yields the camera pose, (b) using an optical tracking system which tracks the camera position by following optical markers or (c) using a hand-held camera and structure-from-motion algorithms for pose estimation. Images (a) and (b) by courtesy of F. Vogt [Vog06].

magnetic resonance imaging (MRI), or ultrasonography, may be superimposed and offer information to the surgeon which is otherwise invisible. In Figure 1.1(e), this was done for the light field of a model of liver and gall bladder, shown in Figure 1.1(d), combined with a CT scan of it.

In both examples the question is where the camera parameters for each image are taken from. In [Vog06], two options for gathering this data are a robot arm, like the one shown in Figure 1.2(a), or an optical tracking system which tracks the endoscope's position and orientation with two cameras, shown in Figure 1.2(b). While these are viable solutions for a clinical environment, they are much too expensive for the first application. For the vendor, a feasible solution would be if he could use a consumer video camera like in Figure 1.2(c) to take an image sequence of an item without having to rely on specialized hardware to gather information about camera pose and parameters.

In principle, the images contain all information required to solve this scenario. The key is the identification of *point correspondences* between individual images of an image sequence. Knowledge of the projections of certain points in a scene into two images—the definition of point correspondences—allows the computation of a mapping of points from one image to the other, so that the possible position in the second image is restricted to a line. If the correspondences are known in three images the mapping can be calculated exactly, and the three relative camera poses can be estimated including even some of the internal parameters of the camera, as long as some assumptions are made. Given correspondences in yet more images it is possible to gradually


Figure 1.3: Example of an image sequence for a dynamic light field: a toy vehicle moves in a static environment while the hand-held camera is moved independently.

reduce these restrictions on the internal parameters.

In practice, camera parameter estimation is complicated by many factors. The point correspondences found are usually slightly shifted due to noisy images, or they are not available for the whole sequence, e. g., because of occlusion. Therefore, robust methods are required which are able to compensate the resulting errors and handle even very long image sequences. These methods usually include the use of (much) more data than strictly required and finding the solution which best fits this data. Outlier detection and probabilistic modeling may further improve the results.

At this point, all data for modeling a simple light field as proposed in the original light field publication [Lev96] is available. However, the parameterization of this light field requires a "dense sampling" of the scene from a regular grid of view points. In case of a hand-held camera, neither a dense sampling nor a sampling from regular view points is possible, since the human hand is ill-suited for such a task. More recent light field models therefore account for these additional requirements [Hei99a, Sch01b, Bue01]. Through sparse sampling, errors due to unknown surface geometry of the scene become visible in images rendered from the light field. Consequently, knowledge about scene geometry reduces these errors. Many methods exist for estimating the depth of a scene from several images. *Structure-from-motion* as well as depth from stereo or multiple images can be applied here.

Finally, going back to the example of the online vendor, he might wish to demonstrate not only a static model of an item, but also the functioning of any moving parts, or, for instance, the movement of vehicles through the environment like the toy vehicle in Figure 1.3.

In medical applications, such as endoscopic surgery [Vog06], it may be required to model organs or medical instruments which are being moved. Again, such a *dynamic* light field is most easily reconstructed using specialized hardware like an array of multiple cameras. If the problem

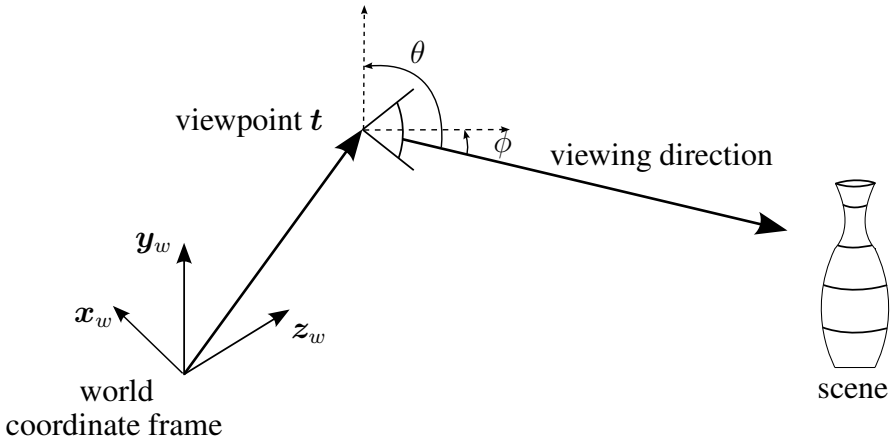



Figure 1.4: The plenoptic function: a single light ray seen by an observer at viewpoint $\boldsymbol{t}$ from the direction denoted by $\theta$ and $\phi$. Wavelength $\lambda_L$ and time $\tau$ are not depicted.

is approached using only one (hand-held) camera, a software solution must be found, and the problem domain is usually not as general as in the former case. By identifying different time steps or separating differently moving objects, a composed light field may be reconstructed and visualized using adapted light field rendering techniques.

This processing chain of reconstructing a static or dynamic light field model from one or several image sequences taken by a hand-held camera is the main topic of this thesis. The subsequent step of correctly rendering images out of such a light field, with its many different approaches, will only be covered shortly as part of the overview over the state of the art in light field modeling in the following section. It is regarded rather as a means of monitoring and evaluating the success of reconstruction.

## 1.2 State of the Art in Light Field Modeling

The idea of the light field was derived from the *plenoptic function* $L(\theta, \phi, \lambda_L, \tau, \boldsymbol{t})$ introduced in [Ade91]. It describes the appearance of a volume in space using seven parameters, namely the viewpoint $\boldsymbol{t}$ of the observer in world coordinates, the two angles $\theta$ and $\phi$ of the viewing direction and the wavelength $\lambda_L$ of the observed light ray at a certain point in time $\tau$. This setup is shown for a single light ray in Figure 1.4.

McMillan and Bishop [McM95] were the first to describe an image-based rendering system using the plenoptic function. Here, a *plenoptic model* is formed using images of the scene—which constitute "samples" of the plenoptic function—taken by a camera which is rotated around its vertical axis. The model is represented using a cylindrical parameterization which corre-

sponds to this two-dimensional panoramic image. Using additional disparity information, arbitrary views of the model can be computed from the inside of the cylinder.

The light field breaks down the high dimensional space of the plenoptic function using a different set of restrictions, which nevertheless allow the placement of the recording camera at arbitrary positions. The observed scene is assumed to be constant over time, and instead of the intensity for every wavelength only one color value is modeled, thus removing two parameters. In addition to that, the air between the observer and the scene surface is assumed to be transparent so that the intensity of the light ray emitted from a surface point in one direction stays constant, no matter where the observer is located on this ray. By selecting a suitable parameterization the plenoptic function is thus reduced to four parameters.

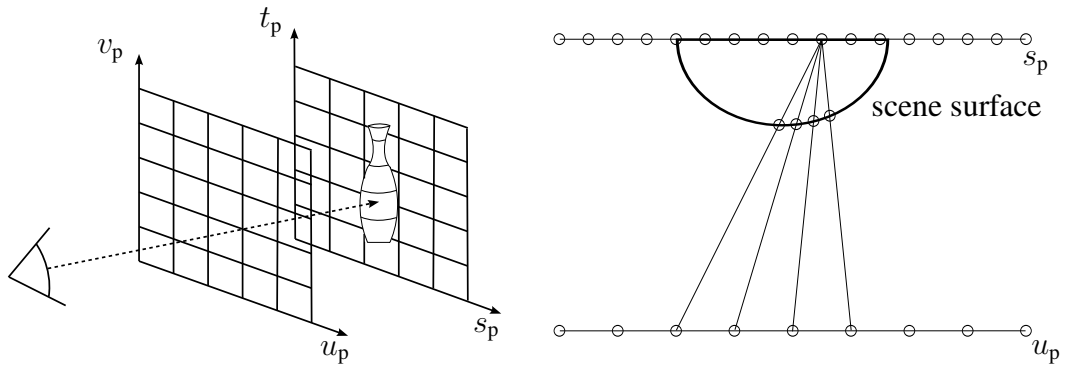
### 1.2.1 Light Field Models and Rendering Techniques

In the following, a short summary of the different parameterizations and rendering techniques for light fields will be given. The techniques are subdivided into two main categories, two-plane and free form parameterizations, while in a third category additional types of light fields are subsumed.

#### Two-plane parameterization

Both the approaches of Levoy et al. [Lev96] and Gortler et al. [Gor96] use a two-plane parameterization $L_{\mathrm{p}}(u_{\mathrm{p}}, v_{\mathrm{p}}, s_{\mathrm{p}}, t_{\mathrm{p}})$ to represent the light field. Here, each light ray of the (reduced) plenoptic function is described by one point on each of the planes $(u_{\mathrm{p}}, v_{\mathrm{p}})$ and $(s_{\mathrm{p}}, t_{\mathrm{p}})$. While in [Lev96] the projection centers of the original cameras are placed on the plane $(u_{\mathrm{p}}, v_{\mathrm{p}})$, called the *camera plane*, and the $(s_{\mathrm{p}}, t_{\mathrm{p}})$ plane is the common *focal plane* of those cameras, there is no such restriction in [Gor96]. In the latter, the two planes are set parallel to each other with, e. g., one plane through the scene surface and the other closer to the observer, as shown in Figure 1.5(a).

The two-plane parameterization simplifies the rendering of synthetic views from the light field considerably. As mentioned before, when using real input images the light field is only sparsely sampled, which means that it is unlikely that any arbitrary light ray was observed before. Gortler et al. create an approximation of the ideal, continuous light field $L_{\mathrm{p}}$ by subdividing the two planes into grids of $M_{u,v} \times M_{u,v}$ and $N_{s,t} \times N_{s,t}$ points, where $x_{i,j,k,l}$ denotes the color value of each pair of grid points. $(i, j)$ and $(k, l)$ index the grid points in the $(u_{\mathrm{p}}, v_{\mathrm{p}})$ and $(s_{\mathrm{p}}, t_{\mathrm{p}})$ grids, respectively. A basis function $B_{i,j,k,l}(u_{\mathrm{p}}, v_{\mathrm{p}}, s_{\mathrm{p}}, t_{\mathrm{p}})$ is associated to each grid point, which is then used to reconstruct the so-called *finite dimensional lumigraph* $\tilde{L}_{\mathrm{p}}$, an approximation to




(a) (b)

Figure 1.5: (a) Two-plane parameterization of the light field: each light ray is parameterized using the four coordinates $s_\mathrm{p}$, $t_\mathrm{p}$, $u_\mathrm{p}$ and $v_\mathrm{p}$. (b) Without depth correction on the $(s_\mathrm{p}, t_\mathrm{p})$ plane, mismatched light rays may be interpolated.

the continuous light field, from the discrete values by

$$\tilde{L}_\mathrm{p}(u_\mathrm{p}, v_\mathrm{p}, s_\mathrm{p}, t_\mathrm{p}) = \sum_{i=0}^{M_{u,v}} \sum_{j=0}^{M_{u,v}} \sum_{k=0}^{N_{s,t}} \sum_{l=0}^{N_{s,t}} x_{i,j,k,l} B_{i,j,k,l}(u_\mathrm{p}, v_\mathrm{p}, s_\mathrm{p}, t_\mathrm{p}) \quad . \tag{1.1}$$

Various choices for the basis function are possible, such as a box function which is $1$ around its associated grid point, and $0$ otherwise, or a quadrilinear function which allows the interpolation of neighboring grid points and thus results in a continuous function $\tilde{L}_\mathrm{p}$. The values $x_{i,j,k,l}$ at the grid points are computed from samples of $L_\mathrm{p}$, i. e., the input images, using the duals of the basis functions. Thus, an interpolation of the closest light rays in the light field to a required ray given by $(u_\mathrm{p}, v_\mathrm{p}, s_\mathrm{p}, t_\mathrm{p})$ is easily computed from $\tilde{L}_\mathrm{p}$. Instead of doing this interpolation ray by ray, i. e., for each pixel of a rendered image, the image patches in each grid of the $(u_\mathrm{p}, v_\mathrm{p})$ plane can be stored as textures and interpolation can be done on graphics hardware for several pixels at a time.

Up to this point, the approaches of Levoy et al. and Gortler et al. are very similar. However, if the recording camera positions were not close together, depth discontinuities in the scene may lead to serious blurring of the output images, the so-called *ghosting artifacts*. The reason for these errors is shown in Figure 1.5(b), namely that even if light rays intersect the $(s_\mathrm{p}, t_\mathrm{p})$ plane at the same point, they may not show the same point on the scene surface if the depth difference is too large. The approach of Gortler et al. incorporates depth information and thus reduces these artifacts.

The advantage of the two-plane approach is its efficiency in rendering, which was further

refined in a number of subsequent publications. Sloan et al. [Slo97] further increase the rendering speed by an improved selection of nodes on the $(u_\mathrm{p}, v_\mathrm{p})$ plane, although at the expense of quality. The same goal is pursued by Schirmacher et al. [Sch00a], who incorporate so-called *image warping* techniques to speed up rendering. This approach is only applicable if dense depth information is available for each image, but may then reduce the number of required samples considerably. Chai et al. [Cha00] examined the number of samples needed for optimal rendering of views from a light field.

However, the major drawback of the two-plane approach is its inflexibility regarding the sampling images, since they are required to be placed on a regular grid with a common focal plane. [Lev96] solves this by using a special camera gantry, while in [Gor96] the images are warped to fit the regular grid, which results in a loss of quality. Therefore, new parameterizations were developed which allow a more flexible camera placement.

### Free form light fields

The first relaxation of the two-plane constraint was proposed by Heigl et al. [Hei99a] for light fields which were recorded using a hand-held camera. Here, the images of the recording cameras are mapped onto the image plane of the virtual camera. The contributing cameras are selected by projecting their camera centers onto the virtual image plane and choosing the three closest projections to the required viewing ray. The virtual image plane is subdivided into triangles and corresponding image patches of the original cameras are mapped onto them. Geometry information is included using depth maps for each image and approximating a plane for each triangle.

The term *free form light field* was introduced by Schirmacher et al. [Sch01b] for a similar, but more efficient approach as the previous one. However, as a trade-off for increased performance, the original cameras are required here to form a convex hull around the scene or object, where each image has a view of the full silhouette of the scene.

In [Bue01], Buehler et al. define a set of eight requirements for light field rendering like *continuity* of color values, *resolution sensitivity* or *unstructured input*. Earlier parameterizations and rendering techniques like those described above are analyzed thereupon and a comparison is given. Accordingly, they propose a rendering algorithm called *unstructured lumigraph* rendering which considers these requirements. The unstructured lumigraph thus constitutes the most generalized free form light field renderer currently available and will be used for all continuative procedures and experiments throughout this thesis.

### Other representations

Analogously to the two-plane light field several alternative representations were introduced as well, using, e. g., a spherical parameterization for one or both of the two planes. However, these approaches never gained such wide-spread use as the two-plane parameterization. Therefore, the reader is referred to the brief summary of these approaches given in [Sch01b].

Another approach to free-form light field rendering is *point-based rendering*. Here, the light field is represented as a set of 3-D points with directional radiance information. The advantage is that instead of storing triangle meshes as geometry information, the connectivity information between the points can be omitted, which results in a smaller model. The disadvantage is that for close-up views and occlusions, gaps may become visible between the rendered points. Point-based rendering for light fields is described in [ES03b] and [Vog05].

The *surface light field*, introduced by Wood et al. [Woo00], is a parameterization which relies on a very accurate geometric model of the represented object. Radiance information for grid points on the surface of the model is stored in so-called *lumispheres*, which represent the light rays emitted from the grid point in any direction. The strength of this approach is its realistic modeling of the reflectance properties of the object, and even highlights are reproduced very well.

Apart from the above mentioned light field models and rendering techniques a number of additional contributions have been published on light field representations. A comprehensive summary of these approaches can be found in [Vog05].

## 1.2.2 Image Acquisition

As explained before, the minimum input of a light field—apart from depth information—consists of a set of images and the corresponding camera parameters. These include intrinsic parameters, e. g., focal length, pixel aspect ratio and other parameters which describe the internal properties of the camera, and extrinsic parameters, i. e., its position and orientation for each image, usually referred to as its *pose*. The camera parameters are introduced in detail in Section 2.1.1. Generally, there are two ways for acquiring these parameters, either using a particular hardware setup like a robot arm, or exclusively from the images themselves. This is true for the intrinsic as well as the extrinsic parameters.

If the intrinsic parameters stay constant over the whole process of image acquisition, their estimation is often done using a calibration pattern with well-known geometric properties. The parameters can then be computed quite accurately using standard calibration algorithms like

[Tsa87]. Camera calibration without a calibration pattern is called *self-calibration*, and is useful if the camera parameters vary during image acquisition or between two recordings. The current state of technology in this area is presented more comprehensively in Section 1.2.3.

In order to acquire the camera pose for each image in a light field, the various approaches summarized in the previous section use a number of different techniques. For the two-plane light field of [Lev96], the camera needs to be at certain discrete grid points on a plane for each image. Therefore, a camera gantry is employed which allows moving the camera on a plane, while the camera was equipped with a pan and tilt mounting which allows to keep the object in view. In order to record a complete fly-around the object was placed on a turntable and rotated by $90$ degrees, generating light fields from four directions. A similar, but much more sophisticated camera gantry is used in [Mat02] for acquiring images of specular and transparent objects. A vertical array of six cameras is mounted above a turntable, while an additionally rotating array of light sources provides varying illumination. In addition to that, monitors below and beside the object provide so-called *environment mattes* [Zon99]. Rendering of the objects is done using a point-based rendering approach [Zwi01].

Another, yet more flexible hardware-based approach to image acquisition is the use of a robot arm. Here, the pose of the camera on the tip of the arm is known via the rotation of the mechanical joints. In medical environments, e. g., robot arms are used to acquire light fields of endoscopic surgeries [Vog04, Vog06].

Instead of one or more movable cameras, the *light field video camera* [Wil02, Wil05] is a multi-camera array of up to $100$ cameras which are capable of recording images simultaneously. The challenge of this approach is to accurately synchronize such a large number of cameras. Although the disadvantage of this setup is that the number of images in the resulting light field is restricted to the number of physical cameras, it is very useful for generating *dynamic* light fields, and will therefore be treated in more detail in Section 1.2.5. The same approach, but on a microscopic scale, was used for the *plenoptic camera* [Ng05]. Here, an array of $296 \times 296$ microlenses was inserted between main lens and optical sensor of a hand-held camera, which then act as multiple individual cameras. Besides moving the observer in macrophotography, this technique allows a refocusing of the image during post-processing.

A less involved approach to calculating the camera pose is the use of a calibration pattern or markers which are visible throughout the whole image sequence. This method was chosen for image acquisition for the first lumigraph [Gor96]. The objects which were visualized here were placed in front and on top of a surface with special markers, whose positions relative to each other were known. Again, the algorithm of Tsai [Tsa87] was used to recover the camera pose.

Similarly, in [Sha02] the camera pose is calculated from two planes carrying four markers each, one plane is located behind the object and a transparent one in front of it.

Especially in medical applications, optical tracking systems are being employed which are based on the same principle of calibration using known markers [Sal01]. For light fields from endoscopic images like in [Vog06], the camera is equipped with a "target", which is recorded by a stereo camera. Thus, the position of the endoscope camera can be calculated quite accurately. However, this technique requires the calculation of a so-called hand-eye transformation between the target and the endoscope tip [Tsa89, Sch04a].

All image acquisition techniques introduced so far included additional hardware apart from the camera itself. However, it is possible to obtain the information required solely from an image sequence, so that no additional hardware is required. The techniques which are employed here are self-calibration for the intrinsic parameters of the camera, and structure-from-motion for the extrinsic ones. Approaches which generate a light field using these techniques and thus require only a single hand-held camera are described in [Koc99a, Hei99a, Bue01, Hei04]. Since this work is based on the same concept, namely the use of only a single hand-held camera for light field reconstruction, the state of the art in this area will be outlined in the next section. Nevertheless, one final light field acquisition system introduced in [Koc02] should be noted here. It constitutes an intermediate solution between the hardware-based and single-camera methods as it uses a hand-held multi-camera rig that consists of two or more cameras whose pose parameters are computed using structure-from-motion.

### 1.2.3 Camera Parameter Estimation

The first step for estimating camera parameters from recorded images is the detection of some kind of features in the images, and the establishment of correspondences between features in different images. These features may be of different kinds, but the most common ones are points or lines. The following section summarizes common approaches concerning point features, since they are also the basis for this work. The summary on camera parameter estimation itself is divided into approaches using different factorization methods, self-calibration methods for intrinsic camera parameter estimation, and continuative techniques handling, e. g., long image sequences which are not covered by the factorization based approaches.

### Point feature detection and tracking

The feature detection algorithms considered in this thesis are generally intensity based, which means that they define an interest operator that considers the intensity values inside a—usually rectangular—window of the image. The most well-known, and still commonly used, interest operators are those by Moravec [Mor77], Förstner [För87] and Harris [Har88]. All these approaches are based on the auto-correlation function, but while the first one only regards the similarity of a window with its neighborhood, the other ones define a matrix which indicates whether a window contains a homogeneous region, an edge or, ideally, a corner. A survey and evaluation of these and many other interest operators can be found in [Sch00b]. In recent years, the so-called SIFT features [Low99] have become quite popular. These rotation and scale invariant feature vectors are commonly used in object recognition or localization, but are also applicable for feature tracking. However, the features defined explicitly for the task of tracking by Tomasi and Kanade [Tom91] are by now a well-established standard. Here, the best features are those with two significant perpendicular gradients inside their window, e. g., corners.

Having detected interesting point features by one of the above algorithms, the next step is the redetection of each feature in another image or the matching of two sets of features found in two images. The latter is often done by hand if only few images with few features are regarded, usually for demonstration purposes, as, e. g., in [Men99]. This method can be automatized if a distance measure between features is defined and the closest ones are matched, as done in [Deu04] using SIFT features [Low99]. However, this matching technique only works well if enough corresponding features are detected in both images.

The alternative of relocating a feature in the next image which has been detected once has become widely used, especially for sequences or sets of many images. The basis of most of these tracking techniques is the so-called Lucas-Kanade tracker [Luc81]. Being originally intended for image registration, it uses a gradient descent minimization to find the best match for each feature independently in the second image. The error to be minimized is the difference between the intensity values in a window around the feature in both images. This method was taken up in [Tom91] and extended by Shi and Tomasi [Shi94] to estimate not only the translational movement, but also an affine deformation matrix for the feature window.

Subsequently, the algorithm experienced many more improvements and modifications. A method for illumination compensation, which was introduced by Hager and Belhumeur [Hag98] for object tracking, was incorporated in the tracker in [Jin01] along with a scheme for outlier detection. The problem of features “drifting away” from the original over long image sequences was addressed in [Mat03]. A considerable speed-up was reached by the so-called *inverse com-*

*positional approach* to updating the motion parameters [Bak01, Bak04]. These improvements were finally combined in the real-time tracking system of Zinßer et al. [Zin04], which is used throughout this thesis.

### Factorization-based structure-from-motion

The factorization-based approach to structure-from-motion was first introduced by Tomasi and Kanade in [Tom92]. Its basic principle is that a so-called measurement matrix, containing all point correspondences in a set of images, is decomposed into the 3-D structure of the scene, i. e., the 3-D points belonging to the image features, and the camera motion. A detailed introduction to these methods will be given in Section 2.2.

Following the first publication, many improvements and variations were proposed. Depending on the type of reconstruction achieved (see Section 2.3.1), these methods can be classified as affine or projective factorization. The former include the original method [Tom92], which assumes an orthographic projection model, and the weak- and paraperspective extensions by [Poe97], which are also applied in [Chr96] in order to iteratively obtain a Euclidean reconstruction from them. Projective reconstructions are estimated by the approach of Sturm and Triggs [Stu96], and an alternative algorithm was given by Oliensis [Oli01]. A recursive approach for projective reconstruction based on factorization was presented by Heyden et al. [Hey99b]. It is capable of integrating more and more images as they become available. A more detailed summary of factorization methods was given in [Kan98], including, e. g., methods using line instead of point features.

More recently, factorization methods were also regarded in a probabilistic framework, by either weighting each feature due to its reliability [Aan02, Aan03], or by modeling each feature with a covariance matrix [Ana02]. Especially if confidence information from feature tracking is available, these methods are capable of improving the robustness of reconstruction from noisy data.

### Self-calibration

A common restriction to the factorization methods summarized above is that they are only able to obtain the camera motion, but not its intrinsic parameters like, e. g., its focal length. The techniques applied for this purpose are called *self-calibration* or *auto-calibration*.

The first solution to the self-calibration problem was given by Faugeras et al. in [Fau92] using the *Kruppa equations*. These quadratic equations are computed from the fundamental matrix of only two views. Although the approach was refined subsequently, e. g., in [Zel96], it is usually

not applied as it is generally considered to be too sensitive to noise, and was found to fail in more situations than other approaches [Stu00].

Another approach which makes likewise use of properties of the fundamental matrix of two views is described in [Men99] and evaluated in [Fus01]. Contrary to the Kruppa equations, it computes the intrinsic parameters of at least three images by iterative minimization of a residual function. Since, for noisy data, it is prone to terminating in a local minimum, several improvements using global optimization were proposed [Rot02, Ben03].

Based mainly on a minimization of the back-projection error of reconstructed 3-D points and camera parameters by Levenberg-Marquardt optimization is the stratified self-calibration approach of Hartley [Har93]. The idea here is to update a projective reconstruction first to quasi-affine and then to metric, while assuming that the intrinsic parameters are the same for each image. An alternative update considering critical camera motions is given in [Dem98].

Triggs uses the so-called *absolute quadric* to recover the update matrix from projective to metric reconstruction [Tri97]. The advantage of this method is that the intrinsic parameters to be estimated need not be constant for all images, as it was shown by Pollefeys [Pol98, Pol99, Pol04]. In [Gib02], the estimation robustness is increased by a RANSAC-like technique. Theoretical requirements for the number of known parameters are investigated by Heyden and Åström in [Hey97, Hey98, Hey99a].

Unfortunately, self-calibration is still very sensitive to noise. Therefore, many of the authors in this field propose to use a technique called *bundle adjustment* to refine the results of self-calibration. Bundle adjustment optimizes 3-D points and camera parameters for a large number of images simultaneously, and is therefore a slow but robust technique as it solves an extremely over-determined system of equations. It was originally developed in photogrammetry [Sla80], but introduced in computer vision in [Har93]. An extension called *interleaved bundle adjustment* [Sze98] is most commonly used.

Since self-calibration is an important but difficult problem, many more contributions have been published on this topic. More extensive summaries than the one presented here can be found in [Pol99, Fus00, Har03].

#### Reconstructing long image sequences

Image-based modeling using light fields usually requires not only a few images, but often large sets of several hundred images. The factorization methods described above share the major drawback that they require all feature points to be visible in all images. Although methods exist to estimate the position of missing features by considering constraints on the subspace spanned

by the features [Tom92, Gue02], these are likely to unduly increase the error in reconstruction if too many features are replaced.

An alternative approach commonly used is to start out with a projectively skewed reconstruction of the camera parameters by successively estimating the fundamental matrix of image pairs [Har93] or the trifocal tensor of image triplets [Bea96], and merging the resulting subsequence reconstructions. Since the errors made for each trifocal tensor estimation accumulate, Fitzgibbon and Zisserman [Fit98] propose to evenly distribute the error to all camera positions for the case that circular camera movements are given. The approach was further improved by using hierarchical merging to achieve the same goal of uniform error distribution, as well as different frame selection schemes [Nis00, Gib02, Sai03]. The so-called *viewpoint mesh weaving* by Koch et al. [Koc99a] generates a 3-D topology of camera positions and thus allows merging of subsequences in several directions.

The restriction of factorization methods to short image sequences is avoided by Heigl and Niemann in [Hei99b], as they combine successive factorizations by merging thus reconstructed overlapping subsequences. Using factorization as an initialization for estimating the camera parameters for the remaining images by non-linearly minimizing the back-projection error is described in [Hei04].

### 1.2.4 Scene Depth

As explained before, sparsely sampled light fields, thus especially free-form light fields from hand-held camera sequences, require scene depth information for accurate rendering. Calculating depth from multiple images has been a very active field of research in computer vision for almost 30 years. Therefore, only a short introduction into the topic will be given with respect to light field modeling.

Apart from the original light field of Levoy, all rendering methods introduced so far make use of additional depth information. For the most part, especially for free-form light fields, depth is stored as one depth map per input image. In the two-plane case, depth information is often integrated globally into the model, like in the original lumigraph, while the light field approach by Isaksen [Isa00] uses a global focal plane that can be adapted dynamically to scene depth. The unstructured lumigraph [Bue01], however, uses a global 3-D mesh to approximate the complete scene geometry.

Numerous possibilities exist to calculate per-image depth maps from a sequence of images. If, as it is done by Heigl [Hei04], a 3-D reconstruction of the scene is calculated from feature correspondences, the resulting 3-D information can be used to infer a sparse depth map for

each image. However, if the camera parameters are known depth can also be calculated from optical flow in stereo image pairs or multiple images. Being mostly based on the optical flow algorithm by Horn and Schunck [Hor81], many different techniques and improvements for the stereo case have been proposed in the following years. Comprehensive summaries can be found in [Bar94, Bea95], although research is still ongoing in this area as can been seen, e. g., in the scale-space based approaches by Alvarez et al. [Alv00, Alv02]. Calculating optical flow from multiple images has only been investigated more recently, e. g., in [Ira99, Kan04].

Similar to Heigl, Kang and Szeliski [Kan96b] use a 3-D reconstruction computed from structure-from-motion to obtain a global 3-D scene model for cylindrical panorama images. For arbitrary camera positions, several algorithms for merging stereo depth maps to a global model have been proposed [Hig94, Cur96, Koc99b]. If only a single object is observed which can be reliably separated from the background a so-called *shape from silhouette* approach can be applied [Pot87, Den98]. Otherwise, the *space carving* [Kut99] technique iteratively generates a voxel volume by checking the consistency of the surface voxels with all images. The common drawback of all these approaches is that they rely on a very accurate calibration of camera parameters in order to generate *consistent* global depth information. Small errors for single images may accumulate enough when all images are used so that the algorithms fail.

### 1.2.5 Dynamic Light Fields

In contrast to static light fields, only few contributions have been published so far on the topic of dynamic light fields. Since the problem domain increases drastically by adding another dimension to the parameter space, only special cases have been addressed so far. Regarding the rendering of light fields of moving objects, Li et al. [Li98] were the first to propose a solution, albeit only for synthetic data sets. Here, a dynamic light field is assembled from a number of static sub light fields which may move and rotate freely in space.

A straightforward approach to rendering dynamic light fields of real scenes is to generate multiple static light fields of different time steps. In an example this was first demonstrated in [Bue01] for the unstructured lumigraph. Here, the periodic movement of an example object is divided by hand into several time steps. The *dynamic textures* approach [Dor03a] aims at modeling the appearance of phenomena such as waves and smoke. In [Dor03b] the authors propose to use a dynamic lumigraph to model dynamic textures, but this approach is not implemented.

The light field video camera mentioned earlier (see Section 1.2.2), which combines several synchronized cameras, greatly facilitates the generation of dynamic light fields. A static light field is thus generated several times per second, depending on the frame rate of the cameras

used, and the sequence of light fields can be viewed like a 3-D movie where the view point can be chosen arbitrarily [Gol02]. Similar camera setups are introduced in [Ooi01, Sch01a, Yan02].

In the case that only a single camera is to be used for dynamic light field acquisition, one feasible approach is to firstly identify the moving entities or regions in the scene. For the special case of rigid, moving objects in an otherwise static scene, several different approaches have been proposed. For restricted motion a 3-D reconstruction can be determined simultaneously for all objects in the scene, i. e., for linear object movement [Avi99, Han00, Sha01, Han03] or object movement along a conic section [Sha99]. However, 3-D movement can only be determined up to scale. For arbitrary object movement, several similar methods have been proposed to separate differently moving objects in the scene [Cos98, Kan01a, Kan03, Vid04]. A 3-D reconstruction is then obtained for each object separately. The dynamic light fields introduced in Section 4.3 are based on these separation methods.

## 1.3 Contribution of this Work

The methods developed in this thesis are based on the work of Heigl [Hei04] and make use of the same implementation framework. Therefore, the basic assumptions made here are similar: Using only one hand-held camera, an image-based model of a scene is generated from a recorded image sequence. The camera parameter reconstruction for each image is done by point feature based structure-from-motion, which requires that the camera moves rather slowly and thus the viewpoint changes only slightly from image to image, so that point feature tracking is possible. No assumption is made on the application domain of the recorded images, which means that scenes of any kind should be processable, be it indoor or outdoor scenes, showing single objects or complex environments. The only requirement here is that the scene surface has a certain texture or structure to allow for the detection of point features. Lighting is assumed to be constant as changes in illumination are not modeled during rendering.

One important restriction of Heigl's work is revoked in this thesis, which is the constancy of the scene. Instead of purely static scenes, dynamic scenes are now considered as well, although no general motion will be allowed, but some restrictions will still be applied: the motion has to be step-wise, repetitive, or rigid.

The contributions of this work can be divided into three different aspects. The first one constitutes improvements and new approaches for reconstructing static light field models, whereas the second aspect is the reconstruction and modeling of dynamic scenes for light field rendering, including adaptations of rendering techniques. The third aspect is the application of light field

models to other fields of pattern recognition and image or video processing.

**Static light fields.** Reconstructing a light field requires a large number of processing steps and the combination of many different techniques. The processing chain proposed in [Hei04] is thus extended by including new approaches for feature tracking [Zin04], factorization [Chr96] and depth map estimation [Alv02]. Additionally, approaches for probabilistic [Kan01b] and robust [Ham86] modeling of different aspects of the reconstruction process are studied. An important contribution is that the linear processing chain was decoupled and different possibilities for feeding back information from later processing steps to previous ones are investigated. Examples are the use of calibration information in order to refine feature tracking, or the use of the final image synthesis for improving the available depth maps. The extension of existing light fields with new images is part of this decoupling of the reconstruction process. Finally, a method is proposed to generate global scene geometry information which increases rendering performance significantly.

Considering these modifications, a way to evaluate the resulting improvements is required. A method is introduced which allows a quantitative measurement of the quality of images rendered from a static light field, and thus offers a possibility to compare different approaches to light field reconstruction.

**Dynamic light fields.** The major contribution of this work regarding dynamic light fields is that, in contrast to the approaches mentioned in Section 1.2.5, only one camera is used for capturing images of the scene, and the light field is reconstructed using structure-from-motion approaches. The restriction to one camera necessitates some constraints on the scene motion. Depending on the reconstruction method, these restrictions are that the scene motion is either only step-wise, or contains only rigid, moving objects and repetitive motion. If step-wise motion is available, reconstruction can be done using common structure-from-motion algorithms. For rigid, moving objects additional segmentation methods [Kan03] are applied. In addition to the reconstruction methods, the available rendering algorithms are extended to enable the rendering of the dynamic light fields.

**Applications of light fields.** Since light fields constitute models of scenes or single objects, it is a logical consequence to use them for different model-driven applications in image processing. Appearance-based object localization, recognition and tracking are some of the areas where light fields have been successfully applied. Here, approaches are introduced which either facilitate the training step of a recognition and localization system or where light fields are applied directly

as object models. Another area where light fields are useful tools is augmented reality, which received increasing attention in recent years. Light fields reconstructed from real images are used here to augment scenes not only by synthetic but real objects.

## 1.4 Overview

In the following, a short overview over the content of this thesis will be given. After introducing the problem domain of light field reconstruction in Chapter 1, Chapter 2 describes the theoretic foundations of projection models, camera calibration and structure-from-motion algorithms such as the factorization method.

In Chapter 3, the reconstruction process of static light fields from image sequences is explained in detail. After an introduction to the reconstruction system layout, the different processing steps are described, beginning with feature tracking, the factorization of subsequences and the extension to a complete image sequence. Additionally, the chapter covers the topics of probabilistic approaches and the computation of scene geometry. The different steps are finally combined in new order by so-called information feedback loops.

The extension of the approaches for static light field reconstruction to dynamic scenes is examined in Chapter 4. After defining the additional requirement for dynamic light fields, several different approaches to reconstructing them from image sequences are introduced. The chapter also deals with the additional requirements which follow for the rendering of dynamic light fields.

After the theoretical descriptions, Chapter 5 presents an experimental evaluation of both static and dynamic approaches to light field reconstruction. A method for quantitative assessment of the rendering quality is introduced.

Following the experiments the next chapter describes several applications of light fields in augmented reality, object recognition and tracking, and medical environments. The last shows their usefulness especially in supporting surgeons in minimally invasive medicine.

The last chapter finally presents a summary of the thesis and an outlook to possible future additions and improvements to the reconstruction system.

# Chapter 2

# Camera Parameter Estimation

As outlined in Chapter 1, a light field is described by a set of input images as well as knowledge on the properties of the recording camera and its pose for each input image. The aim in this work is to estimate the required camera parameters exclusively from information derived from the input images, such as point feature correspondences. The approaches used for this purpose are structure-from-motion for camera pose estimation, and self-calibration for estimating the internal camera parameters.

In the following chapter, the basic principles of camera calibration, such as the mathematical description of a camera and different projection models, will be introduced. In addition to that, the methods used for pose estimation and self-calibration throughout this contribution will be explained, including factorization methods for simultaneously estimating scene structure and camera motion, and non-linear optimization methods for camera parameter estimation.

## 2.1 Camera Calibration

Modern cameras are sophisticated optical devices which include, besides one or several CCD chips, a number of lenses for aberration reduction, zooming and focusing [Smi00]. In order to render the properties of such a camera mathematically manageable, simplifying models have been introduced, of which the pinhole camera model is most widely used and will also be applied in the following.

The projection properties of a camera are described by different parameters, which can be separated into the extrinsic parameters—i. e., the camera pose—and the intrinsic ones, which comprise the internal properties of the camera, which may be fixed or adjustable. These parameters will be described in the following. However, the more parameters are taken into consid-

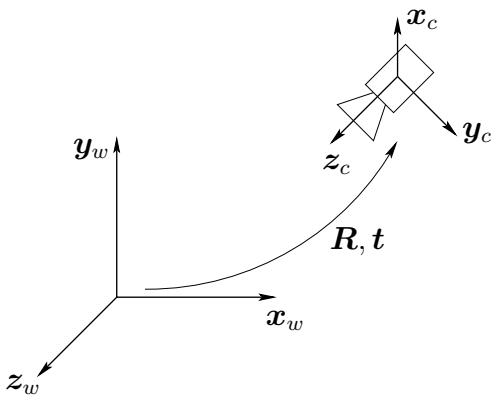



Figure 2.1: Mapping from world to camera coordinates using a rotation $\boldsymbol{R}$ and a translation $\boldsymbol{t}$

eration, the more unstable their estimation will be. Different projection models may be assumed which allow for a choice in the number of parameters. The most important ones of them will be introduced as well.

### 2.1.1 Camera Parameters

Mathematically, a camera denotes a mapping of points in a three-dimensional world coordinate system into the two-dimensional image. This mapping will be described in the following for the perspective projection model. The simplified projection models will be introduced afterwards in Section 2.1.2.

#### Camera pose parameters

The mapping from world to image coordinates can be decomposed into a number of consecutive mappings, i. e., mappings to intermediate coordinate systems. In practice, the camera coordinate system is the most relevant as it describes the position of a 3-D point relative to the camera's position—the position of its optical center—and viewing direction or orientation. Position and orientation together are usually referred to as the *pose* of the camera.

As shown in Figure 2.1, the transformation between world and camera coordinate system is given by a rotation matrix $\boldsymbol{R} \in \mathbb{R}^{3\times3}$ and a translation vector $\boldsymbol{t} \in \mathbb{R}^3$. The axes of the camera coordinate system form a right-handed system, with the viewing direction along the $z$-axis and the image plane of the camera parallel to the $x$- and $y$-axis. In relation to the image, $\boldsymbol{x}_c$ points to the right and $\boldsymbol{y}_c$ to the bottom. Finally, the columns of $\boldsymbol{R}$ are formed by the three axes $\boldsymbol{x}_c$, $\boldsymbol{y}_c$ and $\boldsymbol{z}_c$.

Thus, a 3-D point $\boldsymbol{p}$ in world coordinates would be transformed to $\boldsymbol{p}_c$ in the camera coordinate system by applying the equation

$$\boldsymbol{p}_c = \boldsymbol{R}^{\mathrm{T}}(\boldsymbol{p} - \boldsymbol{t}) \quad . \tag{2.1}$$

This equation is given in Euclidean coordinates, but for reasons that will become apparent later on, it is transformed into so-called projective coordinates. Projective geometry increases the dimensionality of vectors by one, which are then called *homogeneous* vectors. For a short summary on the properties of projective geometry refer to [Hei04]. A more detailed introduction is given in [Har03]. In the following, homogeneous vectors will be indicated by underlining. Here, using homogeneous coordinates for the point $\underline{\boldsymbol{p}}$ in world coordinates and $\underline{\boldsymbol{p}}_c$ in camera coordinates respectively, the mapping can be rewritten as a matrix multiplication

$$\underline{\boldsymbol{p}}_c = \begin{pmatrix} \boldsymbol{R}^{\mathrm{T}} & -\boldsymbol{R}^{\mathrm{T}}\boldsymbol{t} \\ \boldsymbol{0}_3{}^{\mathrm{T}} & 1 \end{pmatrix} \underline{\boldsymbol{p}} \quad . \tag{2.2}$$

It should be noted here that throughout the literature the rotation matrix $\boldsymbol{R}$ is constructed in two different ways. In [Fau01] and [Har03], e. g., $\boldsymbol{R}$ is composed from $\boldsymbol{x}_c$, $\boldsymbol{y}_c$ and $\boldsymbol{z}_c$ as its *row* vectors. The advantage of this representation is that in equations (2.1) and (2.2) $\boldsymbol{R}^{\mathrm{T}}$ can be substituted by $\boldsymbol{R}$. However, in this representation $\boldsymbol{R}$ would not denote a rotation of the camera but of the 3-D scene instead. Therefore, the more intuitive representation was chosen for this work where $\boldsymbol{R}$ is composed of the camera coordinate axes as its *column* vectors, thus denoting a rotation of the camera itself.

### Intrinsic parameters

From camera coordinates a 3-D point is mapped into the coordinate system of the image by a perspective projection. Using homogeneous coordinates again, this mapping can be written as a linear projection matrix. The entries of this matrix are called the *intrinsic* camera parameters as they denote the physical properties of the camera itself. The perspective projection introduced in the following is already a simplification of the true behavior of a real camera, but it is sufficient for most image processing applications. However, further simplified projection models are often required and will be given in Section 2.1.2.

The geometric model usually assumed for perspective projection is the pinhole camera. As shown in Figure 2.2, a 3-D point $\boldsymbol{p}_c$ in camera coordinates is projected along a line through the camera center which corresponds to the origin of the camera coordinate system. The intersection with the image plane yields the 2-D point $\boldsymbol{q}_m$ in image coordinates. The image plane is assumed

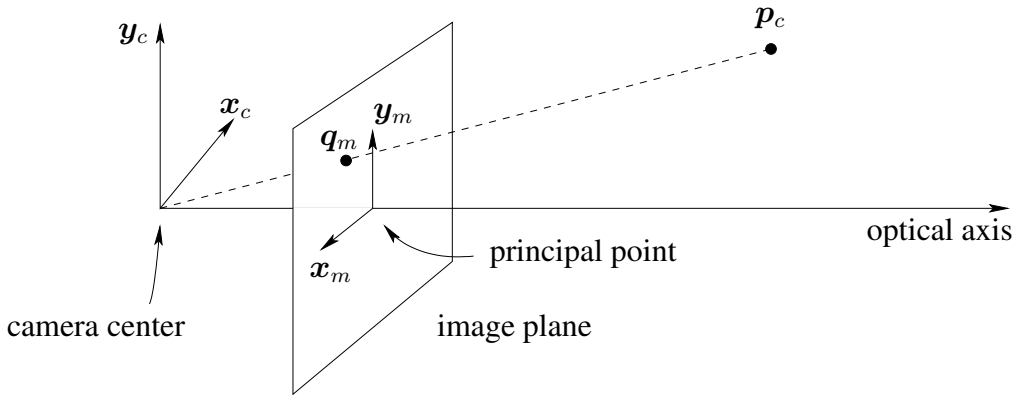



Figure 2.2: The pinhole camera model. The 3-D point $\boldsymbol{p}_c$ in camera coordinates is projected along a line through the camera center into the image plane.

to be parallel to the plane spanned by the $x$- and $y$-axis of the camera coordinate system at a distance $f$, the focal length. The $z$-axis of the camera coordinate system is also called the optical axis and intersects the image plane at the principal point.

For the pinhole model, the projection of a point $\boldsymbol{p}_c$ is given by

$$\boldsymbol{q}_m = \frac{f}{p_{c3}} \begin{pmatrix} p_{c1} \\ p_{c2} \end{pmatrix} \tag{2.3}$$

with $\boldsymbol{p}_c = (p_{c1}, p_{c2}, p_{c3})^{\mathrm{T}}$. This means that the farther $\boldsymbol{p}_c$ is away from the image plane along the optical axis, the closer its projection will be to the principal point in the image. On the other hand, the larger the focal length $f$, the farther the projection will be from the image center.

As indicated before, homogeneous coordinates get important here since this non-linear equation becomes a linear one when they are used, and can be written as a matrix equation:

$$\underline{\boldsymbol{q}}_m = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \underline{\boldsymbol{p}}_c \quad . \tag{2.4}$$

The above equations determine the mapping of 3-D points onto the image plane, which is defined by $\boldsymbol{z}_c = f$. In reality, this plane is where the CCD sensor of the camera is situated, and the mapping to the actual sensor coordinates is given by additional parameters. The elements of the sensor are assumed to be placed on a regular grid with their position being defined by their centers, as shown in Figure 2.3. The origin $\boldsymbol{o}$ of the grid is situated in its upper left corner, with the axes denoted by $\boldsymbol{u}_s$ and $\boldsymbol{v}_s$. The size of one grid element, i. e., one pixel, is given by $d_x$

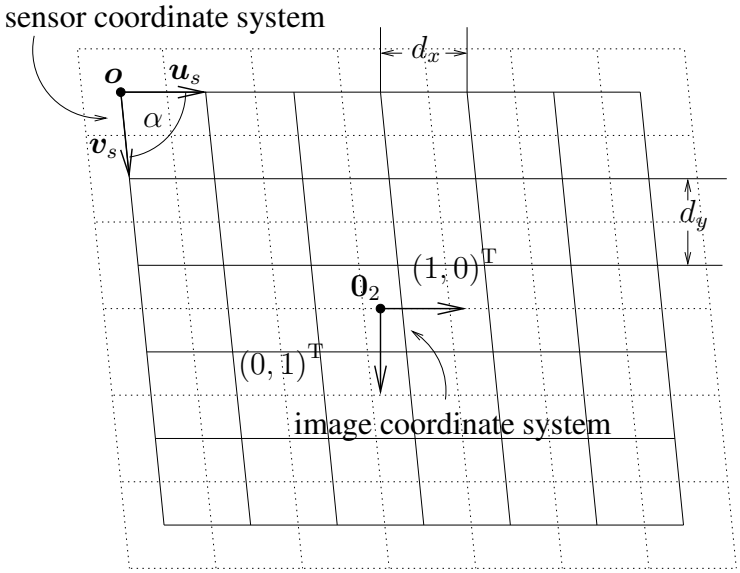



Figure 2.3: Configuration of the sensor and image coordinate system. The solid grid connects the midpoints of the sensor elements whose borders are marked by dotted lines. Figure by courtesy of B. Heigl [Hei04].

horizontally and $d_y$ vertically. Thus, if $\boldsymbol{u}_s = (d_x, 0)^T$, $\boldsymbol{v}_s$ computes to

$$\boldsymbol{v}_s = \begin{pmatrix} \tan(\frac{\pi}{2} - \alpha) d_y \\ d_y \end{pmatrix} \quad , \tag{2.5}$$

with $\alpha$ being the angle enclosed by $\boldsymbol{u}_s$ and $\boldsymbol{v}_s$. The principal point, denoted by $\mathbf{0}_2$ in Figure 2.3 can now be expressed in sensor coordinates by

$$\begin{pmatrix} u_{pp} \\ v_{pp} \end{pmatrix} = \begin{pmatrix} \frac{1}{d_x} & \frac{-\tan(\frac{\pi}{2}-\alpha)}{d_x} \\ 0 & \frac{1}{d_y} \end{pmatrix} (-\boldsymbol{o}) \quad . \tag{2.6}$$

Including the new parameters in the mapping introduced in equation (2.4), the projection of a 3-D point $\underline{\boldsymbol{p}}_c$ in camera coordinates to an image point $\underline{\boldsymbol{q}}$ in sensor coordinates is now given by

$$\underline{\boldsymbol{q}} = \begin{pmatrix} \frac{f}{d_x} & \frac{-f\tan(\frac{\pi}{2}-\alpha)}{d_x} & u_{pp} & 0 \\ 0 & \frac{f}{d_y} & v_{pp} & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \underline{\boldsymbol{p}}_c \quad . \tag{2.7}$$

For convenience, some of the parameters are combined, i. e., $f_x = f/d_x$ and $f_y = f/d_y$ are the effective focal length in horizontal and vertical direction, and the skew $\beta$ of the sensor

coordinate axes is defined as

$$\beta = \frac{-f \tan(\frac{\pi}{2} - \alpha)}{d_x} \quad . \tag{2.8}$$

For the cameras usually used in computer vision, the axes of the sensor grid can be assumed to be orthogonal, thus $\beta$ would become $0$. In fact, for the self-calibration algorithms introduced in Section 2.3 this is even a requirement.

In addition to the intrinsic parameters introduced up to now, a real projection system may contain many more distortions which can be modeled as well. The parameters for correcting radial or tangential distortions are obtained by using a calibration pattern with known world coordinates of its features. A standard algorithm for this problem was given by Tsai in [Tsa87]. However, the goal of this work is to estimate the camera parameters without such a calibration pattern. Self-calibration algorithms are so far unable to obtain these distortion coefficients, therefore they will not be introduced here in more detail.

#### The projection matrix

The left $3 \times 3$ submatrix of the projection of equation (2.7) is called the *calibration matrix* and is defined as

$$\boldsymbol{K} = \begin{pmatrix} f_x & \beta & u_{pp} \\ 0 & f_y & v_{pp} \\ 0 & 0 & 1 \end{pmatrix} \quad . \tag{2.9}$$

It maps a non-homogeneous point $\boldsymbol{p}_c$ in camera coordinates to a homogeneous point $\underline{\boldsymbol{q}}$ in sensor coordinates. Using equation (2.1), the complete projection of a 3-D world point to an image is thus given:

$$\underline{\boldsymbol{q}} = \boldsymbol{K}\boldsymbol{R}^{\mathrm{T}}(\boldsymbol{p} - \boldsymbol{t}) \quad . \tag{2.10}$$

Finally, by using homogeneous coordinates again, the mapping from world to sensor coordinates is rewritten as

$$\underline{\boldsymbol{q}} = \boldsymbol{K}\boldsymbol{R}^{\mathrm{T}} \left( \boldsymbol{I}_{3\times 3} | - \boldsymbol{t} \right) \underline{\boldsymbol{p}} = \boldsymbol{P}\underline{\boldsymbol{p}} \quad . \tag{2.11}$$

Extrinsic and intrinsic parameters are thus combined in a single matrix $\boldsymbol{P}$ which is called the *projection matrix*.

### 2.1.2 Projection Models

The pinhole camera model introduced in the previous section is an approximation of the projection properties of a real camera. Nevertheless, the estimation of its parameters is still complex,

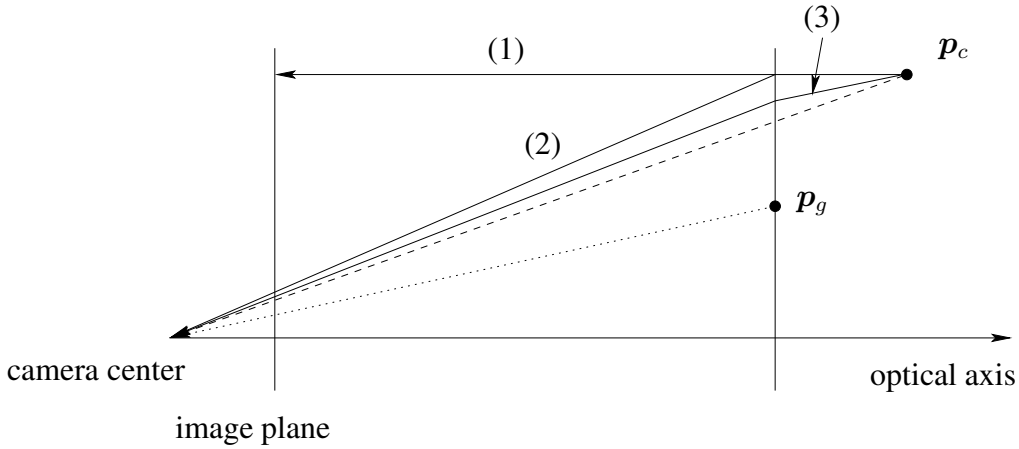



Figure 2.4: Projection of a 3-D point with different projection models. The dashed line shows perspective projection, the dotted line the projection of the center of gravity $\boldsymbol{p}_g$. The solid lines denote (1) orthographic, (2) weak-perspective and (3) paraperspective projection.

and further simplifications likewise simplify the parameter estimation, although at the expense of decreased accuracy.

The projection of a point from camera to image coordinates done by equation (2.4) is called perspective projection. Other projection models are orthographic, weak-perspective and paraperspective projection, although many more models exist. They are of importance in this work and will be discussed in the following. A comparison of these projection models is shown in Figure 2.4.

### Orthographic projection

Orthographic projection—also called parallel projection—is the most basic type of projection. It transforms equation (2.3), the projection of a point $\boldsymbol{p}_c$ in camera coordinates to image coordinates, to the simple form

$$\boldsymbol{q}_m = \begin{pmatrix} p_{c1} \\ p_{c2} \end{pmatrix} \quad . \tag{2.12}$$

In other words, the $z$-coordinate, or depth, of the 3-D point is ignored and the point is thus projected parallel to the camera's optical axis onto the image plane. Orthographic projection is a special case of perspective projection as it assumes that $f \to \infty$ and $p_{c3} \to \infty$. The projection

matrix of equation (2.4) thus changes to

$$\underline{\boldsymbol{q}}_m = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \underline{\boldsymbol{p}}_c \quad . \tag{2.13}$$

### Weak-perspective projection

Weak-perspective projection is essentially a combination of the orthographic and the perspective projection model. As shown in Figure 2.4, a 3-D point is first projected orthographically onto a plane parallel to the image plane through the center of gravity of all 3-D points, followed by a perspective projection onto the image plane. This perspective projection corresponds to an isotropic scaling, therefore, this projection is also called *scaled-orthographic*.

For the weak-perspective case, equation (2.3) for projecting a point in camera coordinates is modified to

$$\boldsymbol{q}_m = \frac{f}{p_{g3}} \begin{pmatrix} p_{c1} \\ p_{c2} \end{pmatrix} \quad . \tag{2.14}$$

Using the $z$-component $p_{g3}$ of the center of gravity $\boldsymbol{p}_g$, i. e., its distance from the center of projection along the optical axis, instead of the true depth of the 3-D point $\boldsymbol{p}_c$, treats every point as having the same depth and thus being projected onto the same plane perpendicular to the optical axis. In matrix notation, the projection is written as

$$\underline{\boldsymbol{q}}_m = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 0 & p_{g3} \end{pmatrix} \underline{\boldsymbol{p}}_c \quad . \tag{2.15}$$

Since this projection is only an approximation to the real projection, it is only valid under certain conditions. In this case, it is required that the relative depth difference between the 3-D points in the scene is at least $20$ times smaller than their average distance from the camera [Tru98]. A survey of the errors made by weak-perspective approximation can be found in [Hei04].

Again, the literature does not agree about the distinction of projection models. While [Tru98] and [Fau01] describe weak-perspective and scaled-orthographic projection as being the same, in [Har03] they are not. There, for weak-perspective projection the focal length is different in $x$ and $y$ direction, while for scaled-orthographic projection there is only one focal length $f$.

### Paraperspective projection

The last projection model introduced here is the paraperspective projection, which is even more accurate than the preceding one. It was first proposed in [Oht81] and finally named in [Alo90]. The difference to weak-perspective projection is that 3-D points are not projected parallel to the optical axis onto the plane through $\boldsymbol{p}_g$, but parallel to the line connecting $\boldsymbol{p}_g$ and the camera center. The effect is again depicted in Figure 2.4.

The projection $\boldsymbol{p}'_c$ of a point $\boldsymbol{p}_c$ onto the plane through $\boldsymbol{p}_g$ is given by

$$\boldsymbol{p}'_c = \boldsymbol{p}_c - \frac{p_{c3} - p_{g3}}{p_{g3}} \boldsymbol{p}_g = \boldsymbol{p}_c - b\boldsymbol{p}_g \quad , \tag{2.16}$$

with $b = (p_{c3} - p_{g3})/p_{g3}$. Modifying equation (2.14) accordingly, $\boldsymbol{p}_c$ is projected to the image plane by

$$\boldsymbol{q}_m = \frac{f}{p_{g3}} \begin{pmatrix} p_{c1} - bp_{g1} \\ p_{c2} - bp_{g2} \end{pmatrix} \quad . \tag{2.17}$$

Again, this projection can be given in matrix notation as well, i. e.,

$$\underline{\boldsymbol{q}}_m = \begin{pmatrix} f & 0 & 0 & f - bp_{g1} \\ 0 & f & 0 & f - bp_{g2} \\ 0 & 0 & 0 & p_{g3} \end{pmatrix} \underline{\boldsymbol{p}}_c \quad . \tag{2.18}$$

The advantage of paraperspective projection over the weak-perspective one is that now objects in the scene with a large depth difference can be modeled as well, i. e., the error induced by such a configuration is less.

The simplified projection models introduced so far are summarized under the term *affine* projection models, since the projection matrices, which can be scaled arbitrarily, are generally of the form

$$\boldsymbol{P} = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{t} \\ \boldsymbol{0}_3{}^{\mathrm{T}} & 1 \end{pmatrix} \quad , \tag{2.19}$$

where $\boldsymbol{A}$ is an arbitrary $2 \times 3$ matrix and $\boldsymbol{t}$ a translation vector.

### Perspective projection

As stated before, the projection models introduced above are approximations of perspective projection, which is the correct projection for the pinhole camera model. One of their most important advantages is that they are described by linear projection equations, as seen in equations (2.12),

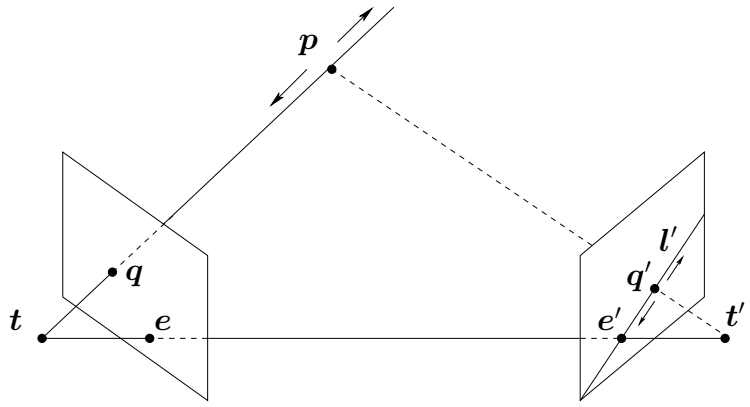



Figure 2.5: Epipolar geometry: given a point $\boldsymbol{q}$ in the left image, its corresponding point $\boldsymbol{q}'$ in the right image is located somewhere along the epipolar line $\boldsymbol{l}'$, since the position of the corresponding world point $\boldsymbol{p}$ is undetermined.

(2.14), and (2.17). Equation (2.3) for perspective projection on the other hand is non-linear since $p_{c3}$ is found in the denominator. Parameter estimation is thus much more difficult for perspective projection, as will become apparent in Section 2.2.

### 2.1.3 Epipolar Geometry

An important first step to 3-D reconstruction for many applications is to obtain information about the relationship between two images, or the recording cameras respectively. This relationship is described by the so-called *epipolar geometry*, which offers an answer to the following question: Given two images, their projection matrices, and a point $\boldsymbol{q}$ in the first image, what can be said about its corresponding point $\boldsymbol{q}'$ in the second image?

The problem is illustrated in Figure 2.5. A point $\boldsymbol{q}$ in one image and its camera center define a ray on which the observed 3-D point $\boldsymbol{p}$ is located. The depth of this 3-D point cannot be determined from one image alone, therefore, its projection into the other image, $\boldsymbol{q}'$, may be anywhere along a line $\boldsymbol{l}'$. This line, called the *epipolar line*, is defined by the intersection of the image plane with a plane which is spanned by the two camera centers and the image point $\boldsymbol{q}$. Such a plane is called an *epipolar plane*. All possible epipolar planes form a pencil of planes around the line between the two camera centers, the baseline of the stereo system. The intersections of this line with the two image planes are called the two *epipoles* $\boldsymbol{e}$ and $\boldsymbol{e}'$ of the stereo system. Every epipolar line passes through the epipole in its respective image.

The relationship between an image point in one image and the corresponding epipolar line in the other image is described by the *fundamental matrix* $\boldsymbol{F}$, which is valid for any such point-line

pair. $\boldsymbol{F}$ is a $3 \times 3$ matrix with arbitrary scaling and rank 2, which means that it has 7 degrees of freedom. For a homogeneous point $\underline{\boldsymbol{q}}$, the epipolar line $\underline{\boldsymbol{l}}'$ is thus calculated as

$$\underline{\boldsymbol{l}}' = \boldsymbol{F}\underline{\boldsymbol{q}} \quad . \tag{2.20}$$

Consequently, for corresponding points in two images, the following equation is valid:

$$\underline{\boldsymbol{q}}'^{\mathrm{T}}\boldsymbol{F}\underline{\boldsymbol{q}} = 0 \quad . \tag{2.21}$$

Considering equation (2.21) and the fact that $\boldsymbol{F}$ has only seven degrees of freedom, it can be shown that $\boldsymbol{F}$ can be computed with a minimum of seven point correspondences. However, the most common way to estimate $\boldsymbol{F}$ is to solve an overdetermined system of equations with at least eight point correspondences using the *normalized 8-point-algorithm* [Har97]. It is described in detail in [Har03], along with several more sophisticated methods, as well as its computation from two projection matrices.

## 2.2 Factorization Methods

As pointed out before, it is not the goal of this work to compute the parameters of a camera using calibration patterns or mechanical devices. In fact, the intrinsic and motion parameters of a camera are to be estimated from the content of the recorded images themselves. A useful tool to achieve this are the so-called *factorization methods*, which were first proposed in [Tom92], assuming orthographic projection. Over time, these methods were adapted to ever more accurate projection models. A description of the method, as well as some of its variations, will be given in the following. An extensive summary of factorization methods can be found in [Kan98].

Factorization methods require that point correspondences are available for the image sequence in consideration. The detection of point features and their tracking over an image sequence will be addressed in Section 3.2. Given $N$ point features $\boldsymbol{q}_{fn}$ observed in all $F$ frames of an image sequence, all factorization methods make use of a *measurement matrix* $\boldsymbol{W}$, which contains all $\boldsymbol{q}_{fn}$ stacked one above another. For all affine projection models, $\boldsymbol{W}$ is given as

$$\boldsymbol{W} = \begin{pmatrix} \boldsymbol{q}_{11} & \boldsymbol{q}_{12} & \cdots & \boldsymbol{q}_{1N} \\ \boldsymbol{q}_{21} & \boldsymbol{q}_{22} & \cdots & \boldsymbol{q}_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{q}_{F1} & \boldsymbol{q}_{F2} & \cdots & \boldsymbol{q}_{FN} \end{pmatrix} \in \mathbb{R}^{2F \times N} \quad . \tag{2.22}$$

In case of the perspective factorization method homogeneous coordinates are used instead. The following details apply for affine projections only, i. e., orthographic, weak- and paraperspective. The perspective factorization method constitutes a special case where the approach is slightly different. The differences will be explained in Section 2.2.3.

For the affine cases, $\boldsymbol{W}$ is preprocessed further. The origin of the world reference frame is shifted to the centroid of the scene points $\boldsymbol{p}_n$ of each image $f$, corresponding to the feature points $\boldsymbol{q}_{fn}$. In affine projection, the centroid of the world points is projected to the centroid of the feature points in each image. Therefore, it is computed for each image as

$$\overline{\boldsymbol{q}}_f = \frac{1}{N}\sum_{n=1}^{N}\boldsymbol{q}_{fn} \quad . \tag{2.23}$$

Each feature point is thus shifted to

$$\widetilde{\boldsymbol{q}}_{fn} = \boldsymbol{q}_{fn} - \overline{\boldsymbol{q}}_f \quad . \tag{2.24}$$

$\widetilde{\boldsymbol{q}}_{fn}$ is called a *registered* image point, and the *registered measurement matrix* $\widetilde{\boldsymbol{W}}$ is constructed from them analogously to $\boldsymbol{W}$ in (2.22).

After this transformation, the entries $\widetilde{\boldsymbol{q}}_{fn}$ in $\widetilde{\boldsymbol{W}}$ are generated by a simple projection

$$\widetilde{\boldsymbol{q}}_{fn} = \boldsymbol{R}_f\widetilde{\boldsymbol{p}}_n \quad , \tag{2.25}$$

where $\boldsymbol{R}_f$ is a $2 \times 3$ rotation and projection matrix. Thus, if all matrices $\boldsymbol{R}_f$ are stacked upon each other in a matrix $\boldsymbol{\Psi}$, and all 3-D points concatenated in a matrix $\boldsymbol{\Phi}$, all entries in $\widetilde{\boldsymbol{W}}$ are computed together as

$$\widetilde{\boldsymbol{W}} = \boldsymbol{\Psi}\boldsymbol{\Phi} \quad . \tag{2.26}$$

Matrix $\boldsymbol{\Phi}$ is called the *shape* of the scene, while $\boldsymbol{\Psi}$ describes the *motion* of the camera for the image sequence.

The main idea of all factorization methods is that $\widetilde{\boldsymbol{W}}$ can be decomposed into its components $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$ as it is highly rank-deficient by construction. For all affine models such as orthographic, weak- and paraperspective projection, $\boldsymbol{\Psi} \in \mathbb{R}^{2F\times 3}$ and $\boldsymbol{\Phi} \in \mathbb{R}^{3\times N}$. Therefore, $\widetilde{\boldsymbol{W}}$ can be at most of rank 3.

In reality, $\widetilde{\boldsymbol{W}}$ is constructed from points which were extracted from the image sequence by feature tracking and are thus noisy. Due to this noise, its rank will not be three exactly. A decomposition is therefore computed using singular value decomposition (SVD). For an introduction

to singular value decomposition see the appendix of [Hei04] or refer to [Pre93]. It yields

$$\widetilde{\boldsymbol{W}} = \boldsymbol{U}\text{diag}(s_1, s_2, \ldots s_N)\boldsymbol{V}^T \quad , \tag{2.27}$$

where the singular values $s_n$ are sorted in decreasing order. If $\widetilde{\boldsymbol{W}}$ were of rank three, the singular values $s_i$ would be zero for $i > 3$. Since this is usually not the case, only the first three singular values are considered and the rest are discarded. Estimates of $\boldsymbol{\Psi}$ and $\boldsymbol{\Phi}$, $\widehat{\boldsymbol{\Psi}}$ and $\widehat{\boldsymbol{\Phi}}$, are thus computed by

$$\begin{aligned} \widehat{\boldsymbol{\Psi}} &= \boldsymbol{U}'\text{diag}(s_1, s_2, s_3)^{1/2} \quad , \\ \widehat{\boldsymbol{\Phi}} &= \text{diag}(s_1, s_2, s_3)^{1/2}\boldsymbol{V}'^T \quad , \end{aligned} \tag{2.28}$$

where $\boldsymbol{U}'$ and $\boldsymbol{V}'$ are the first three columns of $\boldsymbol{U}$ and $\boldsymbol{V}$ respectively.

This factorization of $\widetilde{\boldsymbol{W}}$ is not unique, since for any invertible $3\times3$ matrix $\boldsymbol{Q}$, the factorization

$$\widetilde{\boldsymbol{W}} = (\widehat{\boldsymbol{\Psi}}\boldsymbol{Q})(\boldsymbol{Q}^{-1}\widehat{\boldsymbol{\Phi}}) \tag{2.29}$$

is also valid. Therefore, $\boldsymbol{Q}$ has to be estimated as well in order to get a unique decomposition. The additional constraints required for the computation of $\boldsymbol{Q}$ differ for each projection model and will be introduced in the respective subsection. Note that the correct factorization can only be obtained up to an unknown similarity transformation, which means that the poses of the camera in the world reference frame are always unknown.

The factorization method for perspective projection is substantially more difficult than for the affine cases. It will be discussed in Section 2.2.3.

### 2.2.1 Orthographic Factorization

For orthographic projection, additional constraints for estimating a unique solution for $\widehat{\boldsymbol{\Psi}}$ and $\widehat{\boldsymbol{\Phi}}$ can be derived from the rotation matrix estimates $\widehat{\boldsymbol{R}}_f$, which form $\widehat{\boldsymbol{\Psi}}$. If $\widehat{\boldsymbol{r}}_{f,1}$ and $\widehat{\boldsymbol{r}}_{f,2}$ denote the two rows of $\widehat{\boldsymbol{R}}_f$, $\boldsymbol{Q}$ has to fulfill the following conditions so that $\boldsymbol{R}_f$ forms the upper two rows of a rotation matrix:

$$\begin{aligned} \widehat{\boldsymbol{r}}_{f,1}\boldsymbol{Q}\boldsymbol{Q}^T\widehat{\boldsymbol{r}}_{f,1}^T &= 1 \quad , \\ \widehat{\boldsymbol{r}}_{f,2}\boldsymbol{Q}\boldsymbol{Q}^T\widehat{\boldsymbol{r}}_{f,2}^T &= 1 \quad , \\ \widehat{\boldsymbol{r}}_{f,1}\boldsymbol{Q}\boldsymbol{Q}^T\widehat{\boldsymbol{r}}_{f,2}^T &= 0 \quad . \end{aligned} \tag{2.30}$$

This means that the rows of $\boldsymbol{R}_f$ have to have unit norm and that they are orthogonal to each other. These constraints are called *metric constraints* since the resulting reconstruction is then unique up to a rotation, a translation, and scaling (cf. Section 2.3.1). They yield $3F$ equations which can be solved for the entries of $\boldsymbol{Q}$. The result is used to compute $\boldsymbol{\Psi} = \widehat{\boldsymbol{\Psi}}\boldsymbol{Q}$ and $\boldsymbol{\Phi} = \boldsymbol{Q}^{-1}\widehat{\boldsymbol{\Phi}}$, which yields a solution that is unique up to an arbitrary rotation. Usually, this ambiguity is solved by setting the rotation of the first camera equal to the rotation of the world coordinate system.

$\boldsymbol{\Psi}$ only contains the rotational element of camera motion. The translation for each frame parallel to the image plane is given by the motion of the point centroid $\overline{\boldsymbol{q}}_f$. Note that the motion along the optical axis of the camera cannot be recovered due to the orthographic projection model.

### 2.2.2 Weak- and Paraperspective Factorization

The weak-perspective and paraperspective extensions of the original factorization method are both introduced in [Poe97]. In the *weak-perspective* case, the projection of a registered scene point to an image feature of equation (2.25) is extended to

$$\widetilde{\boldsymbol{q}}_{fn} = \begin{pmatrix} \frac{f}{\overline{p}_{f,3}} & 0 \\ 0 & \frac{f}{\overline{p}_{f,3}} \end{pmatrix} \boldsymbol{R}_f \widetilde{\boldsymbol{p}}_n = \boldsymbol{R}'_f \widetilde{\boldsymbol{p}}_n \quad . \tag{2.31}$$

Constraints for estimating $\boldsymbol{Q}$ and thus a unique solution are similar to those in equations (2.30). For the rows of $\boldsymbol{R}'_f$, $\boldsymbol{r}'_{f,1}$ and $\boldsymbol{r}'_{f,2}$, it is known that

$$\boldsymbol{r}'_{f,1}{\boldsymbol{r}'_{f,1}}^T = \frac{f^2}{{\overline{p}_{f,3}}^2} \quad , \quad \boldsymbol{r}'_{f,2}{\boldsymbol{r}'_{f,2}}^T = \frac{f^2}{{\overline{p}_{f,3}}^2} \quad . \tag{2.32}$$

Since the ratio of focal length and distance of the object centroid to the optical center, $f^2/{\overline{p}_{f,3}}^2$, is unknown, the equations are set equal to form the first constraint. The second constraint is again derived from the requirement that the rows of $\boldsymbol{R}'_f$ be orthogonal. In order to prevent that the trivial solution $\boldsymbol{Q} = \boldsymbol{0}$ is chosen, the last constraint is that, without loss of generality, $\|\boldsymbol{r}'_{1,1}\| = 1$. Thus, the metric constraints for weak-perspective factorization can be summarized as follows:

$$\begin{aligned} \widehat{\boldsymbol{r}}'_{f,1}\boldsymbol{Q}\boldsymbol{Q}^T{\widehat{\boldsymbol{r}}'_{f,1}}^T &= \widehat{\boldsymbol{r}}'_{f,2}\boldsymbol{Q}\boldsymbol{Q}^T{\widehat{\boldsymbol{r}}'_{f,2}}^T \quad , \\ \widehat{\boldsymbol{r}}'_{f,1}\boldsymbol{Q}\boldsymbol{Q}^T{\widehat{\boldsymbol{r}}'_{f,2}}^T &= 0 \quad , \\ \widehat{\boldsymbol{r}}'_{1,1}\boldsymbol{Q}\boldsymbol{Q}^T{\widehat{\boldsymbol{r}}'_{1,1}}^T &= 1 \quad . \end{aligned} \tag{2.33}$$

After updating $\boldsymbol{\Psi}$ and $\boldsymbol{\Phi}$ using $\boldsymbol{Q}$, the rotation for each frame is computed as

$$\widehat{\boldsymbol{r}}'_{f,1} = \frac{\boldsymbol{r}'_{f,1}}{\|\boldsymbol{r}'_{f,1}\|} \quad , \quad \widehat{\boldsymbol{r}}'_{f,2} = \frac{\boldsymbol{r}'_{f,2}}{\|\boldsymbol{r}'_{f,2}\|} \quad . \tag{2.34}$$

Unlike in the orthographic case, the distance of the object along the optical axis, $\overline{p}_{f,3}$, can now be computed from equations (2.32). Since the resulting value may differ for the two equations, the average of the two is used to calculate $\overline{p}_{f,3}$ as

$$\overline{p}_{f,3} = \frac{1}{2f}\left(\frac{1}{\|\boldsymbol{r}'_{f,1}\|} + \frac{1}{\|\boldsymbol{r}'_{f,2}\|}\right) \quad . \tag{2.35}$$

If the focal length of the camera is unknown it is often set to 1. However, this results in a severe distortion of the final reconstruction. Therefore, it is advisable to either estimate the focal length using the self-calibration algorithm based on the essential matrix introduced later in Section 2.3.2, or to use an approximate value which is close to the correct one and to optionally perform the self-calibration based on the absolute quadric afterwards.

For the *paraperspective case* the metric constraints are more sophisticated. As described in Section 2.1.2, points are now projected onto a plane parallel to the image plane through $\boldsymbol{p}_g$ along the line connecting $\boldsymbol{p}_g$ and the camera center. Of $\boldsymbol{p}_g$, the projection into each image $\overline{\boldsymbol{q}}_f$ is known. Accordingly, equation (2.32) is extended to

$$\boldsymbol{r}''_{f,1}{\boldsymbol{r}''_{f,1}}^T = \frac{f^2 + f^2\overline{p}_{f,1}^2}{{\overline{p}_{f,3}}^2} \quad , \quad \boldsymbol{r}''_{f,2}{\boldsymbol{r}''_{f,2}}^T = \frac{f^2 + f^2\overline{p}_{f,2}^2}{{\overline{p}_{f,3}}^2} \quad . \tag{2.36}$$

Again, these two equations are set to equality to form the first constraint, i. e.

$$\frac{\boldsymbol{r}''_{f,1}{\boldsymbol{r}''_{f,1}}^T}{1+\overline{p}_{f,1}^2} = \frac{\boldsymbol{r}''_{f,2}{\boldsymbol{r}''_{f,2}}^T}{1+\overline{p}_{f,2}^2} = \left(\frac{f^2}{{\overline{p}_{f,3}}^2}\right) \quad . \tag{2.37}$$

The second constraint is formed by an angle relationship of $\boldsymbol{r}''_{f,1}$ and $\boldsymbol{r}''_{f,2}$, which is

$$\boldsymbol{r}''_{f,1}{\boldsymbol{r}''_{f,2}}^T = \frac{f^2\overline{p}_{f,1}\overline{p}_{f,2}}{{\overline{p}_{f,3}}^2} \quad . \tag{2.38}$$

Since $f^2/{\overline{p}_{f,3}}^2$ is still unknown, it is substituted by the mean of the two possible values given in equation (2.37)—analogously to equation (2.35). The third constraint is the same as in the weak-perspective case, i. e. $\|\boldsymbol{r}''_{1,1}\| = 1$. In order to calculate $\boldsymbol{Q}$, equations (2.37) and (2.38) are

again reformulated to

$$\frac{\widehat{\boldsymbol{r}}''_{f,1}\boldsymbol{Q}\boldsymbol{Q}^T{\widehat{\boldsymbol{r}}''_{f,1}}^T}{1+\overline{p}^2_{f,1}} - \frac{\widehat{\boldsymbol{r}}''_{f,2}\boldsymbol{Q}\boldsymbol{Q}^T{\widehat{\boldsymbol{r}}''_{f,2}}^T}{1+\overline{p}^2_{f,2}} = 0 \quad , \tag{2.39}$$

$$\widehat{\boldsymbol{r}}''_{f,1}\boldsymbol{Q}\boldsymbol{Q}^T{\widehat{\boldsymbol{r}}''_{f,2}}^T = \frac{\overline{p}_{f,1}\overline{p}_{f,2}}{2}\left(\frac{\widehat{\boldsymbol{r}}''_{f,1}\boldsymbol{Q}\boldsymbol{Q}^T{\widehat{\boldsymbol{r}}''_{f,1}}^T}{1+\overline{p}^2_{f,1}} + \frac{\widehat{\boldsymbol{r}}''_{f,2}\boldsymbol{Q}\boldsymbol{Q}^T{\widehat{\boldsymbol{r}}''_{f,2}}^T}{1+\overline{p}^2_{f,2}}\right) \quad . \tag{2.40}$$

The third constraint is the same as in equation (2.33).

Computing the camera rotations after correcting $\boldsymbol{\Psi}$ with $\boldsymbol{Q}$ is similar, but somewhat more complicated as in the weak-perspective case. For a detailed description the reader is referred to [Poe97].

### 2.2.3 Perspective Factorization Methods

The factorization methods explained in Sections 2.2.1 and 2.2.2 are all based on the assumption of an affine projection mechanism, namely orthographic, weak- or paraperspective projection. While they are mathematically easy to handle, their common drawback is that affine projection is only an approximation to the more accurate perspective projection of the pinhole camera. Therefore, two factorization methods based on the assumption of perspective projection will be introduced in the following.

#### Perspective factorization by Sturm and Triggs

The factorization method by Sturm and Triggs [Stu96] is a straightforward approach to perspective factorization. Consider the projection of a homogeneous 3-D point into an image given in equation (2.11). Since $\underline{\boldsymbol{q}}$ and $\underline{\boldsymbol{p}}$ are given in homogeneous coordinates, their scaling is arbitrary for each individual point. However, combined in a measurement matrix, only the scaling for each image and each 3-D point respectively remains arbitrary. The scaling for each 2-D point is then fixed and has to be given in the projection equation. Thus, equation (2.11) has to be extended to

$$\mu_{fn}\underline{\boldsymbol{q}}_{fn} = \boldsymbol{P}_f\underline{\boldsymbol{p}}_n \quad . \tag{2.41}$$

$\mu_{fn}$ is called the *projective depth* of point $n$ in image $f$.

The projective measurement matrix $\boldsymbol{W}_p \in \mathbb{R}^{3F\times N}$ is now constructed similarly to (2.22), but

incorporating the projective depths and using homogeneous coordinates:

$$\boldsymbol{W}_p = \begin{pmatrix} \mu_{11}\underline{\boldsymbol{q}}_{11} & \mu_{12}\underline{\boldsymbol{q}}_{12} & \cdots & \mu_{1N}\underline{\boldsymbol{q}}_{1N} \\ \mu_{21}\underline{\boldsymbol{q}}_{21} & \mu_{22}\underline{\boldsymbol{q}}_{22} & \cdots & \mu_{2N}\underline{\boldsymbol{q}}_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \mu_{F1}\underline{\boldsymbol{q}}_{F1} & \mu_{F2}\underline{\boldsymbol{q}}_{F2} & \cdots & \mu_{FN}\underline{\boldsymbol{q}}_{FN} \end{pmatrix} = \begin{pmatrix} \boldsymbol{P}_1 \\ \boldsymbol{P}_2 \\ \vdots \\ \boldsymbol{P}_F \end{pmatrix} \begin{pmatrix} \underline{\boldsymbol{p}}_1, \underline{\boldsymbol{p}}_2, \cdots, \underline{\boldsymbol{p}}_N \end{pmatrix} \quad . \tag{2.42}$$

Thus, by construction, the measurement matrix has rank $4$ instead of rank $3$ as in the affine case.

Prior to proceeding to the actual factorization, the question has to be solved how the projective depths can be calculated. A fairly simple solution is to first estimate the fundamental matrices $\boldsymbol{F}_{ij}$ (see Section 2.1.3) for all pairs of consecutive images $i$ and $j$. The respective epipole in the first image of a pair is denoted by $\underline{\boldsymbol{e}}_{ij}$. Next, the following equation is used to relate the projective depths of one point $\underline{\boldsymbol{p}}_n$ in images $i$ and $j$:

$$\mu_{in} = \frac{(\underline{\boldsymbol{e}}_{ij} \times \underline{\boldsymbol{q}}_{in}) \cdot (\boldsymbol{F}_{ij}\underline{\boldsymbol{q}}_{jn})}{\|\underline{\boldsymbol{e}}_{ij} \times \underline{\boldsymbol{q}}_{in}\|^2}\mu_{jn} \quad . \tag{2.43}$$

The projective depths for each point in all images are thus calculated pairwise for all consecutive image pairs, starting each time with $\mu_{1n} = 1$. Derivations of the above relation can be found in [Stu96] and [Hei04].

As stated before, the projective depths of each 3-D point, which corresponds to a column in $\boldsymbol{W}_p$, and for each image, a triplet of rows in $\boldsymbol{W}_p$, may be scaled arbitrarily. This has only an effect on the scaling of the corresponding 3-D point $\underline{\boldsymbol{p}}_n$ or the projection matrix $\boldsymbol{P}_f$, respectively. In order to prevent that the size of the projective depths differs considerably and thus leads to numerical instabilities, the measurement matrix is *weighted* before factorization. Hereby, each column of $\boldsymbol{W}_p$ and each triplet of rows of $\boldsymbol{W}_p$ is rescaled so that its norm is equal to one. If necessary, the rescaling steps are repeated until the entries in $\boldsymbol{W}_p$ do not change significantly any more. A considerable improvement can also be achieved if the coordinates of the image points are normalized according to [Har97], i. e., the point coordinates in each image are shifted and scaled such that their mean is the origin of the coordinate system and their average distance from the origin is $\sqrt{2}$.

Having done all these preparations, $\boldsymbol{W}_p$ can finally be factorized as in equations (2.27) and (2.28), but considering that the rank of $\boldsymbol{W}_p$ is now $4$. Like in the affine case, the decomposition of $\boldsymbol{W}_p$ into $\widehat{\boldsymbol{\Psi}}_p$ and $\widehat{\boldsymbol{\Phi}}_p$ is not unique, but there exists a $4 \times 4$ projective transformation matrix $\boldsymbol{Q}_p$, analogously to equation (2.29), which also leads to a valid solution, $\widehat{\boldsymbol{\Psi}}_p\boldsymbol{Q}_p$ and $\boldsymbol{Q}_p^{-1}\widehat{\boldsymbol{\Phi}}_p$. However, unlike in the affine case, no constraints exist to calculate the correct transformation

$\boldsymbol{Q}_p$, so that a subsequent self-calibration step as described in the Section 2.3 is necessary.

One drawback of the above strategy for calculating the projective depths is that it is prone to error accumulation or propagation in the pair-wise fundamental matrix computations. Therefore, Heigl [Hei04] proposes to use the depth of 3-D points calculated by a preceding weak- or paraperspective factorization. The projective depths then correspond to the distance of the 3-D points along the optical axis from the projection center of the camera.

### Iterative factorization by Christy and Horaud

The idea behind the iterative factorization by Christy and Horaud [Chr96] is to perform one of the affine factorizations of Section 2.2.2 in each iteration and to gradually adapt the 2-D image features to finally fit to a perspective projection. In principle, any affine reconstruction method can be used. However, the algorithm is shown here only for an underlying paraperspective factorization, while it works similarly for the weak-perspective factorization. In [Chr94], the use of further affine reconstruction methods is shown.

Assuming that the intrinsic camera parameters, focal length and principal point, for each camera are known, the image points $\boldsymbol{q}$ are used in the following in camera coordinates:

$$x_c = \frac{q_1 - u_{pp}}{f_x} \quad , \tag{2.44}$$

$$y_c = \frac{q_2 - v_{pp}}{f_y} \quad . \tag{2.45}$$

Considering now the transformation of a point in world coordinates into camera coordinates (cf. equation (2.1)) and its projection to image points, $x_c$ and $y_c$ are given as

$$x_c = \frac{\boldsymbol{r}_1{}^{\mathrm{T}}\boldsymbol{p} + t_1}{\boldsymbol{r}_3{}^{\mathrm{T}}\boldsymbol{p} + t_3} \quad , \tag{2.46}$$

$$y_c = \frac{\boldsymbol{r}_2{}^{\mathrm{T}}\boldsymbol{p} + t_2}{\boldsymbol{r}_3{}^{\mathrm{T}}\boldsymbol{p} + t_3} \quad . \tag{2.47}$$

The vectors $\boldsymbol{r}_i$ denote the columns of rotation matrix $\boldsymbol{R}$ in equation (2.1), $t_i$ the elements of the translation vector $\boldsymbol{t}$.

In order to simplify these equations, both numerator and denominator on the right side are divided by $t_3$, and the following terms are combined: $\boldsymbol{i} = \boldsymbol{r}_1{}^{\mathrm{T}}/t_3$, $\boldsymbol{j} = \boldsymbol{r}_2{}^{\mathrm{T}}/t_3$, $x_0 = t_1/t_3$ and $y_0 = t_2/t_3$ as the projection of the camera coordinate center, and finally $\xi = (\boldsymbol{r}_3{}^{\mathrm{T}}\boldsymbol{p})/t_3$. However, in the following only the equations for the projection in $x$-coordinates will be given, since the equations are formed analogously for the $y$-direction. With these new denotations,

equation (2.46) for perspective projection changes to

$$x_c = \frac{\boldsymbol{ip} + x_0}{1 + \xi} \quad . \tag{2.48}$$

In this notation, the linear, paraperspective approximation to perspective projection only differs in the use of $\xi$. If the object is relatively far away from the camera, i. e., $t_3$ is large compared to $(\boldsymbol{r}_3{}^{\mathrm{T}}\boldsymbol{p})$, the following first-order approximation can be applied:

$$\frac{1}{1 + \xi} \approx 1 - \xi \quad . \tag{2.49}$$

Thus, the equation for paraperspective projection is

$$x_c \approx (\boldsymbol{ip} + x_0)(1 - \xi) \approx \boldsymbol{ip} + x_0 - x_0\xi = x_p \quad , \tag{2.50}$$

where the term $\boldsymbol{ip}\xi$ is neglected. By solving the right-hand, paraperspective part of the equation for $\boldsymbol{ip}$ and inserting the result into the original, perspective equation (2.48), a relationship between perspective and paraperspective projection is found as

$$x_p = x_c(1 + \xi) - x_0\xi \quad . \tag{2.51}$$

Considering $x_c$, and analogously $y_c$, as the perspective projections of a world point $\boldsymbol{p}$, and $x_c(1+\xi) - x_0\xi$ and $y_c(1+\xi) - y_0\xi$ as the paraperspective projection of the same world point, the problem of perspective reconstruction can now be reformulated as incrementally finding suitable $\xi_{fn}$ for the projections of $N$ world points in $F$ images by paraperspective reconstructions. These reconstructions are performed by paraperspective factorization according to Section 2.2.2. A modified measurement matrix $\boldsymbol{W}_{\mathrm{CH}}$ is used where the entries $\boldsymbol{q}_{fn}$ are given as

$$\boldsymbol{q}_{fn} = \begin{pmatrix} (x_{c,fn} - x_{0,f})(1 + \xi_{fn}) \\ (y_{c,fn} - y_{0,f})(1 + \xi_{fn}) \end{pmatrix} \quad . \tag{2.52}$$

The final iterative algorithm is summarized in Figure 2.6. For the first iteration, the $\xi_{fn}$ are initialized with zero, which is equivalent to using the original, unaltered image point coordinates. In each iteration step, $\boldsymbol{W}_{\mathrm{CH}}$ is built using the entries given in equation (2.52) and the current $\xi_{fn}$. The metric results of a paraperspective factorization are then used to estimate the new set of $\xi_{fn}$, thus altering the image coordinates of the projected feature points. The algorithm has converged when the $\xi_{fn}$ do not change anymore. The resulting image features yield the

| | |
|---|---|
| Initialize $\xi_{fn}^{(0)} = 0$, with $1 \leq f \leq F, 1 \leq n \leq N$ | |
| | Set up ${\boldsymbol{W}_{\text{CH}}}^{(i)}$ using the current $\xi_{fn}^{(i)}$ for each entry in equation (2.52) |
| | Perform paraperspective factorization according to Section 2.2.2 using ${\boldsymbol{W}_{\text{CH}}}^{(i)}$ |
| | Estimate new $\xi_{fn}^{(i+1)}$ |
| UNTIL $\forall f, n$: $\xi_{fn}^{(i+1)} \approx \xi_{fn}^{(i)}$ | |

Figure 2.6: The iterative factorization algorithm by Christy and Horaud

perspective reconstruction of camera parameters and 3-D point positions after the application of a paraperspective factorization to them.

Like the other factorization methods introduced in this section, the iterative factorization likewise yields a reconstruction which is only unique up to a projective transformation if wrong internal parameters are used. The next section will give an introduction into the different types of reconstructions possible and how their ambiguities can be resolved using self-calibration.

## 2.3 Camera Self-Calibration

Camera self-calibration, which is also often called auto-calibration, denotes the process of estimating the internal parameters of a camera solely from cues in images which were taken with this camera and which show, above all, no special markers such as a calibration patterns. In other words, the calibration of the camera is done on the fly while recording a natural scene and not, as is often done, before the actual utilization of the camera. On the other hand, self-calibration does not include the estimation of the external parameters of a moving camera, which is usually referred to as *pose estimation* and was already introduced in the previous sections.

A short overview over different self-calibration methods was already given in Section 1.2.3. In the following, the methods applied for the experimental evaluations will be introduced in more detail. The next section in particular deals with the different kinds of ambiguities which are removed by self-calibration.

### 2.3.1 Reconstruction Types

The algorithms for camera pose reconstruction from two or more images described in the previous section, such as the fundamental matrix and the various factorization methods, are only able to solve the problem up to a certain ambiguity. In general, if a projection matrix $\boldsymbol{P}_i$ is found for some image, it is not a unique solution, but there always exists a transformation matrix $\boldsymbol{D}$, so

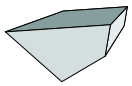
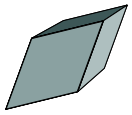
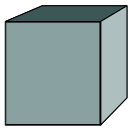
| Type | Ambiguity | Example | DOF | Preserved Properties |
|---|---|---|---|---|
| Projective | $\begin{pmatrix} \boldsymbol{A} & \boldsymbol{t} \\ \boldsymbol{x}^{\mathrm{T}} & x \end{pmatrix}$ |  | 15 | cross ratio, tangency, intersection, collinearity |
| Affine | $\begin{pmatrix} \boldsymbol{A} & \boldsymbol{t} \\ \boldsymbol{0}_3{}^{\mathrm{T}} & 1 \end{pmatrix}$ |  | 12 | parallelism, centroids, area ratio, volume ratio, plane at infinity $\underline{\boldsymbol{m}}_\infty$ |
| Metric | $\begin{pmatrix} \check{s}\boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0}_3{}^{\mathrm{T}} & 1 \end{pmatrix}$ |  | 7 | angles, relative distances, absolute conic $\boldsymbol{C}_a$, absolute quadric $\boldsymbol{Q}_a$ |
| Euclidean | $\begin{pmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0}_3{}^{\mathrm{T}} & 1 \end{pmatrix}$ |  | 6 | absolute distances, area, volume |

Table 2.1: The most important types of transformations in 3-D reconstruction [Pol99, Fau01, Har03]: From projective to Euclidean the degrees of freedom (DOF) decrease while more properties are preserved. For each transformation, the properties of the preceding one are preserved as well. The elements of the ambiguity matrices given in the second column have to be chosen such that the complete matrix is still invertible and are defined as follows: $\boldsymbol{A} \in \mathbb{R}^{3\times3}$ is an arbitrary matrix, $\boldsymbol{x} \in \mathbb{R}^3$ is an arbitrary vector, $x \in \mathbb{R}$ is an arbitrary scalar value; $\boldsymbol{R} \in \mathbb{R}^{3\times3}$ is a rotation matrix, $\boldsymbol{t} \in \mathbb{R}^3$ a translation vector, and $\check{s} \in \mathbb{R}$ is a non-zero scale factor. Table and images by courtesy of J. Schmidt [Sch06].

that for the projection of a 3-D point $\underline{\boldsymbol{p}}_n$ into image $i$ the following equation holds:

$$\underline{\boldsymbol{q}}_{i,n} = \boldsymbol{P}_i \underline{\boldsymbol{p}}_n = (\boldsymbol{P}_i \boldsymbol{D})(\boldsymbol{D}^{-1} \underline{\boldsymbol{p}}_n) \quad . \tag{2.53}$$

The structure of $\boldsymbol{D}$ depends on the algorithm used to estimate the projection matrix. In the most general case the $4 \times 4$ matrix $\boldsymbol{D}$ has 15 degrees of freedom (DOF) and thus constitutes a projective transformation. An example for such a reconstruction is generated by the perspective factorization method in Section 2.2.3. The effect of this ambiguity is that the reconstruction is severely skewed—in fact, only few of the properties of a scene are preserved, as shown in Table 2.1—and further processing is necessary, especially for applications in light field rendering.

The degrees of freedom of the transformation matrix can be reduced by assuming some properties of the scene or the recording camera. Thus, the reconstruction can be upgraded from projective to affine and metric. In the process, more information about the intrinsic camera pa-

rameters are acquired as well. This upgrade is the result of self-calibration and will be detailed in the next section. Table 2.1 summarizes the properties of the transformation matrix $\boldsymbol{D}$ for different types of reconstruction.

However, even if intrinsic camera parameters are known or can be estimated by self-calibration, the ambiguity can only be reduced to a metric transformation, i. e., a similarity transformation consisting of rotation, translation and scale. A Euclidean reconstruction can only be achieved if at least one distance in the scene is known.

More detailed surveys of the possible types of reconstruction can be found in the publications of Hartley [Har03], Faugeras [Fau01] and Pollefeys [Pol99]. One issue that frequently causes some confusion is that the terms *metric* and *Euclidean* reconstruction are not used consistently in the literature. The denotation chosen here, i. e., that a Euclidean reconstruction is unique only up to a Euclidean transformation, is consistent with the denotation in [Har03], [Pol99] and [Sch06], whereas Faugeras uses the reverse denotation.

### 2.3.2 Self-Calibration Methods

In Section 2.2.3 it was stated that after a perspective factorization, the reconstruction of camera parameters and 3-D points is only given up to a projective transformation $\boldsymbol{D}$. Affine factorization methods require the knowledge of some intrinsic parameters to result in an accurate reconstruction. If these intrinsic parameters are estimated wrongly, the reconstruction will be likewise skewed. Of the two self-calibration methods introduced in the following, the first one by Mendonça and Cipolla [Men99] requires only the knowledge of the fundamental matrices $\boldsymbol{F}_{i,j}$ between all pairs of images to estimate the intrinsic parameters. It is thus suitable to be applied before an affine factorization. The second one makes use of skewed projection matrices $\boldsymbol{P}_i$ and properties of the absolute quadric $\boldsymbol{Q}_a$ to correct a projective reconstruction via the transformation $\boldsymbol{D}$. This method was introduced in [Tri97], but is explained in greater detail in [Har03] and [Pol99].

#### Self-calibration using the essential matrix

Given two calibration matrices $\boldsymbol{K}_i$ and $\boldsymbol{K}_j$, as defined in equation (2.9), for two different images, as well as the fundamental matrix $\boldsymbol{F}_{ij}$ linking these two images, the so-called essential matrix is defined as

$$\boldsymbol{E}_{ij} = {\boldsymbol{K}_j}^{\mathrm{T}} \boldsymbol{F}_{ij} \boldsymbol{K}_i \quad . \tag{2.54}$$

The one property of the $3\times3$ matrix $\boldsymbol{E}_{ij}$ that can be exploited for self-calibration is that it is of

rank 2 if the calibration matrices are correct. Thus, the objective is, given fundamental matrices $\boldsymbol{F}_{ij}$ between pairs of images, to find calibration matrices $\boldsymbol{K}_i$ and $\boldsymbol{K}_j$ such that $\mathrm{rank}(\boldsymbol{E}_{ij}) = 2$, with $1 \leq i, j \leq F$, $i \neq j$. The number of images $F$ required for a unique solution depends on the number of parameters that are to be estimated or considered known or fixed, respectively. A detailed calculation can be found in [Pol99].

The self-calibration process can now be formulated as a minimization problem of an error function:

$$\epsilon(\boldsymbol{K}_i) = \sum_{i=1}^{F} \sum_{j=i+1}^{F} \frac{w_{ij}}{\sum_{k=1}^{F} \sum_{l=k+1}^{F} w_{kl}} \frac{s_{ij,1} - s_{ij,2}}{s_{ij,2}} \quad . \tag{2.55}$$

$s_{ij,1}$ and $s_{ij,2}$ denote the first two singular values of essential matrix $\boldsymbol{E}_{ij}$. The weight parameters $w_{ij}$ can be used to assign confidence values to the different fundamental matrices, or can be set to 1 uniformly. The minimum of this error function, which is smooth according to the Wielandt-Hoffman theorem [Gol96], is found using numeric optimization and, specifically, the Nelder-Mead simplex algorithm [Nel65].

### Self-calibration using the absolute quadric

In order to explain this self-calibration method, a few new terms have to be introduced first. The *absolute dual quadric* is a special, degenerate case of a quadric which represents a smooth surface in a three-dimensional projective space $\mathbb{P}_3$, which is the results of using homogeneous coordinates instead of Euclidean coordinates in $\mathbb{R}_3$. The absolute dual quadric $\boldsymbol{Q}_a^*$ is given by a symmetric, homogeneous $4 \times 4$ matrix of rank 3. Its projection into an image using a projection matrix $\boldsymbol{P}$ is the *absolute dual conic* $\boldsymbol{C}_a^*$, a symmetric, homogeneous $3 \times 3$ matrix in two-dimensional projective space $\mathbb{P}_2$. The projection is given as

$$\boldsymbol{C}_a^* = \boldsymbol{P}\boldsymbol{Q}_a^*\boldsymbol{P} \quad . \tag{2.56}$$

Absolute dual quadric and absolute dual conic have two important properties which render them useful for self-calibration. Firstly, in a metric frame, which is the desired type of reconstruction, the absolute dual quadric is of its canonic form $\overline{\boldsymbol{Q}}_a^*$, being

$$\overline{\boldsymbol{Q}}_a^* = \begin{pmatrix} \boldsymbol{I}_{3\times3} & \boldsymbol{0}_3 \\ \boldsymbol{0}_3{}^{\mathrm{T}} & 0 \end{pmatrix} \quad . \tag{2.57}$$

In a projective frame, this quadric is likewise skewed by a projective transformation $\boldsymbol{D}$, so that $\boldsymbol{Q}_a^* = \boldsymbol{D}\overline{\boldsymbol{Q}}_a^*\boldsymbol{D}^{\mathrm{T}}$. Secondly, $\boldsymbol{C}_a^*$ is given as $\boldsymbol{C}_a^* = \boldsymbol{K}\boldsymbol{K}^{\mathrm{T}}$, $\boldsymbol{K}$ being the calibration matrix corre-

sponding to the projection matrix $\boldsymbol{P}$.

There are several ways to solve for $\boldsymbol{Q}_a^*$ and thus finding $\boldsymbol{D}$ using these equations. A linear and a non-linear approach, which can be applied successively, as well as a simplification will be introduced.

**Linear approach.** If, e. g., a projective reconstruction of an image sequence by perspective factorization is given, the transformation $\boldsymbol{D}$ is the same for each image. Thus, using equation (2.56) and some assumptions on the calibration matrix $\boldsymbol{K}$, constraints can be found on the entries of $\boldsymbol{Q}_a^*$. Assuming the general definition of $\boldsymbol{K}$ from equation (2.9), $\boldsymbol{C}_a^* = \boldsymbol{K}\boldsymbol{K}^{\mathrm{T}}$ calculates as

$$\boldsymbol{C}_a^* = \begin{pmatrix} {f_x}^2 + \beta^2 + {u_{pp}}^2 & \beta f_y + u_{pp} v_{pp} & u_{pp} \\ \beta f_y + u_{pp} v_{pp} & {f_y}^2 + {v_{pp}}^2 & v_{pp} \\ u_{pp} & v_{pp} & 1 \end{pmatrix} \quad . \tag{2.58}$$

If it is further assumed that the image skew $\beta$ is zero, a first constraint $c_{12}^* c_{33}^* = c_{13}^* c_{23}^*$, which holds likewise for the entries of $\boldsymbol{P}\boldsymbol{Q}_a^*\boldsymbol{P}$, is given. It defines one linear equation per image for the entries of $\boldsymbol{Q}_a^*$. More equations become available if the principal point is set to the image center, i. e., $u_{pp} = v_{pp} = 0$, and the aspect ratio $f_x/f_y$ is known. Thus, up to four equations per image can be defined.

In general, the symmetric matrix $\boldsymbol{Q}_a^*$ has nine degrees of freedom and thus nine unique entries if it is scaled such that $q_{33}^* = 1$. However, if all constraints on the intrinsic parameters are applied, the absolute dual conic $\boldsymbol{C}_a^*$ of equation (2.58) is reduced to a diagonal matrix, and likewise $\boldsymbol{Q}_a^*$ will then be given as

$$\boldsymbol{Q}_a^* = \begin{pmatrix} q_{11}^* & 0 & 0 & q_{14}^* \\ 0 & q_{11}^* & 0 & q_{24}^* \\ 0 & 0 & 1 & q_{34}^* \\ q_{14}^* & q_{24}^* & q_{34}^* & q_{44}^* \end{pmatrix} \quad . \tag{2.59}$$

Thus, in this simplified version, only five parameters remain to be estimated.

After solving the corresponding linear system of equations, the additional constraint that the rank of $\boldsymbol{Q}_a^*$ is three is enforced using singular value decomposition and setting the last singular value to zero. The resulting estimate for $\boldsymbol{Q}_a^*$ is finally decomposed into $\boldsymbol{Q}_a^* = \boldsymbol{D}\overline{\boldsymbol{Q}}_a^*\boldsymbol{D}^{\mathrm{T}}$ using eigenvalue decomposition. The reconstruction can then be upgraded to metric using $\boldsymbol{D}$ according to equation (2.53).

**Non-linear approach.** Having obtained a solution for $\boldsymbol{Q}_a^*$ from the above linear approach, it can be further refined using a non-linear optimization method such as the Levenberg-Marquardt algorithm. A suitable error function is given by

$$\epsilon_{\text{SC}} = \sum_{f=1}^{F} \left\| \frac{\boldsymbol{K}_f \boldsymbol{K}_f{}^{\text{T}}}{\|\boldsymbol{K}_f \boldsymbol{K}_f{}^{\text{T}}\|} - \frac{\boldsymbol{P}_f \boldsymbol{Q}_a^* \boldsymbol{P}_f{}^{\text{T}}}{\|\boldsymbol{P}_f \boldsymbol{Q}_a^* \boldsymbol{P}_f{}^{\text{T}}\|} \right\| \quad . \tag{2.60}$$

The parameters to be adjusted are the entries of $\boldsymbol{K}_f$ and $\boldsymbol{Q}_a^*$.

A major drawback pertaining the self-calibration using the absolute quadric is that the estimated quadric $\boldsymbol{Q}_a^*$ has to be positive definite. Otherwise, an eigenvalue decomposition into $\boldsymbol{D}\overline{\boldsymbol{Q}}_a^*\boldsymbol{D}^{\text{T}}$ is not possible. However, it is not advisable to enforce this constraint as this would introduce additional errors. Therefore, the self-calibration may frequently fail at this point, especially since the input data in real applications is usually distorted by noise.

### 2.3.3 Bundle Adjustment

Once a reconstruction has been computed by any of the methods described above, bundle adjustment is often used to do a final refinement of the result. In this technique, the camera parameters and 3-D points are adjusted simultaneously to minimize the error made by back-projecting the points into the images and comparing them with the original feature measurements. For $F$ projection matrices $\widehat{\boldsymbol{P}}_f$, $1 \leq f \leq F$, and $N$ 3-D points $\underline{\widehat{\boldsymbol{p}}}_n$, $1 \leq n \leq N$, each point is back-projected into each image as $\underline{\widehat{\boldsymbol{q}}}_{fn} = \widehat{\boldsymbol{P}}_f \underline{\widehat{\boldsymbol{p}}}_n$, and the residual vector $\boldsymbol{\epsilon}_{\text{B}}$ is calculated from the non-homogeneous 2-D points as

$$\boldsymbol{\epsilon}_{\text{B}} = \left( (\boldsymbol{q}_{11} - \widehat{\boldsymbol{q}}_{11})^{\text{T}}, (\boldsymbol{q}_{21} - \widehat{\boldsymbol{q}}_{21})^{\text{T}}, \ldots, (\boldsymbol{q}_{FN} - \widehat{\boldsymbol{q}}_{FN})^{\text{T}} \right)^{\text{T}} \quad . \tag{2.61}$$

The vector $\boldsymbol{\epsilon}_{\text{B}}$ has $2FN$ entries if each point was visible in each image, and fewer if the points are not visible in each image. Parameter estimation is performed by minimizing the sum of squared differences $\boldsymbol{\epsilon}_{\text{B}}{}^{\text{T}}\boldsymbol{\epsilon}_{\text{B}}$. The number of parameters to be estimated depends again on assumptions made on the intrinsic parameters. Referring to equations (2.9) and (2.10), each $\widehat{\boldsymbol{P}}_f$ has 11 degrees of freedom as it depends on the entries in $\widehat{\boldsymbol{K}}_f$, $\widehat{\boldsymbol{R}}_f$ and $\widehat{\boldsymbol{t}}_f$. In this case, the total number of parameters is $11F + 3N$. By making the same assumptions on skew $\beta$, principal point $(u_{pp}, v_{pp})$ and focal length $(f_x, f_y)$ as in the previous section, the number of parameters can be reduced accordingly. If the intrinsic parameters are assumed to be correct, only $6F + 3N$ parameters remain to be estimated.

A short note on the representation of rotation matrices has to be made here. Although a rotation matrix $\boldsymbol{R}$ has nine entries, it has only three degrees of freedom due to the orthonormality of its columns. Therefore, it can be represented using only three variables. Common representations are Euler angles, axis/angle and quaternions. In [Sch01c], these alternatives are described and compared with respect to bundle adjustment. Since for Euler angles singularities exist which cause numerical instabilities, one of the latter two alternatives is chosen wherever rotations in 3-D are processed in this thesis, their performances being very similar to each other.

As for bundle adjustment, estimating such a large number of parameters takes considerable amounts of time. In case of the Levenberg-Marquardt algorithm, the calculation of a Jacobian matrix by numerical differentiation and its inversion are necessary in each iteration, where the Jacobian matrix $\boldsymbol{J}$ is of size $(2FN) \times (11F + 3N)$. Calculating the pseudo-inverse of $\boldsymbol{J}$ using singular value decomposition has a complexity of $O(F^3N + F^2N^2 + FN^3)$. Therefore, two simplifications of the original bundle adjustment have been introduced which reduce the complexity of the problem: interleaved bundle adjustment and the exploitation of a special block structure of the Jacobian $\boldsymbol{J}$.

In interleaved bundle adjustment [Shu99, Har03], optimization of the camera parameters and 3-D points are separated from each other. First, the $11F$ camera parameters are optimized while the positions of the 3-D points are kept fixed. After each iteration of this "outer" loop, the 3-D points are optimized while keeping the camera parameters fixed. The advantage of this "inner" loop is that each 3-D point position can be optimized independently from the other points, as they do not influence each other's back-projection error. Thus, instead of having to calculate one large Jacobian matrix, a large number of small ones, i. e., $N$ matrices of size $2 \times 3$, suffices. This reduces the overall complexity of pseudo-inversion to $O(F^3N)$. According to [Har03], interleaved bundle adjustment yields the same result as the basic approach as it still minimizes the same cost function. However, this is only true if both approaches converge to the same minimum, which is usually not the case in practice. Additionally, interleaved bundle adjustment requires more iterations for convergence.

For the outer iteration loop of optimizing the camera parameters of each image, the Jacobian matrix $\boldsymbol{J}_c$ has a special block structure. Since the camera parameters for each image are inde-

pendent from each other, $\boldsymbol{J}_c$ is block-diagonal, and each block can be inverted independently:

$$\boldsymbol{J}_c^+ = \begin{pmatrix} \boldsymbol{J}_1 & 0 & \cdots & 0 \\ 0 & \boldsymbol{J}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \boldsymbol{J}_F \end{pmatrix}^+ = \begin{pmatrix} \boldsymbol{J}_1^+ & 0 & \cdots & 0 \\ 0 & \boldsymbol{J}_2^+ & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \boldsymbol{J}_F^+ \end{pmatrix} . \tag{2.62}$$

Here, $\boldsymbol{J}_c^+$ denotes the pseudo-inverse of $\boldsymbol{J}_c$. Each block matrix $\boldsymbol{J}_f$ is of size $2N \times 11$ so that the complexity of computing the pseudo-inverse of $\boldsymbol{J}_c$ reduces to $O(FN)$ [Hei04].

Obviously, error minimization by bundle adjustment is a time consuming process since a large number of parameters have to be optimized at once. However, if time is not an issue, bundle adjustment usually yields the best reconstruction results if a suitable initialization was given. A number of methods have been proposed to further speed up bundle adjustment, such as *virtual key frames* [Shu99]. However, they are not included here as bundle adjustment is not a main focus of this thesis. A comprehensive summary of improvements on bundle adjustment, such as interleaving and exploitation of block-diagonal Jacobi matrices, is given in [Tri00].

# Chapter 3

# Reconstruction of Static Light Fields

The reconstruction of a light field from an input image sequence requires a number of processing steps. Starting from the extraction of point features, the camera parameters for each image have to be estimated and scene geometry information has to be generated. The following chapter will give a detailed view of the reconstruction process for static light fields, offering alternatives and new approaches for some of these steps.

## 3.1 System Overview

The whole process of reconstructing a static light field from an image sequence can be subdivided roughly into six steps:

- image acquisition,
- feature extraction and tracking,
- initialization of reconstruction, usually using a factorization method,
- extension of the reconstruction to long image sequences,
- approximation of scene geometry, and
- light field visualization using different rendering methods.

Basically, these steps form a linear processing chain that has to be executed in this order, as proposed in the underlying work of Heigl [Hei04]. However, it is often beneficial to deviate from this strict linearity and use information obtained in later processing steps to refine the results of an earlier step. These iterations are referred to as feedback loops.

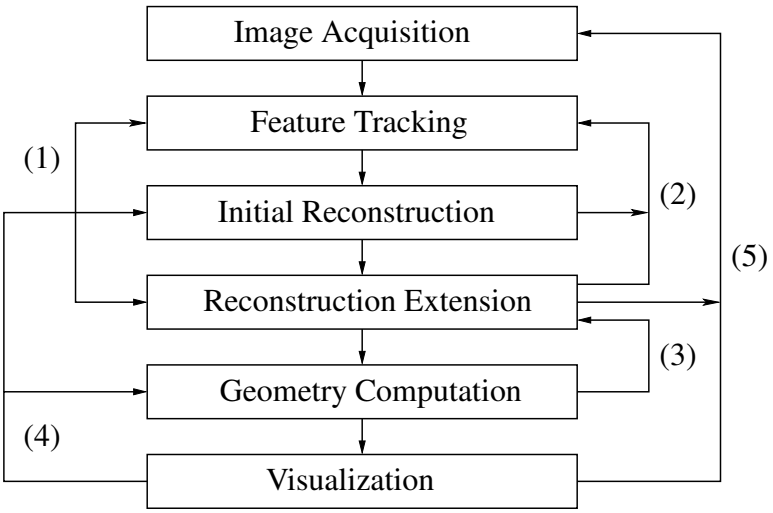



Figure 3.1: System overview: the linear processing chain for static light field reconstruction. The possible deviations from the linear process, (1) to (5), are explained in the text.

Figure 3.1 shows a number of different possibilities to deviate from the linear procedure. In particular, the five possibilities which are examined in this thesis are the following:

1. Image-by-image reconstruction: after initial reconstruction, which requires features to be available in a complete subsequence, feature tracking and reconstruction extension can be done in turn for each image. This method forms the basis for some of the feedback loops of item (2), which then supply additional information to feature tracking. It is introduced in Section 3.7.1.

2. Prediction for feature tracking and cross-relations: feature tracking, as it is applied here, searches for a feature in a subsequent image in an area around a certain starting point. Using the reconstructed 3-D positions of features aids in predicting feature positions more reliably and thus reduces the number of lost features (Sections 3.2.3 and 3.7.1). Knowledge of camera poses additionally helps to establish relationships between images beyond the sequential relationship derived from the recorded sequence (Sections 3.4.2 and 3.7.2).

3. Incremental global geometry: Instead of building a geometry model from the complete 3-D reconstruction, it can be generated incrementally after the processing of individual images (Section 3.6.3).

4. Feedback from synthesis: the images synthesized from a light field incorporate not only the reconstructed 3-D information, but the texture information of all input images. This additional knowledge can be used to improve depth or scene reconstruction (Section 3.7.3).

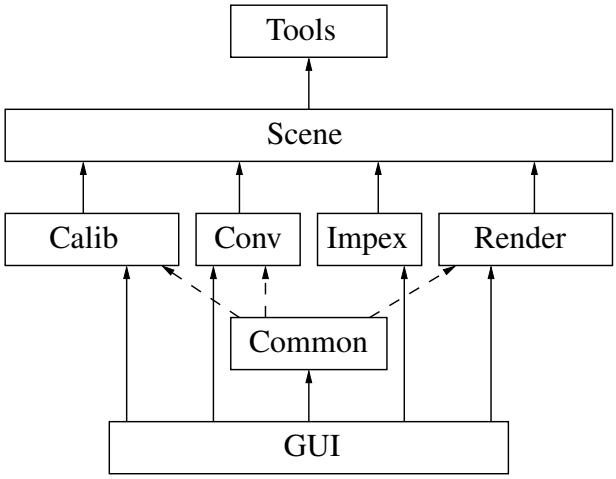



Figure 3.2: The modules of the LGF$^3$ framework according to [Vog05], showing also the additional *Common* module.

5. Adding new images: if a light field is to be constructed from several image sequences, the different 3-D reconstructions for each sequence have to be stitched together consistently. Synthesized images can help here to identify the correct starting point (Section 3.7.4).

Obviously, these many different possibilities for interaction between different processes require a common implementation framework, integrating all the processes. The framework created for this purpose is called LGF$^3$ (**l**umi**g**raph **f**iles, version **3**), a C++ library which incorporates the implementations of both light field reconstruction and visualization [Vog02a]. The novelty of the system is that it combines the two fields of computer vision and computer graphics and provides a common interface for low-level algorithms and data structures.

The framework is structured into eight different modules, which depend on each other as outlined in Fig. 3.2. The basis of the system is formed by the *Scene* module, managing the common data structures for storing scene information such as images, camera parameters and 3-D scene data. It makes use of low-level algorithms in the *Tools* module, while the *Impex* module provides interfaces for importing and exporting data. The algorithms for reconstructing light fields and visualizing them are implemented in the *Calib* and *Render* modules, respectively, and the *Conv* module contains routines for converting data according to the different needs of these modules. The *GUI* module provides a graphical user interface for user applications. A detailed description of these seven modules was given by Vogelgsang in [Vog05]. The last module, *Common*, was added later to accommodate the requirements of the feedback loops described above, namely access to all other modules, apart from *GUI*, in arbitrary order.

## 3.2 Feature Extraction and Tracking

Estimating camera pose and scene geometry for a sequence of images requires knowledge about correspondences between the images. In the case at hand, these correspondences consist exclusively of point-like features on the surface of the scene whose projections in the images are identified. Establishing point correspondences basically requires two steps, first the extraction of the 2-D features from images, and second the tracking of the features to other images or their matching with features in other images. Thus, for each scene surface point in consideration its occurrences in a number of images are detected. In the following, these occurrences of one feature will be called a *feature trail* through the image sequence.

There are two approaches to solving the correspondence problem which are regarded in the following sections. In case of relatively slow camera motion and little image displacement, the procedure of extracting features in one image and tracking them in the following images yields the best results. The most commonly used approach, usually called Kanade-Lucas-Tomasi (KLT) tracking, is described in the following section. If the displacements between images are larger, e. g., if only a set of single photographs is given, tracking features will be more likely to fail. Extracting features in all images and matching them subsequently may perform better. An approach using SIFT features is introduced in Section 3.2.2. For both approaches, the variant working on gray-level images $\boldsymbol{f}^g$ is presented.

Finally, tracking features sequentially through an image sequence is not the only way to establish a feature trail. Deviating from this procedure increases the length of feature trails and therefore often leads to improved subsequent results. This non-sequential tracking will be described in the last part of this Section.

### 3.2.1 Kanade-Lucas-Tomasi Tracking

The KLT tracker used for experiments in this thesis incorporates several approaches from different authors. The feature detection algorithm was introduced by Tomasi and Shi [Tom91, Shi94], while the tracking algorithm is based on the work of Lucas and Kanade [Luc81], originally meant for image registration, with the extension of Shi and Tomasi for affine feature transformation [Shi94]. Illumination compensation and feature drift prevention were added to feature tracking by Jin [Jin01] and Matthews [Mat03], and combined for the feature tracker used here in [Zin04]. Finally, a Gaussian resolution pyramid is commonly used to increase the basin of convergence of the tracking algorithm. However, this component will not be addressed in detail.

**Feature detection**

In order to reliably track a feature to other images, the selection of "good" features is very important. Thus, the question is, what is a good feature for tracking? Obviously, single pixels are not a good choice since their appearance may change drastically due to noise and illumination changes, and a single pixel value may appear very frequently in an image. Therefore, rectangular windows of the image are regarded, especially those containing large gradients, i. e., lines and corners or textured areas instead of homogeneous ones. A single line in an image window suffers from the so-called *aperture problem* (see, e. g., [Tru98]), which means that it can be localized well in the direction of the gradient but not along the line. Hence, the following definition of a good feature requires the presence of two strong gradients in the window, which correspond to an edge or a corner.

For a gray-level image $\boldsymbol{f}^g(\boldsymbol{q}, \tau)$ in an image sequence taken at a discrete time step $\tau$ the gradient vector at each pixel $\boldsymbol{q} = (x, y)^{\mathrm{T}}$ is given as

$$\boldsymbol{g}(\boldsymbol{q}, \tau) = \left( \frac{\partial \boldsymbol{f}^g(\boldsymbol{q}, \tau)}{\partial x}, \frac{\partial \boldsymbol{f}^g(\boldsymbol{q}, \tau)}{\partial y} \right)^{\mathrm{T}} \quad . \tag{3.1}$$

The gradient is computed, e. g., by a Sobel operator [Jäh95] applied for each direction, as it is done here as well. Usually, image noise is reduced by low-pass filtering the image with a Gaussian filter before computing the gradient.

From these gradient vectors a *structure matrix* $\boldsymbol{G}(\boldsymbol{q}, \tau) \in \mathbb{R}^{2\times 2}$ is computed for each pixel which encodes the gradient information at this position:

$$\boldsymbol{G}(\boldsymbol{q}, \tau) = \sum_{k=-r}^{r} \sum_{l=-r}^{r} \boldsymbol{g}\left(\boldsymbol{q} + (k, l)^{\mathrm{T}}, \tau\right) \cdot \boldsymbol{g}^{\mathrm{T}}\left(\boldsymbol{q} + (k, l)^{\mathrm{T}}, \tau\right) \quad . \tag{3.2}$$

The index $r$ denotes the radius of the window around $\boldsymbol{q}$, so that a neighborhood of $(2r + 1)^2$ pixels is considered.

One very useful property of $\boldsymbol{G}$ is that its larger eigenvalue $\lambda_1$ corresponds to the most dominant gradient direction in an image window, while $\lambda_2$ denotes the magnitude of the gradient perpendicular to it. This reflects the requirement for a good feature stated before, namely that two strong gradients are present in a window if both eigenvalues are above a certain threshold $\lambda_{\text{th}}$. The window around a pixel is accepted accordingly if

$$\min(\lambda_1, \lambda_2) > \lambda_{\text{th}} \quad . \tag{3.3}$$

In practice, a minimum distance between two features is defined to avoid clustering around few very good features. From an interest map, which is created by applying the above criterion for each pixel in the image, the strongest respective feature is stored in a list and the interest values in a window around this feature are set to zero before the next feature is selected. The resulting list is then ordered and the first $n$ features can easily be read out as the best features in this image.

Several similar criteria, especially for corner detection, have been defined and some of them are in wide-spread use. A selection of corner detectors is listed in Section 1.2.3.

### Feature tracking

The task of identifying correspondences to a feature in subsequent images of an image sequence is called *tracking* the feature. Formally, this can be described by the equation

$$\boldsymbol{f}^g(\boldsymbol{q},\tau) = \boldsymbol{f}^g(\boldsymbol{q}+\boldsymbol{d},\tau+t) \quad , \tag{3.4}$$

meaning that a feature $\boldsymbol{q}$ is found in a second image at time $\tau+t$ at a position which is displaced by the vector $\boldsymbol{d}$.

As for feature detection, it does not make much sense to regard only single pixels, therefore, a feature window around $\boldsymbol{q}$ is again taken into account. In the ideal case, the difference between corresponding feature windows in two images would be zero, but in reality, the windows will differ due to occlusion, rotations of the surface, changes in illumination, or noise. Therefore, the problem is that of minimizing the following error measure:

$$\epsilon(\boldsymbol{d}) = \sum_{k=-r}^{r}\sum_{l=-r}^{r}\left(\boldsymbol{f}^g\left(\boldsymbol{q}+(k,l)^{\mathrm{T}},\tau\right) - \boldsymbol{f}^g\left(\boldsymbol{q}+(k,l)^{\mathrm{T}}+\boldsymbol{d},\tau+t\right)\right)^2 \quad . \tag{3.5}$$

By approximating the intensity function by its truncated Taylor series,

$$\boldsymbol{f}^g\left(\boldsymbol{q}+\boldsymbol{d},\tau+t\right) \approx \boldsymbol{f}^g\left(\boldsymbol{q},\tau+t\right) + \boldsymbol{g}^{\mathrm{T}}\left(\boldsymbol{q},\tau+t\right)\boldsymbol{d} \quad , \tag{3.6}$$

the derivative with respect to $\boldsymbol{d}$ of (3.5) is computed as follows and set to zero to yield the minimum of the error function.

$$\sum_{k=-r}^{r}\sum_{l=-r}^{r} 2\left(\boldsymbol{f}^g\left(\boldsymbol{q},\tau\right) - \boldsymbol{f}^g\left(\boldsymbol{q},\tau+t\right) - \boldsymbol{g}^{\mathrm{T}}\left(\boldsymbol{q},\tau+t\right)\boldsymbol{d}\right)\boldsymbol{g}\left(\boldsymbol{q},\tau+t\right) = 0 \tag{3.7}$$

Assuming that the displacement $\boldsymbol{d}$ is constant for every pixel in the window, and substituting $\boldsymbol{g}\boldsymbol{g}^{\mathrm{T}}$ with the structure matrix $\boldsymbol{G}$ of equation (3.2), a linear system of equations of the form

$$\boldsymbol{G}\boldsymbol{d} = \sum_{k=-r}^{r}\sum_{l=-r}^{r}\left(\boldsymbol{f}^{g}\left(\boldsymbol{q}+(k,l)^{\mathrm{T}},\tau\right)-\boldsymbol{f}^{g}\left(\boldsymbol{q}+(k,l)^{\mathrm{T}},\tau+t\right)\right) \tag{3.8}$$

is created which yields a unique solution for $\boldsymbol{d}$. Of course, since the intensity function was approximated only by the linear term of the Taylor series, the solution $\widehat{\boldsymbol{q}}$ for the tracked point in the image at time instance $\tau + t$ will be inexact. Therefore, the procedure is iterated with $\widehat{\boldsymbol{q}}$ as starting point until either $\boldsymbol{d}$ is below a certain threshold or a maximum number of iterations is reached. As $\boldsymbol{d}$ is in general a vector of real numbers, the pixel values for the sums in the equations above are interpolated bilinearly. Thus, tracking is performed with sub-pixel accuracy.

One drawback of this tracking method is that its basin of convergence is usually very small, in the order of a few pixels. In order to increase the basin of convergence, a Gaussian resolution hierarchy is usually employed. By low-pass filtering and subsampling the image, tracking is performed first on lower resolutions and then refined at higher resolution levels.

### Affine distortion estimation

A major assumption for the above translational tracking is that the displacement vector $\boldsymbol{d}$ is equal for all pixels in the window. However, this is only true for some very special cases, like windows on planar surfaces combined with translational camera movement parallel to the surface. For most other cases, the image window will get more and more distorted the longer it is tracked through an image sequence. This distortion corresponds to a perspective transformation, but, as for factorization in Section 2.2, it can be approximated by an affine transformation. Thus, the displacement $\boldsymbol{d}$ extends to an affine motion field of the form

$$\boldsymbol{d}_a = \begin{pmatrix} d_{xx} & d_{xy} \\ d_{yx} & d_{yy} \end{pmatrix}\boldsymbol{q} + \boldsymbol{d} = \boldsymbol{D}_a\boldsymbol{q} + \boldsymbol{d} \quad . \tag{3.9}$$

The correspondence error between two feature windows is extended accordingly to

$$\epsilon(\boldsymbol{d}_a) = \sum_{k=-r}^{r}\sum_{l=-r}^{r}\left(\boldsymbol{f}^{g}\left(\boldsymbol{q}',\tau\right)-\boldsymbol{f}^{g}\left(\boldsymbol{q}'+\boldsymbol{D}_a\boldsymbol{q}'+\boldsymbol{d},\tau+t\right)\right)^{2}, \text{with } \boldsymbol{q}' = \boldsymbol{q}+(k,l)^{\mathrm{T}}. \tag{3.10}$$

As before, the intensity function is approximated by a truncated Taylor series as in equation

(3.6), i. e.,

$$\boldsymbol{f}^g\left(\boldsymbol{q}+\boldsymbol{d},\tau+t\right) \approx \boldsymbol{f}^g\left(\boldsymbol{q},\tau+t\right) + \boldsymbol{g}_a{}^{\mathrm{T}}\left(\boldsymbol{q},\tau+t\right)\boldsymbol{a} \quad , \tag{3.11}$$

where $\boldsymbol{a} = (d_{xx}, d_{xy}, d_{yx}, d_{yy}, d_x, d_y)^{\mathrm{T}}$, and $\boldsymbol{g}_a = (q_x g_x, q_x g_y, q_y g_x, q_y g_y, g_x, g_y)^{\mathrm{T}}$. Likewise, the derivative of equation (3.10) is set to zero analogously to equation (3.7), and by using the affine structure matrix $\boldsymbol{G}_a = \boldsymbol{g}_a \boldsymbol{g}_a{}^{\mathrm{T}}$ a linear equation system equivalent to (3.8) is solved for the unknown elements of $\boldsymbol{a}$, the affine distortion matrix $\boldsymbol{D}_a$ and the displacement vector $\boldsymbol{d}$.

Note that for affine distortion estimation, six instead of two parameters have to be estimated, so that a larger window around the feature has to be chosen. Robustness of the estimation increases with additional equations, and each pixel in the feature window adds one equation.

### Illumination compensation and drift prevention

The next factor which influences the appearance of a feature window considerably is illumination and its changes over time. There are a number of sources for illumination change apart from actual variation in lighting, such as shadows, surface reflectance changes due to different viewing angles, or the auto-shutter mechanism of the camera. These effects are modeled by a linear illumination model using two parameters $\alpha$ and $\kappa$, adjusting the contrast and the brightness of the feature window respectively. They are incorporated into the affine correspondence error of equation (3.10) by

$$\epsilon(\boldsymbol{d}_a) = \sum_{k=-r}^{r}\sum_{l=-r}^{r}\left(\boldsymbol{f}^g\left(\boldsymbol{q}',\tau\right) - \left(\alpha \boldsymbol{f}^g\left(\boldsymbol{q}'+\boldsymbol{D}_a\boldsymbol{q}'+\boldsymbol{d},\tau+t\right)+\kappa\right)\right)^2 \quad . \tag{3.12}$$

The minimum of this function is found as in the translational and affine cases by linearizing it with a truncated Taylor series and establishing a linear system of equations using its derivative. The details of this procedure are explained in [Jin01].

Taking advantage of the different strengths of the alternatives for tracking introduced so far, the common procedure is to first perform a translational tracking from image $\boldsymbol{f}_f^g$ to $\boldsymbol{f}_{f+1}^g$. The larger convergence radius of translational tracking allows tracking features for a longer period of time, but the drawback is that the appearance of the feature window may change over time by error accumulation, causing the feature to "drift" away from its original position. Affine tracking is used to either monitor the success of translational tracking by periodically comparing the result of an affine tracking from the first image $\boldsymbol{f}_0^g$ where the feature was detected to image $\boldsymbol{f}_{f+1}^g$, or by refining the result of translational tracking with this affine tracking for every image. While the first method may only discard features that have drifted, the second prevents drifting and leads

to tracking over more images.

### 3.2.2 Establishing Correspondences Using SIFT Features

The main assumption made for feature tracking is that changes between two images, which usually correspond to camera movement, are rather small. The gradient descent technique, which minimizes for instance the error function of equation 3.12, is only a local process and will therefore only detect the next local minimum, which may very well be the wrong feature. Thus, the closer the starting point for this search is to the correct feature location, the higher the probability that the correct feature will be tracked. In practice, this basin of convergence will be around 20 to 30 pixels wide, depending on the resolution hierarchy employed.

In contrast to that, local features such as the SIFT features [Low99, Low04] may overcome the problem posed by large translational movement. For each detected feature a distinctive feature description vector is calculated. This allows for the comparison of two sets of local feature vectors found in two images and the identification of corresponding features by a distance measure between two feature vectors. The SIFT feature approach has been chosen here as it compares favorably with other local descriptors as shown in [Mik05].

In case of the SIFT (Scale Invariant Feature Transform) feature, a difference-of-Gaussian approach in scale-space is used to identify features in one image. The original image is filtered twice by two different Gaussian filters, i. e., with $\sigma = \sqrt{2}$ and $\sigma = 2$. The two result images are subtracted from each other, now emphasizing the image gradients. The second result image is then subsampled with a pixel spacing of $1.5$, which yields already the first filtered image at the next resolution level. Using this efficient technique, the difference-of-Gaussian images are calculated for a resolution pyramid of several levels. Points of interest are extracted by considering pixels in each resolution with a higher gray value than their eight neighbors. If the same is not true for the corresponding pixels in the next higher and lower resolution, the feature is discarded.

The scale-space approach provides scale invariance, but in order to achieve rotation and illumination invariance, the gradient magnitude $\boldsymbol{m}_{\text{SIFT}}$ and rotation $\boldsymbol{r}_{\text{SIFT}}$ for each pixel in the resolution pyramid are calculated as

$$\boldsymbol{m}_{\text{SIFT}}(\boldsymbol{q}) = \sqrt{(\boldsymbol{f}^g(\boldsymbol{q}+\mathbf{1}_x) - \boldsymbol{f}^g(\boldsymbol{q}-\mathbf{1}_x))^2 + (\boldsymbol{f}^g(\boldsymbol{q}+\mathbf{1}_y) - \boldsymbol{f}^g(\boldsymbol{q}-\mathbf{1}_y))^2} \quad , \tag{3.13}$$

$$\boldsymbol{r}_{\text{SIFT}}(\boldsymbol{q}) = \arctan\frac{\boldsymbol{f}^g(\boldsymbol{q}+\mathbf{1}_y) - \boldsymbol{f}^g(\boldsymbol{q}-\mathbf{1}_y)}{\boldsymbol{f}^g(\boldsymbol{q}+\mathbf{1}_x) - \boldsymbol{f}^g(\boldsymbol{q}-\mathbf{1}_x)} \quad . \tag{3.14}$$

$\mathbf{1}_x$ and $\mathbf{1}_y$ denote the vectors $(1,0)^{\mathrm{T}}$ and $(0,1)^{\mathrm{T}}$, respectively. The gradient magnitude is limited to an upper bound, $0.1$ times the maximum magnitude according to [Low99], and for each feature a principal orientation angle is determined from a local window. This is done by Gauss-weighting the rotations of each pixel in the window, as well as weighting them with their thresholded magnitude, and accumulating them in a histogram with $36$ bins. The histogram is smoothed and the dominant bin is selected as orientation for this feature.

The local feature descriptors are calculated likewise from $\boldsymbol{m}_{\text{SIFT}}$ and $\boldsymbol{r}_{\text{SIFT}}$. The gradient orientations in a $16 \times 16$ pixel window around each feature at the scale level it was detected are weighted as above. This window is then subdivided again into $4 \times 4$ subwindows, and an orientation histogram with eight bins is calculated for each subwindow. The histograms are rotated by the principal orientation of the feature to achieve rotation invariance. The entries of these $16$ histograms are then concatenated and normalized to form the $128$-dimensional feature vector $\boldsymbol{c}$.

Applied to a sequence or set of images, the SIFT feature detector yields a set of feature vectors and their coordinates for each image. For further processing in 3-D reconstruction, corresponding features in all images have to be found and combined to form again feature trails as in the previous section. In [Grä05] such a trail is defined as a set of feature vectors $\mathcal{C}_i$ with

$$\mathcal{C}_i = \left\{ \boldsymbol{c} \mid \text{idx}\left(\boldsymbol{c}_k\right) \neq \text{idx}\left(\boldsymbol{c}_l\right) \wedge d(\boldsymbol{c}_k, \boldsymbol{c}_l) < \epsilon_{\text{SIFT}} \right\}, k \neq l;\ i \neq j \Rightarrow \mathcal{C}_i \cap \mathcal{C}_j = \emptyset \quad . \tag{3.15}$$

The operator $\text{idx}\left(\cdot\right)$ returns the image index where the feature vector was detected, so that $\boldsymbol{c}_k$ and $\boldsymbol{c}_l$ are two features found in different images. $d(\cdot,\cdot)$ denotes the Euclidean distance between two vectors. $\epsilon_{\text{SIFT}}$ is a threshold value that was found empirically by considering the Euclidean distance of different, not corresponding feature vectors found in one image. If a feature vector is allocated to more than one trail, the trails are discarded as being ambiguous. Thus, no two trails may share a common feature.

Both types of feature trails, those originating from feature tracking and those from SIFT feature detection, may be used equivalently for the reconstruction methods described in the following sections. Differences in their applicability and the quality of reconstruction results will be examined in Section 5.2.

### 3.2.3 Non-Sequential Tracking

Very often, tracking features over an image sequence is perceived as starting from the beginning of the sequence and finishing at its end. However, much benefit can be gained from abandoning

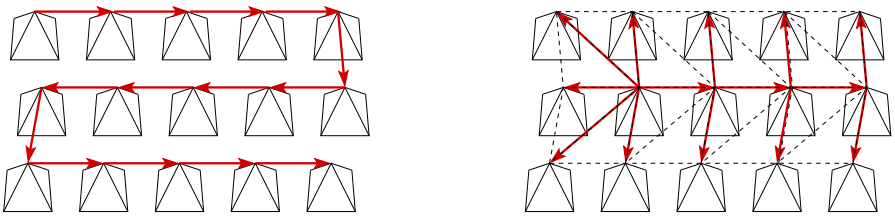


Figure 3.3: Non-sequential tracking in an image sequence. Left: Sequential tracking along the movement path of the camera. Camera positions are depicted as pyramids with the center of projection at the tip of the pyramid. Right: Meshed camera positions and non-sequential tracking from a starting image outward.

this notion. Consider the case of a camera moving parallel to a surface. In each image, features will be lost because they moved out of the camera's field of view, and a replacement feature is detected. This replacement is not necessarily found at the opposite image border to where the previous feature was lost, but anywhere in the image. Thus, its motion path may not span the whole width of the camera's field of view, but only a certain part of it, i. e., from its point of first detection to the image border. However, if the movie were played backwards, the feature would be tracked to the opposite image border. Therefore, by processing the image sequence in reverse order in a second pass and picking up all features at their initial frames, the length of their trails can be increased considerably. This technique will be called *bilinear tracking* and is used whenever features are tracked through the whole sequence before any further processing.

If additional information is available on the neighborhood relationship of the camera positions in an image sequence, it can be exploited similarly to generate longer feature trails. For the example of the camera movement in the left sketch of Figure 3.3, a mesh on the camera positions may be available as shown in the right hand side. This mesh information may be available in advance if the camera performed a controlled movement, or may be generated as will be described later on in Section 3.4.2. Tracking can now be performed not only to the succeeding and preceding image, but to all neighboring images as shown in the right hand sketch. This decreases the length of the path to the farthest camera position and thus the likelihood of losing the features, as well as the amount of accumulated errors.

Tracking through such a viewpoint mesh can be done in numerous ways, but the following scheme is employed here. Starting at an arbitrary image, $N_{\text{init}}$ features are selected. These are tracked to all neighboring views and then outward iteratively to the next distant views, but without replacing lost features. Tracking from one view onward is stopped if less than $N_{\text{min}}$ features have been tracked to it. If no more features can be tracked, the image $f$ with the lowest number of visible features, $N_f$, is selected as the new starting frame and tracking is initialized

with $N_{\text{init}} - N_f$ features. This is repeated until at least $\nu N_{\text{init}}$ features have been tracked in each image, where $\nu$ is a ratio between zero and one. This tracking scheme will be denoted as *mesh tracking*.

## 3.3 Automatic Selection of Factorization Methods

In Section 2.2 the principles of several different factorization methods were described. Each of these methods has its benefits and disadvantages regarding robustness and accuracy. Therefore, the decision which method is the best may vary between different data sets, and usually, it cannot be determined beforehand.

This section deals with the problem of automatically selecting a suitable or even the best factorization method in each application case. However, there are different possibilities for improving the results of each method before a selection is performed, a topic that will also be dealt with in the following. Finally, selecting the best factorization method requires quality measures which allow the selection in the first place. Besides the back-projection error, the peak signal-to-noise ratio based on a reconstructed light field will be introduced.

### 3.3.1 Robust Estimation Using Outlier Detection

In the following, a method for increasing the robustness and accuracy of 3-D reconstruction, Least Median of Squares (LMedS), is introduced. Being a robust estimation technique similar to *Random Sample Consensus* (RANSAC) [Fis81], it is used for outlier detection and is applied in different aspects to reconstruction.

#### Least Median of Squares (LMedS)

Given a noisy set of data with a number of outliers, such as point features tracked in an image sequence, Least Median of Squares is used to identify these outliers for a given estimation task, such as the estimation of the fundamental matrix from eight point correspondences. Thus, if for each point an error value can be computed, LMedS minimizes the median of squared errors for the whole set of points. Given a set $\mathcal{Q}$ of $N$ points $\boldsymbol{q}_n$, the algorithm is outlined as follows for the example of robust fundamental matrix estimation:

1. Draw $M$ sets $\mathcal{Q}_m$ of $p$ points from $\mathcal{Q}$, where $p$ is the minimum number of points required for estimation. In case of the fundamental matrix, an estimate is possible using only seven point correspondences (see [Har03] for algorithm details). However, this yields up to three

solutions which have to be disambiguated. Instead, the linear 8-point-algorithm is applied which always yields a unique solution, so that $p = 8$.

2. For each set $\mathcal{Q}_m$ compute the fundamental matrix $\boldsymbol{F}_m$. Then, apply each fundamental matrix to each point in the complete set $\mathcal{Q}$ according to equation (2.21), which yields the error $r_n(\mathcal{Q}_m) = \underline{\boldsymbol{q}}_n'^{\mathrm{T}} \boldsymbol{F}_m \underline{\boldsymbol{q}}_n$. Finally, compute the median of the squared errors, $R_m = \underset{n=1\ldots N}{\mathrm{median}}(r_n^2(\mathcal{Q}_m))$.
3. Select the set with the smallest median error, $R_{min}$, to compute outliers in $\mathcal{Q}$. An outlier is given if the error of a point with respect to $\mathcal{Q}_{min}$, $r_n(\mathcal{Q}_{min})$, is larger than $2.5\widehat{\sigma}$, where

$$\widehat{\sigma} = 1.4826(1 + p/(N - p))\sqrt{r_n(\mathcal{Q}_{min})} \quad . \tag{3.16}$$

4. Compute the fundamental matrix again using all remaining inliers.

For this algorithm, only one parameter remains to be selected by the user, i. e., the number of random sets $M$ to be drawn from $\mathcal{Q}$. For the algorithm to produce the desired result, at least one of the random sets has to consist only of inliers. In order to determine $M$, the user supplies an estimate of the number of outliers in the point set, $\kappa$, and the desired probability $P$ that at least one subset consists entirely of inliers. Using these values, $M$ computes as

$$M = \frac{\log(1 - P)}{\log(1 - (1 - \kappa)^p)} \quad . \tag{3.17}$$

For further information about the LMedS algorithm, see [Rou87] for a more theoretic definition, and [Zha95] for a comprehensive description of its application to conic fitting.

**Application in factorization**

For the factorization methods of Section 2.2, especially for perspective factorization, there are two ways in which robust outlier detection is applied here. The first one, the application to fundamental matrix estimation, is only used in perspective factorization. It was already described above and is used for estimating the projective depths of each feature point with increased accuracy.

The second application of outlier detection is to the process of factorizing the measurement matrix $\boldsymbol{W}$ itself. This is possible for both affine and perspective factorization. In the affine case, the minimum number of points to perform the factorization is $p = 3$, for the perspective case it is $p = 4$. The decomposition of each measurement submatrix $\boldsymbol{W}_m$ results in estimates $\widehat{\boldsymbol{\Psi}}_m$

and $\widehat{\boldsymbol{\Phi}}_m$ for the projection matrices and 3-D point positions. In order to get an estimation of the point positions for the whole data set, the complete measurement matrix is multiplied with the pseudo-inverse of the estimated motion, i. e.,

$$\widehat{\boldsymbol{\Phi}}^{(m)} = \widehat{\boldsymbol{\Psi}}_m^{+} \boldsymbol{W} \quad . \tag{3.18}$$

From this, a distance matrix $\boldsymbol{\Delta}_m$ is computed which holds the distances between the back-projected 3-D points, $\widehat{\boldsymbol{\Phi}}^{(m)}$, and the measurements $\boldsymbol{W}$:

$$\boldsymbol{\Delta}_m = \boldsymbol{W} - \widehat{\boldsymbol{\Psi}}_m \widehat{\boldsymbol{\Phi}}^{(m)} \quad . \tag{3.19}$$

The average back-projection error is now computed from the columns of $\boldsymbol{\Delta}_m$ for each 3-D point, and outlier detection proceeds as above for the fundamental matrix.

Note, however, that application of LMedS to affine factorization methods is often not sensible. The error induced by the approximated projection model increases the farther a feature is away from the optical center (weak-perspective) or the projection of the center of mass of the corresponding 3-D points (paraperspective). Thus, LMedS tends to discard features close to the image border and to favor those which are close to the center of the image. This problem of perspective distortions does not occur for the perspective factorization method of Sturm and Triggs.

### 3.3.2 Quality Measures

The automatic selection of a factorization method is based on two different quality criteria, back-projection error and peak signal-to-noise ratio. The two measures offer different benefits and drawbacks, so that a combination of them is advantageous.

#### Back-projection error

The back-projection error was already introduced in Section 2.3.3 for bundle adjustment. For each feature in each image, it denotes the difference between the projection of the corresponding, estimated 3-D point into the image, and the feature itself. Using the residual vector of equation (2.61) for all features used during factorization, the average back-projection error is computed as

$$\overline{\epsilon}_{\text{bp}} = \frac{1}{FN} \sqrt{{\boldsymbol{\epsilon}_{\text{B}}}^{\text{T}} \boldsymbol{\epsilon}_{\text{B}}} \quad . \tag{3.20}$$

As is already known from Section 2.2, the result of any factorization method is unique only up to an unknown transformation $\boldsymbol{Q}$ or $\boldsymbol{Q}_p$, respectively. However, the selection of this transformation has no effect at all on the back-projection error. Nevertheless, whether the update to a metric reconstruction, i. e., self-calibration in the perspective case, is successful or not has a huge impact on the quality of images rendered from the resulting light field later on. Rendering a light field requires metric reconstruction. Therefore, the second quality measure is defined with respect to the quality of rendered images.

### Peak signal-to-noise ratio

In order to measure the quality of a reconstruction result, ground truth data is always the best standard to compare with. Thus, if the quality of an image rendered from a light field with the virtual camera at a certain position is to be measured objectively, it should be compared with a real image which was actually recorded with the camera at this very position. This configuration can be achieved by rendering for each image $\boldsymbol{f}_f$ in the original sequence the corresponding image using the calculated camera parameters $\boldsymbol{P}_f$. The original image in question is not used for rendering so that the rendered image $\widehat{\boldsymbol{f}}_f$ is only composed from contributions of neighboring images. The average squared pixel difference of original and rendered image is computed by

$$\overline{d_f^2} = \frac{1}{3M} \sum_{j=1}^{M} \left( |\boldsymbol{f}_f^r(j) - \widehat{\boldsymbol{f}}_f^r(j)|^2 + |\boldsymbol{f}_f^e(j) - \widehat{\boldsymbol{f}}_f^e(j)|^2 + |\boldsymbol{f}_f^b(j) - \widehat{\boldsymbol{f}}_f^b(j)|^2 \right) \quad , \tag{3.21}$$

where $M$ is the number of pixels in each image, and $\boldsymbol{f}_f^r$, $\boldsymbol{f}_f^e$ and $\boldsymbol{f}_f^b$ are the red, green and blue channels of the color image $\boldsymbol{f}_f$, respectively.

This squared difference value $\overline{d_f^2}$ represents the noise on the rendered image compared to the original image. The signal is given by the average squared pixel value in the original image, $\overline{f_f^2}$, and thus the *signal-to-noise ratio* (SNR) for one image is defined as

$$\text{SNR} = 10 \log_{10}(\overline{f_f^2}/\overline{d_f^2}) \quad [dB]. \tag{3.22}$$

The one fault of this definition is that the resulting value depends on the overall brightness of the image. Thus, a dark image with the same level of noise will have a much lower SNR value as a bright one. Therefore, the *peak signal-to-noise ratio* (PSNR) will be used in the following, which is defined as

$$\text{PSNR} = 10 \log_{10}(f_{\text{max}}^2/\overline{d_f^2}) \quad [dB] \tag{3.23}$$

and allows the comparison of methods across images from different data sets. The value $f_{\max}$ denotes the maximum possible pixel value, for one color channel in case of color images. For a color image with eight bits per channel, this would be $255$. PSNR is also widely used for measuring quality in image compression [Kou95].

Apart from the automatic selection of factorization methods in this section, the PSNR will also be used in Section 3.6 for optimizing depth maps and global proxies, and generally in Chapter 5 for evaluating experimental results. In the literature, the use of the mean absolute pixel difference between rendered and original image was first mentioned in [ES03a], the use of SNR and PSNR for this purpose were formally introduced in [Nie05] and [Vog06], respectively. In [Fec06], PSNR is used to measure the effects of depth map compression on light field rendering quality.

### 3.3.3 Alternative Processing Chains

Basically, the goal of the procedure introduced in this section is to select the best of the factorization methods described in Section 2.2 for a given initial image sequence. In practice, not all of these factorization methods need to be considered, since orthographic factorization is generally inferior to all other methods due to its very simplified projection model—unless an orthographic camera is actually used, which is the case here. Additionally, paraperspective factorization is usually more accurate than the weak-perspective one, so that the latter is not considered either.

On the other hand, the accuracy of camera calibration and 3-D reconstruction also depends on subsequent processing steps, like outlier detection as described above, or self-calibration as described in Section 2.3. Therefore, the selection algorithm will consider four different processing chains, outlined in Figure 3.4, which will be described in the following.

#### Paraperspective factorization

As shown in Section 2.2.2, a paraperspective factorization method is first applied to the features found in the initial image subsequence. This method does not include an estimation of intrinsic camera parameters. Focal length, principal point and image skew have to be calibrated beforehand or else roughly estimated or defined arbitrarily. As paraperspective factorization yields a rotation matrix and a translation vector for each camera, these can be combined with the estimated calibration matrix to form metric projection matrices as in equation (2.11).

Due to the simplified projection model, this metric reconstruction will be skewed nevertheless. Therefore, a factorization is followed by a non-linear optimization step similar to bundle

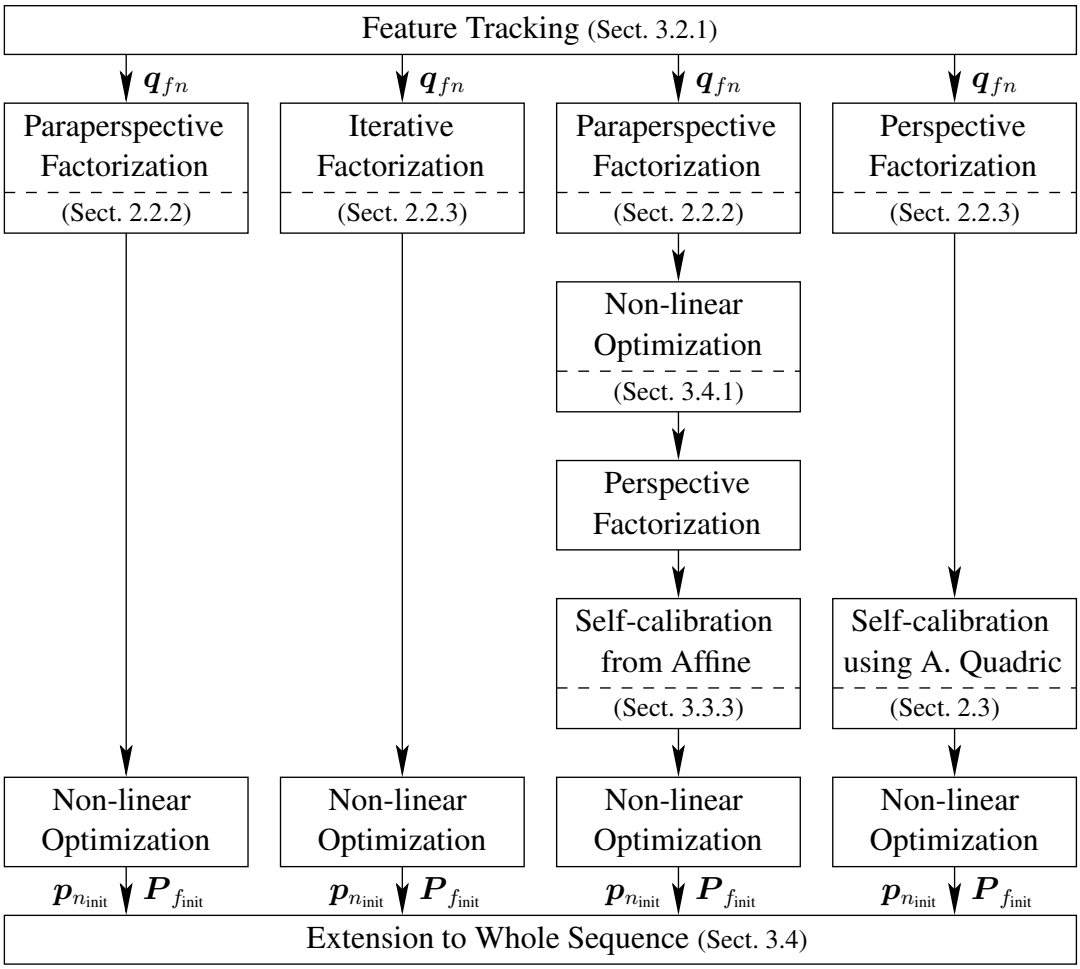



Figure 3.4: Alternative processing chains for selecting the most suitable initial reconstruction

adjustment of Section 2.3.3. The residual vector of equation (2.61) is minimized alternatingly for the camera parameters in $\widehat{\boldsymbol{P}}_f$ and the 3-D point coordinates in $\underline{\widehat{\boldsymbol{p}}}_n$. In this case, however, only the six camera parameters for rotation and translation are optimized. At this point, LMedS outlier detection may be applied to either exclude single 2-D features of a trail, or the complete 3-D point. Back-projection error is used as error measure in both cases. The same process of non-linear optimization is also used for the subsequent extension to the rest of the image sequence. Therefore, it will be treated in more detail in Section 3.4.1.

### Iterative factorization

The second processing chain using the iterative factorization method by Christy and Horaud (Section 2.2.3) is similar to the previous one. Likewise, the intrinsic camera parameters are

not estimated by the factorization method and have to be calibrated or estimated beforehand. Likewise, the factorization is followed by a non-linear optimization step. However, iterative factorization is usually more accurate than the paraperspective one as it tries to approximate the perspective solution.

### Perspective factorization with self-calibration using an affine solution

The remaining two processing chains both implement the perspective factorization method by Sturm and Triggs, but differ in the subsequent self-calibration step which is required to find the projective update matrix $\boldsymbol{D}$. In the first case, $\boldsymbol{D}$ is determined using the affine solution of a preceding paraperspective factorization, thus extending the first processing chain.

This self-calibration using an affine solution was introduced in [Hei04]. Affine and perspective factorization yield the same 3-D point clouds, although the latter is skewed by $\boldsymbol{D}$, while the former already resides in a metric framework. If a 3-D point of the affine solution is denoted by $\underline{\widehat{\boldsymbol{p}}}_n$, and the corresponding one of the skewed solution by $\underline{\widehat{\boldsymbol{p}}}'_n$, the following equation results:

$$\chi_n \begin{pmatrix} \widehat{p}'_{n1} \\ \widehat{p}'_{n2} \\ \widehat{p}'_{n3} \\ 1 \end{pmatrix} = \underbrace{\begin{pmatrix} d_{11} & d_{12} & d_{13} & d_{14} \\ d_{21} & d_{22} & d_{23} & d_{24} \\ d_{31} & d_{32} & d_{33} & d_{34} \\ d_{41} & d_{42} & d_{43} & d_{44} \end{pmatrix}}_{\boldsymbol{D}} \begin{pmatrix} \widehat{p}_{n1} \\ \widehat{p}_{n2} \\ \widehat{p}_{n3} \\ 1 \end{pmatrix} \quad . \tag{3.24}$$

The projective scale factor $\chi_n$ is readily computed from the last row of this equation system. In order to compute $\boldsymbol{D}$, its entries are concatenated in one column vector and a linear system of equations of size $3N \times 16$ is formed by all $N$ 3-D point pairs.

After applying the projective transformation $\boldsymbol{D}$, a non-linear optimization of the result is performed once again. This type of self-calibration has the advantage of increased stability compared to the self-calibration method of the fourth processing chain. However, it inherits some of the inaccuracy of the underlying paraperspective factorization result.

### Perspective factorization with self-calibration using the absolute quadric

For the last alternative processing chain no preceding affine factorization is required as opposed to the previous one. Self-calibration of the perspective factorization result is done using the absolute quadric as described in Section 2.3.2. This is again followed by a non-linear optimization of the extrinsic and, if desired, intrinsic camera parameters and 3-D points.

Due to the exclusive use of the accurate, perspective projection model this processing chain potentially yields the best reconstruction results, but is also most vulnerable to noisy point features. However, this drawback can be alleviated partly by using robust outlier detection for fundamental matrix computation and factorization.

### 3.3.4 Automatic Selection

The automatic selection algorithm is implemented fairly straightforward. A 3-D reconstruction is performed using each of the four alternative processing chains, and the results are evaluated using back-projection error and PSNR. For the latter, depth information is required which is usually computed using local or global proxies (cf. Section 3.6) for their efficiency in computation and rendering. The primary selection criterion is the PSNR. However, as it will be shown experimentally in Section 5.1, the PSNR does not represent a monotonous function, but is subject to minor variabilities. Therefore, from those alternatives which lie within 1 dB of the best processing chain regarding PSNR, the one with the lowest back-projection error is finally selected.

As is apparent from Figure 3.4, the different methods contain some redundant processing steps, which can be exploited in order to speed up selection. Thus, paraperspective factorization is part of the first as well as the third processing chain, while the results of perspective factorization can be used for both of the last two processing chains.

One problem concerning outlier detection has to be mentioned as well. The purpose of outlier detection is to remove unsuitable points from the set of points used for reconstruction. While this automatically decreases the back-projection error—since this is the error measure applied—it does not necessarily lead to an increase of PSNR if local or global proxies are used for visualization. These resulting triangle meshes are built from the point set used for reconstruction, and will likewise contain fewer nodes and triangles after outlier removal. However, a more sparsely sampled mesh may generate new artifacts in rendering, and thus increase PSNR.

## 3.4 Extension to Long Image Sequences

So far, using the methods described in the previous sections, only a subsequence of the whole image sequence has been reconstructed, namely the one chosen for the initial factorization method. This subsequence may be located at the beginning of the sequence, or somewhere in the middle where a large number of stable features was found. The following section describes how to continue the reconstruction from this point, using a non-linear optimization scheme similar to

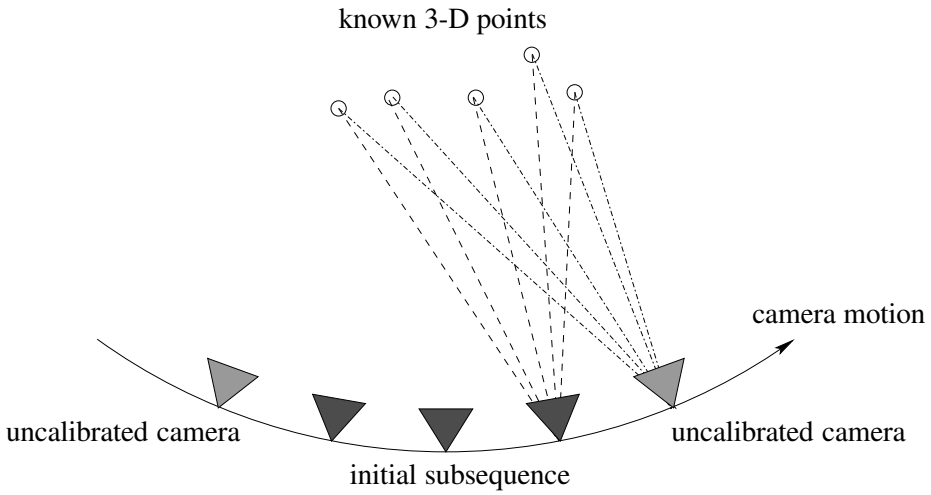



Figure 3.5: Extension of the reconstruction from an initial subsequence to the rest of the images. The dark triangles denote camera positions which have already been reconstructed, the light ones yet unknown camera positions in the image sequence.

bundle adjustment. Alternatively, the process can be regarded rather as being applied to a selection of images than an image sequence, where a mesh-like relationship is given between the images instead of a linear one. This will be considered in the second part of this section.

### 3.4.1 Extension by Non-Linear Optimization

The extension scheme described here was introduced in [Hei04], therefore, only its basic principles will be detailed here. The main idea is that the 3-D points reconstructed in the previous step form a kind of calibration pattern for the yet uncalibrated images. They represent known world coordinates for the corresponding projections into the next image to be calibrated, in those cases where these features were tracked successfully. The reconstruction process therefore consists of the following iterative steps:

1. Use triangulation to determine the 3-D coordinates of all features which are visible in at least $N_t$ images of the known subsequence.

2. Select the next camera position to be reconstructed, alternatingly before or after the already known subsequence.

3. Determine the parameters of the next camera by minimizing the back-projection error of the known points visible in this image, as illustrated in Figure 3.5.

4. Repeat steps 1 to 3 until all cameras have been calibrated.

**Triangulation.** Given the projection matrices $\boldsymbol{P}_t$ already known from the initial factorization or later iterations of the algorithm, with $\boldsymbol{p}_{ti}, 1 \leq i \leq 3$, being their three row vectors, two equations of the form

$$q_{t,i} \cdot (\boldsymbol{p}_{t3}\underline{\boldsymbol{p}}) - \boldsymbol{p}_{ti}\underline{\boldsymbol{p}} = 0, \quad i = 1, 2 \tag{3.25}$$

are formulated for each point projection $\underline{\boldsymbol{q}}_t$ of the 3-D point $\underline{\boldsymbol{p}}$ to be triangulated (see [Hei04]). $q_{t,i}$ are the elements of $\underline{\boldsymbol{q}}_t$. The resulting system of linear equations in the elements of $\underline{\boldsymbol{p}}$ is solved using SVD, yielding the best solution in the least-squares sense. However, the solution is further refined by a subsequent non-linear optimization of the back-projection error, as this constitutes a more geometrically meaningful measure.

**Camera parameter estimation.** Determining the parameters of the next camera position in step 3 is similar to the camera parameter estimation step of bundle adjustment in Section 2.3.3. Likewise, the residual error of equation (2.61) is minimized by adjusting some or all extrinsic and intrinsic parameters. This estimation is alternated with an additional refinement of the 3-D points visible in the current camera in order to incorporate the new information now available.

A simple means to discard bad features is to set a threshold on the maximum allowed back-projection error for each 3-D point. After adding each new camera, points which have been discarded before can be rechecked and accepted, which may be the case if only a few features of a long trail are erroneous. However, instead of this hard decision robust estimation using LMedS can be applied here again. Two possible ways to do this is to either look for outliers in the 3-D points used for camera parameter estimation, or for erroneous 2-D features during triangulation and its refinement. For parameter estimation, the minimum number of projections required depends on the number of parameters to be estimated, i. e., three if only the extrinsic parameters are optimized, and six if all 11 parameters are estimated. For triangulation, two projections of the 3-D point form a minimal set. In each case, outlier detection is done as described in Section 3.3.1. In Section 3.5, two additional approaches to increase robustness against outliers and image noise will be introduced.

### 3.4.2 Non-Sequential Reconstruction

Up to this point, it was assumed that nothing more is known about the relationship of camera positions to each other than their order in the image sequence. Thus, like for feature tracking, only sequential reconstruction from the initial subsequence forward or backwards was possible.

However, since the camera poses are available after a first reconstruction, it is possible to generate neighborhood relationships in form of a camera mesh from this information. This procedure and its utilization for subsequent reconstruction are described in the following.

**Generating the camera mesh.** The camera mesh is calculated exclusively from the projection matrices $\boldsymbol{P}_f$ of all reconstructed camera poses $f = 0, \ldots, F-1$. First, the average distance $\overline{\Delta_t}$ and angle $\overline{\alpha}$ between two consecutive camera poses is calculated:

$$\overline{\Delta_t} = \frac{1}{F-1} \sum_{i=0}^{F-2} \|\boldsymbol{t}_{i+1} - \boldsymbol{t}_i\| \quad , \tag{3.26}$$

$$\overline{\alpha} = \frac{1}{F-1} \sum_{i=0}^{F-2} \arccos\left(\boldsymbol{r}_{i,3}{}^{\mathrm{T}} \boldsymbol{r}_{i+1,3}\right) \quad , \tag{3.27}$$

where $\boldsymbol{t}_i$ is the translation vector of camera $i$ and $\boldsymbol{r}_{i,3}$ denotes the third column vector of rotation matrix $\boldsymbol{R}_i$, the viewing direction of the camera.

Next, each pair of camera poses $k$, $l$, $k \neq l$ is tested regarding four different conditions in order to be accepted as an edge in the camera mesh. The first two are that the distance of their projection centers is lower than a user-defined multiple of the average, i. e., $\Delta_{t,k,l} < \mu_t \overline{\Delta_t}$, and that their viewing direction difference is not larger than a multiple of the average between two consecutive frames, $\alpha_{k,l} < \mu_\alpha \overline{\alpha}$. Both conditions together ensure that the two corresponding images show approximately the same part of the scene. The third condition is that the two images have a certain distance in the consecutive image sequence, $|k-l| \geq \mu_i$, $\mu_i \in \mathbb{N}$. This reduces the number of edges, eliminating camera pairs which are connected via a short path along the sequence, where the probability of features reappearing is low. Additionally or alternatively, the fourth condition can be applied which states that the angle $\gamma_{k,l}$ between the current camera motion direction and the vector between the two positions has to be above a threshold $\gamma_{min}$, as illustrated in Figure 3.6. The thus selected edges form the camera mesh which will be used for mesh tracking (Section 3.2.3) and non-sequential reconstruction.

**Non-Sequential Reconstruction.** Reconstructing an image sequence if a camera mesh is given does not differ substantially from the sequential case. The two steps of triangulation and camera parameter estimation are applied in the same way as described in the previous section. Step two of the algorithm, selecting the next camera position to be reconstructed, has to be altered, just like the selection of the initial image subsequence—or *subset*—to be reconstructed using a factorization method. The initial subset is chosen as the subgraph of the camera mesh where

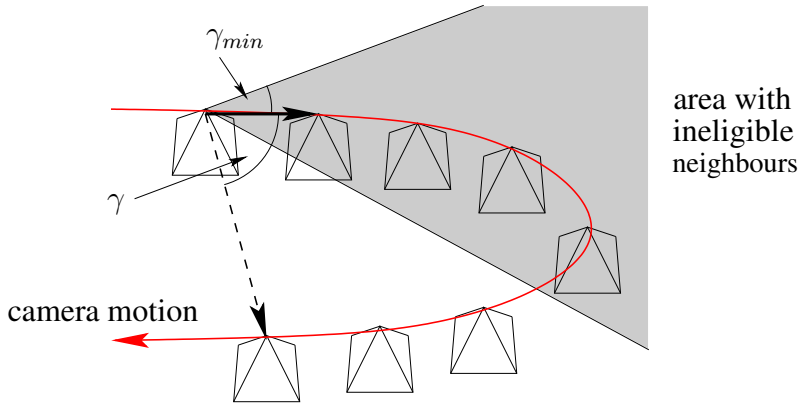



Figure 3.6: By considering the angle $\gamma$ between the current motion direction of the camera and the vector to the potential neighbor, redundant edges in the camera mesh along the motion path of the camera are avoided.

the most feature trails are visible completely. Starting from this submesh, the subsequent camera positions to be calibrated are chosen as in Section 3.2.3, by processing them in order of increasing distance to the initial submesh. Any additional options and processing steps, as described for sequential reconstruction, apply here as well.

## 3.5 Robust and Probabilistic Modeling

The major limiting factor for 3-D reconstruction or, generally, for computer vision, is image quality and thus inaccurate localization of image features. Apart from the inevitable sensor noise, tracked features may be mislocated, captured by similar features, or drift along edges. In Section 3.3.1, the LMedS outlier detection method was introduced which is able to identify such erroneous features and to remove them subsequently. A simpler, straightforward approach is the application of a threshold on the back-projection error of a feature. It was described briefly in Section 3.4.1.

In the following, two additional approaches are introduced which are designed to consider feature quality already during the process of parameter estimation. For the first one, a priori knowledge from feature tracking is applied in order to weight features according to their quality. The second approach makes use of so-called *robust statistics*, applying different estimators than the common least-squares estimator so that the influence of outliers on the final result is reduced.

### 3.5.1 Weighted Parameter Estimation

In order to estimate camera parameters and 3-D point positions in Section 3.4, as well as for bundle adjustment described in Section 2.3.3, a non-linear minimization is applied on the back-projection error $\boldsymbol{\epsilon}_\mathrm{B}$ of point features (cf. equation (2.61)). This residual vector combines the Euclidean distances between measured and estimated feature locations, and the residual which is finally minimized is the *sum of squared differences* $\epsilon_\mathrm{SSD} = {\boldsymbol{\epsilon}_\mathrm{B}}^\mathrm{T}\boldsymbol{\epsilon}_\mathrm{B}$.

However, if the reliability of measured feature locations is known, it can be incorporated into the residual using its covariance matrix $\boldsymbol{\Sigma}_\mathrm{B}$, as pointed out in [Har93, Sze97]. The expression to be minimized corresponds to the squared *Mahalanobis distance*

$$\epsilon_\mathrm{ML} = {\boldsymbol{\epsilon}_\mathrm{B}}^\mathrm{T}{\boldsymbol{\Sigma}_\mathrm{B}}^{-1}\boldsymbol{\epsilon}_\mathrm{B} \quad , \tag{3.28}$$

which leads to a maximum likelihood (ML) estimation of the parameters. If for each feature $n$ in image $f$ a $2 \times 2$ covariance matrix $\boldsymbol{\Sigma}_{\mathrm{B},fn}$ is given, the combined covariance $\boldsymbol{\Sigma}_\mathrm{B}$ has block-diagonal structure. If the error distribution is assumed isotropic, the covariance of each feature is $\boldsymbol{\Sigma}_{\mathrm{B},fn} = \sigma^2_{\mathrm{B},fn}\boldsymbol{I}_{2\times2}$ and $\boldsymbol{\Sigma}_\mathrm{B}$ is diagonal. Under the assumption that all feature errors are equally distributed, $\boldsymbol{\Sigma}_{\mathrm{B},fn} = \sigma^2_\mathrm{B}\boldsymbol{I}_{2\times2}$, equation (3.28) reverts to the (scaled) Euclidean distance used in bundle adjustment. In all cases, the errors are assumed to be normally distributed.

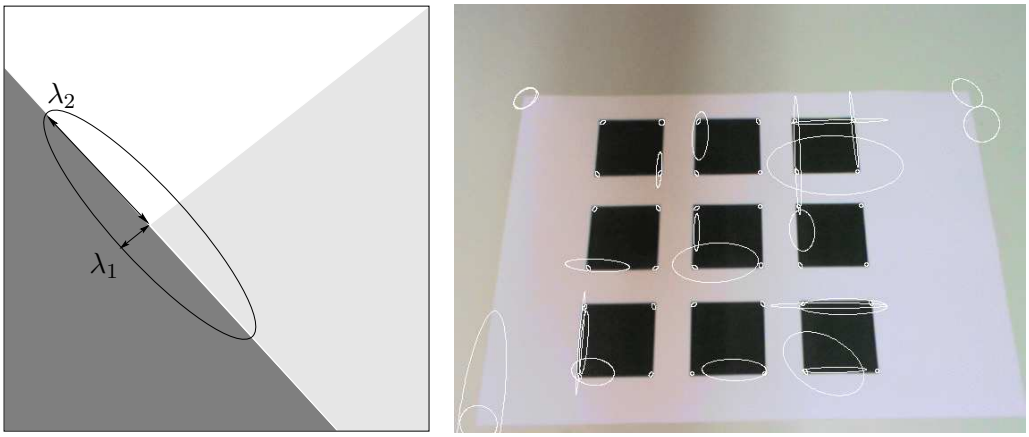
#### Feature Weights

Using the Mahalanobis distance to estimate the reconstruction parameters requires some a priori knowledge about feature quality. Without any further knowledge about scene geometry, this information needs to be derived from image content, which leads to the criteria used during feature tracking. From this information, three different weights are derived:

- **Smaller eigenvalue of structure matrix.** For feature detection, the smaller eigenvalue of the structure matrix $\boldsymbol{G}$ of equation (3.2) is used as quality criterion. Here, it acts as indicator for the reliability of the whole feature trail, yielding

$$\boldsymbol{\Sigma}^{-1}_{\lambda,fn} = \min(\lambda_{1,n}, \lambda_{2,n})\boldsymbol{I}_{2\times2} = w_{\lambda,n}\boldsymbol{I}_{2\times2} \quad . \tag{3.29}$$

- **Intensity difference.** The criterion for deciding whether a feature can be tracked any further is the intensity difference $\epsilon(\boldsymbol{d}_a)$ between the feature windows in the original and the current image, calculated using equation (3.12). This weight for the current feature is




(a) (b)

Figure 3.7: (a) The eigenvalues $\lambda_1$ and $\lambda_2$ of $\boldsymbol{\Sigma_G}$ represent the uncertainty perpendicular ($\lambda_1$) and along ($\lambda_2$) the most prominent edge in the feature window. (b) A selection of features in a real image. Features on corners of the pattern show low uncertainty, and high uncertainty on edges (aperture problem). Eigenvalues are scaled for visualization.

therefore calculated as

$$\boldsymbol{\Sigma}^{-1}_{\boldsymbol{d},fn} = \frac{w_{\lambda,n}}{\epsilon_{fn}(\boldsymbol{d}_a)} \boldsymbol{I}_{2\times 2} \quad . \tag{3.30}$$

Since for the original image the difference is 0 by construction, the eigenvalue weight $w_{\lambda,n}$ from above is used to discern different feature trails. The feature point in base image $f_0$ is weighted with $\boldsymbol{\Sigma}^{-1}_{\boldsymbol{d},f_0 n} = a w_{\lambda,n}$, where $a$ is an arbitrary weight factor.

- **Structure matrix.** As Kanazawa points out in [Kan01b], the (normalized) covariance matrix is equivalent to the inverse of the second derivative of a small neighborhood around the feature point. The second derivative corresponds to the inverse of the structure matrix $\boldsymbol{G}$ computed in equation (3.2). A detailed justification is given in [Kan01b]. This third feature weight, given as

$$\boldsymbol{\Sigma}^{-1}_{\boldsymbol{G},fn} = \boldsymbol{G}(\boldsymbol{q}_{fn}, \tau_f) \quad , \tag{3.31}$$

not only represents a scalar quality value as the preceding two, but encodes reliability along and perpendicular to the strongest edge in the feature window via its eigenvectors and -values. This property is outlined in Figure 3.7(a), while Figure 3.7(b) shows a real-world example.

In [Kan01b], weighting using the structure matrix $\boldsymbol{\Sigma_G}$ is applied to the estimation of homographies and fundamental matrices. As a conclusion, the authors state that only minimal

improvement is achieved if feature detection is already based on this criterion. Nevertheless, its performance will be compared to thresholding, outlier detection, scalar weights $\boldsymbol{\Sigma}_\lambda$ and $\boldsymbol{\Sigma}_{\boldsymbol{d}}$, and the following robust estimators with regard to the reconstruction of long image sequences in Section 5.2.

### 3.5.2 Robust Estimators

The method of minimizing the sum of squared differences (SSD) of an error vector in order to estimate the parameters of a non-linear equation, which is used in many instances during 3-D reconstruction, has one distinct disadvantage. It assumes a normal distribution of measurements, as is the case if only Gaussian noise is present. However, if outliers occur, such as mismatched point correspondences, their squared error has a disproportionate influence on the solution. LMedS is able to identify and discard such outliers even if they constitute up to 50 percent of the data. Its drawback is the computational cost of repeatedly evaluating random samples and the, albeit low, probability of missing outliers.

An alternative approach for handling outliers is *robust statistics* [Ham86]. Here, the quadratic error norm, $\boldsymbol{\epsilon}^{\mathrm{T}}\boldsymbol{\epsilon}$, is replaced by different norms, $\sum_{n=1}^{N}\rho(\epsilon_n)$, called *M-estimators*, which limit the influence of bad point correspondences. The two most well-known of these estimators are *truncated quadratic*,

$$\rho_{\mathrm{tq}}(x) = \begin{cases} x^2 & \text{if } |x| < \sqrt{k} \\ k & \text{otherwise} \end{cases} \quad , \tag{3.32}$$

and *Huber's minimax*,

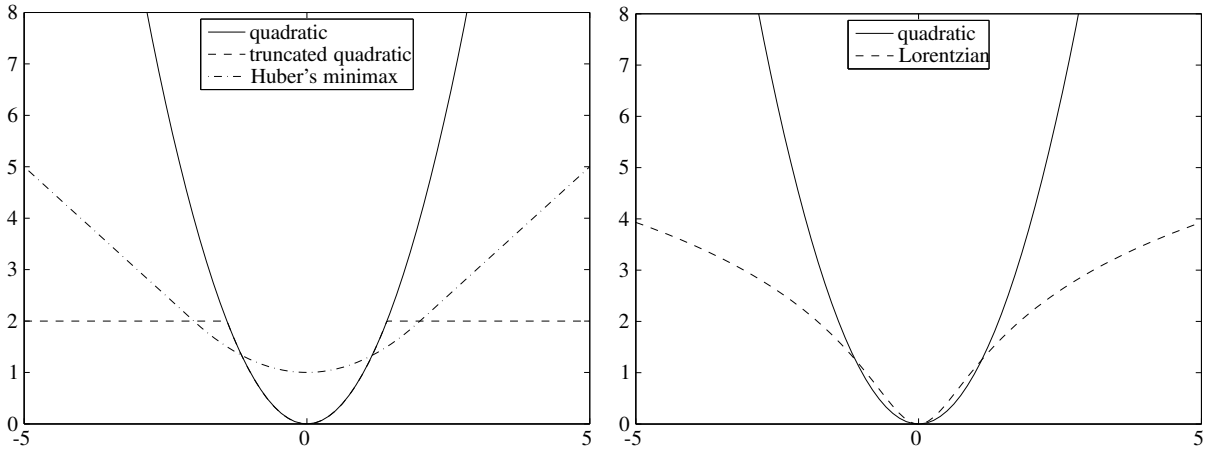
$$\rho_{\mathrm{H}}(x) = \begin{cases} x^2/2k + k/2 & \text{if } |x| \leq k \\ |x| & \text{otherwise} \end{cases} \quad , \tag{3.33}$$

which are both quadratic below a certain threshold and reduce the weight of $x$ above. A third function considered here is the logarithmic *Lorentzian* estimator

$$\rho_{\mathrm{L}}(x) = \log\left(1 + \frac{1}{2}\left(\frac{x}{k}\right)^2\right) \quad . \tag{3.34}$$

The three estimators are plotted in Figure 3.8 in comparison to the quadratic norm. A more comprehensive selection of robust estimators can be found in [Bla96].

Robust estimators may be applied anyplace where the back-projection error is minimized



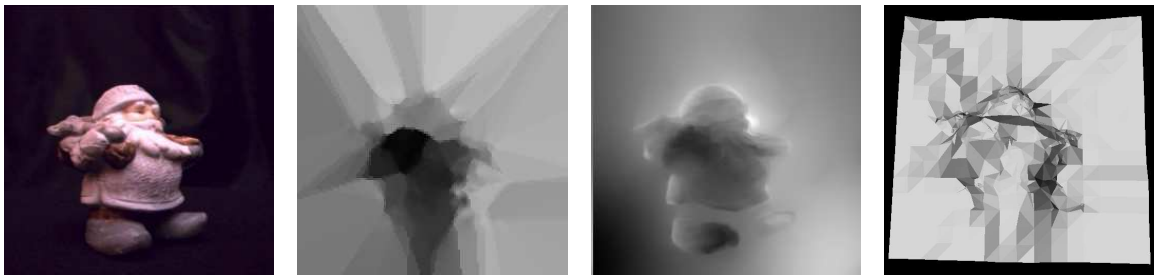

(a) (b)

Figure 3.8: (a) Truncated quadratic and Huber's minimax with $k = 2$ in comparison to least squares estimator, (b) Lorentzian estimator with $k = 0.5$.

in order to estimate scene or camera parameters. More precisely, this is the case during non-linear optimization of factorization results (cf. Section 3.3.3), extension to long image sequences (Section 3.4.1), i. e., camera parameter estimation as well as triangulation, and bundle adjustment (Section 2.3.3). Aanæs et al. [Aan02] proposed a factorization method based on Christy and Horaud's iterative approach which allots a weight to each feature according to one of the above robust estimators. However, this approach is not discussed further here, as robust non-linear optimization of factorization results has a similar effect.

## 3.6 Scene Geometry Reconstruction

As already described in Section 1.2.1, rendering a light field requires depth information in addition to the camera parameters, unless the scene was sampled densely as it is done in [Lev96]. Otherwise, ghosting artifacts may occur if the real depth differs too much from the assumed one (cf. Figure 1.5, page 7). The required sampling density for preventing aliasing or ghosting artifacts was analyzed by [Cha00]. For a sparse sampling when using a hand-held camera additional depth information is indispensable. It can be supplied either as dense depth maps or as a geometric proxy. The computation of these alternatives will be introduced in the following.



(a) (b) (c) (d)

Figure 3.9: Different depth representations of image (a): (b) interpolated from 3-D points, (c) variational approach and (d) a local proxy

### 3.6.1 Depth Maps

Computing a dense depth map for each of the input images using the 3-D points obtained during scene and motion reconstruction is straightforward. The 2-D features $\boldsymbol{q}_{fn}$ in an image $f$ with known 3-D position $\boldsymbol{p}_n$ yield a sparse depth map, where the depth is calculated as

$$z_{fn} = (\boldsymbol{p}_n - \boldsymbol{t}_f)^{\mathrm{T}} \boldsymbol{r}_{3,f} \quad . \tag{3.35}$$

$\boldsymbol{t}_f$ denotes the camera translation vector of this image, $\boldsymbol{r}_{3,f}$ the third column of the respective rotation matrix, which is the viewing direction of the camera. For every pixel in the image, a depth value is then calculated by determining the three closest points and averaging the corresponding depth values, weighted by the pixel's distance to each point. The result of such a simple and fast depth map computation is shown in Figure 3.9(b) for the input image of Figure 3.9(a). It can be seen that the resulting depth map is quite detailed in structured areas of the image, while it is coarse in rather homogeneous areas, where no features were detected and reconstructed, like in the background of the image.

The computation of a depth or disparity map from stereo images has been investigated in numerous publications and may lead to better results than the method above, although often at a higher computational cost. As an example we will use depth maps generated by a variational approach by [Alv02] as seen in Figure 3.9(c), which includes a linear scale-space approach as mentioned in [Wei99]. It makes use of the fundamental matrix $\boldsymbol{F}_{ij}$ for disparity estimation between two images with camera projection matrices $\boldsymbol{P}_i$ and $\boldsymbol{P}_j$ which is readily available from the preceding camera parameter reconstruction. If $\boldsymbol{P}_i$ (and similarly $\boldsymbol{P}_j$) is decomposed into

$\boldsymbol{P}_i = (\boldsymbol{X}_i|\boldsymbol{x}_i)$, with $\boldsymbol{X}_i \in \mathbb{R}^{3\times3}$ and $\boldsymbol{x}_i \in \mathbb{R}^3$, $\boldsymbol{F}_{ij}$ is computed as

$$\boldsymbol{F}_{ij} = [\boldsymbol{x}_j - \boldsymbol{X}_j\boldsymbol{X}_i^{-1}\boldsymbol{x}_i]_\times(\boldsymbol{X}_j\boldsymbol{X}_i^{-1}) \quad . \tag{3.36}$$

$[\boldsymbol{a}]_\times$ denotes the antisymmetric matrix performing an outer left multiplication by $\boldsymbol{a}$:

$$[\boldsymbol{a}]_\times = \begin{pmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{pmatrix} \quad . \tag{3.37}$$

However, computing $\boldsymbol{F}_{ij}$ from the reconstructed projection matrices has a drawback. The estimation process of the projection matrices is a least-squares minimization which is applied to the parameters of a large number of camera positions at once. By minimizing the overall error, the error for each individual camera increases with every additional image. Therefore, it is usually advantageous to compute $\boldsymbol{F}_{ij}$ directly using only the image features detected in the corresponding images by applying, e. g., the robust 8-point-algorithm of Section 3.3.1.

The algorithm returns two smooth disparity maps which encode the horizontal and vertical pixel shift from the first, the reference image, to the second. From these, a depth map is computed by applying triangulation to each pixel in the reference image. In order to decrease processing time, the analytical solution is applied here instead of the least-squares solution of an over-determined system of equations used in Section 3.4.1. It makes use of the fact that the shortest connection between two lines is orthogonal to both, i. e.,

$$(\eta\boldsymbol{d}_i + \boldsymbol{t}_i - \upsilon\boldsymbol{d}_j - \boldsymbol{t}_j)^{\mathrm{T}}\boldsymbol{d}_k = 0 \quad , \tag{3.38}$$

where $\boldsymbol{d}_k$ is the direction vector from the camera center $\boldsymbol{t}_k$ through image point $\boldsymbol{q}_k$, and $k = i, j$ for the left and right image, respectively. These two equations are solved for the unknowns $\eta$ and $\upsilon$. The mid point of the shortest connection is selected as the intersection of the two lines.

In some cases, the "artificial" smoothness of the disparity map leads to the undesired effect that the viewing rays through corresponding pixels are almost parallel, which results in areas with very large depth values. This is handled by setting a limit to the allowed depth range. This depth range is available from the reconstructed 3-D points. The resulting holes in the depth map are either set to the maximum depth value, or ignored entirely during rendering, which is achieved by a so-called *confidence map*, encoding the reliability of each pixel in an image. In the current state of the system, only reliabilities of either zero or one are handled.

An important factor for the quality of a depth map from disparity is the width of the stereo

base between the two images (cf. Section 2.1.3). Camera pairs with a narrow stereo base result in small disparities and thus near-parallel viewing rays for corresponding image points, which leads to unreliable depth estimates. In continuous video sequences, which entail a rather high image frequency, displacements tend to be small, so that it is advisable not to select adjacent frames as image pairs, but pairs which are farther apart. Additionally, quality can be increased by selecting multiple image pairs for one reference frame and averaging over the resulting depth maps. These properties are accounted for by the following basic selection scheme: for each image, two corresponding frames are selected which appear earlier and later in the image sequence and are as much as $10$ frames away if possible. The resulting depth maps are averaged pixel-wise, where pixels outside the above depth range are not considered.

### 3.6.2 Local Geometry Models

The term *geometric proxy*, for representing the geometric properties of a scene, was introduced in [Bue01] for unstructured lumigraph rendering. Instead of a depth map for each image, depth information is provided as a global geometric model. This allows much higher frame rates for rendering on graphics hardware since depth at each sampling point of the synthetic image is then obtained by casting a ray through the triangle mesh of the global proxy. Functions such as this are implemented very efficiently on graphics hardware. For local depth maps, i. e., one depth map per image, determining depth is done by first selecting a number of images which contribute most to the current view, intersecting each of these depth maps with the viewing ray of the sample point, and then determining the depth value which is closest to the observer. A more detailed description of this process can be found in [Vog05], but it is obvious that it is more time-consuming than if a proxy were available.

After scene reconstruction, geometry information is only available as a point cloud which is not necessarily globally consistent. For long sequences, the same feature may be present multiple times at different positions if it is lost and selected again several times during feature tracking, complicating the computation of a single geometry model. Generic methods exist for generating a triangle mesh from point clouds, such as [Wag01, Wag03]. However, they are ill-equipped for handling sparse point clouds with large numbers of, at times considerable, outliers. Holes in the geometry model are often the result, appearing again during light field rendering.

Therefore, instead of a global mesh the geometry is supplied analogously to local depth maps as so-called *local proxies*, one triangle mesh for each image. They are generated from the 3-D points visible in the respective image by first applying a 2-D Delaunay triangulation (cf. [For97]) to the corresponding 2-D features. For this step, the publicly available TRIANGLE

library, described in [She96], is used. The resulting 2-D mesh is transferred to 3-D space by adding a depth value to each vertex using equation (3.35). Optionally, a regular grid may be added to the local proxy before triangulation in order to extend it to the image borders and provide a more regular sampling. The depth values of the grid vertices are interpolated as in the previous section for dense depth maps. An example for such a local proxy is shown in Figure 3.9(d).

Local proxies constitute a compromise between depth maps and global proxies. They are not as accurate as the stereo depth maps introduced before, but they accelerate rendering since depth lookup in a triangle mesh is much faster than in a depth map. However, this lookup still has to be done in several meshes for each sample point, therefore computational cost is higher than for global meshes. A first comparison of the depth models introduced so far regarding light field quality and computation times was done in [Nie05].

### 3.6.3 Global Geometry Models

Generating a global proxy from a point cloud is, as mentioned previously, difficult due to inconsistency of the points, sparseness and outliers. However, the reconstruction process provides additional information which can be exploited to generate such global meshes, namely that the neighborhood relation of subsets of points in the point cloud is known from their visibility in common images. This greatly simplifies the generation of a global proxy for initial subsequences used for the initial factorization. The subsequent extension of such initial proxies to longer image sequences is discussed in the second part of this section.

#### Proxies for Initial Sequences

A characteristic of the factorization methods of the factorization methods of Section 2.2 is that all points $\boldsymbol{p}_n$ in the measurement matrix $\boldsymbol{W}$ have to be visible in each of the $F_{\boldsymbol{W}}$ images of the subsequence. From this follows that none of the $\boldsymbol{p}_n$ are occluded and that a valid 2-D triangle mesh for one image remains valid if the points are mapped to any of the other images of the subsequence. Valid means here that no intersecting edges exist in the mesh. Therefore, any local proxy for the images of the subsequence is also a global proxy for the whole subsequence if only the points in $\boldsymbol{W}$ are used.

The resulting mesh forms, in 2-D, a convex hull around the feature points. For light field rendering, this is insufficient since no depth information is given between the image borders and the mesh, and the virtual image is only rendered at points where depth is available. Therefore,

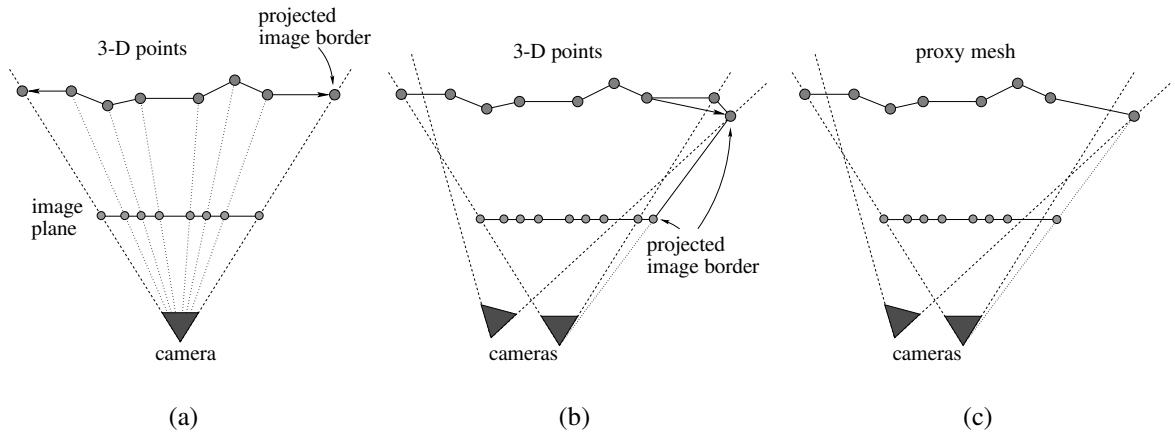



Figure 3.10: Generating a proxy for initial sequences: (a) initial local proxy with projected image border, (b) added image border for second camera, (c) final global proxy with border vertices inside the mesh removed.

the 3-D proxy has to be extended outwards until the projections of the borders of all images in the subsequence intersect the mesh.

In order to generate such a mesh, the following process is applied. First, an image $\boldsymbol{f}_0$ is selected from the subsequence, e. g., the first image. For this, a local proxy is constructed as in the previous section, with the difference that not a complete point grid is added to it, but only a number of equally spaced points on the image border. This is depicted in Figure 3.10(a) for a 2-D slice. For the following image, $\boldsymbol{f}_1$, the border points are projected onto the existing mesh. If the projection intersects the mesh, nothing more has to be done. If it misses the mesh, a depth value for the projection is extrapolated from the closest 3-D points in the mesh, and the resulting 3-D point is back-projected into image $\boldsymbol{f}_0$ using equation (2.11), as shown in Figure 3.10(b). Using all points in image $\boldsymbol{f}_0$, a new proxy is generated using Delaunay triangulation. In the last step, seen in Figure 3.10(c), all previous border points now located inside the proxy mesh are removed. This process is repeated for all images $\boldsymbol{f}_f$, $f = 2, \ldots, F_{\boldsymbol{W}} - 1$.

While the resulting global proxy is not usable for the whole sequence of input images, it is nevertheless very useful for the factorization method selection algorithm outlined in Section 3.3.3. Although it is not as detailed as a local proxy, it is generated quickly, requires little memory and thus takes little time to be transferred to graphics memory. There, it allows high frame rates for rendering so that the required quality measure can be computed efficiently.

### Proxies for Long Image Sequences

For the above proxy, features which are not part of the initial subsequence have not been considered. However, in the process of reconstruction it is desirable to use shorter feature trails as well, and it is necessary to use those visible only in different parts of the sequence. This second strategy of generating a global geometry model therefore deals with the insertion of new features into the model. It does not, however, demand visibility of the proxy for the whole of all images.

As opposed to conventional methods for generating triangle meshes from point clouds, neighborhood information about features is derived solely from the images they have been detected in. This has the advantage that these neighborhood relationships may assumed to be correct as long as no feature mismatches occur.

The initial point of proxy generation is, again, a local proxy generated for an arbitrary image $\boldsymbol{f}_0$ which is part of the initial subsequence. In the following, $\mathcal{F}_{\text{GP}}$ denotes the set of images whose features were already considered for the local proxy. It is thus initialized as $\mathcal{F}_{\text{GP}} = \{\boldsymbol{f}_f \,|\, 0 \leq f < F_{\boldsymbol{W}}\}$.

For adding a new 3-D feature point $\boldsymbol{p}_n$, two cases are distinguished, i. e., the new feature is located inside or outside the existing proxy. Which of these cases applies is determined by inspecting each triangle ("face") $\mathcal{T}^i$ of the existing proxy with respect to each image $\boldsymbol{f}_f \in \mathcal{F}_{\text{GP}}$. A triangle consists of a triple of vertices $(\boldsymbol{v}_{j_1}, \boldsymbol{v}_{j_2}, \boldsymbol{v}_{j_3})$. If all four points, $\boldsymbol{v}_{j_1}$, $\boldsymbol{v}_{j_2}$, $\boldsymbol{v}_{j_3}$ and $\boldsymbol{p}_n$ are visible in $\boldsymbol{f}_f$, the barycentric coordinates [Sla02] of $\boldsymbol{q}_{f,n}$ with respect to the projection of $\mathcal{T}^i$ into $\boldsymbol{f}_f$ are calculated as

$$b_1^i = ((v_{j2,0}^f - q_{f,n,0})(v_{j3,1}^f - q_{f,n,1}) - (v_{j3,0}^f - q_{f,n,0})(v_{j2,1}^f - q_{f,n,1}))/a \quad , \tag{3.39}$$

$$b_2^i = ((v_{j3,0}^f - q_{f,n,0})(v_{j1,1}^f - q_{f,n,1}) - (v_{j1,0}^f - q_{f,n,0})(v_{j3,1}^f - q_{f,n,1}))/a \quad , \tag{3.40}$$

$$b_3^i = 1 - b_1^i - b_2^i \quad , \tag{3.41}$$

where

$$a = (v_{j2,0}^f - v_{j1,0}^f)(v_{j3,1}^f - v_{j1,1}^f) - (v_{j3,0}^f - v_{j1,0}^f)(v_{j2,1}^f - v_{j1,1}^f) \tag{3.42}$$

and $\boldsymbol{v}_j^f$ denotes the projection of $\boldsymbol{v}_j$ into image $\boldsymbol{f}_f$. $\mathcal{T}^i$ is stored in a set of intersected faces $\mathcal{T}_{\text{in}}$ if all three coordinates $b_1^i, b_2^i, b_3^i \geq 0$. If at least one face was intersected the new point is treated as being inside the mesh.

However, the reverse conclusion, that the point is outside the mesh if no face was intersected, is not true. Therefore, a second test is performed, i. e., the 2-D feature $\boldsymbol{q}_{f,n}$ is projected onto the proxy for each image $\boldsymbol{f}_f$ it is visible in. If this yields an intersection of any triangle, the point is

| | | |
|---|---|---|
| Initial set: points $\{\boldsymbol{p}_{n'}\}$, $0 \leq n' < N_{\text{out}}$, outside mesh | | |
| | Select image $\boldsymbol{f}_{f_{\text{out}}}$ and $\{\boldsymbol{p}_m\} \subset \{\boldsymbol{p}_{n'}\}$ of points visible in $\boldsymbol{f}_{f_{\text{out}}}$ | |
| | Determine all vertices $\boldsymbol{v}_{b,i}$ of the proxy border which are visible in $\boldsymbol{f}_{f_{\text{out}}}$ | |
| | IF | Border edges $\mathcal{E}_b = \{(\boldsymbol{v}_{b,i}, \boldsymbol{v}_{b,j})\}$ connecting the $\boldsymbol{v}_{b,i}$ are not continuous |
| | THEN | Remove $\{\boldsymbol{p}_m\}$ from $\{\boldsymbol{p}_{n'}\}$ |
| | ELSE | Apply Delaunay triangulation to $\boldsymbol{p}_m$ and $\boldsymbol{v}_{b,i}$ (1) |
| | | Insert edges $\mathcal{E}_b$ into new mesh, reconnect intersected edges (2) |
| | | Identify subgraphs separated by edges in $\mathcal{E}_b$ (3) |
| | | Merge largest subgraph with proxy mesh and remove added vertices from $\{\boldsymbol{p}_{n'}\}$ (4) |
| | | Remove any points from $\{\boldsymbol{p}_{n'}\}$ which are now inside the proxy and add them appropriately (5) |
| UNTIL $\lvert\{\boldsymbol{p}_{n'}\}\rvert = 0$ | | |

Figure 3.11: Adding new features to the outside of a proxy mesh.

discarded. Only otherwise is it considered being outside the mesh.

**New feature inside mesh.** Entering a new feature inside the mesh is now straightforward. From $\mathcal{F}_{\text{GP}}$, the image $\boldsymbol{f}_{f_{\text{in}}}$ where the most faces from $\mathcal{T}_{\text{in}}$ are visible in is selected to operate in. All edges common to two faces are removed, so that the remaining, surrounding edges form a so-called *planar straight line graph* (PSLG, [She96]) together with the new point $\boldsymbol{q}_{f_{\text{in}},n}$. Performing a *constrained* Delaunay triangulation on this PSLG yields a new submesh including $\boldsymbol{q}_{f_{\text{in}},n}$, which can be inserted seamlessly into the existing proxy. The old triangles $\mathcal{T}_{\text{in}}$ are removed from the mesh.

**New feature outside mesh.** Instead of adding the $N_{\text{out}}$ new features outside the mesh one at a time, they are added together if possible. The objective is to attach a triangle mesh connecting the new points $\boldsymbol{p}_{n'}$, $0 \leq n' < N_{\text{out}}$, to the rim of the existing proxy mesh. This border of the mesh consists of all edges adjoining only one face, and all border edges form one, or several, circular graphs in 3-D. As this process will operate again in 2-D image space using Delaunay triangulation, the image $\boldsymbol{f}_{f_{\text{out}}}$ is selected from $\mathcal{F}_{\text{GP}}$ where most of the border edges are visible along with as many new points as possible, though at least one.

An overview over the following process is given in the structure chart of Figure 3.11. For image $\boldsymbol{f}_{f_{\text{out}}}$, the border features $\boldsymbol{v}_{b,i}$, $1 \leq i \leq V_b$, visible in it are determined as well as the subset of new points $\{\boldsymbol{p}_m\} \subset \{\boldsymbol{p}_{n'}\}$ visible in $\boldsymbol{f}_{f_{\text{out}}}$. The set of border edges $\mathcal{E}_b$ is composed of all pairs of vertices $(\boldsymbol{v}_{b,i}, \boldsymbol{v}_{b,j})$ forming an edge in the existing proxy mesh having only one

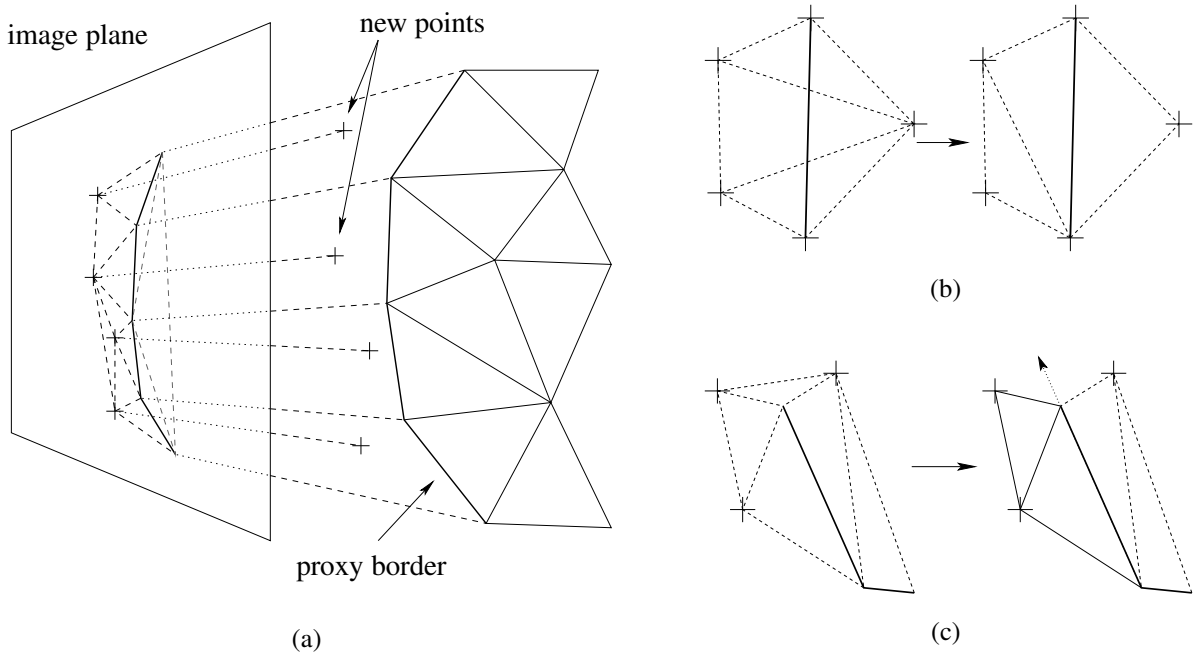



Figure 3.12: (a) Projection of the proxy border and points outside the mesh into the image plane. New points and border vertices are connected using Delaunay triangulation. Border edges are inserted afterwards. (b) Intersected edges are removed and a correct mesh is created. (c) Special case: partitioning of the mesh into subgraphs is forced if border ends inside the mesh.

adjacent face. A requirement for the algorithm to work is that $\mathcal{E}_b$ forms a single graph in 3-D, and is not partitioned into subgraphs. Otherwise, the undesired result of one edge having more than two adjacent faces may occur. In this case, the currently selected points are not added, and the algorithm continues with the remaining features $\{\boldsymbol{p}_{n'}\} \setminus \{\boldsymbol{p}_m\}$.

The remaining steps are performed on the 2-D feature positions of points $\{\boldsymbol{p}_m\}$, i. e., $\{\boldsymbol{q}_{fm}\}$, and border vertices $\boldsymbol{v}_{b,i}^f$ as depicted in Figure 3.12. The process is subdivided into the following five tasks:

1. In the 2-D space of $\boldsymbol{f}_{f_{\text{out}}}$, a Delaunay triangulation is applied on $\{\boldsymbol{q}_{fm}\}$ and $\boldsymbol{v}_{b,i}^f$ as shown in Figure 3.12(a).

2. The old border edges $\mathcal{E}_b$ are now inserted into the new mesh (Figure 3.12(a)). Since Delaunay triangulation is ambiguous, some of the edges in $\mathcal{E}_b$ may not exist in the new mesh, but intersect existing edges, as, e. g., in Figure 3.12(b). In this case, the same method as for inserting points inside the mesh is applied, i. e., the intersected edges and faces are removed and a PSLG consisting of the surrounding edges and the border edge is triangulated.

3. Usually, edges or even vertices are located on both sides of the projected image border. Therefore, the new mesh is partitioned into subgraphs delimited by $\mathcal{E}_b$. The special case sketched in Figure 3.12(c) has to be considered here: if an end vertex of the graph described by $\mathcal{E}_b$ is inside the mesh—being the case if the vertex has as many neighboring edges as faces—the mesh has to be subdivided nonetheless. The terminal border edge is therefore extended and all intersected edges and faces are removed from the mesh.

4. The subgraph containing the most points from $\{\boldsymbol{p}_m\}$ is now selected and merged with the existing global proxy mesh. If this selection were not done, the resulting mesh would end up with edges having three adjacent faces in 3-D, which is not admissible. The points added to the global proxy are removed from the set of remaining points $\{\boldsymbol{p}_n\}$.

5. The remaining points are tested again with respect to their location inside or outside the proxy and are added accordingly in the former case. For the points still outside the proxy the process is repeated until no points remain.

After thus adding the new points found in each consecutive image, there remains an optional check to be performed on the proxy border. As Delaunay triangulation creates a convex hull around the vertices, a concave outline of an object may be padded with additional faces. In order to identify these faces at the rim of the proxy, it is tested whether they occlude features which should otherwise be visible in some image. If so, the border face is removed. This process may be repeated until either no more faces are removed, or until a certain maximum number of iterations is reached.

As mentioned before, the advantage of this approach is that neighborhood relationships between vertices in the proxy are defined in 2-D image space from features actually visible in a common image. However, this also imposes quite stringent requirements on the features used. The longer a feature has been visible, the higher the possibility of finding an image where all neighboring points are visible, too. Therefore, it is recommended to use either feature prediction by back-projection, as described in Section 3.7.1, or non-sequential tracking as outlined in Section 3.2.3 to increase the length of feature trails. Additionally, it is advisable to set a lower limit on the length of these trails.

At present, the main limitation of these global proxies is that neither multiple, independent objects nor holes in one object can be modeled. Therefore, the algorithm requires further refinement and should be regarded as a starting point for further research.

| | |
|---|---|
| Initialize frame number: $f := 0$ | |
| | Track point features to frame $f$ and detect new ones |
| | $f := f + 1$ |
| UNTIL min. number of features $N_{\text{min}}$ visible in all frames reached or $f = F_{\text{init}} - 1$ | |
| Apply factorization method to frames $0, \ldots, f - 1$ | |
| WHILE $f < F$ | |
| | Track point features to frame $f$ and detect new ones |
| | Triangulate 3-D points and calibrate frame $f$ |
| | $f := f + 1$ |

Figure 3.13: Linear tracking and calibration over $F$ images in two steps: factorization of initial subsequence of maximum size $F_{\text{init}}$ and calibration of subsequent images

## 3.7 Information Feedback Loops

In order to reconstruct a light field from an image sequence, a number of processing steps are required, as outlined in Section 3.1. As it was also stated there, it is often possible to use the results of later steps to refine preceding ones in so-called feedback loops. Examples encountered so far are the automatic selection of factorization methods, as well as non-sequential tracking and reconstruction. In the following, four more examples will be outlined, the former two using information from the calibration step, the latter two using the reconstructed light field.

### 3.7.1 Feature Prediction by Back-Projection

The issue of features being lost due to occlusion or moving out of the camera's field of view was addressed already in Section 3.2.3 using non-sequential tracking. The drawback of this approach is that it requires a viewpoint mesh connecting the centers of projection of neighboring camera positions. If this mesh is generated using the algorithm of Section 3.4.2, tracking and calibration have to be done twice over. In addition to that, the mesh generation algorithm does not consider whether the images actually show similar views of the scene, but rather infers it from the distance of the respective cameras and their difference in viewing angles. However, the selection of the angle threshold is a heuristic choice and does not take into account that it depends on many factors like distance of the cameras, distance from the scene surface, and focal length.

Therefore, now the information which is actually seen by the camera, i. e., the feature points on the scene surface, is used to gain knowledge about image similarity. In addition, the sequence of image processing is changed in order to reconstruct the scene in a single pass, while information feedback from calibration to feature tracking remains possible. As outlined in Figure 3.13,

tracking and estimation of the next camera position are now done alternatingly image by image, instead of first tracking features over all images followed by an independent reconstruction for all images. Initialization by factorization is still done for a set of images at once, but always starting at the first image.

The problem of features moving out of the camera's field of view and reappearing is now handled by considering the back-projection of the corresponding 3-D point $\underline{\boldsymbol{p}}_n$ to the image $f_c$ currently processed. This is done after a first calibration of the current image's camera parameters, since the projection matrix $\boldsymbol{P}_{f_c}$ is required for back-projection. Features which were lost at an earlier stage and therefore not tracked to frame $f_c$, but whose back-projections fall within the camera's field of view, are then tracked again using the back-projected point $\widehat{\underline{\boldsymbol{q}}}_{f_c n} = \boldsymbol{P}_{f_c} \underline{\boldsymbol{p}}_n$ as initialization.

The last issue that needs to be solved is the selection of the source image the predicted feature is to be tracked from. Usually, not only one, but several features are to be retracked at once, and for each one an image has to be selected to be used as a template for feature window comparison. However, there may be no single image where all features have been visible before. Therefore, this question is important as it is desirable to use as few different source images as possible while successfully tracking as many features in as little time as possible. Three different strategies have been implemented and tested:

1. *Exhaustive search.* All images where at least one predicted feature was visible in are used. As alternatives, either all features are tracked from each source image, or only those are selected whose 2-D coordinates are closest to the prediction. In either case, the complexity increases linearly with the number of images processed in the worst case, as the cost for tracking a single feature is low compared to the overhead required for tracking from a new image due to the Gaussian pyramid that has to be computed.

2. *Highest number of features first.* The image $f_0$ with the highest number of features $\underline{\boldsymbol{q}}_{f_0 n}$ visible in is selected first. For the following selections, the features that have already been attempted to be tracked are not counted anymore, but nevertheless tracked again if necessary. The selection is stopped after a predefined number of images have been chosen. This guarantees that complexity stays constant with the number of images calibrated.

3. *Weighted selection.* Like before, a predefined number of source images is chosen, but using a weight value as selection criterion. Each image is given a weight

$$w_f = \sum_{n=0}^{N} \frac{1}{h_n + 1} \frac{d_{\max} - d(\widehat{\boldsymbol{q}}_{f_c n}, \boldsymbol{q}_{fn})}{d_{\max}} \quad , \tag{3.43}$$

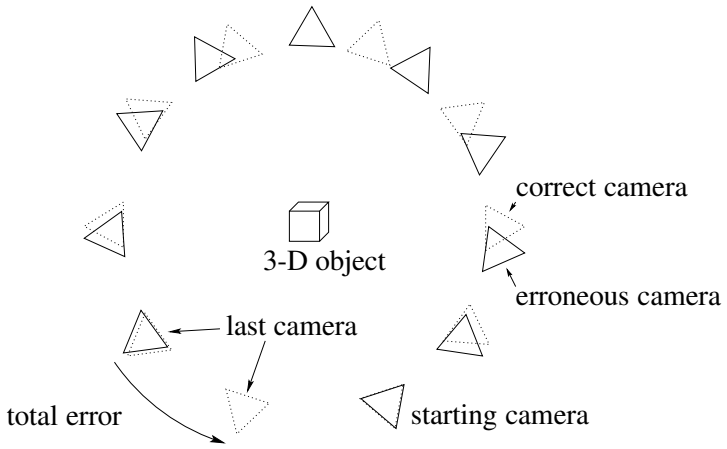



Figure 3.14: Example reconstruction of a camera path around an object. Correct camera positions are denoted by dotted triangles, erroneous ones by solid triangles.

where $h_n$ is the number of images previously selected where feature $n$ was visible in already, thus reducing the weight of features which were unsuccessfully tracked before. $d(\cdot,\cdot)$ denotes Euclidean distance and $d_{\max}$ is the maximum possible distance of two points in an image. The image given the highest weight is selected.

After tracking the predicted features, the reconstruction algorithm continues with feature tracking to the next image $f_c + 1$.

### 3.7.2 Closing Loops in Camera Movement

The main problem arising during the reconstruction of a long image sequence using the method of Section 3.4 is that, though errors may be small from one image to the next, they accumulate over a large number of images leading to inconsistencies in the geometry reconstruction. In the following, it is assumed that the camera is moved in loops around a scene, e. g., to view an object from every direction or to get a dense sampling. The following approach, which was already introduced in [Sch04c], was inspired by solutions in the field of simultaneous localization and mapping (SLAM) for robot navigation. Here, the goal is to generate a globally consistent map of the surroundings of a robot [Lu97], while the data from the robot's sensors, e. g., odometry and a camera, are unreliable. Consistency of the map can be established when the robot returns to a previous position and recognizes landmarks it has seen before. The accumulated error can then be determined and the rest of the map corrected accordingly. For the case of 3-D reconstruction, loops in camera movement are used to update the pose estimation for all previous images.

The problem of accumulating errors is demonstrated in Figure 3.14 where a camera moves

in a circle around an object taking $10$ images in the process. The correct camera positions are equally spaced around the object, but an error of only about four degrees from each camera to the next adds up to more than $35$ degrees. In order to get a correct reconstruction the circle must be closed again by removing this inconsistency.

We assume that $F$ camera positions form a loop and that camera $0$ follows again on camera $F-1$. Assuming further that a viewpoint mesh for non-sequential tracking as in Section 3.2.3 is available, features are tracked from image $F-1$ to image $0$, thus establishing a relationship between the two images. By applying the algorithm of Section 3.4 the displacement between these camera positions, $\Delta\boldsymbol{t}_{F-1}$, is calculated with a much higher accuracy than before when the accumulated error was included.

Using this new information the task of closing the loop is again formulated as an optimization process. For now, only the translation vector of each camera, $\boldsymbol{t}_f$, is considered. The displacement vector between two cameras is denoted by $\Delta\boldsymbol{t}_f = \Delta\widetilde{\boldsymbol{t}}_f = \boldsymbol{t}_{f+1} - \boldsymbol{t}_f$ for $0 \leq f < F-1$. Additionally, $\Delta\widetilde{\boldsymbol{t}}_{F-1}$ constitutes the current displacement vector between last and first camera while $\Delta\boldsymbol{t}_{F-1}$ is the corresponding target displacement calculated above. Thus, for $0 \leq f < F$ the $\Delta\boldsymbol{t}_f$ form the desired set of displacements while the $\Delta\widetilde{\boldsymbol{t}}_f$ are the displacements to be optimized. The residual vector is defined as

$$\boldsymbol{\epsilon}_L = \left( (\Delta\boldsymbol{t}_0 - \Delta\widetilde{\boldsymbol{t}}_0)^{\mathrm{T}}, (\Delta\boldsymbol{t}_1 - \Delta\widetilde{\boldsymbol{t}}_1)^{\mathrm{T}}, \ldots, (\Delta\boldsymbol{t}_f - \Delta\widetilde{\boldsymbol{t}}_f)^{\mathrm{T}} \right)^{\mathrm{T}} \tag{3.44}$$

and using the Levenberg-Marquardt algorithm the camera positions $\boldsymbol{t}_f, f > 0$, are optimized by minimizing the residual ${\boldsymbol{\epsilon}_L}^{\mathrm{T}}\boldsymbol{\epsilon}_L$. The first camera position $\boldsymbol{t}_0$ is kept unchanged.

This approach considers only the translation vector of each camera position, but does not alter the rotation of the cameras. For the example of Figure 3.14, the cameras now do not face exactly towards the center of the circle anymore. Therefore, the missing rotation to a full circle is incorporated into the computation of the residual vector. It is calculated as the rotation difference between the last and the first camera pose, $\Delta\boldsymbol{R} = \boldsymbol{R}_0{\boldsymbol{R}_{F-1}}^{\mathrm{T}}$. Lacking any other knowledge we assume that the $\frac{f}{F}$th part of this rotation, $\Delta\boldsymbol{R}_f$, is missing in each displacement vector. $\Delta\boldsymbol{R}_f$ is computed using spherical linear interpolation (cf. [Wat92]) on a quaternion representation of $\Delta\boldsymbol{R}$. Thus the new displacement vectors are computed as

$$\Delta\widehat{\boldsymbol{t}}_f = \Delta\boldsymbol{R}_f\boldsymbol{R}_f(\boldsymbol{t}_{f+1} - \boldsymbol{t}_f) \quad . \tag{3.45}$$

Usually an image sequence does not consist of exactly one revolution around an object. More circular camera movements may follow the first one, and in such cases it is not desired to change

the camera positions in a loop already closed before. From there on, the position of a camera once adjusted is kept untouched, and the algorithm above is only applied to later cameras.

Changing the camera positions renders the 3-D point positions invalid, so that they have to be recalculated. This is done by again minimizing the back-projection error during an optimization of the 3-D points.

Finally the result is again optimized globally using bundle adjustment as described in Section 2.3.3. The intrinsic parameters are assumed to be correct and bundle adjustment is only applied for the extrinsic parameters. If only some cameras of a loop were adjusted in the closing step before, only those are optimized now, too.

As mentioned before, if it is not known when a camera loop has been completed and the closing algorithm should be applied, a viewpoint mesh can be generated using the algorithm of Section 3.4.2 to gain this knowledge. An unsolved problem using this method is that large displacements are not detected, while the closing algorithm makes the more sense the larger the accumulated error. In those cases, prior knowledge about the image neighborhoods has to be used.

### 3.7.3 Optimizing Scene Geometry

Reduced quality of images rendered from a light field has two sources: inaccurately estimated camera parameters and wrong depth information. The resulting errors—blurring, distortions and superpositions—were already introduced briefly in Section 1.2.1. After introducing two feedback loops which are meant to reduce the error in camera parameters, the following approaches deal with improving depth information by means of rendered image quality, measured by the average squared pixel difference of equation (3.21). The objective is to improve the quality of either depth maps or a global proxy, two instances of a similar approach treated in [Sün05] and [Hop06] respectively, which will be summarized in the following.

#### Optimizing Depth Maps

Like in most optimization processes introduced so far, the Levenberg-Marquardt algorithm is used here again, although a number of adjustments have to be applied. The residual vector in this case contains the pixel-wise differences between a rendered image and the unused original image with the same camera pose (cf. Section 3.3.2). Basically, the residual vector for one depth

map to be optimized is given as

$$\boldsymbol{\epsilon}_{\text{DO}} = \begin{pmatrix} \boldsymbol{f}(\boldsymbol{x}_0) - \widehat{\boldsymbol{f}}(\boldsymbol{x}_0, z_0) \\ \boldsymbol{f}(\boldsymbol{x}_1) - \widehat{\boldsymbol{f}}(\boldsymbol{x}_1, z_1) \\ \vdots \\ \boldsymbol{f}(\boldsymbol{x}_{M-1}) - \widehat{\boldsymbol{f}}(\boldsymbol{x}_{M-1}, z_{M-1}) \end{pmatrix} \quad , \tag{3.46}$$

where $M$ is the number of pixels $\boldsymbol{x}_m$ in the image, and each pixel in the rendered image depends on the corresponding depth value $z_m$ at the same location. Thus, the residual vector is modified by changing the depth values.

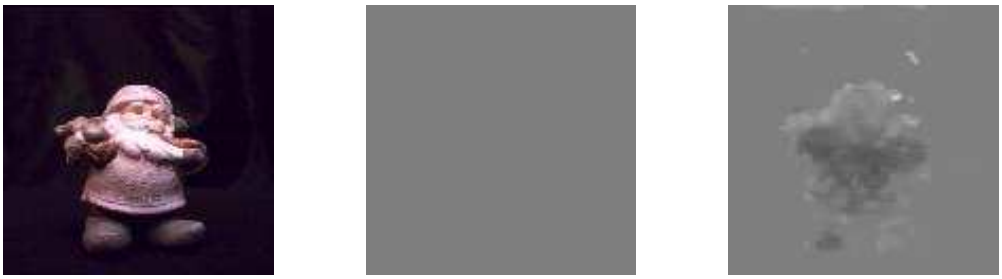
Two major problems have to be considered using this approach. Firstly, the Levenberg-Marquardt algorithm minimizes the residual vector by computing its Jacobian matrix $\boldsymbol{J}$, which is of size $M \times M$. Even for small images, this matrix becomes far too big in terms of memory requirement and computation time. Therefore, the image is subdivided into small rectangular windows and each one is processed independently. This reduces both the size of each individual Jacobian matrix and the number of parameters to be optimized at once.

The second problem is that while the Levenberg-Marquardt algorithm works on continuous functions, the residual vector $\boldsymbol{\epsilon}_{\text{DO}}$ will only contain discrete values, since pixel values represent quantized intensities. This fact affects the numerical calculation of the Jacobian matrix using finite differences, i. e., each entry is computed as

$$\frac{\partial \epsilon(\boldsymbol{x}_m, z_m)}{\partial z_m} = \frac{\epsilon(\boldsymbol{x}_m, z_m + \partial z_m) - \epsilon(\boldsymbol{x}_m, z_m)}{\partial z_m} \quad . \tag{3.47}$$

If $\partial z_m$ is chosen too small, the difference will not suffice to change the color value of pixel $\boldsymbol{x}_m$, so that $\epsilon(\boldsymbol{x}_m, z_m + \partial z_m) - \epsilon(\boldsymbol{x}_m, z_m) = 0$ and the optimization algorithm is stuck. Therefore, a larger depth difference $\partial z_m$ is chosen. In [Sün05], a guideline was given that the pixel displacement in the source images for rendering resulting from a depth displacement in the rendered image should be in the order of one pixel.

Although this method may generate good results in some cases, as shown in Figure 3.15, it suffers generally from the fact that a local optimization algorithm is applied to a problem with too many parameters. In many cases, it runs into a local minimum quickly and does not improve the results significantly, even if it is initialized using, e. g., one of the depth maps from Section 3.6.1. Therefore, the next step is to apply a similar method to a global proxy mesh which contains only very few nodes, thus reducing the size of the parameter vector.



(a) (b) (c)

Figure 3.15: A depth map (c) generated for the $128 \times 128$ pixel image (a) from a flat initial map (b). The image was subdivided into windows of size $4 \times 4$ pixels and the result was median filtered. Images taken from [Sün05].

### Optimizing Global Proxies

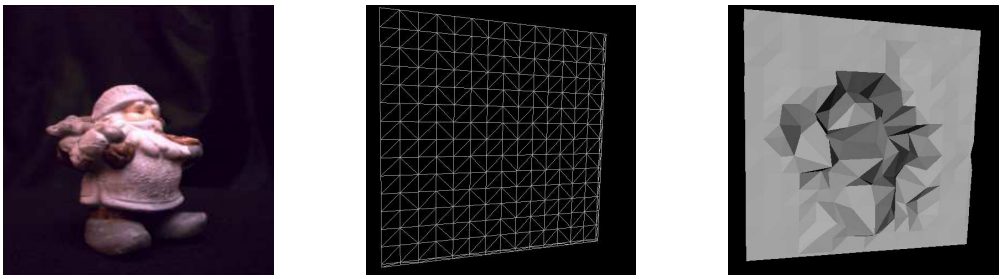
A global proxy, as defined in Section 3.6.3, constitutes a triangle mesh of $V$ vertices $\boldsymbol{v}^j$, $1 \leq j \leq V$. Instead of optimizing $M$ depth values $z_m$, one for each pixel, it is now sufficient to adjust the positions of the vertices in space. This leads to a $3V$-dimensional parameter vector

$$\boldsymbol{v}_J = (v_1^1, v_2^1, v_3^1, \ldots, v_1^V, v_2^V, v_3^V)^{\mathrm{T}} \quad . \tag{3.48}$$

In [Hop06], global proxies are considered which mostly consist of a regular grid of between $8 \times 8$ and $16 \times 16$ vertices. The size of the parameter vector is thus two to three orders of magnitude smaller than in the case of depth map optimization. The locally convergent Levenberg-Marquardt algorithm is much better equipped to handle these reduced parameter vector sizes. On the other hand, the error vector $\boldsymbol{\epsilon}_{\mathrm{GP}}$ is defined analogously to $\boldsymbol{\epsilon}_{\mathrm{DO}}$ given in equation (3.46), containing the same number of entries as well. Consequently, the Jacobian matrix $\boldsymbol{J}_{\mathrm{GP}}$ retains its size of $M \times M$ entries. However, the sparsity of $\boldsymbol{J}_{\mathrm{GP}}$ is now exploited which is beneficial regarding its inversion, applied once during each optimization step.

Contrary to the optimization of local depth maps, it is now possible to optimize a proxy at once for a complete image, using all its vertices. The example in Figure 3.16 shows the optimization of a global proxy from a regular, flat initialization for the same sequence as in Figure 3.15, though the image size of $256 \times 256$ is not reduced here. Initialization with an approximate geometry model is possible as well and likely to yield even better results.

Although the optimization of a global proxy regarding image difference yields good overall results, the current approach retains some drawbacks with potential for considerable improve-



(a) (b) (c)

Figure 3.16: A global proxy (c) generated from a flat initial proxy (b) with $16 \times 16$ vertices. A single reference image (a) was used and $5$ iterations of the Levenberg-Marquardt algorithm were applied.

ment. Currently, no precautions are taken to prevent vertices to be moved in such a way that the connected triangles overlap. While this has no visual effect on rendering quality, it is an undesired effect. The relative influence of vertices is shifted, up to the point where individual vertices have no influence at all on image quality. This effect may be prevented by displacing vertices only perpendicularly to the plane of the image currently considered, thereby also reducing the size of the parameter vector $\boldsymbol{v}_J$. However, this approach remains to be implemented and verified.

### 3.7.4 Extending Existing Light Fields

So far, the described methods for light field reconstruction, especially feature tracking, assume that camera motion between two consecutive images is quite low and that the search range can thus be restricted considerably. The drawback of this assumption is that once the recording of an image sequence has been completed, finding feature correspondences in any new images or image sequences which were taken later from a different view point poses a problem. For the light field, this means that once it has been reconstructed from one image sequence, it is difficult to extend it in order to, e. g., cover a broader viewing range or improve its quality by adding more images. It is therefore necessary to at least find the most similar image in the light field to the one to be added. Since the disparity between those two images may be too large for robust feature tracking, it is desirable to additionally supply a good estimate for the new feature positions.

In the following, two methods originally introduced in [Deu04] are presented which are able to determine both the closest image and a feature position estimate. The first one is based on SIFT features as described in Section 3.2.2. The second approach uses the rendered images from the initial light field as feedback for a parameter search. As position estimates, the SIFT feature matching yields a 2-D homography which describes the transformation from the closest to the

new image. The parameter search returns a camera pose estimation which can be used to predict features by back-projection of known 3-D points.

### Matching with SIFT Features

In this approach, determining the most similar image to a new one will be done by a full comparison with all known images already in the light field. The 2-D homography for subsequent tracking is calculated using the SIFT feature correspondences.

**Image Matching.** The technique for estimating which of the images of the first sequence is most similar to the first image $\widetilde{\boldsymbol{f}}$ (test image) of the second sequence is based on majority voting. For every image $\boldsymbol{f}_i, i = 1, 2, \ldots, F$, in the first sequence, a set of SIFT features $\mathcal{C}_i = \{\boldsymbol{c}_i^1, \boldsymbol{c}_i^2, \ldots, \boldsymbol{c}_i^{N_i}\}$ is calculated. The value $N_i$ denotes the number of SIFT features which were detected by the SIFT feature point detector in image $\boldsymbol{f}_i$. Similarly, the set $\widetilde{\mathcal{C}} = \{\widetilde{\boldsymbol{c}}^1, \widetilde{\boldsymbol{c}}^2, \ldots, \widetilde{\boldsymbol{c}}^{\widetilde{N}}\}$ is computed for the test image $\widetilde{\boldsymbol{f}}$. For counting the votes, an accumulator

$$a_f = \sum_{n=1}^{\widetilde{N}} \delta(f - \operatorname*{argmin}_i \min_j d(\boldsymbol{c}_i^j, \widetilde{\boldsymbol{c}}^n)) \tag{3.49}$$

is used, where $\delta$ is the Kronecker delta function, $a_f$ is the accumulator entry for image $\boldsymbol{f}_f$ and $d(\cdot, \cdot)$ is the Euclidean distance of two SIFT features. The index $b$ for the best matching image $\boldsymbol{f}_b$ is retrieved from the accumulator by $b = \operatorname*{argmax}_f a_f$.

**Homography Estimation.** Even if $\boldsymbol{f}_b$ is found to be the most similar image in the light field to the new image $\widetilde{\boldsymbol{f}}$, feature tracking may still have to solve large displacements which surpass the usual basin of convergence. In order to deal with any rotation, translation and scale differences between the two images, the SIFT feature correspondences found previously are used to estimate a homography (cf. [Har03]), which yields a position prediction for tracking feature points in image $\widetilde{\boldsymbol{f}}$. For this purpose, we calculate a set of the coordinates of the $N_b$ best matching SIFT features in image $\boldsymbol{f}_b$ and $\widetilde{\boldsymbol{f}}$, i. e., $\mathcal{M} = \{(\underline{\boldsymbol{x}}_1, \underline{\widetilde{\boldsymbol{x}}}_1), (\underline{\boldsymbol{x}}_2, \underline{\widetilde{\boldsymbol{x}}}_2), \ldots, (\underline{\boldsymbol{x}}_{N_b}, \underline{\widetilde{\boldsymbol{x}}}_{N_b})\}$. Assuming that the transformation of $\underline{\boldsymbol{x}}_i$ to $\underline{\widetilde{\boldsymbol{x}}}_i$ is a projective transformation, the homography is given as

$$\boldsymbol{M}_2 = \boldsymbol{H}\boldsymbol{M}_1 \quad , \tag{3.50}$$

where matrix $\boldsymbol{H} \in \mathbb{R}^{3\times3}$ is the homography matrix and

$$\boldsymbol{M}_1 = (\underline{\boldsymbol{x}}_1, \underline{\boldsymbol{x}}_2, \ldots, \underline{\boldsymbol{x}}_{N_b}) \quad , \tag{3.51}$$

$$\boldsymbol{M}_2 = (\underline{\widetilde{\boldsymbol{x}}}_1, \underline{\widetilde{\boldsymbol{x}}}_2, \ldots, \underline{\widetilde{\boldsymbol{x}}}_{N_b}) \quad . \tag{3.52}$$

The overdetermined linear equation system of equations (3.50) is solved using a least squares estimation, namely the pseudo-inverse (cf. [Pre93]), provided that $N_b > 3$. Finally, $\boldsymbol{H}$ is used to estimate the positions of feature points in $\widetilde{\boldsymbol{f}}$ by mapping their 2-D positions from $\boldsymbol{f}_b$ to $\widetilde{\boldsymbol{f}}$.

### Parameter search by rendering feedback

In the previous approach, the camera parameter estimation has only used the original images of the existing light field for image matching. The following method, however, makes use of images rendered from arbitrary view points using the light field itself.

For any set of camera pose parameters $\boldsymbol{\rho}$, the light field can be used to generate the image $\boldsymbol{f}_{\boldsymbol{\rho}}$ corresponding to those parameters. $\boldsymbol{\rho}$ contains the six extrinsic parameters of the camera, i. e., its rotation and translation. This generated image can be used to search for the camera parameters $\widetilde{\boldsymbol{\rho}}$ of the image $\widetilde{\boldsymbol{f}}$ to be added by searching for the $\boldsymbol{\rho}$ such that $\boldsymbol{f}_{\boldsymbol{\rho}}$ is *closest* to $\widetilde{\boldsymbol{f}}$.

Using a distance metric to compare images, the parameter search becomes a global minimization problem. For this, the sum-of-squared-differences (SSD), analogous to equation (3.21), is used. SSD's main drawback is its lighting dependence, however to extend a light field the new images are assumed be recorded with the same lighting conditions.

Several optimization algorithms can be applied to this problem, of which two (related) approaches were adopted: adaptive random search (ARS) [Tör89], and the particle filter (PF), specifically the Condensation algorithm [Isa98]. Both approaches maintain a current set $\mathcal{R}_t$ of camera parameter *hypotheses* $\boldsymbol{\rho}_{t,i}, i = 1, \ldots, |\mathcal{R}_t|$, where $t$ is an iteration index and $|\mathcal{R}_t|$ the fixed size of the set. Each hypothesis $\boldsymbol{\rho}_{t,i}$ also has a scalar rating $\varrho_{t,i}$ which is derived from the image comparison. The initial set $\mathcal{R}_0$ can be derived from the camera parameters of the first sequence for a global search, or clustered around an initial estimate. Both approaches iterate over this set several times to fine-tune it, by generating new hypotheses $\boldsymbol{\rho}_{t+1,i}$ and ratings $\varrho_{t+1,i}$ from the current ones. The methods differ in how the new hypothesis set is generated.

For ARS, the rating is the same as for the SSD distance measure. $\mathcal{R}_{t+1}$ is generated from $\mathcal{R}_t$ by discarding the worse rated half of all $\boldsymbol{\rho}_{t,i}$, and replacing them with diffused copies of the better rated half. The diffusion is an additive Gaussian noise, the standard deviation of which is derived from the spread of the original camera parameters. This standard deviation is reduced on

each iteration to decrease the search area.

Unlike ARS, the PF algorithm uses a probabilistic framework. The rated hypothesis set represents the probability density function (PDF) of the camera position, $p(\boldsymbol{\rho}_t|\widetilde{\boldsymbol{f}})$, given the target image. Such rated hypotheses are also called *particles*. Applying Bayes' formula yields

$$p(\boldsymbol{\rho}_t|\widetilde{\boldsymbol{f}}) = \frac{1}{c}p(\widetilde{\boldsymbol{f}}|\boldsymbol{\rho}_t)p(\boldsymbol{\rho}_t) \quad , \tag{3.53}$$

with $c$ a normalizing constant. Thus, we seek information about the camera parameters $\boldsymbol{\rho}_t$ corresponding to the new image $\widetilde{\boldsymbol{f}}$. As with ARS, the initial a priori density $p(\boldsymbol{\rho}_0)$, represented by $\mathcal{R}_0$, can be uniformly distributed over the search space or clustered around an initial estimate $\boldsymbol{\rho}_0$.

The a priori density $p(\boldsymbol{\rho}_t)$ is derived from the previous a posteriori density $p(\boldsymbol{\rho}_{t-1}|\widetilde{\boldsymbol{f}})$ through

$$p(\boldsymbol{\rho}_t) = \int p(\boldsymbol{\rho}_t|\boldsymbol{\rho}_{t-1})p(\boldsymbol{\rho}_{t-1}|\widetilde{\boldsymbol{f}})\, d\boldsymbol{\rho}_{t-1} \quad . \tag{3.54}$$

In typical particle filter usage, this models the noisy state transition over time. Since the state is not expected to change, this is merely a diffusing process, as with ARS.

The *likelihood* $p(\widetilde{\boldsymbol{f}}|\boldsymbol{\rho}_t)$ is derived from the image comparison by constructing a Gibbs distribution [Bla92]:

$$p(\widetilde{\boldsymbol{f}}|\boldsymbol{\rho}_t) = \frac{1}{k}\exp\left(-\mu E(\widetilde{\boldsymbol{f}}|\boldsymbol{\rho}_t)\right) \tag{3.55}$$

with $k$ a normalizing constant. For each hypothesis $\boldsymbol{\rho}_{t,i}$, the rating is $\varrho_{t,i} = p(\widetilde{\boldsymbol{f}}|\boldsymbol{\rho}_{t,i})$. The term $E(\widetilde{\boldsymbol{f}}|\boldsymbol{\rho}_t)$ is an error energy, comparing the target image with the image corresponding to the hypothesis $\boldsymbol{\rho}_t$. The better the hypothesis image matches the target, the lower the energy should be. The image comparison metric used, the SSD over the image space, has such a property, and is used unchanged for $E(\widetilde{\boldsymbol{f}}|\boldsymbol{\rho}_t)$. However, this may result in very similar energies for all compared images, which erodes the particle filter's effectiveness. Multiplying the SSD by a scalar value $\mu > 1$ requires a hypothesis to be much closer for a good rating.

Using this likelihood definition as a hypothesis rating, the PF solves the combination of equations (3.53) through (3.55) using Monte Carlo integration. A new set $\mathcal{R}_{t+1}$ is derived from $\mathcal{R}_t$ by sampling from the latter, using the ratings as a sampling probability, and then re-rating $\mathcal{R}_{t+1}$. This new set represents the PDF whose main mode is the hypothesis $\widetilde{\boldsymbol{\rho}}$ with the lowest image discrepancy from $\widetilde{\boldsymbol{f}}$.

There are two caveats with this method. If the optimization starts from a single initial state estimate, the standard deviation of the particle diffusion must be chosen large enough so that the particles search beyond any local minima. Secondly, the results may by biased due to the

rendering of light field images. An image rendered from a light field will often exhibit local distortions, causing the minimum to diverge slightly from the true camera parameters of $\widetilde{\boldsymbol{f}}$.

Thus, both ARS and PF determine an estimate $\widehat{\boldsymbol{P}}_{\widetilde{\boldsymbol{f}}}$ for the projection matrix of $\widetilde{\boldsymbol{f}}$. Using back-projection, features in $\widetilde{\boldsymbol{f}}$ can now be predicted. The question remaining is which base image is to be used to track the features from. Several approaches are possible, i. e., using the estimate $\boldsymbol{f}_b$ of the previous SIFT approach, determining the most similar camera pose using the method for building a camera mesh of Section 3.4.2, or the most similar image based on the difference of back-projected features of equation (3.43).

# Chapter 4

# Dynamic Light Fields

In this chapter, one of the restrictions made so far will be suspended, namely that the recorded scene has to be static, with no movement being visible at all. Instead, it will now be possible that parts of the scene or single objects in it are in motion between or during recordings. These *dynamic* scenes open up an entirely new domain of problems to be solved, especially since it is still assumed that only one hand-held camera is available for sampling the scene. The resulting light fields are termed, consequently, *dynamic light fields*.

A characterization of this new problem domain will be given in the first section of this chapter, where the new requirements to reconstruction as well as rendering of the final light field are defined. The following two Sections, 4.2 and 4.3, will introduce and deal with the reconstruction of different kinds of dynamic light fields. Step-wise static light fields assume that several image sequences of a scene were recorded, where each represents one time step of the motion. In contrast to that, the second type of light field allows motion during recording, but restricts it to objects which are rigid in themselves. The final section of this chapter introduces techniques for extending existing rendering techniques to accommodate these new kinds of light fields.

## 4.1 From Static to Dynamic: Additional Requirements

While the static light field constitutes a simplification of the plenoptic function—as introduced in Section 1.2—from a 7-D function to 4-D, the dynamic light field reintroduces the time parameter $\tau$ and thus increases the parameter space again to 5-D. In order to handle this increased complexity, new techniques have to be implemented in terms of reconstruction, data management, and rendering, as characterized in the following.

**Reconstructing dynamic light fields.** As before, it is still assumed that the images for light field reconstruction are recorded with a single hand-held camera. If nothing is known about scene motion, classic structure-from-motion and calibration algorithms such as factorization or bundle adjustment are no longer applicable. On the other hand, if nothing is known about the recording camera no information about scene geometry can be inferred through, e. g., triangulation. Some additional knowledge about either scene or camera motion has to be presumed in order to overcome this deadlock.

Both step-wise and rigid object approaches to dynamic light field reconstruction introduced in this chapter assume that the recorded scene is only partially in motion. Thus, the images show a considerable part of the background which is static. Feature points detected on the background therefore constitute reliable data for 3-D reconstruction, and the task is reduced to discerning *static* features on the background from *dynamic* features on objects in motion.

A further simplifying constraint is applied for step-wise dynamic light fields concerning the recording process. Here, it is assumed that scene and camera motion are mutually exclusive, i. e., the scene changes only if the camera is motionless and vice versa.

**Memory management.** As with video sequences in contrast to still images, dynamic light fields require a considerably greater amount of memory or storage space than static light fields due to the higher dimensionality. How much more memory is required depends mainly on the amount of motion in the scene, and thus the number of time steps the motion is divided into.

Again, like for video, the information stored in a dynamic light field may contain a lot of redundancy for those parts of the scene which are not in motion. In order to cover a certain amount of time and motion for the dynamic parts of the scene, sampling of the static parts will increase likewise. Identifying this redundancy would be the first step to more efficient coding of dynamic light fields.

However, even though encoding of dynamic light fields turns into an ever more important area of research, it will not be considered in this thesis. Efficient compression strategies for static light fields with images positioned on a regular grid have been examined, e. g., by Magnor in [Mag00, Mag01]. A similar approach was extended to dynamic light fields recorded with multiple but immobile cameras in [Fec05]. The irregular camera position setup resulting from hand-held camera movement has so far not been considered in current research, but offers a challenge for its own branch of research in video coding.

**Visualization.** Visualizing a dynamic light field requires some additional functionality. As a first approach, the reconstructed light fields will be divided into time steps which show consecu-

tive, previously observed states of the dynamic scene. In order to visualize these, the light field renderer has to be able to switch back and forth efficiently between these time steps. This requires a suitable labeling of the input images or subimages with time stamps, and the ability of the renderer to switch between these subsets of the data with little delay.

Starting from these rendering techniques, it is desirable that even time steps between the labeled ones were synthesized by the renderer, so that a continuous viewing of the scene through time is possible. On the other hand, if the motion of rigid objects as in Section 4.3 is considered, being able to visualize them independently from each other is another option. However, in-depth techniques for rendering dynamic light fields will neither be a topic of this thesis, although some suggestions will be made how to handle light field data labeled accordingly during 3-D reconstruction.

## 4.2 Step-Wise Static Light Fields

The first type of dynamic light field described here, which was first introduced in [Sch02] and [Sch04b], is called *step-wise static* as the dynamic scene is divided into $k$ time steps where each time step is modeled with a complete static light field of the scene. Thus, no assumption is made about the rigidity of the objects in the scene—rigid as well as deformable objects may thus be modeled.

The input images used in the following to reconstruct a dynamic light field need to fulfill two main requirements. Firstly, one image sequence must be available for each time step so that the $k$ static light fields can be reconstructed from them. Secondly, for two consecutive image sequences the last camera of the first sequence must have approximately the same pose as the first camera of the second sequence, which means that the two sequences have one camera position in common. In practice, the camera is moved over the scene while the objects in it stay immobile, and is kept steady while the objects are moved to the next position. This second part, where the objects are moved, is cut away, leaving the desired input image sequences. Thus, the images seen from the common camera position of two sequences show a different state of the moving object in the scene. This different image content poses the main difficulty for registration.

The dynamic light field is reconstructed from this input data by first calibrating and reconstructing the individual image sequences and then registering the resulting threads of camera positions with each other. Finally a refinement step can be applied which calibrates all cameras together. The assumption which must be made in order for this last step to work is the one already made in Section 4.1, being that dynamic objects only occupy the lesser part of the visible

scene. This means that the background of the scene covers the major part of each image so that more point correspondences are found on static parts of the scene than on moving ones. The three reconstruction steps will be described in the following.

### 4.2.1 Static Light Field Reconstruction

Each image sequence is calibrated independently using the reconstruction techniques described in Chapter 3. Following feature detection and tracking for the complete sequence, an initial subsequence is calibrated using one of the factorization processes of Section 3.3.3. Usually, the third processing chain of perspective factorization with self-calibration using an affine solution is selected as it is both reliable and accurate. This is followed by an extension of the reconstruction to the whole image sequence according to Section 3.4.1.

For simplicity, the intrinsic camera parameters are assumed to be the same for all images in all sequences, and, furthermore, they are assumed to be known at least approximately. The error arising from this assumption has proved to be tolerable, even if deviations were well above ten percent of the true value.

After 3-D reconstruction for each of the image sequences, the information for reconstructing the individual static light fields is available. However, the coordinate systems of the reconstructed camera poses for two image sequences now differ from each other by a rotation, a translation and an unknown scale factor. Before the individual light fields can be combined to form a dynamic light field, they need to be registered with each other using the steps introduced in the following.

### 4.2.2 Registration

Rotation and translation can be determined using the fact that two camera poses of a pair of consecutive image sequences are approximately the same. The transformation is done by first mapping one of the two cameras into the origin of its coordinate system and then to the pose of the other camera. If the $3 \times 4$ projection matrix of the second camera is given as $\boldsymbol{P}_2$, and the rotation matrix and translation vector of the two cameras as $\boldsymbol{R}_1$, $\boldsymbol{R}_2 \in \mathbb{R}^{3\times3}$ and $\boldsymbol{t}_1$, $\boldsymbol{t}_2 \in \mathbb{R}^3$, respectively, the transformation is obtained as follows:

$$\boldsymbol{P}'_2 = \boldsymbol{P}_2 \begin{pmatrix} \boldsymbol{R}_2^{\mathrm{T}} & -\boldsymbol{R}_2^{\mathrm{T}}\boldsymbol{t}_2 \\ \boldsymbol{0}_3^{\mathrm{T}} & 1 \end{pmatrix}^{-1} \begin{pmatrix} \boldsymbol{R}_1^{\mathrm{T}} & -\boldsymbol{R}_1^{\mathrm{T}}\boldsymbol{t}_1 \\ \boldsymbol{0}_3^{\mathrm{T}} & 1 \end{pmatrix} \quad . \tag{4.1}$$

The result $\boldsymbol{P}'_2$ denotes the new projection matrix in the coordinate system of the first sequence. The same transformation is then applied to all remaining $F_2 - 1$ cameras of the second

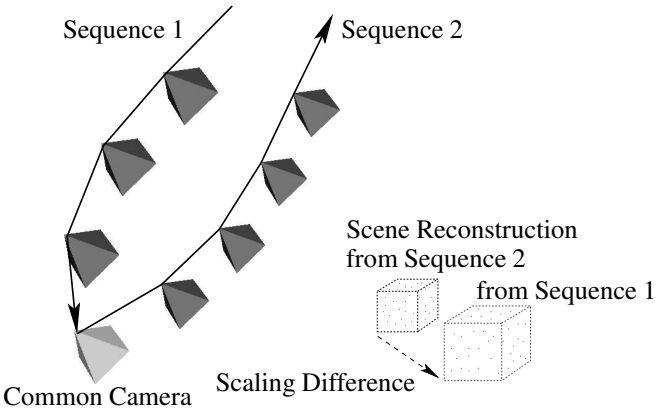



Figure 4.1: Registration of two image sequences. The incorrect scaling results in different distances of the reconstructed point clouds of the scene from the camera positions.

sequence, denoted in the following by $\boldsymbol{P}'_{f_2}$, $f_2 \in \{2, \ldots, F_2\}$. The inverse of this transformation is applied to each of the $N_2$ 3-D points $\underline{\boldsymbol{p}}_{n_2}$, $n_2 \in \{1, \ldots, N_2\}$, of the second sequence to convert them in the same way:

$$\underline{\boldsymbol{p}}'_{n_2} = \begin{pmatrix} \boldsymbol{R}_1^{\mathrm{T}} & -\boldsymbol{R}_1^{\mathrm{T}}\boldsymbol{t}_1 \\ \boldsymbol{0}_3^{\mathrm{T}} & 1 \end{pmatrix}^{-1} \begin{pmatrix} \boldsymbol{R}_2^{\mathrm{T}} & -\boldsymbol{R}_2^{\mathrm{T}}\boldsymbol{t}_2 \\ \boldsymbol{0}_3^{\mathrm{T}} & 1 \end{pmatrix} \underline{\boldsymbol{p}}_{n_2} \quad . \tag{4.2}$$

Figure 4.1 depicts such a registration of two image sequences using a common camera pose. It also shows the effect of the scaling step which has not yet been applied: on the one hand it results in different distances of the two 3-D point clouds, one for each image sequence, from the camera positions. On the other hand, the distances between the cameras positions are scaled by the same factor, which is indicated by smaller camera symbols for the second sequence in addition to the smaller spacing between them.

The scale factor is obtained by considering the centers of mass of the 3-D points in each image sequence. As the sequences were taken of the same scene, the centers of mass are assumed to be 3-D points which are roughly at the same position in the scene. If the camera is moved on approximately the same path for two consecutive image sequences this assumption holds, since the features selected by the tracking algorithm will be similar. The scale factor $\check{s}$ is computed as the ratio of the distances of the centers of mass from the two equal camera poses of two consecutive sequences.

Once this ratio is known all cameras and 3-D points of the respective second sequence are scaled to be registered correctly with the first sequence. Again, the projection matrices of all cameras are first transformed such that the common (first) camera position is moved to the origin

of the coordinate system, scaled by a matrix

$$\boldsymbol{S} = \begin{pmatrix} \check{s} & 0 & 0 & 0 \\ 0 & \check{s} & 0 & 0 \\ 0 & 0 & \check{s} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.3}$$

and then moved back by the inverse first transformation. This way, the first camera pose stays the same, which is equal to that of its counterpart in the first sequence. Thus, this transformation can be written as follows:

$$\boldsymbol{P}''_{f_2} = \boldsymbol{P}'_{f_2} \begin{pmatrix} \boldsymbol{R}_2^{\mathrm{T}} & -\boldsymbol{R}_2^{\mathrm{T}}\boldsymbol{t}_2 \\ \boldsymbol{0}_3^{\mathrm{T}} & 1 \end{pmatrix}^{-1} \boldsymbol{S} \begin{pmatrix} \boldsymbol{R}_2^{\mathrm{T}} & -\boldsymbol{R}_2^{\mathrm{T}}\boldsymbol{t}_2 \\ \boldsymbol{0}_3^{\mathrm{T}} & 1 \end{pmatrix} \quad . \tag{4.4}$$

It is applied again to each of the $F_2$ camera poses in the second sequence. $\boldsymbol{R}_2$ and $\boldsymbol{t}_2$ without the index $f_2$ denote the rotation matrix and translation vector of the first (common) camera. The scaling of the 3-D points is done analogously to equation (4.2).

### 4.2.3 Refinement

After transformation of the camera poses of all individual static light fields to the same coordinate system, a further refinement of the calibration is performed. This refinement compensates for the errors made during registration considering the assumptions made during scaling and the fact that the respective two camera poses used for registration will not be exactly identical.

As it was proposed in Section 3.4.2, the camera positions are combined in a 3-D mesh in which neighbors with similar views on the scene can be identified easily. In Figure 4.2 such a resulting mesh is shown schematically. Parts of two image sequences are shown on the left side with four cameras each. The neighborhood connections that were established automatically are denoted by dashed lines, provided that these cameras are close enough to each other.

Using these neighborhoods a second tracking step similar to mesh tracking introduced in Section 3.2.3 is invoked. This time, no new features are added but only those are tracked further that were used for the first calibration. Since during the calibration process features with a too high back-projection error are removed, only the more robust ones are left at this stage. Tracking in this case is performed in a two-step loop for each image sequence:

1. The existing features in the current sequence are tracked to the other image sequences following the neighborhood links established before. The tracking algorithm is implemented

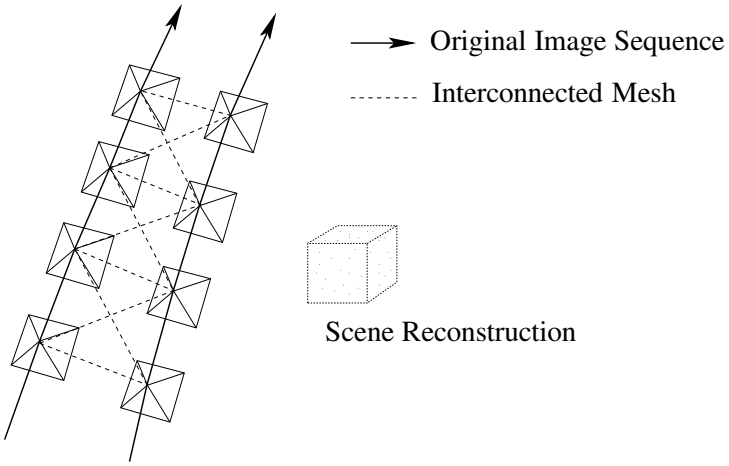



Figure 4.2: Creating a mesh between the original image sequences. Two different sequences are sketched along the solid lines, while the detected neighborhood relations are drawn as dashed lines.

in such a way that features can be tracked from one image to any other image available. No reordering of the image sequences is required.

2. These additional features are now propagated through the other image sequences. Depending on the effort to be spent this can be just the preceding and the following image sequence of the current one, or all other image sequences. The complexity in the latter case of course increases quadratically with the number of image sequences.

Through this process, the formerly mostly unrelated sequences—except for the one frame which is common in each pair of consecutive sequences—are now linked together through these new feature correspondences.

In a second calibration step the camera parameters for the whole set of images of all sequences can now be calculated together. Like for each individual image sequence before, an initial subset of images is sought for now which have the highest number of feature correspondences in common. Unlike before, this subset does not need to be a subsequence of consecutive frames anymore, but constitutes a subgraph of the neighborhood mesh constructed for the feature tracking step (cf. Figure 4.2). Calibration is then performed as before, using a factorization for the initial subset and adding the remaining frames one by one. Which frame to add next is again determined by the neighborhood mesh.

Afterwards, all camera parameters are available in the same coordinate system, and no second refinement step is necessary. The error which occurs if the corresponding cameras of each pair of sequences were not exactly the same after all—which is usually the case since the hand holding the camera trembles while the object is moved—is thus removed.

The disadvantage of the new feature correspondences is that any of them could be positioned on a moving, i. e., dynamic, part of the scene. These *dynamic features* can be considered equivalent to erroneously tracked points, and can severely perturb the calibration result. Nevertheless, by postulating that only a minor part of the scene is actually in motion the calibration algorithm proved to be robust enough to handle these outliers. They are removed during the non-linear optimization steps described in Section 3.4.2, where features with a high back-projection error are discarded using a threshold value or robust outlier detection.

On the other hand, features on a dynamic part of the scene are unproblematic as long as they are not tracked to another image sequence at a different time step. Within one sequence, all features are static.

At this point, the dynamic light field is completely reconstructed except for depth information. This can be generated, e. g., by using one of the local depth map algorithms described in Section 3.6.1. Since rendering of the next type of dynamic light fields is done in the same way as step-wise static ones, depth map generation will be described in Section 4.4 for both types of light fields.

## 4.3 Moving but Rigid Objects

The video recording technique used for the dynamic light fields of the previous section obviously poses a strong limitation regarding possible applications. In order to simplify the recording process, the second type of dynamic light fields allows continuous movement of one or possibly several objects in the scene. The restriction here is that the objects are rigid in themselves. This approach was first published in [Sch05].

### 4.3.1 Segmentation of Feature Points

The starting point of dynamic light field reconstruction is an image sequence of a scene containing at least one permanently moving but rigid object in front of a static background. Two images of an example for such a sequence are shown in Figure 4.3, a toy crawler which moves in a circle on a desk, while the camera is likewise in permanent motion. The point segmentation algorithm applied in the following relies on object motion over a short period of time, therefore, a lack of motion is allowable for at most a few frames at a time, and objects and background have to be visible at all times.

In the following, the background will be treated as another rigid object, since no distinction can be made a priori between foreground and background. Thus, $K$ objects are assumed, where

Figure 4.3: Two images of the *Crawler* example sequence

$K \geq 2$.

The main difference between the reconstruction process here and the reconstruction of a static light field as outlined in Figure 3.1 of Section 3.1 is an additional segmentation step after feature tracking, which assigns each feature point to one object. This segmentation allows a separate reconstruction for each object, which are merged afterwards as described in Section 4.3.2.

Like in the factorization method for reconstruction, feature segmentation requires a *measurement matrix* $\boldsymbol{W}$ containing the features visible in the frames of an initial subsequence. As it is known from Section 2.2, the *registered* measurement matrix $\widetilde{\boldsymbol{W}}$ is constrained to rank 3 in case of a static 3-D scene and assuming an affine projection model. The preprocessing step of registering the measurement matrix reduces its rank by 1, so that the *unregistered* measurement matrix $\boldsymbol{W}$ has rank 4. Consequently, each additional, independently moving 3-D object increases the rank of $\boldsymbol{W}$ by 4, if enough features on the object have been detected. In order to assign each point to one object or the other, Costeira and Kanade make use of the so-called *shape interaction matrix* $\boldsymbol{Q}$ [Cos98]. From the singular value decomposition $\boldsymbol{W} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}}$, $\boldsymbol{Q}$ is obtained as

$$\boldsymbol{Q} = \widehat{\boldsymbol{V}}\widehat{\boldsymbol{V}}^{\mathrm{T}} \quad , \tag{4.5}$$

where $\widehat{\boldsymbol{V}}$ corresponds to the first $4K$ columns of $\boldsymbol{V}$. Its most useful property is that each entry in $\boldsymbol{Q}$ describes whether the corresponding columns in $\boldsymbol{W}$, two features, belong to the same subspace, i. e., object, or not. The higher the value the greater the correlation. A proof of this property of $\boldsymbol{Q}$ as well as an overview over its additional properties are given in [Cos98].

Segmenting the features which were used to compose $\boldsymbol{W}$ now comes down to sorting the

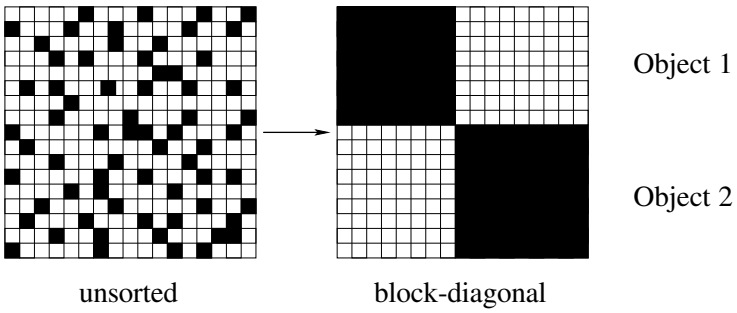



Figure 4.4: Shape interaction matrix: the entries are sorted to form a block-diagonal matrix, where each block denotes the features of one object.

rows and columns of $\boldsymbol{Q}$ such that a block-diagonal structure emerges, as depicted in Figure 4.4. Sorting $\boldsymbol{Q}$ is equivalent to permuting the columns in $\boldsymbol{W}$, transforming it to its canonical form $\boldsymbol{W}^*$, where

$$\boldsymbol{W}^* = (\boldsymbol{\Psi}_1|\boldsymbol{\Psi}_2) \begin{pmatrix} \boldsymbol{\Phi}_1 & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Phi}_2 \end{pmatrix} \quad . \tag{4.6}$$

Here, $\boldsymbol{\Psi}_1$ and $\boldsymbol{\Psi}_2$ denote the motion of the camera relative to objects 1 and 2, while $\boldsymbol{\Phi}_1$ and $\boldsymbol{\Phi}_2$ contain the 3-D feature positions on either object, respectively.

In order to sort $\boldsymbol{Q}$, Costeira and Kanade apply a greedy algorithm which minimizes a cost function $C_j^m$. In each sorting step $m$, one column and row permutation is performed, increasing the size of the already sorted matrix $\boldsymbol{Q}^{m^*}$ by 1. The permutation is selected such that the energy of interaction between each candidate column $j$ and the features already in $\boldsymbol{Q}^{m^*}$ is maximized. Thus, $C_j^m$ is given as

$$C_j^m = \sum_{i=1}^{m} q_{ij}^2 \quad , \tag{4.7}$$

where $j = m+1, \ldots, N$. The columns and rows $j$ and $m+1$ of $\boldsymbol{Q}$ are permuted in order to increase the sorted part $\boldsymbol{Q}^{(m+1)^*}$ of $\boldsymbol{Q}$. In the ideal case, the algorithm terminates yielding a sorted shape interaction matrix $\boldsymbol{Q}^*$. The remaining task is to identify the blocks of $\boldsymbol{Q}^*$, using the fact that the square of the Frobenius norm of each block $\boldsymbol{Q}_k^*$ is equal to its rank.

Unfortunately, this greedy approach to sorting the entries in $\boldsymbol{Q}$ is quite susceptible to noise, and thus fails frequently to provide the correct segmentation. Therefore, Kanatani introduced three subsequent improvements of the original segmentation algorithm. These methods are termed *subspace separation and model selection* [Kan01a], *affine space separation* [Kan02] and *multi-stage optimization* [Kan03], and will be described shortly in the following. For detailed derivations, proofs and analyses refer to the original publications.

**Subspace separation and model selection.** In order to group features together in four-dimensional submatrices, the same energy as in equation (4.7) is maximized. However, grouping is started with $N$ initial groups $\mathcal{L}_n$ of one feature each which are then merged successively, instead of grouping all features of one object together before continuing to the next object.

As soon as more than four features are grouped together, a four-dimensional subspace is fit to them using, e. g., SVD, and the feature positions are recalculated. Through this *dimension correction*, the noise in the features is reduced, as long as the features are grouped correctly. Merging of groups and dimension correction are repeated until only $K$ groups remain.

Furthermore, in deciding which groups of features to merge, Kanatani does not rely exclusively on the energy $C$ which is based on the shape interaction matrix. Additionally, the residual $J$, which denotes the squared difference between original feature positions and those calculated from the subspace fit, is regarded. It increases considerably if two mismatched groups are merged, and less if they belong to the same subspace. However, the residual $J_{i\oplus j}$ of two merged groups $\mathcal{L}_i$ and $\mathcal{L}_j$ will always be larger than the sum of the individual residuals, $J_i + J_j$.

The decision whether to merge two groups is now based on a criterion which is called the *geometric AIC*, an extension of the original *Akaike Information Criterion*. The G-AIC quantizes how well a model, i. e., a subspace, fits a set of data by also taking into account the degrees of freedom of a subspace, as two separate subspaces possess more degrees of freedom than one combined subspace. The G-AIC for the merged groups of features is computed as

$$\text{G-AIC}_{i\oplus j} = J_{i\oplus j} + 2d(N_i + N_j + 2F - d)\eta^2 \quad , \tag{4.8}$$

where $d$ is the rank of the subspace (4 in this case), $N_i$ and $N_j$ are the number of features in $i$ and $j$, respectively, and $F$ is the number of images. Thus, $2F$ is the number of rows of the vector for one feature. Finally, $\eta$ denotes the noise level, which is estimated using the residual $J_{i\oplus j}$ (cf. [Kan01a]). The combined G-AIC for two individual feature groups is given by

$$\text{G-AIC}_{i,j} = J_i + J_j + 2d\left(N_i + N_j + 2(2F - d)\right)\eta^2 \quad . \tag{4.9}$$

A formal derivation for the G-AIC, and the AIC in general, is given in [Kan96a].

In principle, two subspaces should be merged if $\text{G-AIC}_{i\oplus j} < \text{G-AIC}_{i,j}$. Since this would disregard the information in $\boldsymbol{Q}$, Kanatani proposes the following measure which incorporates both criteria:

$$c_{i,j} = \frac{\text{G-AIC}_{i,j}}{\text{G-AIC}_{i\oplus j}} \max_{\alpha\in\mathcal{L}_i,\beta\in\mathcal{L}_j} |q_{\alpha\beta}| \quad . \tag{4.10}$$

Replacing the energy criterion of equation (4.7), the two subspaces with the largest similarity measure $c_{i,j}$ are grouped together until only $K$ groups remain.

A final step in this segmentation algorithm is a robust detection of outliers using LMedS, as it was already described in Section 3.3.1. Here, points far away from the optical center are favored as random samples, since they are segmented more reliably. The detected outliers in one group are then reassigned.

**Affine space separation.** As it was pointed out in Section 2.2, the registered measurement matrix $\widetilde{\boldsymbol{W}}$ has only rank 3, since it is assumed that it is generated by affine projection. The same assumption can be made for feature segmentation, and thus, the above algorithm changes accordingly. In [Kan02], this different assumption is implemented by fitting three-dimensional affine subspaces to the grouped features instead of four-dimensional ones. Likewise, the geometric AIC of equations (4.8) and (4.9) are adjusted. The stability of the resulting segmentation increases as long as perspective effects are small. This is in accordance with the observations made for different factorization methods.

**Multi-stage optimization.** The last addition of Kanatani to his segmentation algorithm entails two contributions: an unsupervised learning step which fits a Gaussian distribution to the affine subspaces, and the consideration of degenerate motion. These two extensions are combined to form the multi-stage optimization.

Unsupervised learning means that for each class, i. e., subspace of $\boldsymbol{W}$, or moving object, respectively, a Gaussian distribution consisting of mean vector and covariance matrix is estimated. Estimation is performed using the expectation maximization (EM) algorithm [Nie03], an iterative procedure which approximates the parameters of the distribution by maximizing the likelihood of each class.

Degenerate motions and, likewise, object geometries, are one issue that has not been mentioned here so far. However, as Costeira and Kanade already pointed out and considered in their approach, the subspace spanned by one object does not need to have the maximum rank. One reason, which they treated, is that the features on the object themselves only form a subspace in 3-D, i. e., they are situated on a plane or a line only. The second reason, which is addressed by Kanatani at this point and the possibly more important one, is degenerate motion. This is the case if an object only moves parallel to the image plane and rotates only about the optical axis of the camera. In case of the assumption of affine subspaces, the rank of the subspace would be reduced to 2. If, in addition, the objects do not rotate at all, they form parallel, two-dimensional

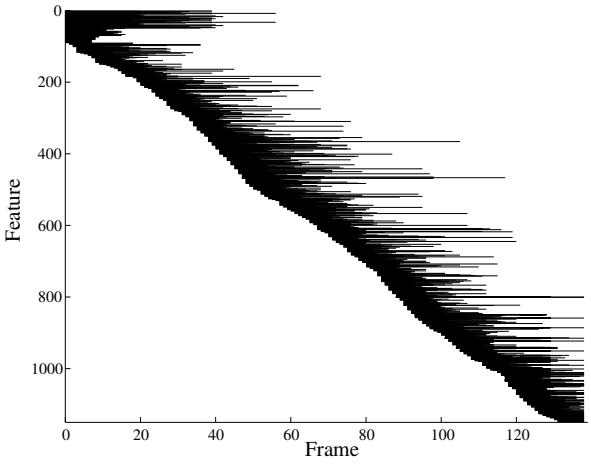



Figure 4.5: Features tracked on the *Crawler* object

subspaces. If such degenerate motions exist, the affine space separation algorithm from above has to be altered accordingly.

Unfortunately, it is generally not known beforehand whether an image sequence contains degenerate motion, or not. Therefore, Kanatani proposes a three-step approach to take into account all kinds of motion. First, an initial segmentation is determined using affine space separation assuming degenerate motion. This segmentation is used to initialize an unsupervised learning step assuming, again, degenerate motion. The final step consists of unsupervised learning assuming general motion. This multi-stage method proved to yield the best segmentation results[1].

Once the feature points on each object have been identified, the 3-D structure and camera poses relative to each object are determined by applying, e. g., a paraperspective factorization method, followed by non-linear optimization (cf. Section 3.3.3).

The fact that feature trails are not long enough to reconstruct a scene with only one application of a factorization method is even more true for dynamic scenes. As features are divided up between several objects, which may be small or moving quickly, the average length of a feature trail may be quite low, possibly as few as 10 to 20 frames. As an example, the features found on the moving object in the *Crawler* sequence are plotted in Figure 4.5. Thus, the iterative extension process of Section 3.4.1 has to be applied here as well, but interweaved with repeated segmentations of the new features to be added to the reconstruction. In order to increase the underlying amount of data, and thus robustness, the already assigned features are used to initialize

[1]The source code was kindly provided by the authors at http://www.suri.it.okayama-u.ac.jp/e-program.html

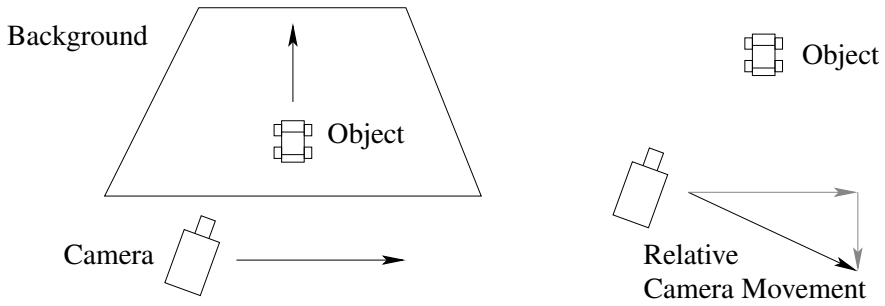



Figure 4.6: Camera motion relative to a moving object. If both camera and object are in motion (left), the camera motion relative to the object consists of both motion components (right).

Kanatani's algorithm with the known segmentation, thereby also decreasing computational cost.

Assigning new features to an object has to be done before their 3-D position is triangulated. In order to do so, the set of new features is subdivided according to the number of calibrated frames they are visible in. For each subset, a measurement matrix is constructed, adding all features which are already segmented and visible in these images. The segmentation algorithm is initialized with $K$ predefined groups of sorted features, and one group for each new feature.

Concerning the identification of wrongly assigned features, the usual methods for discarding erroneous points—a threshold on the back-projection error or outlier detection—take effect here as well.

### 4.3.2 Registration of Object Reconstructions

After calibration, the camera motion relative to each object is available and can be used to infer the motion of the object. As depicted in Figure 4.6, the reconstructed camera motion not only depends on the motion of the object, but also includes the motion of the camera itself. In order to get the motion of the object itself relative to the background or other objects, the camera's own motion has to be eliminated. The remaining motion component will be required in the next subsection for dividing the motion into time steps.

Since, after reconstruction, no common world coordinate system is available, the reconstruction for each object will differ from the others by an arbitrary rotation, translation and scaling. This issue was encountered likewise in the previous section for step-wise static light fields. The solution proposed here is therefore similar to the approach applied before.

Without loss of generality, the object containing the highest number of features is selected as the background of the scene and denoted as object $0$. Likewise without loss of generality, it is assumed that the camera poses for the background and any object $k \geq 1$ for the first image, $\boldsymbol{P}_{0,1}$

and $\boldsymbol{P}_{k,1}$, are the same. Then, any object camera pose can be transformed to the background coordinate system analogously to equation (4.1):

$$\boldsymbol{P}'_{k,j} = \boldsymbol{P}_{k,j} \begin{pmatrix} \boldsymbol{R}_{k,1}^{\mathrm{T}} & -\boldsymbol{R}_{k,1}^{\mathrm{T}} \boldsymbol{t}_{k,1} \\ \boldsymbol{0}_3^{\mathrm{T}} & 1 \end{pmatrix}^{-1} \begin{pmatrix} \boldsymbol{R}_{0,1}^{\mathrm{T}} & -\boldsymbol{R}_{0,1}^{\mathrm{T}} \boldsymbol{t}_{0,1} \\ \boldsymbol{0}_3^{\mathrm{T}} & 1 \end{pmatrix} \quad . \tag{4.11}$$

The inverse transformation is applied to each (homogeneous) 3-D object point.

The remaining scale factor is determined in the same way as for step-wise light fields, by assuming that the objects should be at the same distance from the camera positions. Therefore, the scaling is calculated as the ratio between the distances of the centers of mass of the 3-D point clouds of object and background, and again applied to each camera pose as in equation (4.4):

$$\boldsymbol{P}''_{k,j} = \boldsymbol{P}'_{k,j} \begin{pmatrix} \boldsymbol{R}'^{\mathrm{T}}_{k,1} & -\boldsymbol{R}'^{\mathrm{T}}_{k,1} \boldsymbol{t}'_{k,1} \\ \boldsymbol{0}_3^{\mathrm{T}} & 1 \end{pmatrix}^{-1} \boldsymbol{S} \begin{pmatrix} \boldsymbol{R}'^{\mathrm{T}}_{k,1} & -\boldsymbol{R}'^{\mathrm{T}}_{k,1} \boldsymbol{t}'_{k,1} \\ \boldsymbol{0}_3^{\mathrm{T}} & 1 \end{pmatrix} \quad . \tag{4.12}$$

Both background and object reconstruction are now in the same coordinate system, although the transformation is not exact since the scaling is calculated only by a heuristic measure. An exact solution to this problem seems to be an open issue in 3-D reconstruction, since it is neither addressed by Costeira and Kanade in [Cos98], although two independent reconstructions are calculated there as well, nor in any other contribution dealing with segmentation of moving objects in 3-D reconstruction, such as [Avi99], [Han00] or [Sha01]. In cases where the inexact approximation of the scale factor is inadequate for further processing, it has to be adjusted by hand. An accurate calculation of the scale factor remains subject to future work.

Finally, the camera movement relative to the object is calculated as the transformation between two corresponding camera poses, the one relative to the background, given by $\boldsymbol{R}_{0,j}$ and $\boldsymbol{t}_{0,j}$, and the transformed object motion $\boldsymbol{R}''_{k,j}$, $\boldsymbol{t}''_{k,j}$ extracted from $\boldsymbol{P}''_{k,j}$. This pose difference is transformed back to the world coordinate system using the reference pose $\boldsymbol{R}_{0,1}$, $\boldsymbol{t}_{0,1}$:

$$\boldsymbol{P}^*_{k,j} = (\boldsymbol{K}|\boldsymbol{0}_3)\boldsymbol{M}_{0,1}\boldsymbol{M}_{0,j}^{-1}\boldsymbol{M}''_{k,j} \tag{4.13}$$

where

$$\boldsymbol{M}_{k,j} = \begin{pmatrix} \boldsymbol{R}_{k,j}^{\mathrm{T}} & -\boldsymbol{R}_{k,j}^{\mathrm{T}} \boldsymbol{t}_{k,j} \\ \boldsymbol{0}_3^{\mathrm{T}} & 1 \end{pmatrix} \quad . \tag{4.14}$$

As can be seen from the above equation, the calibration matrix $\boldsymbol{K}$ is assumed to be the same for all images, the intrinsic parameters being constant for the whole sequence. If they are estimated alongside the pose parameters, they are still the same for all objects $k$ in one image $j$.

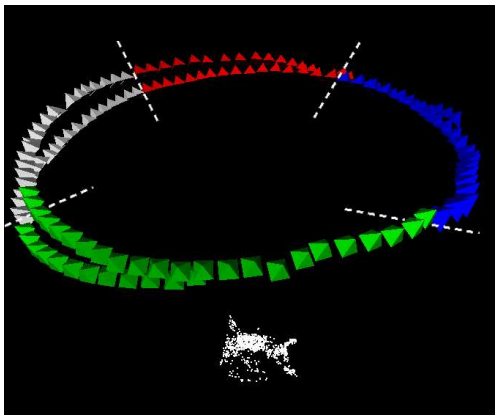


Figure 4.7: Reconstruction of the moving object in the *Crawler* sequence and partitioning of its movement into four time steps, using the object-relative camera positions. The camera positions are marked in a different shade for each time step and emphasized additionally by dashed lines.

Like for equation (4.11), the inverse transformation of equations (4.12) and (4.13) is applied to the 3-D object points, although separately for each camera pose. In this way, 3-D point trajectories are generated which are relevant later on for calculating depth maps for visualization.

### 4.3.3 Time Step Identification

In case of the step-wise static light fields introduced previously, different time steps in scene motion are available due to the adapted recording method. Visualizing these light fields is a simple matter, as will be shown in Section 4.4. Therefore, it is desirable to establish the same preconditions for the light fields at hand. Here, time steps are equivalent to similar poses of an object in the course of its motion. Thus, the goal is to identify and combine images with similar object positions and orientations to form individual time steps.

Using the projection matrices $\boldsymbol{P}^*_{k,j}$, $k \geq 1$, calculated previously, similar object positions are calculated by applying a vector quantizer [Ger82] to the camera positions, i. e., the translation vectors $\boldsymbol{t}^*_{k,j}$. In order to generate the codebook, the standard Linde-Buzo-Gray (LBG) algorithm [Lin80] is employed. This iterative algorithm determines an optimal partition of a set of vectors by grouping them around one codebook vector for each class, thus minimizing the intra-class distance. The codebook vectors are given by the centroids of each class. Both partitioning and codebook estimation are updated in each iteration. Thus, for each desired time step, one codebook vector is specified. An example for a resulting quantization is shown in Figure 4.7 for the *Crawler* sequence. Here, the camera positions are subdivided into four time steps.

Unfortunately, this partitioning scheme for discrete time steps introduces a spatiotemporal

dilemma for later rendering. If a small number of time steps is selected, as in Figure 4.7, the object positions belonging to one time step will be widely spread. This leads to serious blurring during rendering of the object as images are interpolated which do not fit together well. Likewise, the object will seem to move slightly even inside of one time step as different images are selected as reference while the observing, virtual camera is moved.

On the other hand, if a high number of time steps is chosen, only few images are assigned to each one. Since the light field relies on a dense sampling of the plenoptic function, this leads to a bad reproduction of viewpoint-dependent effects, such as specular lighting, shadowing or occlusion, and errors in geometry information become more pronounced. Therefore, the resulting dynamic light field gets better with either more input images, or smaller and repetitive object motion.

## 4.4 Extended Rendering Techniques

Both step-wise static and rigid object light fields are divided into time steps, arising directly from reconstruction or resulting from a final processing step. Therefore, both types of light fields can be visualized using the rendering techniques for step-wise light fields, introduced in [Sch02, Sch04b]. This technique will be described in the following subsection. However, the latter type of light fields offers more information about motion in the scene, which is exploited by the rendering technique introduced in Section 4.4.2. The final Section 4.4.3 of this chapter provides an outlook to the future of dynamic light field rendering for rigid object motion by introducing the concept of *object light fields*.

### 4.4.1 Multiple Light Fields

For step-wise static light fields, each time step is well separated from all others by the fact that for each of them, one input image sequence is available. Thus, for each time step, one light field is readily available, consisting of input images, camera parameters, and depth information in the form of depth maps or local proxies. As scene reconstructions for all time steps are registered with each other, so are the resulting light fields.

Visualizing these multiple light fields is a rather simple matter. The light field renderer, be it the free-form light field [Sch01b] used originally in [Sch04b], or the unstructured lumigraph [Bue01] used later, is extended to handle an arbitrary number of light fields. By switching to another light field, different time steps are visualized.

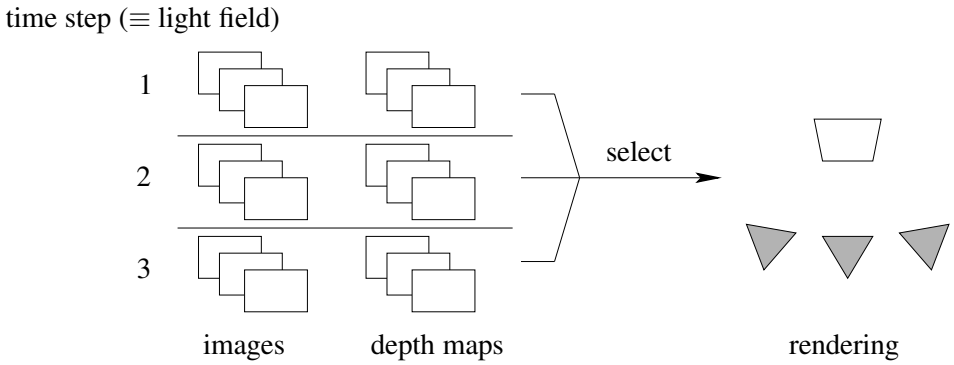



Figure 4.8: Rendering time steps for step-wise static light fields. From one light field per time step, the time step currently viewed is selected and images are rendered for this time step from the corresponding light field.

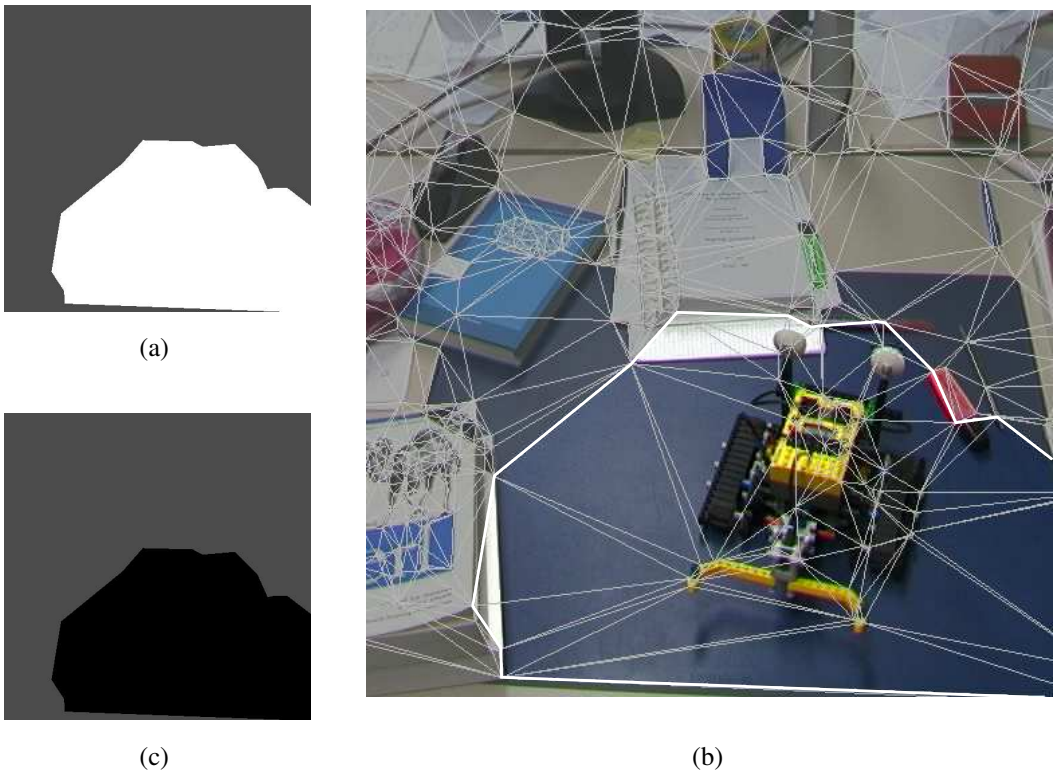
This approach is outlined in Figure 4.8. As can be seen there, the sets of input images are disjoint for each time step, and an additional set of depth maps is required for each time step. Consistent information across time steps, such as for the static background, is not handled. Consequently, large amounts of data have to be stored, and switching between light fields may require extensive memory access operations.

The same visualization technique can be applied for dynamic light fields of rigid, moving objects. However, in order to create one light field for each time step, the set of input images available has to be split up between the light fields. The resulting light fields possess a very small data base, causing low image quality. The next rendering technique therefore exploits the availability of an object segmentation, improving quality at least for parts of the scene.

### 4.4.2 Using Confidence Maps

The observation of the previous section was that, for both kinds of dynamic light fields regarded here, the background of the scene is static and parts of it are visible in all input images. Nevertheless, for each time step only the input images allocated to it were utilized for rendering the background, which means that a lot of information is discarded. With the segmentation of features on the moving foreground objects, a selective usage of image content for rendering is now possible.

The key to this selective usage are so-called *confidence maps*. They were originally intended to provide a confidence value between $0$ and $1$ for the depth values in the corresponding depth maps. However, they were so far only used to mask out areas of an image where no depth was available at all, or where image information was unusable. This is the case, e. g., in medical



(a) (c) (b)

Figure 4.9: Example confidence maps of the *Crawler* sequence where the object is visible (a) and invisible (c). The confidence map is generated from a Delaunay triangulation of the segmented features in an image (b).

applications where highlights are caused on the scene due to wet surfaces of organs. The benefit of removing these highlights using binary confidence maps was shown in [Vog02b, Vog02c].

For visualizing dynamic light fields of moving objects, confidence maps $\boldsymbol{h}_{f,\tau}$ are calculated for each image $\boldsymbol{f}_f$ in the sequence and for each time step $\tau = 0, 1, \ldots$. In this case, the confidence maps may contain three different values, $\boldsymbol{h}_{f,\tau}(\boldsymbol{q}) = 0$ for pixels on the foreground object if it is invisible, $\boldsymbol{h}_{f,\tau}(\boldsymbol{q}) = 1$ if it is visible, and $\boldsymbol{h}_{f,\tau}(\boldsymbol{q}) = 0.5$ for all background pixels. The values are chosen arbitrarily. Figures 4.9(a) and 4.9(c) show two example confidence maps for the *Crawler* sequence, the first where the object is visible, and the second where it is not.

In order to partition each image into a foreground and a background area, the segmented feature points are connected by a mesh using Delaunay triangulation as in Section 3.6.2. A triangle is assigned to the foreground if at least one of its vertices belongs to the moving object. Otherwise it is allocated to the background. Such a partitioning is visualized in Figure 4.9(b), showing the relationship between original image, point feature triangulation, and confidence

map. It also shows that this segmentation is very inexact and masks the object quite generously. Obviously, an improved segmentation would be desirable, but, as it will be demonstrated in Section 5.3, erroneously masking the background is preferable to the opposite, which would result in artifacts during rendering.

Transforming the triangle mesh to a confidence map can be done very efficiently using graphics hardware, which is a logical choice for handling meshes. Each triangle is assigned a depth value according to its classification, and after rendering using OpenGL the confidence map is read from the depth buffer of the graphics card.

In order to make the effect of confidence maps more plausible, the general approach to the rendering of images from a light field shall be summarized briefly. Initially, the virtual image to be created is subdivided by a regular triangle mesh with $m \times n$ vertices. For each of these sampling points $\check{\boldsymbol{q}}_{mn}$, a fixed number $f_r$ of contributing input images is selected according to different criteria. The first two of those are the angular difference of viewing rays passing through the proxy surface point corresponding to the sampling point, and the distance of the camera position to that of the virtual camera.

At this point, a third selection criterion incorporates the confidence values. For any kind of light field, static or dynamic, the input image is discarded if $\boldsymbol{h}_{f,\tau}(\check{\boldsymbol{q}}_{mn}) = 0$. In the dynamic case, this occurs if the image shows the moving object at a point in time where it is invisible. Furthermore, two sets of closest images are maintained, the first one,

$$\mathcal{F}_{\text{bg}}(\check{\boldsymbol{q}}_{mn}) = \{\boldsymbol{f}_f \,|\, 0 < \boldsymbol{h}_{f,\tau}(\check{\boldsymbol{q}}_{mn}) < 1\} \quad , \tag{4.15}$$

where the background is visible, and the second,

$$\mathcal{F}_{\text{fg}}(\check{\boldsymbol{q}}_{mn}) = \{\boldsymbol{f}_f \,|\, \boldsymbol{h}_{f,\tau}(\check{\boldsymbol{q}}_{mn}) = 1\} \quad , \tag{4.16}$$

showing the object in a desired position for time step $\tau$. Images are selected until $|\mathcal{F}_{\text{bg}}| + |\mathcal{F}_{\text{fg}}| = f_r$. If no images are found showing the object, i. e., if $|\mathcal{F}_{\text{fg}}| = 0$, only the background will be rendered from $\mathcal{F}_{\text{bg}}$. However, if $|\mathcal{F}_{\text{fg}}| > 0$, $\mathcal{F}_{\text{bg}}$ is discarded entirely and only $\mathcal{F}_{\text{fg}}$ is used for rendering. This prevents the blending of background and foreground, which would result in a translucent appearance of the object in motion.

The final rendering step is initialized by reevaluating the weights for the input images at each vertex of the regular triangle mesh superimposed on the virtual image using some additional criteria. The resulting *blend field* determines how corresponding texture triangles from the input images are blended to form the new, virtual image.

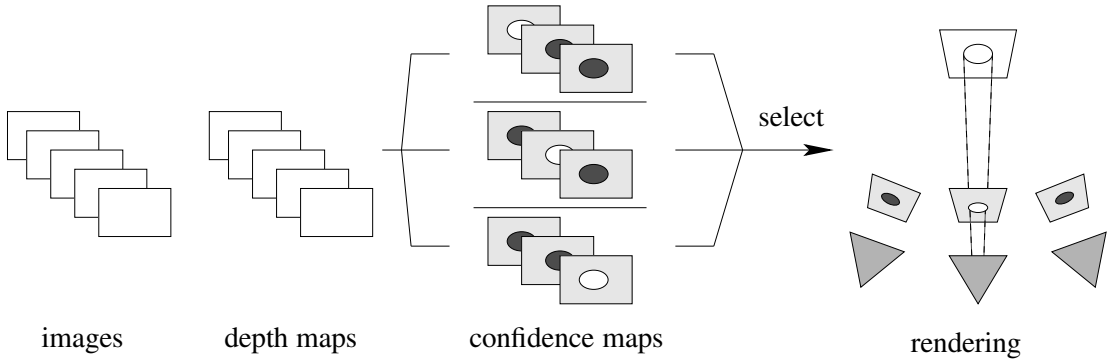



Figure 4.10: Visualizing dynamic light fields using confidence maps. A light field thus consists of one set each of images and depth maps, and one set of confidence maps for each time step.

The benefit of this improved rendering method for step-wise light fields is clarified by Figure 4.10. Now, all input images and depth maps are used for each time step. A restriction is only applied for parts of the input images where moving objects are present. This leads to a considerably improved rendering quality for background areas, while the foreground image quality remains the same as before. However, the improvement is paid for with a higher amount of data to be handled as the total number of images and depth maps stays the same, while confidence maps are required additionally.

While the use of confidence maps improves rendering quality, the principle drawback of dividing object motion into discrete steps, mentioned in Section 4.3.3, remains. The more steps are used, increasing time resolution, the lower the number of images available for each. The concept of *object light fields* introduced in the following constitutes an important step towards remedying this tradeoff.

### 4.4.3 Object Light Fields

The idea behind object light fields is to make use of additional information which was neglected so far. Because of feature point segmentation, a 3-D reconstruction is available for each of the $K$ objects, i. e., background and moving objects. Now, using the original camera parameters $\boldsymbol{P}_{0,j}$ of the background, and the registered parameters $\boldsymbol{P}''_{k_o,j}$, $k_o = 1, \ldots, K-1$, of equation (4.12), a separate light field is created for each object. The confidence maps calculated in the previous section are used to completely mask all image information not belonging to the current object light field. The values for $\boldsymbol{h}_{f,k}(\boldsymbol{q})$, $k = 0, \ldots, K-1$, have to be selected accordingly.

In order to correctly visualize the dynamic light field, the $K$ individual light fields have to be rendered together, with an adjustable transformation between background and moving object,

$M_{0,k_o}$.

Two main issues exist which currently prevent automatic reconstruction of object light fields with sufficient quality:

1. Object segmentation for confidence map calculation making use of feature segmentation is not accurate enough for correct visualization of individual objects. An exact segmentation is required for rendering without residual artifacts showing parts of the background around objects.

2. Joint rendering of multiple light fields requires the consideration of multiple issues which have not occurred before. Collision detection or correct occlusion handling depending on independent sets of depth maps or proxies, casting of shadows as well as efficiency aspects have to be treated additionally.

These problems are not addressed as part of this thesis and left open for further research. However, Section 5.3.4 offers a first example as outlook to the potential use of object light fields. Another side-effect which can already be utilized is that in the background light field thus created, the moving objects are effectively removed. Thus, if an object passing through the image is considered a disturbance, the undisturbed static scene is made visible again.

# Chapter 5

# Experiments

In the following chapter, the techniques introduced in Chapters 3 and 4 for reconstructing static and dynamic light fields will be evaluated experimentally. The methods of evaluation are described first in Section 5.1, which consist of the already known measures of back-projection error and PSNR, as well as ground truth data obtained by an optical tracking system and known scene geometry. The experiments themselves are divided, again, into two parts for static and dynamic light fields, respectively, given in Sections 5.2 and 5.3.

## 5.1 Light Field Quality Assessment

Objective evaluation of light field quality is a difficult task. Two criteria, back-projection error and PSNR, were introduced already in Section 3.3.2. In the following, an assessment of these quality criteria themselves will be given. It will be shown that both of these methods exhibit different advantages and drawbacks, so that the information deduced from them has to be treated with care. Therefore, a third evaluation method is introduced based on ground truth data. This is obtained either by an optical tracking system or through the use of objects with known geometry. While it does not give a direct measure of light field rendering quality, it offers a good estimate to the accuracy of either camera pose or object geometry reconstruction.

### 5.1.1 Back-Projection Error and PSNR

Apart from a comparison with ground truth data, if available, back-projection error and PSNR, as defined in equations (3.20) and (3.23), are the only possibilities found so far to obtain an indication for the success of a reconstruction. Although often used, they should nevertheless be

| | Back-projection error | PSNR |
|---|---|---|
| Independent of image content | no | no |
| Measures $\boldsymbol{P}_f$ and $\boldsymbol{p}_{fn}$ independently | no | no |
| Independent of scene geometry type | yes | no |
| Measures quality of metric upgrade | no | yes |
| Smooth error measure | yes | no |

Table 5.1: Comparison of back-projection error and PSNR as quality measures for light fields

regarded with caution for reasons outlined in the following. A summary of the properties of the two measures is given in Table 5.1.

First of all, it has to be stated that neither measure constitutes an absolute criterion. Both depend on the content of the source image sequence and the circumstances of recording. Thus, if, e. g., an affine projection model is assumed, the back-projection error of a feature depends on the distance of the corresponding point from the camera and its position in an image, as features closer to the optical axis are less distorted (cf. Figure 2.4, page 27). On the other hand, the PSNR of a light field will be higher if the images contain large, homogeneous areas which disappear in difference images.

With both measures, the results of camera parameter and 3-D point position estimation are measured together, reflecting the success of a reconstruction as a whole. Thus, it is usually not possible to give a quality value for either camera parameters $\boldsymbol{P}_f$ or 3-D points $\boldsymbol{p}_{fn}$ independently. As back-projection error is computed before scene geometry reconstruction it remains the same for any of the geometry types introduced in Section 3.6 which may be used later on. This is not the case for PSNR, which renders it suitable for comparing the different types of geometry.

The main reason for introducing PSNR as a second selection criterion for factorization methods in Section 3.3 is its ability to measure the success of an upgrade from projective to metric reconstruction. This is demonstrated with the left plot of Figure 5.1, where a $100$ images subsequence is processed applying perspective factorization with self-calibration using an affine solution as described in Section 3.3.3 on page 66. The entries of upgrade matrix $\boldsymbol{D}$ given in equation (3.24) are modified with Gaussian noise of different degrees to distort the resulting reconstruction. Consequently, the PSNR value deteriorates rapidly with increasing noise, while the back-projection error is completely unaffected. The results for each noise level were averaged over five trials. The example on the right hand side of Figure 5.1 shows reconstruction errors of a paraperspective factorization of the same image subsequence for different assumptions for focal length $f$. In this case, back-projection error and PSNR are obviously correlated.

Finally, Figure 5.2 shows the experiments of Figure 5.1 for a very short example sequence of

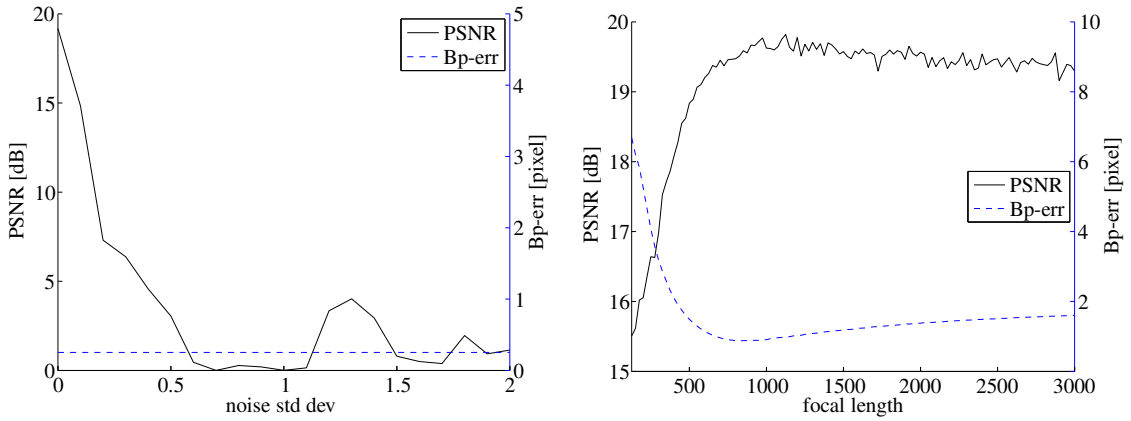



Figure 5.1: Back-projection error and PSNR for the reconstruction of a 100 images initial subsequence. (a) Variation of noise on the metric upgrade matrix; (b) different focal length presettings for paraperspective reconstruction.

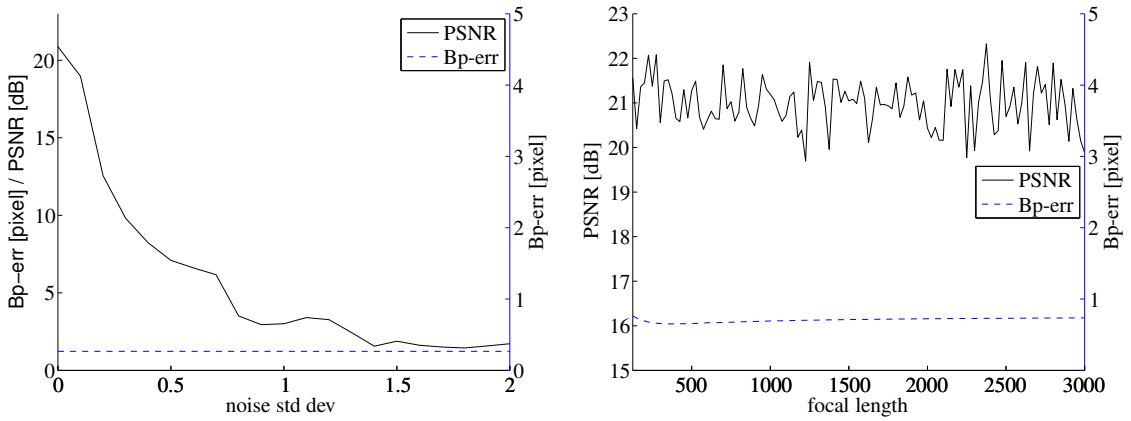



Figure 5.2: Back-projection error and PSNR for the reconstruction of an 8 images example sequence. (a) Variation of noise on the metric upgrade matrix; (b) different focal length presettings for paraperspective reconstruction.

only 8 images. The right hand plot documents one additional drawback of PSNR, its sensitivity to small changes in parameters, especially for few images where little averaging occurs. In contrast to that, the back-projection error yields a smooth error curve. For factorization selection of Section 3.3, this effect is the reason why the final decision is made based on back-projection error.


(a) (b)

Figure 5.3: (a) Sony DV camera equipped with a target consisting of five reflective, spherical markers, (b) ARTtrack2 cameras of the smARTtrack1 optical tracking system

### 5.1.2 Relative Pose and Geometry Error from Ground Truth Data

The evaluation methods introduced here were devised by Vogt for experimental evaluation of light fields from endoscopic images in [Vog06]. As already mentioned in Section 1.1, a very reliable way to acquire camera pose information for light fields is the use of an optical tracking system. Such a tracking system consists of two or more infrared cameras with integrated infrared light sources, and a target equipped with reflective markers. For the experiments in this chapter, such a target was fixed to a Sony DSR-PD100AP DV camera as shown in Figure 5.3(a). In Figure 5.3(b), two cameras of the tracking system by Advanced Realtime Tracking GmbH are depicted. Since the relative poses of the two cameras and the geometry of the target are calibrated, the target's position can be triangulated in a Euclidean world reference frame (cf. Table 2.1, page 41).

The optical tracking system yields ground truth data consisting of rotation $\boldsymbol{R}_f$ and translation $\boldsymbol{t}_f$ for each image $f$ after a so-called *hand-eye calibration* is applied. Obviously, the tracking system only estimates the position of the target, but not the pose of the camera which is denoted by the camera center. This transformation—translation and rotation—which is fixed for all images, is calculated using a number of reference images of a calibration pattern. The transformation is obtained using the vector quantization based data selection scheme introduced in [Sch04a, Sch06]. Estimated camera poses from reconstruction algorithms are denoted by $\widehat{\boldsymbol{R}}_f$ and $\widehat{\boldsymbol{t}}_f$.

As proposed in [Vog06], the difference between reconstruction and ground truth data is given as average relative errors between camera pose pairs $(f, g)$ which are chosen randomly. Ad-

ditionally, a constraint may be imposed stating that the minimum frame distance in the image sequence is above a certain threshold, i. e., $|f - g| \geq \Delta_f$. A usual number of point pairs is $100$ or $1000$.

Calculating the relative translation between two camera poses is straightforward, the error is given as

$$\epsilon_{\boldsymbol{t},\text{rel}} = \frac{\|\boldsymbol{t}_{f,g} - \widehat{\boldsymbol{t}}_{f,g}\|}{\|\boldsymbol{t}_{f,g}\|} \quad , \tag{5.1}$$

where $\boldsymbol{t}_{f,g}$ is the translation between camera poses $f$ and $g$. The error thus denotes a percentage value of deviation from the true translation distance. However, the two translation values $\boldsymbol{t}_{f,g}$ and $\widehat{\boldsymbol{t}}_{f,g}$ are only comparable if the two reconstructions are scaled equally. In order to achieve this, the algorithm proposed in [Zin05] for registering two point sets with integrated scale estimation is applied. It uses SVD to minimize the distance between the two sets of camera poses with respect to the sum of squared differences. Thus, not only the correct scale factor is calculated, but also rotation and translation. A rough pre-alignment is achieved by mapping one camera pose of the estimation to the corresponding ground truth pose, analogously to equation (4.1), page 100.

For measuring the rotation error, the rotation difference between the two camera poses of a pair is calculated as

$$\boldsymbol{R}_{\text{diff},f,g} = \widehat{\boldsymbol{R}}_{f,g}^{\mathrm{T}} \boldsymbol{R}_{f,g} \quad . \tag{5.2}$$

In order to get an angular error, the rotation difference matrix is transformed to axis/angle representation, with $\boldsymbol{r}_{\text{diff},f,g}$ the rotation axis and $\phi_{\text{diff},f,g}$ the rotation angle. The relative rotation error is then computed as

$$\epsilon_{\boldsymbol{R},\text{rel}} = \frac{\|\phi_{\text{diff},f,g}\|}{\|\phi_{f,g}\|} \quad . \tag{5.3}$$

Contrary to the translation error $\epsilon_{\boldsymbol{t},\text{rel}}$, the relative rotation error is independent of a prior registration of the two camera pose sequences. The average errors over a large number of camera pose pairs are denoted by $\overline{\epsilon}_{\boldsymbol{t},\text{rel}}$ and $\overline{\epsilon}_{\boldsymbol{R},\text{rel}}$, respectively.

A second method for acquiring ground truth about camera poses is to place an object to be recorded on a turntable, and to fix the camera on a robot arm which can be moved in a vertical arc over the turntable. From the position angles of turntable and robot arm, the camera poses on a hemisphere over the object are derived. Relative translation and rotation errors are then computed as above.

So far, ground truth data was only acquired for the camera poses, but not for the 3-D object or scene being recorded. The usual way for providing ground truth for scene geometry is to use only objects with known properties. The most prominent of these are calibration patterns.

Most calibration patterns, however, are ill-suited for structure-from-motion as they often show repetitive patterns, leading to wrongly assigned features, or possess only a single surface which constitutes a degenerate case. Instead, spherical objects with a more "natural" surface texturing are used. A sphere is fitted to the reconstructed scene points by solving an over-determined linear system of equations for its radius and center for each 3-D point. The 3-D points are then scaled so that the radius of the sphere is normalized to $1$. The normalized relative shape error $\epsilon_{\text{shape}}$ is calculated as the mean distance of the 3-D points to the surface of the sphere.

### 5.1.3 Comparison of Evaluation Methods for Static Light Fields

Of the three options introduced previously for evaluating the quality of a light field reconstruction, back-projection error, PSNR, and ground truth data, the last one is clearly preferable to the former two, if available. Nevertheless, some restrictions have to be considered. Optical tracking itself has a limited accuracy, which is specified by the manufacturer to be $0.19$mm perpendicular to the optical axes of the tracking camera, and $0.36$mm parallel to them at a distance of up to $4$m. The rotational error is $0.14°$.

An additional error is introduced by hand-eye calibration. Comparing optical tracking with calibration using a calibration pattern, the error after hand-eye calibration is given in [Vog06] with $\overline{\epsilon}_{\boldsymbol{t},\text{rel}} = 2.5\%$ and $\overline{\epsilon}_{\boldsymbol{R},\text{rel}} = 2.2\%$ for the Sony DV camera shown in Figure 5.3(a). No such measurements are available for the turntable sequences.

In many cases however, optical tracking is not possible or desired. One drawback of these sequences is that a synchronization signal is required which is only available if S-Video is used for transmission. This analog signal is, however, inferior to the digital DV signal which is ideally provided, e. g., by the above Sony DV camera. In addition, S-Video images are generated by interlacing two fields of half the resolution of the whole image. This requires deinterlacing and interpolation of every second scan line, which further reduces image quality in contrast to progressive scan recording. In [Vog06], the sensor noise of S-Video is quantified with a mean standard deviation of $2.4$ averaged over the three color channels as opposed to DV with a value of $1.8$. These differences are of course noticeable for reconstruction.

Additionally, the optical tracking system is unsuitable for outdoor applications or $360°$ recordings where the tracking target is frequently occluded. In these cases, back-projection error and PSNR remain for quality measurement. They constitute meaningful measures as long as the characteristics stated in Section 5.1.1 are considered. Thus, their expressiveness increases if used together, and, in case of PSNR, the longer the image sequence under examination. For comparing different types of scene geometry, only the PSNR is meaningful, and both measures

depend on image content and should therefore not be used to compare methods across image sequences.

### 5.1.4 Quality Measures for Dynamic Light Fields

Evaluating the reconstruction of a dynamic light field is considerably more difficult than in the static case, and different methods have to be applied for step-wise static and rigid-object case. In the first case, back-projection error and PSNR may be likewise applied for the light field of each single step, but this yields no information about how well the light fields are registered. In order to measure this, the back-projection error of points visible in several time steps on one hand, and the average color value difference per pixel for the same view of the scene rendered for different time steps on the other hand are considered. The specific implementations of the two measures are described in Section 5.3.2.

In case of light fields of moving but rigid objects, the back-projection error for each object is as meaningful as before, but calculating the PSNR yields no additional information, as object and background in a rendered image are usually not comparable to a single source image. This leaves only the subjective impression of the observer as a means for comparison of rendering quality.

## 5.2 Evaluation of Static Light Fields

In this section, the methods for reconstructing a static light field, introduced in Chapter 3, are evaluated experimentally. First, the following subsection introduces all image sequences which are used later on in the experiments, describing the circumstances of their recording and explaining the naming conventions. The remainder of the section deals with different aspects of static light field reconstruction in the respective subsections.

### 5.2.1 Test Sequences and Hardware Description

The image sequences used for experimental evaluation can be divided into four different categories according to the method of recording. The identifiers given to each sequence reflect this method of recording, as well as the number of images in the sequences. The four categories are

- SET: An unordered collection of photographs taken with a Casio QV-3000 EX/Ir 3.3 Mpixel digital camera. The sets of images are used to evaluate reconstruction using SIFT features.

Figure 5.4: Test image collections: two images from *FountainSET-35* (first row) and *GlobeSET-17* (second row), and a complete example reconstruction for each collection in the right-hand image showing all camera poses and reconstructed 3-D points on the scene

- HH: Images taken with a ***h***and***h***eld camera, specifically the Sony DV camera shown in Figure 5.3(a) but without the reflective markers, using its non-interlaced progressive scan mode for recording. The images are thus of high quality and in sequential order, suitable for feature tracking using the KLT approach.

- TT: Objects placed on a ***t***urn***t***able with the camera mounted on a robot arm. This allows the recording of a hemisphere of images around the objects. The camera used is a Pulnix TMC-9700, which is also capable of progressive scan recording.

- ART: Image sequences with additional pose information obtained by the optical tracking system of Advanced Realtime Tracking GmbH (ART). As described in Section 5.1.2, the images are recorded using S-Video which requires deinterlacing, thus reducing image quality in comparison to the hand-held sequences.

The category of recording is part of the name of each image sequence. The length of each sequence is annotated in the name as well, so that each sequence name is of the form `<name><category>-<length>`.

Of the collections of photographs (SET), two instances are used for evaluating SIFT feature tracking in Section 5.2.2: *FountainSET-35* and *GlobeSET-17*, shown in Figure 5.4. As can be seen from the example reconstructions, the first collection, showing a fountain in the garden of

the Erlangen residence, represents a relatively dense sampling of the scene with camera positions closely together and at the same distance from the central object, but with a large depth difference between foreground and background. In the second collection, camera positions were quite far apart, laterally as well as in distance from the scene. All images of both sets are of size $1024 \times 768$ pixels.

Five of the test sequences, depicted in Figure 5.5, were taken with a hand-held camera (HH). The first one, *MilkHH-190*, is used throughout most of the following experiments. *ArtifactHH-160* shows an ancient Greek vase which was, although placed on a turntable, recorded with the Sony DV camera mounted on a tripod. Since the vertical position of the tripod was not measured, the sequence is nevertheless categorized as hand-held. It is used in Section 5.2.3 to point out the effects of non-sequential tracking and reconstruction. For each of the remaining three sequences, *OfficeHH-109*, *CandyHH-113*, and *SantaHH-207*, a second sequence is available which is combined with the first one in Section 5.2.7 to demonstrate the extension of existing light fields. The image size for all sequences is $512 \times 512$ pixels cut out symmetrically from the original PAL resolution since images of size $2^n \times 2^n$ can be rendered most efficiently.

The next category of sequences is those of objects placed on a turntable (TT). Figure 5.6 exemplifies two of the three sequences available. While *SantaTT-752* covers almost the whole hemisphere around the object with a total of $752$ images, *ColaTT-40* represents only one revolution of the turntable. The third sequence, *SantaTT-200*, constitutes the lower five revolutions of the turntable of sequence *SantaTT-752*. It is thus very similar to *SantaTT-752* and therefore not depicted itself. While *SantaTT-752* is used in Section 5.2.3 dealing with viewpoint meshes, the latter two sequences are utilized in Section 5.2.7. The images of these sequences are the smallest in the test set, with only $256 \times 256$ pixels.

The last sequences to be introduced are those which were taken with a hand-held camera and an optical tracking system in order to gain information about the camera poses. The first sequence depicted in Figure 5.7, *GlobeART-501*, is used in most of the following sections as it supplies a measure for the correctness of 3-D reconstruction, depending on the deviation from a sphere for the reconstructed globe. Three more sequences are derived from it, *GlobeART-200*, *GlobeART-100*, and *GlobeART-50*, by leaving out images at regular intervals. These are used in the next section only for comparing SIFT features and KLT tracking. The remaining sequences were included to test various special properties: *CirclesART-501* and *CrawlerART-510* contain many recurring camera poses, and *GlassART-150* includes some transparent portions which are likely to generate unreliable features. While *CirclesART-501* is again used in different experiments, *GlassART-150* is applied to test robust methods in Section 5.2.5, and *CrawlerART-510* is only


Figure 5.5: Test sequences taken with a hand-held camera: (from top to bottom) *MilkHH-190*, *ArtifactHH-160*, *OfficeHH-109*, *CandyHH-113*, and *SantaHH-207*. The right-hand image shows an example reconstruction for each sequence.

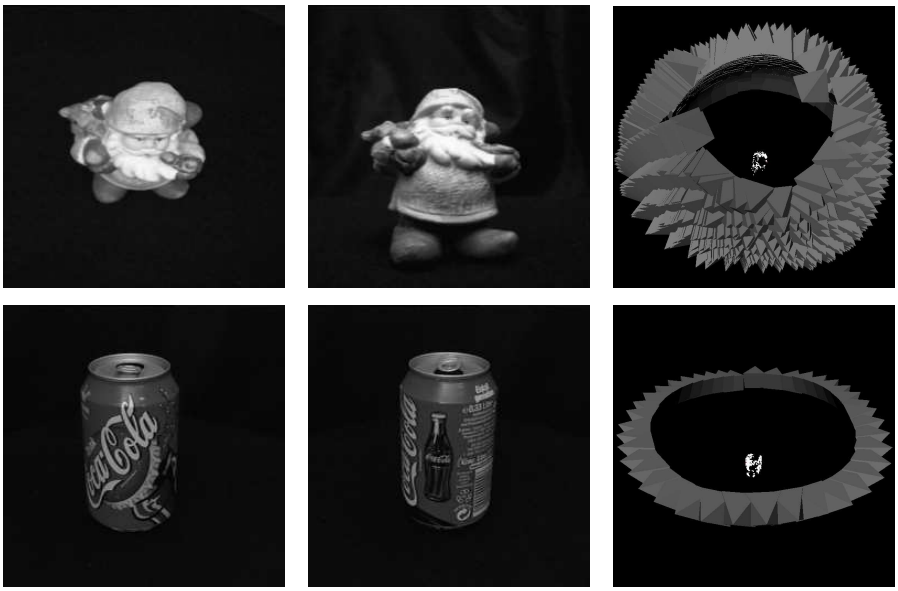


Figure 5.6: Test sequences of objects on a turntable: (top row) *SantaTT-752* and (bottom row) *ColaTT-40*, with an example reconstruction for each

used in Section 5.2.7. Images are generally of size $512 \times 512$ pixels, only in case of *CrawlerART-510* the native PAL format of $768 \times 576$ pixels is retained. In this case, this is required to improve feature tracking.

All experiments, for static as well as for dynamic light fields, are performed on a Linux PC with an AMD Athlon 64 3000+ CPU and 2 GByte of main memory. The graphics hardware is based on the NVidia GeForce 6800 chipset. The results shown in the following may vary if the software used to run the experiments is compiled with different compilers, compiler versions, or optimizations for different types of CPU architecture. In general, however, it has been found that these variations are small and do not alter the conclusions drawn. Only in cases where techniques showed an unstable behaviour for certain input data, distinct differences were observed at times.

### 5.2.2 SIFT Features and KLT Tracking

In Section 3.2 two alternatives for generating feature correspondences have been introduced: the matching of SIFT features which are described by equations (3.13) and (3.14) on page 57, and feature tracking using the Kanade-Lucas-Tomasi (KLT) approach as defined by equation (3.12), page 56. The advantage of SIFT features is that by matching features from each image with all others, a sequential order of the images is not required. In this section, the properties of SIFT and KLT features are compared regarding accuracy of reconstruction, but also their runtime


Figure 5.7: Test sequences with pose data from optical tracking: (from top to bottom) *GlobeART-501*, *CirclesART-501*, *GlassART-150*, and *CrawlerART-510*. An example reconstruction is given in the right-hand image for each.

requirements.

The advantage of SIFT features, their reliability despite large image disparities, is shown by the first experiment where SIFT feature correspondences were generated for the two sets of photographs *GlobeSET-17* and *FountainSET-35*. In order to estimate the camera pose parameters, a variant of non-sequential reconstruction (Section 3.4.2) is used, where the camera mesh constitutes a fully meshed graph. The next camera pose to be estimated is selected as the one for which the highest number of visible features have already been triangulated. Only the extrinsic cam-

| Sequence | PSNR | back-proj. err. | sphere dev. | time |
|---|---|---|---|---|
| *GlobeSET-17* | 14.9 dB | 0.548 | 0.139% | 296s |
| *FountainSET-35* | 15.7 dB | 1.73 | — | 1242s |

Table 5.2: Reconstruction results for two sets of photographs using SIFT features. A reconstruction using tracked features was not possible.

Figure 5.8: Images rendered from the *GlobeSET-17* and *FountainSET-35* light fields reconstructed using SIFT feature correspondences

era parameters are estimated, the true intrinsic parameters were determined beforehand using a calibration pattern.

The results of reconstructing the two sets of images are shown in Table 5.2. In both cases, reconstruction was successful for SIFT features with relatively low back-projection errors, while results for KLT tracking are not given since from the feature correspondences generated, no reconstruction was possible. The low PSNR values, however, are explained easily. In both cases, reconstruction was based on very few features, which are then used to generate local proxies as scene geometry. But since both scenes possess a large depth range, this scene geometry forms only a very rough approximation. This is apparent in the rendered example images of Figure 5.8. Especially for the globe image, the patch where feature correspondences were available stands out because of its high contrast, while the image gets increasingly blurred farther outside. This effect is amplified by the very low number of images which were taken from far apart viewpoints. Although successful, generating the feature correspondences took $4.8$ and $20.9$ minutes, respectively, which is about two orders of magnitude slower than KLT tracking in this case.

The second experiment was designed to compare the accuracy of SIFT features and KLT tracking when applied to image sequences with little disparity between consecutive images. The results shown in Table 5.3 for sequences *GlobeART-200* and *MilkHH-190* suggest that both types

| Sequence | Type | PSNR | back-proj. err. | sphere dev. | trans. err. | rot. err. |
|---|---|---|---|---|---|---|
| *GlobeART-200* | **KLT** | 30.4 dB | 0.393 | 0.217% | 8.42% | 11.2% |
| | **SIFT** | 30.5 dB | 0.470 | 1.63% | 8.56% | 9.55% |
| *GlobeART-100* | **KLT** | 26.8 dB | 0.667 | 1.22% | 9.88% | 14.4% |
| | **SIFT** | 26.8 dB | 0.547 | 1.28% | 17.6% | 9.45% |
| *GlobeART-50* | **KLT** | 8.75 dB | 1.51 | 1.74% | 751% | 127% |
| | **SIFT** | 22.9 dB | 0.443 | 0.623% | 7.93% | 9.36% |
| *MilkHH-190* | **KLT** | 25.4 dB | 0.954 | — | — | — |
| | **SIFT** | 26.8 dB | 0.900 | — | — | — |

Table 5.3: Comparison of reconstructions using SIFT features and KLT tracking. While results are comparable for image sequences, the SIFT features get better the farther the images are apart.

of feature correspondences are comparable regarding accuracy, since all error measures are consistently low. This accuracy, however, is gained at the expense of exponential complexity in case of the SIFT features, as matching is done for all possible pairs of images. For *GlobeART-200* for instance, SIFT feature matching takes several hours for all images, while KLT tracking requires only $56$ seconds.

Sequences *GlobeART-100* and *GlobeART-50* are intended to show the effect of increasing disparity between images. Therefore, every second image for *GlobeART-100* and three out of four images for *GlobeART-50* were removed from *GlobeART-200*. Consequently, error rates increase drastically using KLT tracking, so much that *GlobeART-50* could not be reconstructed properly. For SIFT features, the reconstruction errors remain roughly the same, although PSNR decreases due to the greater baseline between images and thus increased influence of incorrect depth.

**Summary:** While accuracy may be comparable between these two types of feature generation, linear tracking using the KLT method should be chosen if image sequences with little disparity between consecutive images are processed because of its much lower complexity. However, for collections of photographs or sequences with low frame rates, SIFT features may be the only possible option, even if processing time increases rapidly with the number of images. Thus, a combination of the two methods is a logical choice for generating a light field from multiple image sequences, as it is described in Section 3.7.4.

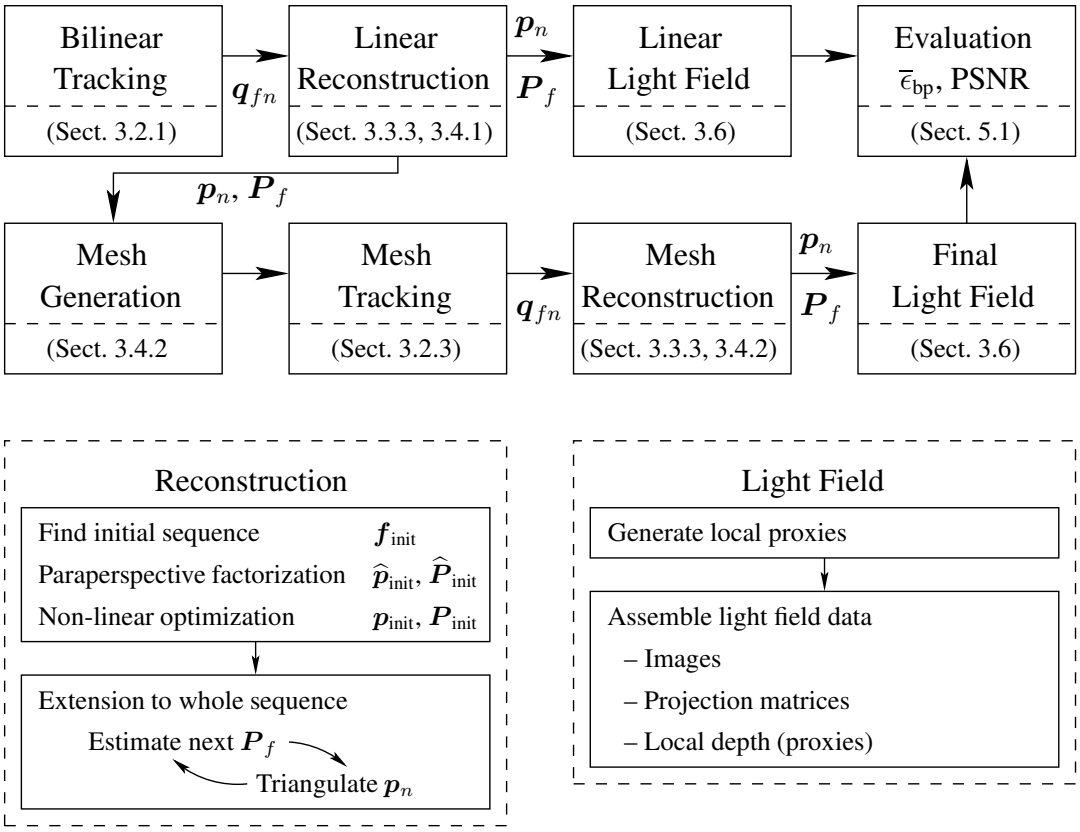



Figure 5.9: Processing steps for linear and non-sequential (mesh) reconstruction of a light field. The two steps of camera parameter and 3-D reconstruction and light field generation are outlined in detail. Since they are mostly identical for both types of reconstruction, they are depicted only once.

### 5.2.3 Non-Sequential Tracking and Reconstruction

It was pointed out before that processing an image sequence in sequential order may lead to error accumulation. Therefore, non-sequential processing was introduced for feature tracking (Section 3.2.3) and camera parameter estimation (Section 3.4.2). The creation of a camera mesh from the results of a preceding, sequential reconstruction, which is described by equations (3.26) and (3.27) on page 70, is required as a preprocessing step.

In order to compare sequential (linear) and non-sequential (mesh) reconstruction, five image sequences with different properties were selected. The first experiment was conducted in the same way for all sequences, i. e., bilinear tracking was applied and after initial factorization, images are processed according to Section 3.4.1. For the second experiment, a camera mesh was constructed as described in Section 3.4.2. Depending on the kind of information available for a sequence, the mesh was either generated from the previous, linear reconstruction (sequences *GlobeART-501*, *CirclesART-501*, *MilkHH-190*) or from known turntable positions (sequences

| Sequence | Type | features | avg. trail length |
|---|---|---|---|
| *GlobeART-501* | **linear** | 10507 | 52.3 |
| | **mesh** | 4568 | 83.8 |
| *CirclesART-501* | **linear** | 14746 | 31.7 |
| | **mesh** | 4183 | 88.1 |
| *SantaTT-752* | **linear** | 57041 | 8.64 |
| | **mesh** | 2166 | 201 |
| *ArtifactHH-160* | **linear** | 18885 | 6.55 |
| | **mesh** | 3837 | 25.8 |
| *MilkHH-190* | **linear** | 4455 | 42.2 |
| | **mesh** | 1808 | 85.7 |

Table 5.4: Features tracked with sequential and non-sequential tracking

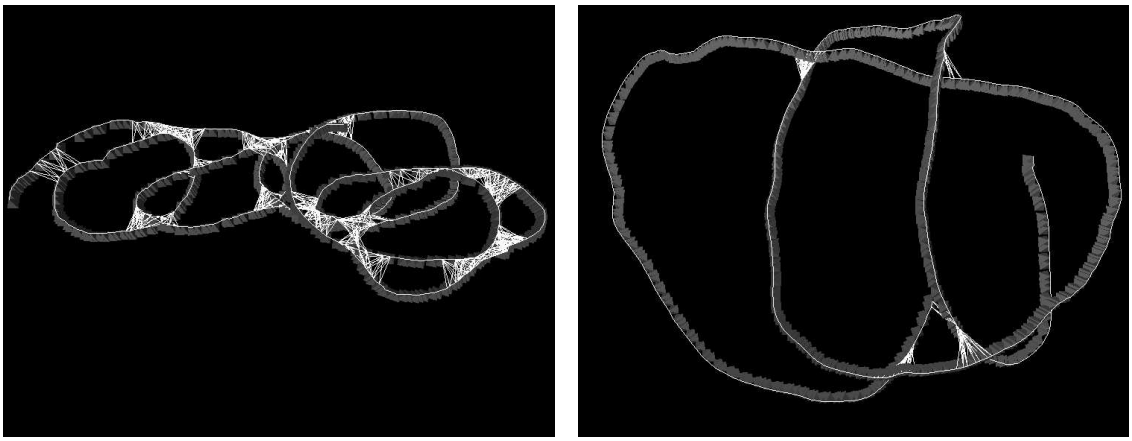
| Sequence | Type | PSNR | back-proj. err. | sphere dev. | trans. err. | rot. err. |
|---|---|---|---|---|---|---|
| *GlobeART-501* | **linear** | 31.1 dB | 0.376 | 1.96% | 10.5% | 16.4% |
| | **mesh** | 29.9 dB | 0.548 | 0.586% | 10.1% | 15.0% |
| *CirclesART-501* | **linear** | 31.7 dB | 0.941 | — | 14.8% | 23.4% |
| | **mesh** | 30.9 dB | 1.23 | — | 25.2% | 61.1% |
| *SantaTT-752* | **linear** | 19.3 dB | 0.820 | — | 97.0% | 62.7% |
| | **mesh** | 28.8 dB | 0.842 | — | 17.3% | 13.4% |
| *ArtifactHH-160* | **linear** | 14.2 dB | 0.832 | — | — | — |
| | **mesh** | 26.7 dB | 1.04 | — | — | — |
| *MilkHH-190* | **linear** | 26.7 dB | 0.685 | — | — | — |
| | **mesh** | 25.5 dB | 1.04 | — | — | — |

Table 5.5: Comparison of sequential and non-sequential reconstruction

*SantaTT-752*, *ArtifactHH-160*). Feature tracking and reconstruction were then performed again using this mesh information. The two procedures are outlined in Figure 5.9. As can be seen, they share a number of processing steps. Camera pose and 3-D reconstruction as well as light field generation are mostly identical in both cases, such that the detailed itemizations given for those processing steps in two separate blocks of the diagram are valid for linear as well as mesh reconstruction. Only the sequence of processing the images will usually differ for reconstruction.

The most obvious benefit of mesh tracking can be observed in Table 5.4. Using the camera mesh, the number of features tracked for one image sequence decreases considerably, while the average length of a trail, i. e., the number of images a feature was visible in, increases at almost the same rate.

From the evaluation results for 3-D and camera pose reconstruction of Table 5.5 on the other hand, a number of ambiguous conclusions can be drawn. In all cases, back-projection errors



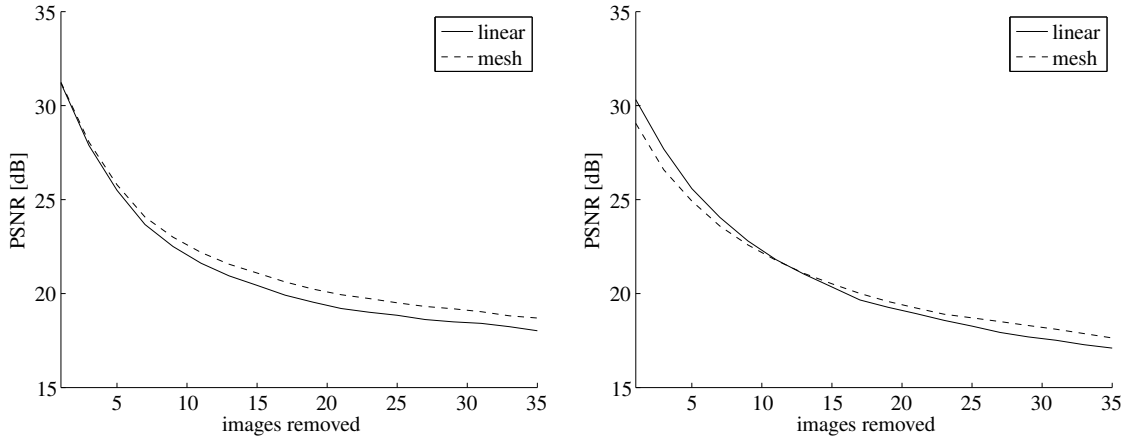
(a) *CirclesART-501* (b) *GlobeART-501*

Figure 5.10: Camera meshes for two example sequences. Sequence *CirclesART-501* represents a noticeably denser sampling of the scene. The interconnecting mesh is denoted by lines connecting the centers of projection of neighboring cameras (pyramid tips).

increased after the application of mesh reconstruction. Considering that this error reflects the fit of a 3-D point to its 2-D projections, the increase is not surprising since a larger number of 2-D features is available for each 3-D point.

The remaining error measures show clearly that the success of non-sequential reconstruction depends strongly on the type of image sequence. In case of the two turntable sequences, *SantaTT-752* and *ArtifactHH-160*, an obvious improvement is achieved. They represent a dense sampling of the scene, which results in short distances from one camera position to its neighbors, and therefore very similar images in all directions. This prerequisite is beneficial for KLT tracking, leading to small errors during tracking. The other three sequences all constitute sparse samplings of the scene to varying degrees, which can be seen in the comparison between two of them in Figure 5.10. They are suited less for non-sequential reconstruction, which is reflected in a worsening of both PSNR and pose errors in the non-sequential case.

Nevertheless, this result is still misleading, mainly because PSNR, as it is used here, measures local light field quality, while non-sequential processing improves the results globally—many features are re-detected which are otherwise lost, and thus valid for a larger number of images. However, if only the current image is removed for calculating the image difference, only a local neighborhood of images is drawn on for rendering. The more images in the neighborhood of the current image are removed, the farther away the images used for rendering will be, thus reflecting better the global correctness of reconstruction.




(a) *CirclesART-501* (b) *GlobeART-501*

Figure 5.11: Local and global error in non-sequential reconstruction using PSNR, measured by an increasing number of images removed during PSNR calculation

This is exemplified in Figure 5.11, where an increasing number of images was removed for PSNR calculation for sequences *CirclesART-501* and *GlobeART-501*. In both cases, PSNR gets better eventually for non-sequential reconstruction, showing that the result is globally more correct. Additionally, the number of local proxies of neighboring images which were used to calculate depth for each image was increased for this experiment, which improved PSNR for the non-sequential case. This shows that local proxies are more consistent as well, as they are calculated from fewer, non-ambiguous 3-D points.

**Summary:** The benefits of non-sequential tracking and reconstruction depend strongly on the way the input images were acquired. In case of a dense sampling of the scene, it may considerably increase the average number of correspondences for each feature, thus improving the reconstruction results. Nevertheless, for less densely sampled scenes, the greater global consistency achieved is not necessarily reflected in light field quality. In those cases where, e. g., the image sequence contains only few images, the additional steps of mesh tracking and reconstruction are not necessary and can be omitted.

### 5.2.4 Automatic Selection of Factorization Methods

As was pointed out in Section 3.3, the factorization methods used to process an initial subsequence show different properties regarding accuracy and robustness. In general, the more

| Sequence | Type | PSNR | back-proj. err. | sphere dev. | trans. err. | rot. err. |
|---|---|---|---|---|---|---|
| *GlobeART-501* | **para** | **29.2 dB** | **0.184** | 0.0369% | 8.46% | 13.1% |
| | **christy** | 29.1 dB | 0.314 | 0.0508% | 12.9% | 20.3% |
| | **aff. upd.** | 4.44 dB | 64.3 | 99.7% | 2280% | 310% |
| | **quadric** | — | — | — | — | — |
| *CirclesART-501* | **para** | 29.1 dB | 1.10 | — | 46.5% | 74.4% |
| | **christy** | **29.9 dB** | **1.03** | — | 30.6% | 62.6% |
| | **aff. upd.** | 3.82 dB | 1.17 | — | 165% | 212% |
| | **quadric** | 22.0 dB | 1.31 | — | 708% | 89.2% |
| *MilkHH-190* | **para** | 26.2 dB | 0.544 | — | — | — |
| | **christy** | 26.1 dB | 0.595 | — | — | — |
| | **aff. upd.** | **26.3 dB** | **0.269** | — | — | — |
| | **quadric** | 6.47 dB | 0.721 | — | — | — |

Table 5.6: Automatic selection of the initial factorization method for three different image sequences. The method finally selected according to PSNR and back-projection error is emphasized in each case.

accurate a process, the greater its shortcomings in robustness.

The selection scheme introduced in Section 3.3.4 determines the best factorization method from four alternatives by judging PSNR and back-projection error of the reconstruction results of the initial subsequence they were applied to. A block diagram of the selection scheme was already provided in Figure 3.4, page 65, where the four alternative processing chains are shown in direct comparison. They are designated here according to their main factorization and self-calibration methods, respectively, i. e., paraperspective (*para*), as described in Section 2.2.2, iterative by Christy and Horaud (*christy*), Section 2.2.3, perspective with update to a metric reconstruction using the affine solution (*aff. upd.*), explained in Sections 2.2.2 and 3.3.3, and perspective factorization with self-calibration using the absolute quadric (*quadric*). Details of the last method can be found in Sections 2.2.3 and 2.3.2.

In Table 5.6, the application to three of the example sequences is shown. In each case, the length of the subsequence was preset, and the best subsequence was determined with respect to the highest number of features. The three examples selected here show that indeed, the most suitable method depends on the image sequence and should not be fixed beforehand. In case of the globe sequence, the spherical setup of the features poses a degenerate case for perspective reconstruction, so that self-calibration failed completely—i. e., the estimated quadric was not positive definite—and the affine update yielded no usable result either. For the second sequence, the affine approximations perform better as well, mostly again because the features are located on a roughly flat surface (the keyboard), being badly suited for reconstruction. However in this case, the choice is not as obvious as before regarding only back-projection error, which is

comparable for all four methods. Translation and rotation errors confirm on the other hand that the selection regarding PSNR was correct. The third sequence shows that, if image quality is sufficient, perspective factorization may clearly outperform the affine ones, although the solution using the absolute quadric failed here again. The final selection was done regarding the average back-projection error.

Reconstruction times for the automatic selection algorithm include, of course, time for evaluating the PSNR of each individual result. Thus, automatic selection for the *CirclesART-501* sequence requires a total of $4.8$ minutes, while the sum of reconstruction times alone computes to $2.5$ minutes. On the other hand, the user is not required to manually inspect the results if one of the algorithm fails.

**Summary:** The success of a factorization method depends on many different factors so that it is not obvious which method is most suitable for a certain image sequence. The automatic selection scheme using PSNR and back-projection error reliably determines those attempts where reconstruction failed completely and selects the best reconstruction from the successful ones. Since a successful overall reconstruction depends highly on a correct treatment of the first subsequence, using this selection procedure is recommended whenever time enough is available for a number of trials.

### 5.2.5 Robust Methods

Robust estimation techniques are usually recommended if the underlying data are of low quality, contaminated by noise or prone to other errors which manifest themselves as outliers. In Chapter 3, a number of these techniques have been described, the most important ones of which will be evaluated here regarding their effectiveness for 3-D reconstruction.

Four robust techniques, which are all applied to the extension of reconstruction after an initial sequence has been processed using a factorization method, are compared here using three different image sequences. The four methods are:

- *Outlier detection:* the LMedS technique described in Section 3.3.1 is applied as outlined in Section 3.4.1 for identifying outliers in 3-D point positions.

- *Feature weighting:* the third of the feature weighting approaches described in Section 3.5.1, which uses the structure matrix of each feature defined by equation (3.31), page 73, is evaluated, being the most accurate one of these methods.

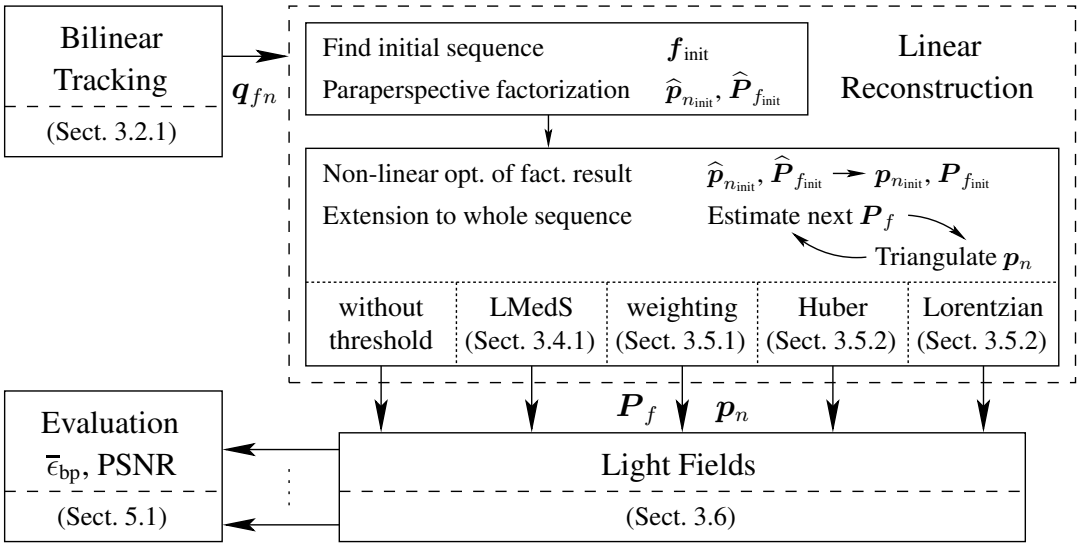



Figure 5.12: Processing steps for the comparison of outlier detection, weighting of features, and robust estimators with the standard reconstruction method without error threshold. The five methods tested are applied during non-linear optimization as part of the linear reconstruction method.

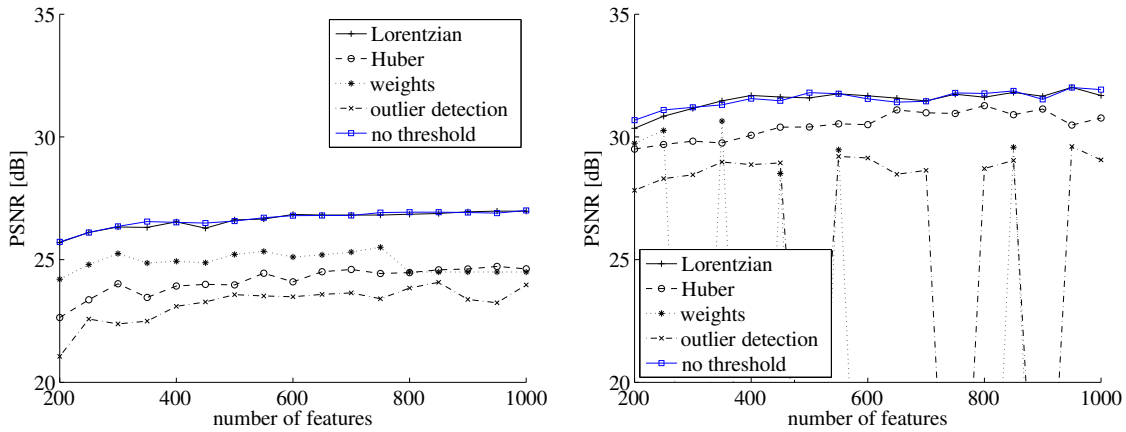
- *Huber's minimax estimator:* this robust estimator which was introduced in Section 3.5.2 using equation (3.33), page 74, is applied with the threshold parameter $k = 2.0$.

- *Lorentzian estimator:* for this second robust estimator, the regulating parameter of equation (3.34) is set to $k = 0.5$. This value for $k$ and the one for Huber's estimator are commonly used as they approximate the quadratic error well up to an error of about one pixel and are thus suitable for most input image sequences. Further, sequence-dependent optimization is possible but not examined as part of this thesis.

The complete progression of the experiments conducted here is shown in Figure 5.12. As can be seen there, all the robust techniques are applied to the non-linear extension method after factorization of the initial subsequence, including the non-linear refinement step of the factorization result. The factorization method is applied identically for each technique, but different settings are used for different image sequences.

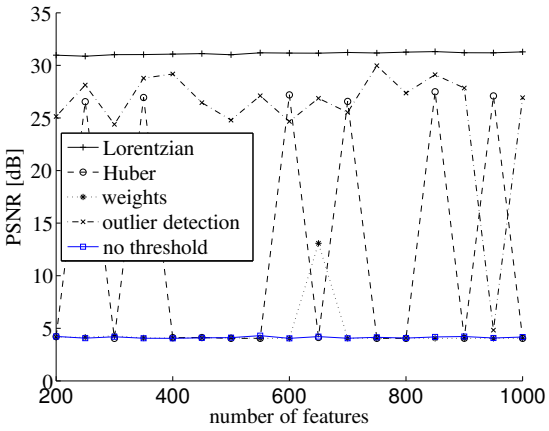
The three image sequences selected for these experiments are *MilkHH-190*, *GlassART-150*, and *GlobeART-501*. The sequence *GlassART-150* was included in order to demonstrate the effect of bad point correspondences in a sequence with large areas of potential outliers, as is the case here with the transparent and reflecting surfaces.

In Figure 5.13, the four robust methods are compared to the standard reconstruction method without discarding any bad feature correspondences above a given back-projection error thresh-




(a) *MilkHH-190*

(b) *GlassART-150*




(c) *GlobeART-501*

Figure 5.13: Comparison of PSNR values of different robust reconstruction methods for an increasing number of point features

old. This method is labeled “no threshold”. The reconstructions are performed for a varying number of tracked features per image, ranging from 200 to 1000 features, and the PSNR is given for quality measurement. In case of the first two sequences, the features detected are obviously good enough so that neither a threshold for discarding features is necessary for good reconstruction, nor do the robust methods improve the results. Except for the Lorentzian estimator, the robust methods even decrease the resulting light field quality.

The third sequence, *GlobeART-501* imposes greater demands on the reconstruction method, which is demonstrated by Figure 5.13(c), since the normal method fails here. Again, the Loren-

tzian estimator yields consistently good results, outlier detection has a positive effect and Huber's minimax yields a good reconstruction in some cases. Using the structure matrix as feature weight again has no effect on reconstruction quality, a result which was obtained by Kanazawa and Kanatani in [Kan01b] as well—the features detected automatically are good enough so that weighting does not yield further improvement.

**Summary:** The experiments above show that while robust estimators like the Lorentzian do not necessarily improve the reconstruction quality, they may indeed help to increase the stability, i. e., robustness, of estimation. While the Lorentzian estimator with $k = 0.5$ seems to be a good choice in any case, this is not the case for the other methods tested. Here, the conclusion of [Kan01b] that probabilistic modeling is not efficient if feature correspondences are already well selected holds, too. However, the reconstruction algorithms proved to be unperturbed by the transparent and reflecting objects in *GlassART-150* and likewise in *MilkHH-190*, as long as enough diffuse surfaces are available.

### 5.2.6 Scene Geometry

In Section 3.6 four different types of scene geometry were introduced: depth maps by interpolating reconstructed 3-D points, a variational stereo disparity approach, local proxies, and the generation of one global proxy. Now, these types of geometry are compared to each other regarding rendering quality and efficiency.

The experiments are performed on two different image sequences, *MilkHH-190* and *ColaTT-40*. In case of the first sequence, only an initial subsequence of $30$ images was used for geometry reconstruction so that a global proxy of the first type, i. e., with the mesh extended up to the image borders, can be generated. The second sequence constitutes a complete rotation of the camera around an object, so that here, the initial sequence has to be short since all features will be lost eventually. In fact, the initial subsequence which also creates the initial proxy mesh is only ten images long, and for the rest of the images, the extension technique for long image sequences is applied to the proxy as described in Section 3.6.3.

Figure 5.14 shows examples for the variational approach, local, and global proxies for both sequences. The left and middle images each represent depth corresponding to one image, the geometry meshes on the right are valid for the whole sequence. As can be seen in the second row, the local and global proxies of the second sequence only model the object present in the scene, but not the background. While for the global proxy this is a consequence of the algorithm used, it is an option for the local proxies chosen here to provide comparability for the two approaches.

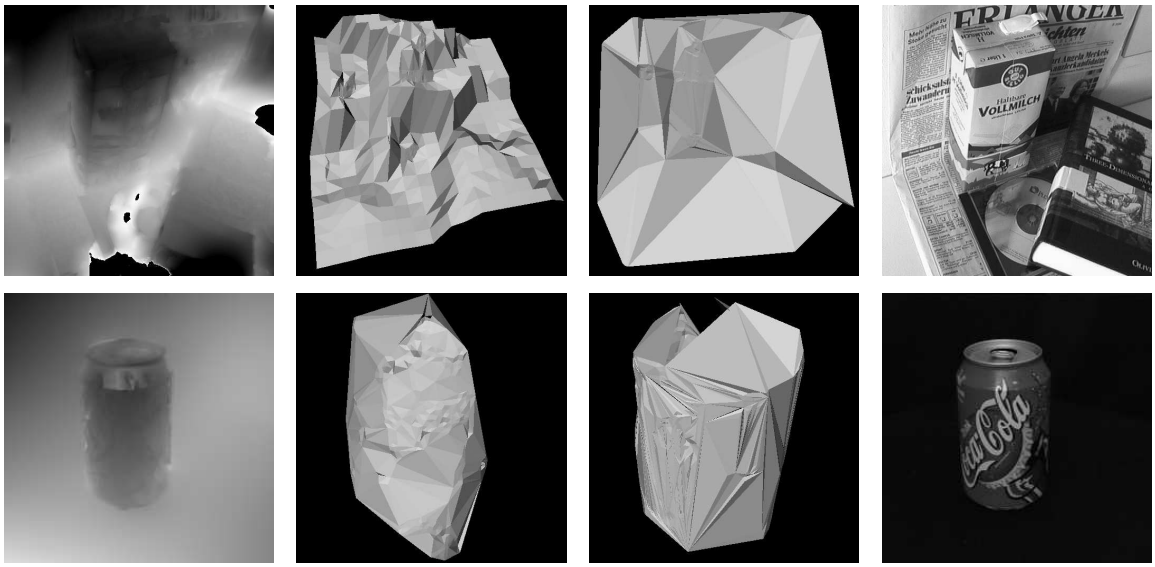



Figure 5.14: Examples for different types of geometry for the two image sequences *MilkHH-190* (top row) and *ColaTT-40* (bottom row): variational approach by Alvarez et al. (left), local proxy (middle left), global proxy (middle right), and an original input image for comparison (right)

| | *MilkHH-190* | | *ColaTT-40* | |
|---|---|---|---|---|
| **Type** | **PSNR** | **time/frame** | **PSNR** | **time/frame** |
| **3-D points** | 24.3 dB | 0.298s | 27.1 dB | 0.370s |
| **variational** | 22.0 dB | 0.352s | 27.6 dB | 0.662s |
| **local proxies** | 24.7 dB | 0.0643s | 21.5 dB | 0.0193s |
| **global proxy** | 24.6 dB | 0.0649s | 21.4 dB | 0.0192s |

Table 5.7: Comparison of rendering quality and speed for different types of geometry: depth maps interpolated from 3-D points, using a variational image disparity approach, local proxies, or one global proxy

The figures of Table 5.7 compare the four resulting light fields for each sequence regarding rendering quality in terms of PSNR, as well as rendering time in seconds per frame. The obvious conclusion is that both types of proxies, local and global ones, are considerably more efficient for rendering than depth maps. The variational approach loses additional time as it utilizes confidence maps to mask out those areas where no depth information is available.

These areas are also responsible for the bad quality results of the variational approach for the *MilkHH-190* sequence. Appearing preferably at the image border, areas may remain during rendering where no depth was found at all, resulting in a high error compared to the original image. This effect is visible in the difference image between original and rendered image shown in Figure 5.15(a). Additionally, the algorithm generates wrong depth values for reflecting surfaces

(a) (b) (c)

Figure 5.15: Problems occurring for different kinds of geometry: (a) missing borders result in high image differences for the variational approach; (b) parts of the object are missing if local proxies are not extended to the image borders; (c) global proxies may be incomplete.

like the CD case in sequence *MilkHH-190*, where the other algorithms just use an average depth of the surrounding features. For sequences where these problems do not occur, as in the second case, the accuracy of the variational approach consequently yields the best results.

The low quality results of local and global proxies for the *ColaTT-40* sequence can be ascribed to the lack of depth information outside and on the rim of the central object in the scene. Although local proxies are easily extended to each respective image border, this is not done here to ensure comparability with the global proxy in both quality and run-time. The result is a somewhat truncated but well rendered object as shown in Figure 5.15(b).

The global proxy of sequence *ColaTT-40* exhibits another of the remaining drawbacks of the algorithm used for generating it. As can be seen in Figure 5.15(c), closing the final gap on its backside where the algorithm finished did not work out. While the algorithm is theoretically able to do this, it is too restrictive in this case regarding the visibility of features in the pertaining images to succeed here.

**Summary:** Regarding rendering speed as well as quality, local proxies currently represent the best choice for light field geometry. Even regarding their generation, constructing a 2-D Delaunay mesh can be done very efficiently, and storage requirements are very low. Dense depth maps using optical flow, such as the variational approach used here, yield potentially better results, but are less reliable, less efficient during rendering, and require significantly more time for their computation. Global proxies would pose a serious alternative to local ones once their reconstruction could be done in a stable manner. Rendering performance benefits, however, can only be expected for much larger sets of input images.

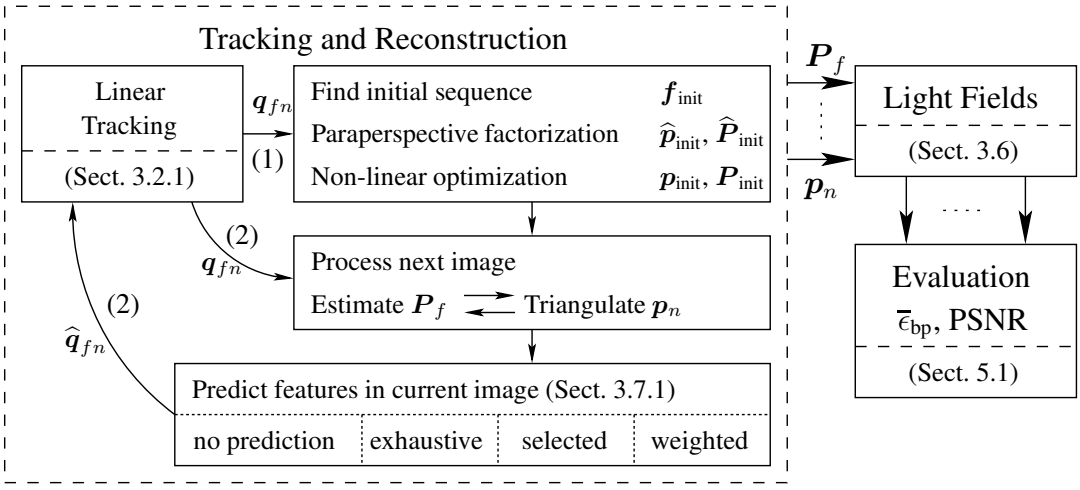



Figure 5.16: The integrated process of tracking and reconstructing an image sequence image by image. Processing step (1) is only performed once for the initial subsequence, while the iteration (2) is repeated until all images have been processed. The distinction between no feature prediction and the exhaustive, selected, and weighted prediction schemes is located inside the prediction step.

### 5.2.7 Feedback Loops

For the experiments in this section, the reconstruction algorithm integrating tracking and calibration in a single pass is applied according to Figure 3.13, page 85. This reconstruction scheme is equivalent to linearly tracking features through the sequence without the usually applied backwards tracking step, followed by linear calibration. Therefore, reconstruction quality is lower in general, but the process offers the benefit of a streaming-like processing of incoming images.

#### Prediction by Back-Projection

The prediction of features for tracking by back-projecting previously reconstructed points into the most recent image as introduced in Section 3.7.1 is similar in its effects to non-sequential reconstruction of Section 3.4.2. Two image sequences, *CirclesART-501* and *CrawlerART-510*, were selected because of their length and repetitive motion paths, and reconstructed using four different schemes: no prediction, prediction using exhaustive search of source images, and source image selection using the highest number of features ("selected") as well as weighting according to equation (3.43) on page 86 ("weighted"). For the last two schemes, the number of images to track features from was limited to five.

Figure 5.16 offers a detailed overview over the reconstruction process for this experiment. The iteration of tracking, reconstruction, and prediction which is repeated for each additional image after processing the initial sequence is labeled with (2). The four different prediction

| | *CirclesART-501* | | *CrawlerART-510* | |
|---|---|---|---|---|
| **Type** | **features** | **avg. trail length** | **features** | **avg. trail length** |
| **no prediction** | 14716 | 20.4 | 28138 | 10.9 |
| **exhaustive** | 5769 | 63.3 | 8561 | 37.7 |
| **selected** | 5138 | 70.7 | 9245 | 34.2 |
| **weighted** | 5244 | 69.3 | 10380 | 30.3 |

Table 5.8: Features tracked without prediction and with different selection schemes for prediction

| **Sequence** | **Type** | **PSNR** | **b.-proj. err.** | **trans. err.** | **rot. err.** | **time/image** |
|---|---|---|---|---|---|---|
| *CirclesART-501* | **no pred.** | 30.6 dB | 0.884 | 18.0% | 30.8% | 5.58s |
| | **exhaustive** | 30.8 dB | 25.9 | 15.8% | 34.3% | 12.6s |
| | **selected** | 30.0 dB | 43.8 | 14.2% | 37.3% | 9.49s |
| | **weighted** | 29.8 dB | 43.7 | 30.1% | 51.2% | 9.61s |
| *CrawlerART-510* | **no pred.** | 26.0 dB | 0.697 | 8.35% | 10.8% | 6.46s |
| | **exhaustive** | 24.8 dB | 24.6 | 9.45% | 9.65% | 34.9s |
| | **selected** | 23.5 dB | 41.7 | 10.1% | 10.7% | 10.2s |
| | **weighted** | 23.9 dB | 42.5 | 11.2% | 12.1% | 10.3s |

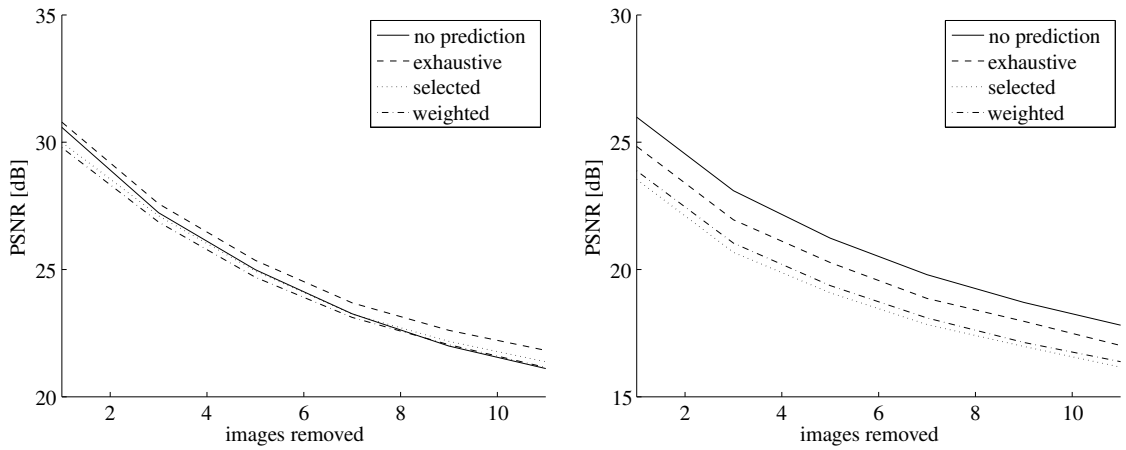
Table 5.9: Comparison of different selection schemes for feature prediction by back-projection with linear reconstruction

schemes are all located in the same processing step in this loop.

As can be seen in Table 5.8, the average feature trail length is increased as substantially as in the experiments for non-sequential reconstruction (Section 5.2.3), by a factor of more than three. Surprisingly, this effect is not necessarily the most pronounced for exhaustive search. In case of sequence *CirclesART-501* it is more distinct for the selective schemes.

Just like for non-sequential reconstruction, the results shown in Table 5.9 suggest that local correctness is not necessarily increased by prediction. Likewise, but on a much larger scale, back-projection error increases which is due to the fact that reused features may badly fit their projections in the most recent image and are discarded, but continue to contribute to the total error. This effect is less pronounced in the non-sequential case since there, the order of feature tracking and reconstruction are both determined by the view mesh. However, errors compared to ground truth and PSNR stay at the same level compared to normal reconstruction if prediction by back-projection is applied.

An increase of global correctness, which is examined by again leaving out a higher number of neighboring images during PSNR calculation, was only observable for sequence *CirclesART-501*. PSNR for different numbers of images removed is plotted in Figure 5.17 for both sequences. The runtime behavior is as expected, as computation time increases most for exhaustive search,




(a) *CirclesART-501* (b) *CrawlerART-510*

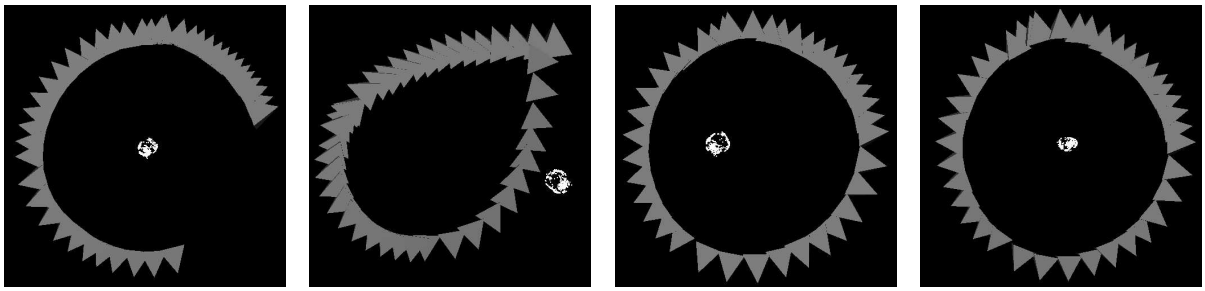
Figure 5.17: Local and global error for linear reconstruction and feature prediction with back-projection using PSNR

even distinctly more so for *CrawlerART-510* where the same section of the scene is visible in more images. The selective schemes on the other hand increase runtime by a fixed amount since the number of images for retracking is fixed, too. In general, the "selection" scheme performs slightly better than weighting.

**Summary:** While considerably increasing the length of feature trails being tracked, prediction by back-projection does not necessarily improve the overall accuracy of reconstruction. The approach is thus interesting if common information is required for many different images, such as if a global proxy were reconstructed, or if bundle adjustment or self-calibration are to be applied subsequently. In that respect it also provides an advantage over non-sequential tracking and reconstruction as it requires only one, even streaming-like pass over the sequence. In addition, this one pass can be done in a fixed amount of time per frame using one of the selective schemes. In that case, the simple selection proved superior to the weighting scheme as it yielded lower absolute errors.

### Closing Loops in Camera Movement

In order to exemplify the results of the closing loops algorithm introduced in Section 3.7.2, the two turntable sequences *ColaTT-40* and *SantaTT-200* are chosen. While the first consists of only one revolution of a turntable with $40$ images, the second sequence comprises five revolutions.



(a) (b) (c) (d)

Figure 5.18: Steps of the closing algorithm for the *ColaTT-40* sequence: (a) linear, erroneous reconstruction, (b) loop closed without considering rotation, (c) loop closed considering rotation, and (d) final, optimized reconstruction with 3-D point positions corrected as well

As can be seen in Figure 5.18(a), the linear reconstruction of the *ColaTT-40* sequence yields a considerable gap between the first and the last image. The information that the sequence is circular is provided by adding the first image to the end of the sequence and then registering first and last image with each other. The process of closing the loop is demonstrated by the rest of Figure 5.18, with the intermediate steps of closing the loop considering only translation, minimizing the residual vector of equation (3.44), page 88, and with rotations included, using equation (3.45). For the final result, the 3-D point positions are recalculated.

For *SantaTT-200*, the viewpoint mesh algorithm of Section 3.4.2 is used to identify the end of a loop. Thus, loops were detected automatically every fifth image after reconstruction of the first revolution, which was selected as minimum length of a loop.

As mentioned in Section 3.7.2 the closing of loops makes the more sense the larger the accumulated error. This issue is reflected by the results shown in Table 5.10. PSNR, back-projection as well as pose errors are given for the reconstruction using only bundle adjustment on identified loops and for the whole process of closing loops, as opposed to the purely sequential camera calibration and 3-D reconstruction. This linear reconstruction of sequence *ColaTT-40* has a large accumulated error, therefore, closing loops has a great effect on the position difference while just applying bundle adjustment is insufficient to reduce this error. Closing the loop on the other hand has a large effect, especially on pose error.

For sequence *SantaTT-200* on the other hand, the accumulated error is rather small, and thus, the camera position difference is still lower without the closing step. The inaccuracies introduced by closing, reflected by the increased back-projection error in both sequences, were not compensated sufficiently by bundle adjustment.

| Sequence | Type | PSNR | back-proj. err. | trans. err. | rot. err. |
|---|---|---|---|---|---|
| *ColaTT-40* | **linear** | 27.4 dB | 1.51 | 33.5% | 31.6% |
| | **bundle** | 27.5 dB | 1.45 | 33.1% | 31.3% |
| | **closing** | 26.9 dB | 3.06 | 20.1% | 19.3% |
| *SantaTT-200* | **linear** | 18.9 dB | 1.31 | 71.3% | 65.8% |
| | **bundle** | 30.0 dB | 4.52 | 26.0% | 26.4% |
| | **closing** | 26.8 dB | 6.13 | 26.1% | 27.4% |

Table 5.10: Comparison of linear reconstruction, bundle adjustment, and closing of loops for *ColaTT-40* and *SantaTT-200* sequences.

**Summary:** Since the closing introduces some error on each camera position it works well for the compensation of large errors, but for small displacements, using only bundle adjustment may yield better results. Thus, the main issue for future work would be the reduction of errors introduced during the closing process.

### Extending Light Fields

In order to test the methods introduced in Section 3.7.4 for extending existing light fields with new input images, the experiments of [Deu04] are repeated here. The three image sequences recorded for this purpose are *OfficeHH-109*, *CandyHH-113* and *SantaHH-207*. For each of them, a second sequence was recorded starting at an arbitrary camera position within, or close to, the convex hull of the camera positions of the first sequence.

The principle setup of the following experiments is outlined in Figure 5.19. After feature tracking and linear reconstruction, a light field is generated for the first image sequence. While the parameter search by rendering feedback requires this first light field, it is not necessary for SIFT feature matching. Both approaches yield an image ID determining where to initialize feature tracking to the additional images, and either a homography according to equation (3.50), page 93, or a camera pose estimate for feature prediction. The pose estimate is obtained using either adaptive random search, or the particle filter, solving equations (3.53) through (3.55).

Both approaches, matching with SIFT features, and parameter search by rendering feedback, are tested with the same two experiments each. For the first one, only the first image sequence is used. One of the images is removed from the sequence and fed into the search algorithm to perform a search for the nearest image. This was done for ten images for each of the three scenes. The second set of experiments deals with attaching a second sequence to the original sequence. Again, ten images are used, this time from the beginning of the second sequence.

In case of the SIFT feature method, an image neighboring the missing image in the sequence

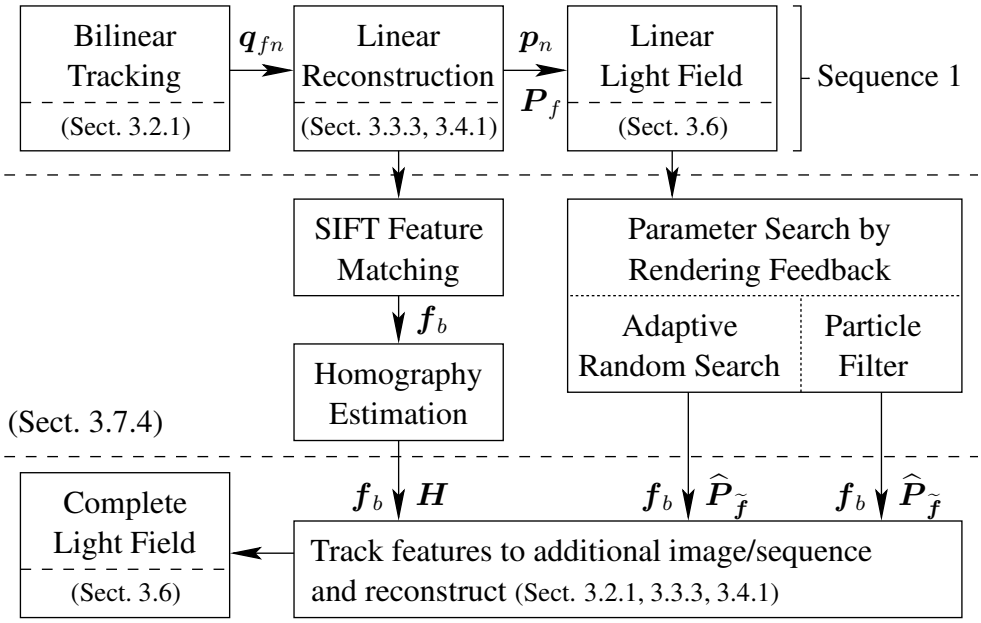



Figure 5.19: Processing steps for the three different approaches to adding new images or image sequences to an existing light field: SIFT feature matching and parameter search by rendering feedback using either adaptive random search (ARS) or the particle filter (PF)

| | *OfficeHH-109* | | *CandyHH-113* | | *SantaHH-207* | |
|---|---|---|---|---|---|---|
| **Type** | **ARS** | **PF** | **ARS** | **PF** | **ARS** | **PF** |
| **Cardan** $\alpha$ | 0.642° | 6.30° | 0.529° | 3.07° | 2.81° | 6.96° |
| **Cardan** $\beta$ | 0.654° | 4.52° | 0.963° | 2.71° | 2.32° | 5.68° |
| **Cardan** $\gamma$ | 0.561° | 2.22° | 0.653° | 1.87° | 2.92° | 9.12° |
| **translation** | 145% | 872% | 108% | 423% | 328% | 921% |

Table 5.11: Correctness of estimation of closest image for adaptive random search (ARS) and particle filter (PF)

is found for all images in each of the experiments. Since the method does not calculate any pose parameters for the missing camera position, no further statement of accuracy can be made.

For the use of rendering feedback on the other hand, the calculated camera poses can be compared to the pose of the camera computed originally during a reconstruction of the whole sequence. Table 5.11 shows the average rotational and translational error. In this case, the rotational error is given as Cardan angles in absolute degrees as in the original publication. The translational error for removed image $\boldsymbol{f}_i$ is calculated as $|\widehat{\boldsymbol{t}} - \boldsymbol{t}_i|/|\boldsymbol{t}_{i+1} - \boldsymbol{t}_i|$, where $\widehat{\boldsymbol{t}}$ is the translation found and $\boldsymbol{t}_i$ the previously reconstructed translation of image $\boldsymbol{f}_i$. Thus, the translational error is given as a percentage of the camera distance around the removed image or, in other words, the error is $100\%$ if, instead of the true position, that of a neighboring image is returned.

| Type | *OfficeHH-109* | *CandyHH-113* | *SantaHH-207* |
|---|---|---|---|
| **feature dist. ($I$)** | 49.0 | 58.6 | 41.6 |
| **feature dist. ($H$)** | 2.11 | 2.31 | 1.59 |
| **% tracked ($I$)** | 77.7% | 35.3% | 75.4% |
| **% tracked ($H$)** | 91.3% | 90.0% | 91.3% |

Table 5.12: Tracking accuracy of SIFT features in pixels and percentage of features tracked without ($I$) and with ($H$) using the homography matrix for prediction

| | *OfficeHH-109* | | *CandyHH-113* | | *SantaHH-207* | |
|---|---|---|---|---|---|---|
| **Type** | **ARS** | **PF** | **ARS** | **PF** | **ARS** | **PF** |
| **feature dist. (all)** | 30.5 | 44.6 | 31.3 | 37.5 | 28.0 | 24.8 |
| **feature dist. (init)** | 38.1 | 33.6 | 20.1 | 33.8 | 29.7 | 34.5 |
| **% tracked (all)** | 87.6% | 86.9% | 76.0% | 75.2% | 84.4% | 84.7% |
| **% tracked (init)** | 86.9% | 86.8% | 76.0% | 70.2% | 84.8% | 84.4% |

Table 5.13: Tracking accuracy of adaptive random search (ARS) and particle filter (PF) in pixels and percentage of features tracked for a full search (all) or initialized with the closest image (init)

As can be seen, the adaptive random search (ARS) method outperforms the particle filter (PF) approach for all sequences. The translational distances are generally within a few neighboring images, and the rotational ones within a few degrees. Though this error is larger than if the closest image as per the SIFT method had been used, the results are quite usable for a global search. Compared to the original publication in [Deu04], the results are much improved. This can be ascribed to distinct improvements in rendering quality due to a modified selection algorithm for contributing images, the use of local proxies instead of depth maps, and minor optimizations of the calibration process.

For the second experiment, the SIFT feature and rendering feedback methods can be compared directly. The extension was evaluated by measuring the average feature distance from the predicted position, and the fraction of successfully tracked features. In case of the SIFT feature approach, a distinction is made between using the estimated homography matrix between the image to be added and the closest one in the first sequence, and not using it. From Table 5.12, it is obvious that the homography matrix maps the features very accurately from one image to the other as the average feature distance is below five pixels in all cases. Consequently, feature tracking is very successful in this case. Without this mapping, the tracking algorithm has to match features which are very far apart, and obviously, the basin of convergence is not large enough for distances beyond $50$ pixels, as documented by the low rate of tracked features for sequence *CandyHH-113*.

(a) (b) (c)

Figure 5.20: Extended reconstruction of the *SantaHH-207* image sequence (a) and rendered images without (b) and with (c) the additional input images as seen from exactly the same camera position

For evaluation of the rendering feedback methods, initialization of the search was tested both from camera parameters of all images in the first sequence, and from the closest camera as per the SIFT neighbor search. The resulting proposed camera parameters were then passed to the feature tracking process by using the back-projections of every known 3-D point as an estimate for the feature positions similar to the homography $\boldsymbol{H}$.

The results are listed in Table 5.13. Here, an advantage of adaptive random search over the particle filter cannot be observed as in the first experiment or in [Deu04]. The reason may be that the higher quality of rendered images promotes the speed of convergence of the particle filter approach. While the accuracy of feature prediction is lower than in the SIFT feature case, it is still high enough to allow for a percentage of tracked features above $70\%$ for all methods and situations. However, it has to be mentioned that extension failed in two out of ten cases for adaptive random search with initialization applied to sequence *SantaHH-207*.

Regarding runtime, the SIFT feature approach outperforms rendering feedback, even though the efficient unstructured lumigraph rendering is used together with local proxies. In case of sequence *SantaHH-207*, e. g., SIFT features require a total of $96.7$ seconds for finding the closest neighboring image, of which homography calculation makes up only $1$ second. In contrast to that, adaptive random search takes about $6.5$ minutes for the same task without initialization. The condensation algorithm requires even $8.9$ minutes in average. In both cases, $20$ iterations with $207$ particles were used.

Figure 5.20 finally shows the result of adding the second sequence to *SantaHH-207*. Image (a) depicts the reconstructed camera positions of both sequences with the positions of the additional sequence highlighted. Images (b) and (c) show two images rendered from the resulting

light field without (b) and with (c) the additional images, demonstrating the increased viewing range of the extended light field.

**Summary:** The experiments show that either method is able to reliably solve the problem of extending a light field. The SIFT feature approach, however, offers a much higher accuracy in predicting feature positions, while the rendering feedback is sufficient and fails only in very few cases. Additionally, SIFT feature calculation is more efficient here unless the number of particles is reduced unreasonably.

**Recommendations:** The best approach to generating a static light field from an image sequence always depends on various factors, such as the type of recording, the time available, and the purpose of the light field. The most general scenario is probably the availability of an image sequence recorded with a hand-held camera, and no real-time constraints, although the result should be available within several minutes. In order to achieve a good result soon with little or no parameter adjustments, the recommended procedure is a linear reconstruction process according to Figure 5.9 including bilinear tracking. Linear reconstruction itself should be substituted by automatic selection of a factorization method as shown in Figure 3.4, page 65, where the subsequent extension to the whole sequence should make use of the robust Lorentzian estimator of equation (3.34), page 74, as shown in Figure 5.12. The fastest generation and best rendering results for scene geometry are achieved with local proxies according to Section 3.6.2 on page 78.

Non-sequential tracking and reconstruction following Figure 5.9 constitutes an optional, successive processing step. It should only be applied if linear reconstruction was successful so that the camera mesh, which is generated using equations (3.26) and (3.27), page 70, contains sensible interconnections of neighbouring viewpoints. This step is recommended, however, if a global proxy is to be generated from the reconstruction. Finally, if the existing light field is to be extended by new images, the approach using SIFT feature matching and homography estimation, shown in Figure 5.19, should be preferred for the reasons described in the summary above.

## 5.3 Evaluation of Dynamic Light Fields

Two different modeling techniques for dynamic light fields have been introduced in Chapter 4, step-wise static light fields and the modeling of rigid object motion. These two models are evaluated in the following. As in the previous section, the different example sequences are introduced first. As an outlook to future work, an example of an object light field is presented.

Figure 5.21: Example images from the test sequences for step-wise static light fields: *HandSWS-468* (top row), *HeadSWS-482* (middle row), and *RotorSWS-576* (bottom row)

### 5.3.1 Example Sequences

In case of dynamic light fields, the example sequences are subdivided into just two categories, one for each type of modeling. The categories given in the sequence identifiers are SWS for step-wise static, and ROM for rigid object motion. The sequence length for step-wise static sequences denotes the sum of images in all sequences, as one sequence was recorded for each time step. All sequences were recorded using the hand-held Sony DV camera already used previously.

For step-wise static light fields, three sets of image sequences were recorded. Each of them shows a different dynamic object in front of a static background. Example images for each of these scenes are shown in Figure 5.21. The numbers of image sequences (time steps) for each scene are $5$, $6$, and $8$ for *HandSWS-468*, *HeadSWS-482*, and *RotorSWS-576*, respectively. The number of images per sequence varies between $64$ and $108$.

Another three image sequences were chosen for the experiments on light fields of moving

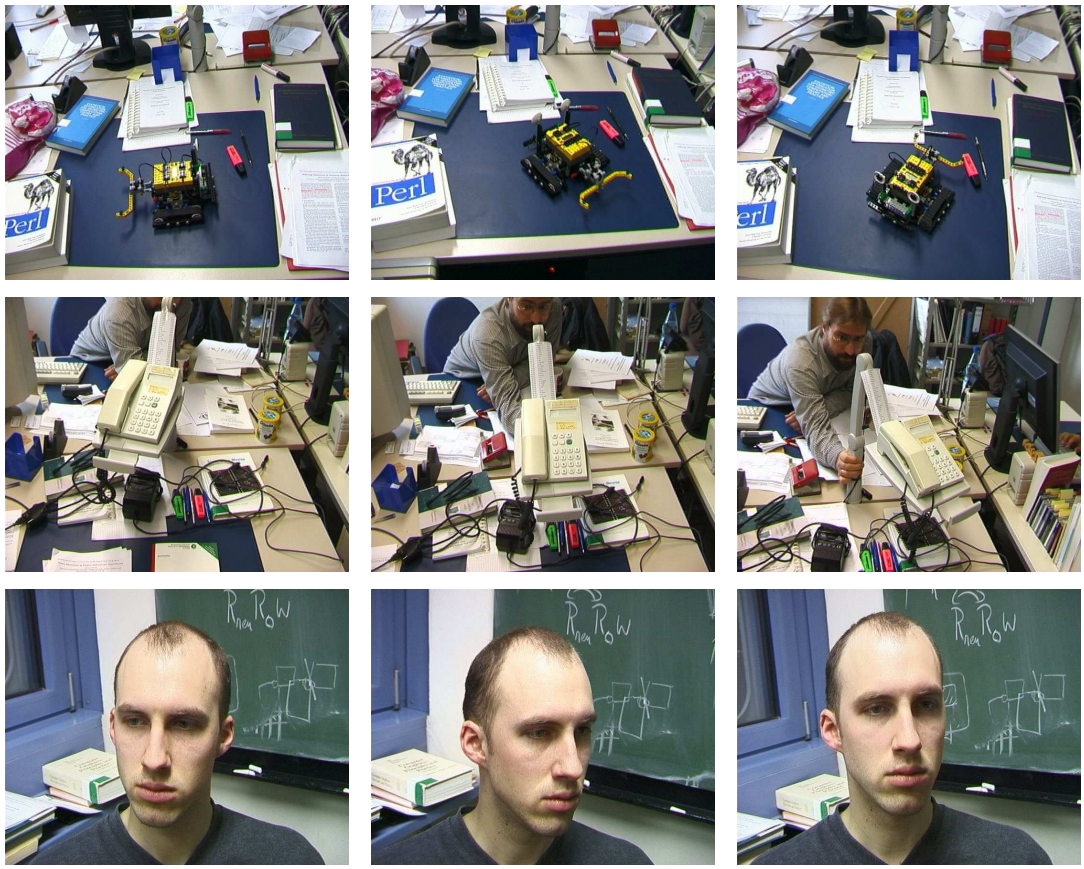


Figure 5.22: Example images from the test sequences for light fields of rigid, moving objects: *CrawlerROM-139* (top row), *PhoneROM-145* (middle row), and *HeadROM-200* (bottom row)

but rigid objects. Two example images of each sequence are shown in Figure 5.22. Besides the rotating toy crawler already introduced in Section 4.3, the sequences show a telephone arm being rotated repeatedly, and a person turning his head several times. The sequence identifiers are *CrawlerROM-139*, *PhoneROM-145*, and *HeadROM-200*.

Finally, the example of an object light field which is given in Section 5.3.4 as a preview is generated from sequence *PhoneROM-145* without any modifications to it.

### 5.3.2 Step-Wise Static Light Fields

The experimental evaluation of step-wise static light fields is done following the same procedure as in [Sch04b]. Although the same image sequences were examined there, results may differ considerably. The reasons are that instead of the feature tracking scheme of Heigl [Hei98], the more accurate and efficient implementation by Zinßer [Zin04] is used. Furthermore, the

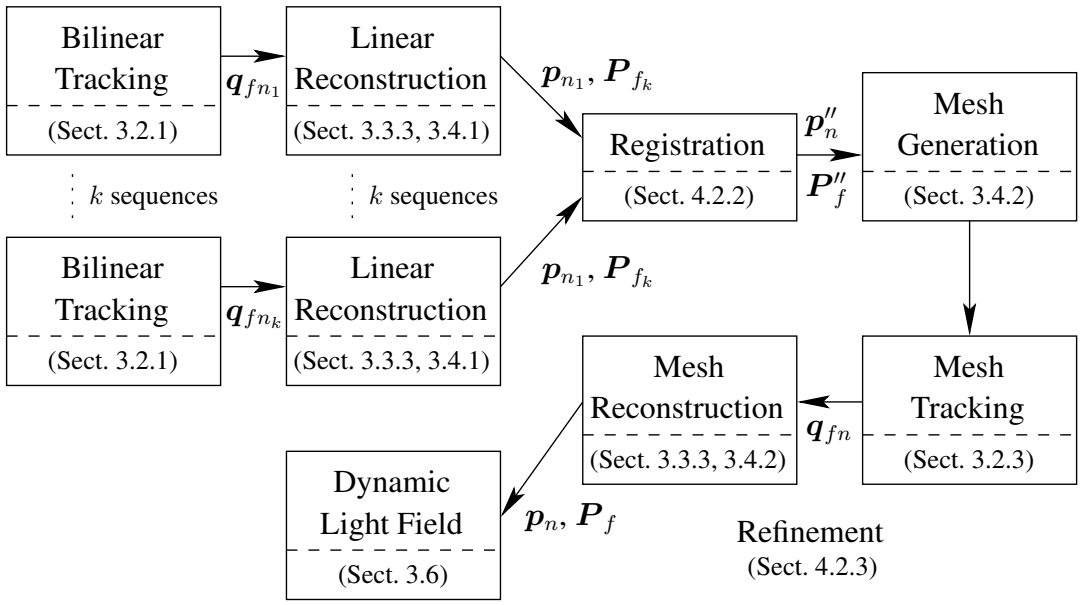



Figure 5.23: Processing steps for generating a step-wise static light field from $k$ input image sequences. After independent reconstruction of each sequence, the dynamic light field is assembled after registration and refinement.

unstructured lumigraph renderer [Bue01], which is used throughout this thesis, is now employed instead of the free form renderer introduced in [Sch01b].

The experimental setup is outlined in Figure 5.23. The processing steps which are of special interest here are registration, which combines the independent reconstructions of the input image sequences, and refinement, which consists of mesh tracking and reconstruction for all sequences together. Registration is explained in Section 4.2.2 and applies the transformations given in equations (4.1) to (4.4) on pages 100 to 102. Refinement is described in detail in Section 4.2.3.

An assessment of the quality of registration using the back-projection error is shown in Figure 5.24 for the *HandSWS-468* sequence. In order to ensure the comparability between registration by concatenation and the following refinement, the same set of point correspondences was used for both. These consist exclusively of features which were observed in more than one of the image sequences. Unlike in the normal case, the corresponding 3-D points were not taken from the calibration process, but were triangulated from the 2-D features. This way, features which are usually discarded because of a back-projection threshold are included as well.

The comparison in Figure 5.24 shows clearly that the back-projection error is considerably smaller in each image of all sequences after the refinement than before. The reduction of the back-projection error of the two other examples is given in Table 5.14, and is likewise considerable.

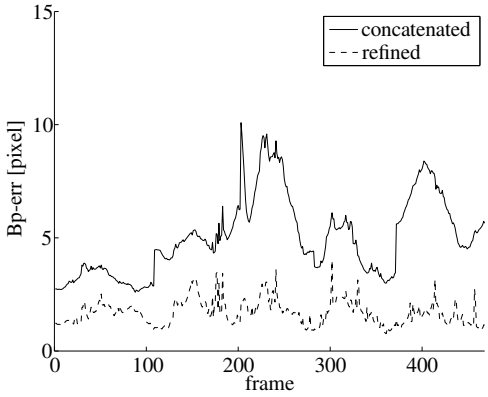



Figure 5.24: Back-projection error of point correspondences with features in images of at least two sequences. The error in each frame of sequence *HandSWS-468* is given in pixels, before (solid line) and after (dashed line) the refinement step.

| **Sequence** | **back-proj. err.** | | **img. diffs** | **mean diff.** | |
|---|---|---|---|---|---|
| | **concat** | **refine** | | **concat** | **refine** |
| *HandSWS-468* | 5.17 | 0.629 | 45.0 | 35.9 | 18.5 |
| *HeadSWS-482* | 5.02 | 1.69 | 30.0 | 21.3 | 14.1 |
| *RotorSWS-576* | 25.0 | 0.824 | 84.0 | 49.4 | 19.5 |

Table 5.14: Comparison of back-projection error in pixels and background shifts using average color value difference per pixel for the example scenes before and after refinement. In column "img. diffs", the number of image comparisons for average color value difference is given.

The second measure for the quality of image sequence registration is the shift of the background for two rendered images of different time steps, but as seen with exactly the same virtual camera pose. Putting this shift into numbers is difficult, so the average absolute color value difference per pixel was chosen as a measure. While this lacks some quantitative expressiveness, it can still give a good qualitative impression. In order to ensure the validity of this comparison, the test images were always rendered with approximately the same visible object sizes. An exact match was not possible since the coordinate systems differ before and after refinement. The values in column "img. diffs" of Table 5.14 denote the number of image comparisons performed.

Since only the background shift was to be considered, the dynamic objects in the foreground were removed by hand-coloring them in black. These and any other colorless parts of the images—they appear if no reference images for interpolation are close enough to this area—were ignored in the difference. The last two columns of Table 5.14 show the pixel differences for the simple image sequence concatenation compared to the value after the refinement step. In all scenes the refinement step clearly improves the registration. This can be seen in the example in

Figure 5.25: Difference images for time steps 0 and 3 of the *HeadSWS-482* sequence from similar camera positions before and after refinement

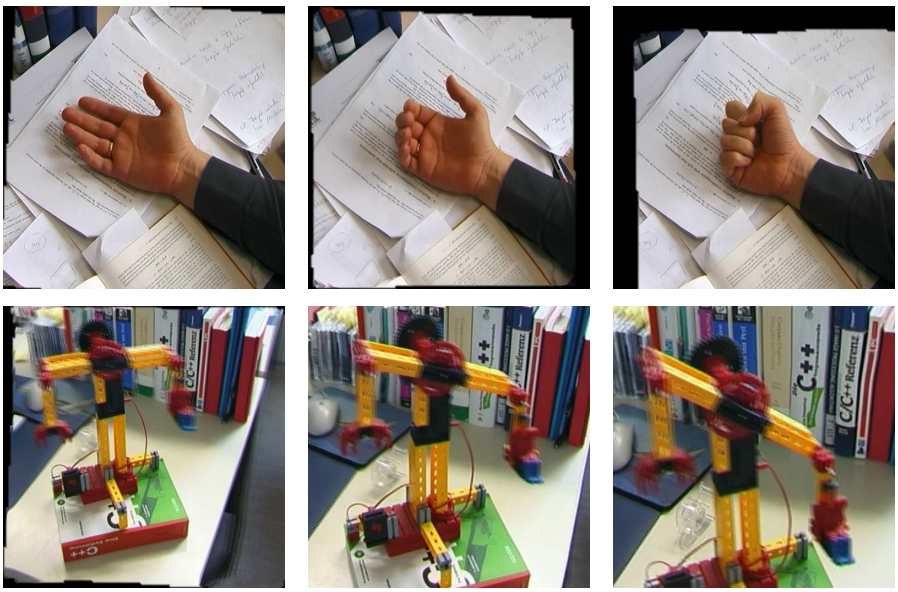


Figure 5.26: Example images rendered from the light fields of *HandSWS-468* and *RotorSWS-576* showing different time steps. The images of the first row were rendered using a constant camera pose, while for the second row the camera was moved towards the scene.

Figure 5.25 where the difference images of two time steps are plotted, before (left) and after the refinement (right) for sequence *HeadSWS-482*. The hand-colored regions appear as uniformly black rectangles.

Finally, in Figure 5.26 a series of example images is rendered from the light fields of sequences *HandSWS-468* and *RotorSWS-576* respectively, which were both generated after the refinement step. For the three different time steps rendered the observing camera's pose was kept constant in the first case, to demonstrate the stability of background. In the second case, the camera is moved towards the object while advancing in time as well.

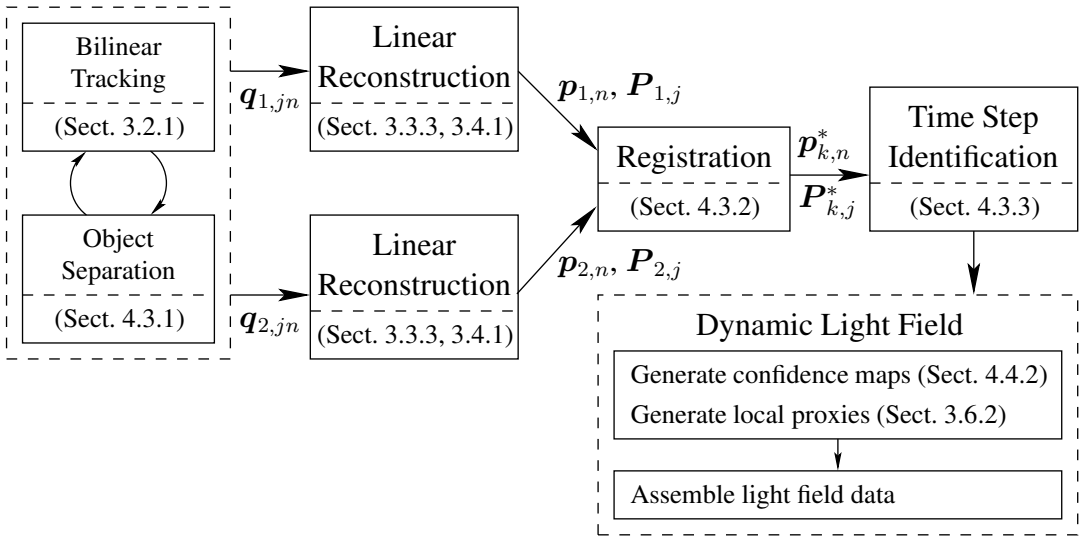



Figure 5.27: Processing steps for generating a dynamic light field with rigid, moving objects. In this case, the process is shown for background and one object only.

### 5.3.3 Rigid Object Motion

The block diagram of Figure 5.27 shows how the processing steps of object separation, registration and time step identification are integrated into the usual process of light field generation. They were introduced in Sections 4.3.1 to 4.3.3 and denote the main techniques employed to reconstruct this kind of dynamic light field. Object separation makes use of the shape interaction matrix $\boldsymbol{Q}$ given in equation (4.5), page 105, and the separation criterion of equation (4.10), page 107. Registration follows equations (4.11) to (4.14) on page 111. Tracking, 3-D reconstruction and light field assembly are reused from static light fields, although the last is extended by confidence map generation which is required for visualization of the dynamic light field.

One aspect of reconstructing light fields of moving but rigid objects which was emphasized in Section 4.3.1 is the problem of finding enough features on both background and foreground of the scene, which may be difficult if the object is either too small or moving too quickly. The three example sequences used in the following were selected keeping this issue in mind. However, as can be seen in Table 5.15, the total number of features assigned to the object during reconstruction may be considerably smaller than for the background, or otherwise, the average trail length may be very short, as in case of sequence *CrawlerROM-139*. The initial factorization was therefore done on as few as $10$ (*CrawlerROM-139*) to $35$ (*HeadROM-200*) images. For the remaining extension to the whole sequence, each feature had to be visible in at least $8$ images to be included.

| Sequence | images | features bg. | features fg. | avg. trail length |
|---|---|---|---|---|
| *CrawlerROM-139* | 139 | 2181 | 1049 | 15.5 |
| *PhoneROM-145* | 145 | 2207 | 266 | 38.0 |
| *HeadROM-200* | 200 | 1160 | 710 | 90.9 |

Table 5.15: Some statistics on the example sequences: the columns denote the total number of images, the total number of features on background and foreground, and the average length of feature trails on the foreground object.

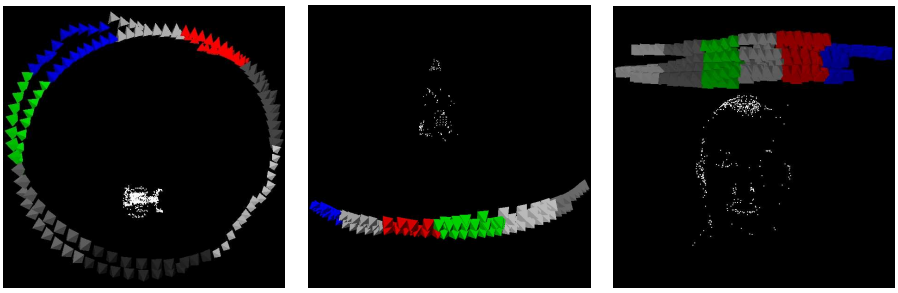


Figure 5.28: Reconstruction and quantization of the three objects into 8 time steps for the first, and 6 time steps for the other two sequences. From left to right: *CrawlerROM-139*, *PhoneROM-145*, and *HeadROM-200*.

The final results of the object reconstructions for each sequence are depicted in Figure 5.28. Here, the final camera path is visible which results from deducting the camera's own motion. The quantization by camera position, as described in Section 4.3.3, is illustrated by different shades for each "time" step.

Figure 5.29 shows four images rendered from the resulting light fields for each of the three sequences, using the rendering method introduced in Section 4.4.2. For all four images of each sequence, the camera pose was the same, which is reflected by the identical background in each image. The object, however, was rendered for four different time steps, and is thus at different positions. Note that the camera poses for the rendered images were not part of the original sequence, but chosen arbitrarily. For the *crawler* light field the image sequence was subdivided into eight time steps, while the other two light fields consist of six time steps each.

The most obvious errors in rendering are apparent for sequence *HeadROM-200*, where artifacts are visible around the head to varying degrees in all images. This is the result of a less than optimal partitioning of the input images into foreground and background, due to background features lying too close to the object. Thus, if parts of the object are not masked out by a confidence map, they are rendered as background and lead to the observed effects. For the other two sequences, the generous partitioning is sufficient to prevent these errors.

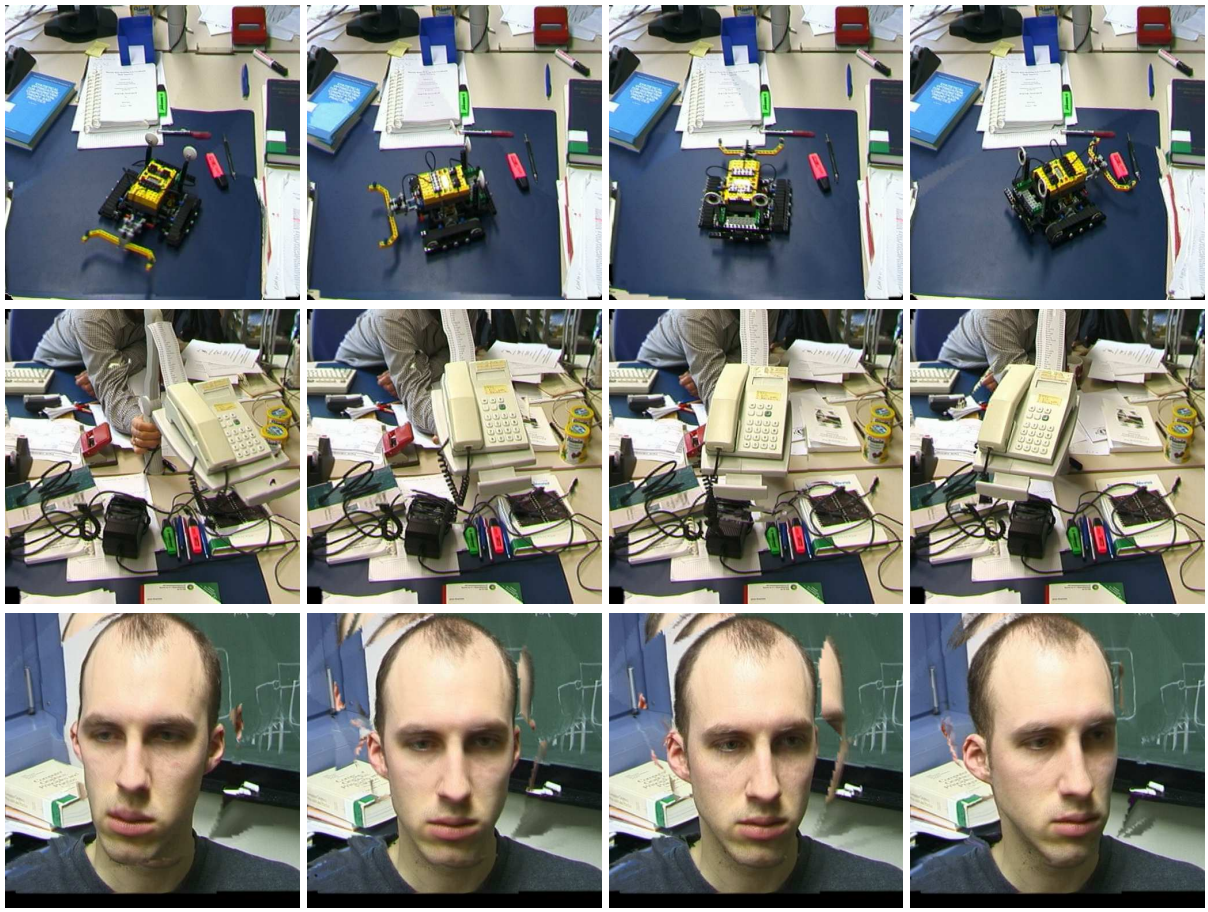


Figure 5.29: Rendered images for four different time steps of the rigid object light fields created from the *CrawlerROM-139* (top row), *PhoneROM-145* (middle row), and *HeadROM-200* sequence (bottom row), seen from the same camera position

### 5.3.4 Object Light Fields

As a further development of the dynamic light fields of rigid objects, the object light field was introduced in Section 4.4.3. It makes use of the same segmentation algorithm, but instead of partitioning the images into time steps, an individual light field is generated for background and object each.

For the preliminary example of an object light field shown here, sequence *PhoneROM-145* is reused. From its previous reconstruction shown in Figure 5.28, the two individual light fields for background and foreground object are generated. For the background light field, the data already available can be reused without further processing, except that those confidence maps are discarded which allow for the foreground object to be rendered. Thus, the telephone arm is deleted from the scene and not visible anymore during rendering, as is demonstrated in Figure 5.30(a).

(a) (b) (c) (d)

Figure 5.30: Example rendering of an object light field: background (a) and object (b) of the *PhoneROM-145* sequence are rendered individually using hand-segmented confidence maps (c), and together in one renderer (d).

For the foreground light field, the object is cut out too generously in the existing confidence maps so that they have to be refined by hand. For the light field rendered for Figure 5.30(b), confidence maps such as the one of Figure 5.30(c) are used.

The combination of background and foreground light fields to the final object light field is shown in Figure 5.30(d). Here, the rendering software was extended to allow the independent manipulation of either background or foreground. Although there are noticeable artifacts around the object which result from inaccuracies in object segmentation, this finally permits a continuous representation of a dynamic light field without discrete time steps.

# Chapter 6

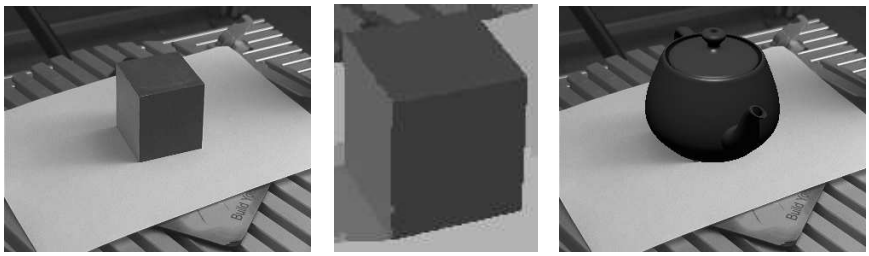
# Applications of Light Fields

In principle, light fields constitute another type of geometric model, with the special property that they are easily computed from real images. Applying them as models for real-world applications suggests itself, and examples were already shown in [Hei04]. There, light fields were applied to two tasks in image analysis, viewpoint selection and self-localization. The following chapter will briefly introduce a number of additional applications in augmented reality, object recognition, and minimally invasive surgery.

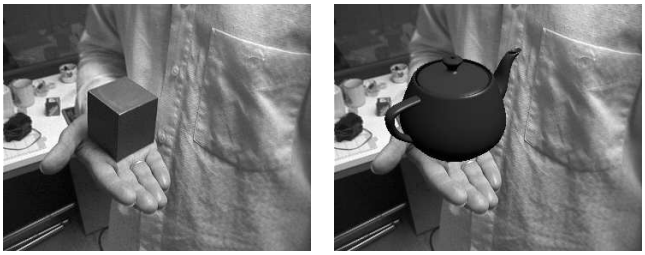
## 6.1 Augmented Reality

Augmented reality is a relatively young field of research with the aim of combining real environments with computer-generated, virtual objects. The impression the user should get is that of a seamless integration of both reality and virtuality. In its simplest form, augmented reality is used to enhance photographs with, e. g., models of new buildings or archaeological reconstructions. In its most sophisticated form, the user wears a head-mounted display in which his environment and virtual objects are combined on-the-fly in real-time. His surroundings may be recorded by cameras that are mounted on his head as well, his own head pose has to be reconstructed either from these camera images or from additional sensors, such as optical tracking by external cameras. A comprehensive survey on the state of the art in augmented reality has been published in [Bim06].

An example for a system which was designed for application in head-mounted displays has been introduced in [Sch01d] and [Sch01e]. It makes use of a metal cube with a side length of $6$ cm that is painted with a different color on each side, such that its position and orientation can be determined unambiguously. A real scene can thus be augmented by rendering a virtual object



(a) (b) (c)



(d) (e)

Figure 6.1: Two examples of augmentation: Original (a), CSC segmented cut-out (b), and augmented image (c) on a turntable; original (d) and augmented image (e) of cube held in hand.

in the same pose, thus replacing the cube. Head tracking is not necessary using this approach.

The cube is detected by exploiting the known colors of its sides. Applying a resolution hierarchy, the image is color segmented using the Color Structure Code (CSC) algorithm [Reh98]. Candidate regions are identified in each resolution regarding their color and verified in the next resolution level. This process is exemplified in Figures 6.1(a) and 6.1(b). The corners of the cube are identified next, and used to estimate the camera pose for each image, assuming perspective projection and making use of the known shape of the cube. The augmentation of the cube with a virtual object using the computed pose is shown in Figures 6.1(c) and 6.1(e).

Using a light field instead of the synthetic, virtual object above is a small step which has been implemented for a slightly different scenario in [Mül02]. Here, the "calibration object", which was the cube in the system before, is the face of a person, and the virtual object is replaced by the light field of a different person's head. Thus, the goal here is to exchange a person's face in a movie with that of a different person.

Two main tasks have to be solved here. Firstly, the respective faces in both the original image sequence and the images used for reconstructing the light field have to be identified. This is done by detecting the faces based on skin color [Cha98], and only features inside these image

(a) (b) (c)

Figure 6.2: An image of a person's face from the original sequence (a) and the corresponding view from the light field of a different person (b). Augmentation is done by matching the eye positions and warping the light field image (c). Images taken from [Mül02].

regions are used later on for calibration and reconstruction. Secondly, a face with the same pose as the one in the current, original image has to be generated from the light field. This is achieved by calibrating the original image sequence in the same way as the images for the light field had to be, using, in both cases, the methods described in Chapter 3. By registering these two reconstructions, the respective image is easily rendered by using the camera parameters of the original image sequence. Registration, however, is done here mostly manually.

In Figure 6.2, the process is demonstrated for one example. For the view from the original image sequence of Figure 6.2(a), the corresponding view is rendered from the light field of a second person, as shown in Figure 6.2(b). In order to achieve the correct mapping of light field image over the original face in Figure 6.2(c), it is necessary to warp the light field image slightly, using the Beier-Neely warping algorithm [Bei92]. As a corresponding pair of lines, the connecting line between the eyes is detected in each image using their difference in color from the rest of the face.

This application constitutes a special kind of augmented reality, as it combines a real environment with "real" images, stored in a light field model. While the example above leaves room for improvement, it shows that if seamless augmentation were realized, using a light field as virtual object model may enhance realism with little effort.

## 6.2 Object Localization and Recognition

A number of applications of light fields in object recognition or robot navigation have been introduced before by Heigl in [Hei04]. With the advances in both light field reconstruction and rendering as well as object recognition techniques, different approaches have become feasible. Two of them, object tracking using particle filters and fast training for object recognition, are described in the following.

### 6.2.1 Visual Object Tracking

Object tracking is the task of sequentially estimating the internal, unknown state of a moving object from a sequence of observations. Model based object tracking exploits the fact that information about the object is available in form of a model, from which it is possible to predict what observations should be expected, if a certain state of the object is assumed. This predicted observation is compared to the actual observation and the estimated state may be updated accordingly. In 3-D object tracking, the state of the object is a vector that usually contains the 3-D position of the object, its velocity and acceleration. Additionally, sometimes the pose of the object is also of interest and is therefore included in the state vector. The use of a light field as object model for 3-D object tracking with pose estimation was examined in [Fri02] and [Zob02] and will be summarized here.

In order to obtain a suitable object model which provides information about an object from all viewing directions, the recording setup of a camera mounted above a turntable as described in Section 5.2.1 is used here. Unlike the light fields described so far in this thesis, the light fields used here are not reconstructed using structure-from-motion algorithms. Instead, the camera poses are derived from the positions of turntable and robot arm. This way, it is guaranteed that a light field can be reconstructed for any object. On the other hand, no depth maps are as readily available from triangulated feature points. Therefore, only flat depth maps are used here.

#### Particle Filter Approach

The visual 3-D object tracking algorithm makes use of the particle filter (PF) technique, which was already described for the extension of existing light fields in Section 3.7.4. It uses a set of hypotheses $\mathcal{R}_t$ for the camera parameters, where each of the hypotheses $\boldsymbol{\rho}_{t,i}, i = 1, \ldots, |\mathcal{R}_t|$, represents the probability density function (PDF) $p(\boldsymbol{\rho}_t|\widetilde{\boldsymbol{f}})$ for the camera pose $\boldsymbol{\rho}_t$, i. e., the view of the object model, given the observed object $\widetilde{\boldsymbol{f}}$. Like before, the particle filter requires an estimate

for both the a priori density $p(\boldsymbol{\rho}_t)$ and the likelihood $p(\widetilde{\boldsymbol{f}}|\boldsymbol{\rho}_t)$ as stated by equation (3.53) which decomposes the PDF using Bayes' formula.

Here, both a priori density and likelihood are estimated similar to Section 3.7.4. For the latter, a Gibbs distribution is used as approximation. In this case, the light field object model predicts an image of the object assuming a certain state. This image has to be compared to the actual observation. Therefore, the energy term $E(\widetilde{\boldsymbol{f}}|\boldsymbol{\rho}_t)$ of the Gibbs distribution given in equation (3.55) has to measure the similarity between the predicted observation and a certain target region in the actually observed image.

The energy $E(\widetilde{\boldsymbol{f}}|\boldsymbol{\rho}_t)$ is defined by means of the *normalized correlation coefficient* (NCC). NCC is a common similarity metric between images used for template matching. Because the likelihood definition of equation (3.55) requires a high positive energy for bad matches, the reciprocal of the absolute NCC is used. Additionally, the absolute NCC is increased to the power of $\breve{\alpha}$ with $\breve{\alpha} > 1$. $\breve{\alpha}$ is called a *strictness* parameter because the higher this parameter is chosen, the better the match has to be in order to finally achieve a high likelihood.

### Experiments

In [Zob02], three different objects were used for experiments: beside the Santa Claus figure already seen in Chapter 5 these are a stuffed elk and a tape cassette. For each of these objects a light field object model is constructed from $817$ camera positions on a hemisphere with the turntable-arm assembly described before.

In order to evaluate the accuracy of the proposed tracking technique, ground truth data is obtained by recording the objects again from camera positions on the hemisphere that are different from those used for the generation of the light field object models. In Figure 6.3(a), the $687$ camera positions for the *elk* sequence are plotted. Object motion was then simulated by defining different paths through those camera positions. The three different paths of length 72 for the *elk* sequence are charted in Figure 6.3(a) as well.

For tracking the objects on their paths (i.e., actually, the robot arm and turntable positions are estimated) a particle filter using $200$ samples for representing the posterior density is employed. At each time step the estimated state is taken to be the sample mean of the posterior, hence a minimum mean square estimator is realized. In this case, the state is restricted to contain only the 3-D pose parameters, namely, the two angles $\theta$ and $\phi$, their velocity, and their acceleration.

For each of the three objects two tracking experiments were conducted, setting the strictness parameter $\breve{\alpha}$ to either $\breve{\alpha} = 1$ or $\breve{\alpha} = 5$. For a qualitative impression of the results, the tracking of path $1$ for the *elk* object with $\breve{\alpha} = 5$ is compared to the original path in Figure 6.3(b). The

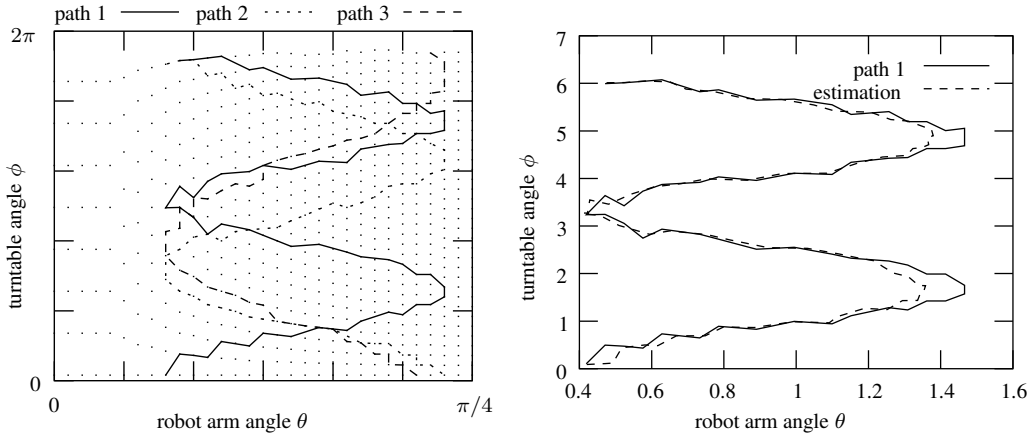



Figure 6.3: (a) Three different motion paths through the camera positions on the hemisphere (dots). (b) Original path and estimated object motion path for path 1.

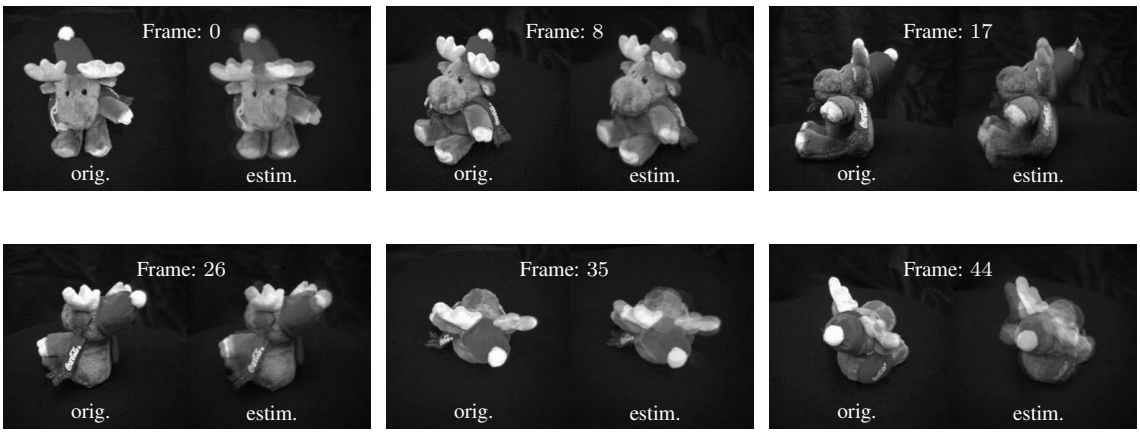



Figure 6.4: Comparison between the original image (left) of the test sequence for object *elk* on path 1 and $\breve{\alpha} = 5$, and the rendered image (right) from the light field according to the estimated state.

remaining experiments yielded similar results, although the system performed better in all cases for $\breve{\alpha} = 5$. For the exact figures of all experiments refer to [Zob02].

The actual similarity of the rendered images to the original ones is demonstrated in Figure 6.4 for the same experiment as above. For time steps at regular intervals, the rendered image corresponding to the estimated state is placed beside the respective original image to allow for visual comparison. It can be seen that the object's pose is estimated quite accurately despite the low rendering quality which results from the lack of accurate depth information.

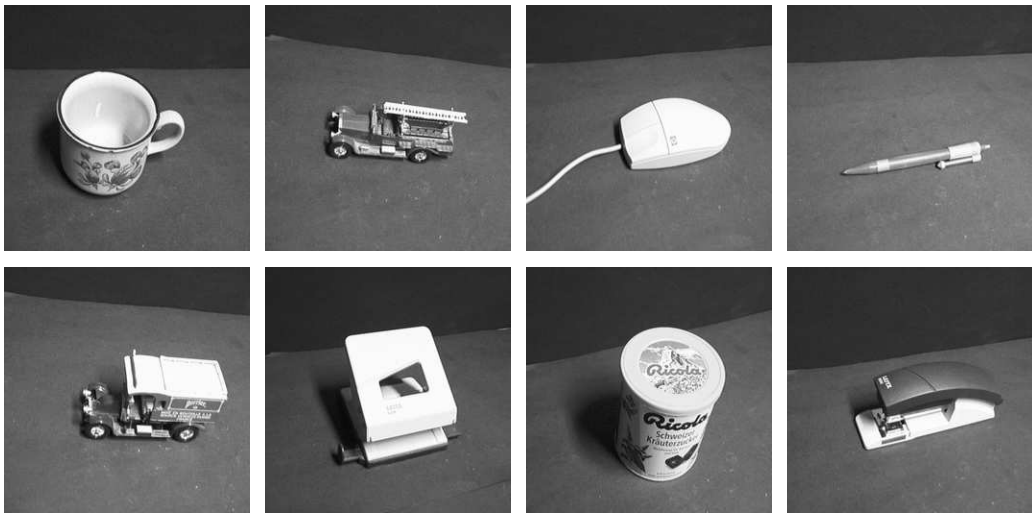



Figure 6.5: Object classes used for recognition: (first row) cup, toy fire engine, mouse, pen, (second row) toy passenger car, hole puncher, candy box, stapler. Each image shown is part of the respective training sequence.

For these experiments scaling was not considered. The distance of the cameras on the hemisphere to the center of gravity of the objects is assumed to be fixed. In [Fri02], the scenario is extended and the tracker is applied to images taken by the camera of a mobile robot platform (MOBSY). The state vector is extended to include the 3-D position information instead of just the orientation angles, but velocity and acceleration have to be omitted to reduce the dimension of the state space. Tracking, however, is performed successfully as well for this application.

### 6.2.2 Fast Training for Object Recognition

In [Grz04], [Grz05], and [Grz07b], a different aspect of light field reconstruction was applied to object recognition. Instead of the rendered images as in the application before, the reconstruction process for image sequences from hand-held cameras is now utilized to quickly generate new object models. It thus replaces the procedure commonly used in appearance-based object recognition of recording images using a turn-table setup such as the one described previously.

Object model generation is done by placing each object on a black, homogeneous surface and recording an image sequence which covers a wide variety of viewing directions. The objects used in [Grz04] are shown in Figure 6.5. After camera calibration and 3-D reconstruction using bilinear tracking and reconstruction as described in Chapter 3, the camera poses are available in the well-known form derived from equation (2.11). At this point, a parameter transformation is

required since the object recognition system uses a different parameter representation. It comprises two translations and one rotation in the image plane, two rotations outside the image plane and one translation along the optical axis. Only four of these parameters apply since the objects are assumed to be upright and at a fixed distance to the camera.

At the beginning of statistical modeling, one image of each class is selected as a reference image, and the pose parameters for the remaining images are calculated relative to the pose of the object in this image. For all images, features are computed using a wavelet transformation, where the wavelet coefficients are calculated on a multi-scale grid on each image. Features on the background are discarded. The computed feature vectors are interpreted as random variables and their components modeled as normally distributed, thus comprising the statistical object model. For localization and classification, the same feature vectors are calculated for the test image. Object pose and class are determined using maximum likelihood estimation. For detailed information on this procedure refer to [Grz07a].

In the experiments conducted for [Grz04], [Grz05], and [Grz07b], a training sequence was recorded and 3-D reconstructed for each of the eight test objects with more than $200$ images each. As test images, a second sequence was recorded for each object with more than $120$ images and likewise reconstructed. Depending on the number of training images used, the best average classification rate achieved was $98.8\%$. The best localization rate amounted to $45.5\%$, assuming a maximum deviation of $10$ pixels for translation and $15$ degrees for rotation, respectively.

The two most prominent benefits, however, are the time required for acquiring the object models, and the lack of need for specialized hardware. Recording $200$ images with a hand-held camera takes about $13$ seconds (at a rate of $15$ frames per second), feature tracking and 3-D reconstruction require about three minutes. Together with preprocessing steps, generating one object model from the images can be done in about $35$ seconds. Thus, from recording to the final model takes less than four minutes. Compared to this, the recording of the same number of images on a turntable and robot arm setup requires about $20$ minutes, and additional time for model generation. The proposed approach is thus well suited for the ad-hoc setup of a recognition system, even if no specialized hardware is available.

## 6.3 Medical Applications

Light fields have found their way into a number of different medical disciplines. One of these areas is endoscopic surgery, where some of the methods introduced in the previous chapters have been applied to address a number of problems. These issues, highlight substitution, progress

Figure 6.6: Highlight substitution on light field images of a gall bladder in laparoscopic surgery: original view (left) and substitution (right). An improvement is mainly visible in the image center and upper right corner. Images taken from [Vog06] using improved rendering techniques.

monitoring of operations, and reconstruction for flexible endoscopes, will be introduced in the following.

### 6.3.1 Highlight Substitution

In endoscopic surgery, the common approach is to enter an endoscope into a cavity of the human body, such as the nose, or the abdomen in laparoscopy, where the operation is performed. The major problem the surgeon faces here are a limited field and quality of vision, and a limited access to the operation area, since his tools have to be entered through the same or nearby openings.

A first issue that was approached using light fields is that of highlights frequently occurring on internal organs in laparoscopy. Since the light source is usually placed right beside the optical device, the light is often reflected by wet surfaces perpendicular to the camera lens, thus leading to saturation of the optical sensor. In [Vog02b] and [Vog02c], the property of light fields of using images of several viewing directions to reconstruct a novel view is exploited, similar to the methods for rendering dynamic light fields introduced in Section 4.4. In the original images, highlight regions are detected using, e. g., thresholds on the values of saturation and value in HSV color space. These regions are marked in confidence maps for each image, just like moving objects for certain time steps in dynamic light field rendering. As a result, the surface regions in question are rendered from images close-by, which were recorded from different viewing angles. In Figure 6.6, an example for highlight substitution in this way on a gall bladder is shown.

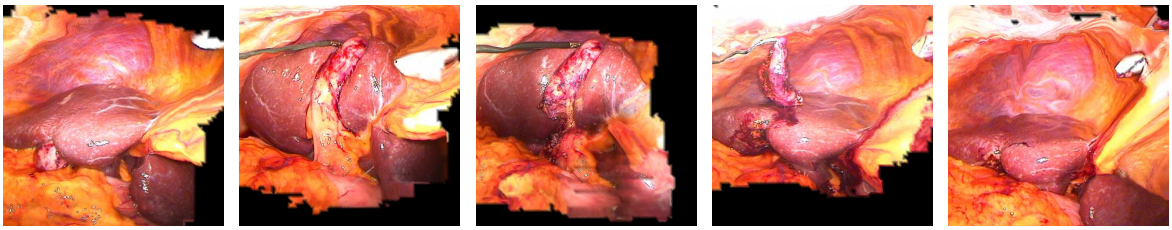


Figure 6.7: Dynamic light field reconstruction in endoscopic surgery: five image sequences were acquired during a gall bladder removal (cholecystectomy), showing the progress of the operation. Images taken from [Vog06].

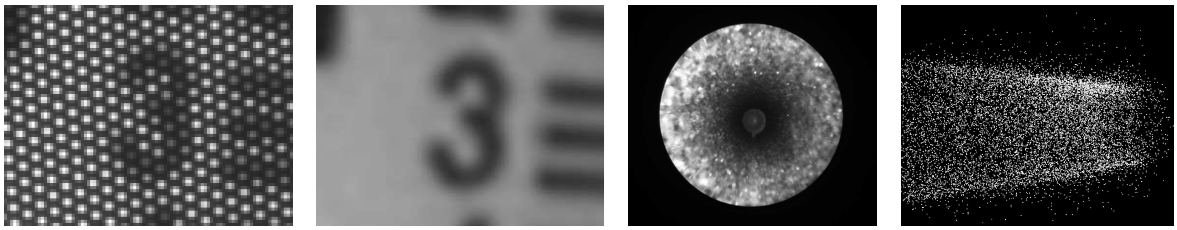
### 6.3.2 Progress Monitoring using Dynamic Light Fields

The area of operation in endoscopic surgery is intrinsically non-static: apart from continuous movement due to heart beat, respiration, and other functions of the body which are not suppressed during anesthetization, the operation area itself is changed in the process by the surgeon. Neglecting the first kind of dynamics, it is still not sensible to generate one static light field including images of different stages of the operation. Therefore, [Vog06] makes use of step-wise static light fields as described in Section 4.2, and especially the rendering techniques introduced in Section 4.4.1.

Figure 6.7 presents a dynamic light field with five time steps which was recorded during the laparoscopic removal of a gall bladder. As camera pose information is obtained by the optical tracking system introduced in Section 5.1.2, registration of the different time steps is unproblematic, and the problem of unknown depth is solved by calibrating the intrinsic parameters of the camera beforehand. Even if the large interval between time steps does not allow for an impression similar to 3-D video, this approach is nevertheless useful for physicians to document the progression of an operation.

### 6.3.3 3-D Reconstruction for Flexible Endoscopes

So far, the endoscopes used in this section to record the images for light field reconstruction were rigid, possessing lenses with short focal length which yield a distorted, but nevertheless correctable image using standard calibration algorithms. In [Win05b] on the other hand, a flexible glass-fiber endoscope (a so-called fiberscope) is used, which may be as small in diameter as a static one (below 3 mm) and can be used to enter even complex cavities, which are otherwise not accessible. On the other hand, a fiberscope consists of only 3,000 to 50,000 fibers where each transmits only one intensity value, comparable to one pixel. These intensities are recorded



(a) (b) (c) (d)

Figure 6.8: Example image acquired with a fiberscope exhibiting honeycomb structure (a) and the result of filtering (b); example image of the bore sequence (c) and 3-D reconstruction (d)

by a camera at the outside end of the fiberscope, resulting in an over-sampling of the information available. Thus, the final image exhibits a honeycomb structure, since the image point of each fiber is surrounded by a dark ring, owed to the transmission properties of glass fibers. An enlarged example image is shown in Figure 6.8(a).

The goal in [Win05b] was to generate a 3-D model of a cylindrical bore despite the deficiencies of the available images. This was achieved by first eliminating the honeycomb structure using a spectral filter. Based on the spectrum of a fiberscopic image, an optimal band-rejection mask is generated automatically, using the method described in [Win05a]. As can be seen in Figure 6.8(b), this process is able to remove the comb structure almost completely, making the images usable for further processing.

Next, the camera is calibrated in order to determine its intrinsic parameters, as well as radial and tangential distortion coefficients. The distortion corrected images are then suitable for 3-D reconstruction. In order to reconstruct the bore shown in Figure 6.8(c), bilinear feature tracking as described in Section 3.2.3 is applied and the extrinsic camera parameters, as well as surface 3-D points, are estimated using the techniques of Sections 3.3.3 and 3.4.1. The resulting, cylindrical point cloud is depicted in Figure 6.8(d). In order to refine the available features, some knowledge on the observed objects is applied, using only feature points in pre-defined image regions.

An interesting effect which can be observed here is that the originally completely cylindrical tube appears conical in reconstruction. Observations indicate that this is due to a systematic error made during feature point tracking. The feature window transformations are approximated by an affine matrix, although in reality, the transformations are perspective ones. This effect is particularly conspicuous here where the camera moves parallel to the surface in view. The result is a systematic distortion of the reconstruction.

In [Wit06], the above approach is used to reconstruct the sphenoidal sinus, a cavity of the skull behind the eyes, of a skull phantom. Another refinement of point feature quality for endoscopic images is studied in [Pop06]. By replacing the conventional ring-shaped illumination at the endoscope tip with four point-shaped light sources, surface irregularities are highlighted, especially if the lights are turned on alternatingly. The resulting features can be tracked longer, are less noisy and more numerous. The reconstruction of a metal tube as above thus shows a distinct improvement in 3-D point localization.

# Chapter 7

# Summary and Outlook

In the following chapter, the main aspects of this thesis and its conclusions are summarized in Section 7.1. Section 7.2 describes continuative work and offers some possibilities for extending upon the results of different methods and applications introduced in the preceding chapters.

## 7.1 Summary

The concept of image-based rendering, and as an implementation of it that of the light field, was introduced in order to supply a means for modeling real objects and scenes and to reproduce them in photo-realistic quality. While early publications on light fields made use of dedicated, often stationary camera mountings, the desire to use commercial, hand-held cameras soon emerged in order to easily capture the necessary input images. This requires the computation of camera pose information for each image and knowledge about the projection properties of the camera, as well as some knowledge about the geometry of the scene.

In this thesis, different methods for computing these data exclusively from the input images using point feature tracking and (self-)calibration methods were examined and extended. Two different kinds of light fields were considered, static and dynamic ones, where the latter forms an extension of the first in that it resolves a major constraint, the static nature of the recorded scene.

Regarding static light fields, the methods introduced can be divided again into those which increase robustness of the reconstruction methods employed, and those which aim at creating a globally valid 3-D reconstruction of the underlying scene. A general tool for achieving these goals is to feed back information about the current state of reconstruction in a loop to earlier stages of the process, and thus to refine the results iteratively. In order to measure the quality of reconstruction, the peak signal-to-noise ratio (PSNR) was introduced for this application. It is

utilized beside the back-projection error which is commonly employed for this purpose but does not necessarily reflect the objective quality of a light field.

Concerning robustness of reconstruction, a comparison of SIFT features with the widely-used Kanade-Lucas-Tomasi (KLT) feature tracking algorithm showed that a reconstruction using SIFT features is on a par with those based on KLT features, working even much better if camera positions are far apart. E. g., while increasing the spacing between camera positions by a factor of four in a common image sequence increased the translational error based on ground truth for KLT tracking drastically from $8.42\%$ to $751\%$, it remained unperturbed at around $8\%$ for SIFT features. However, as computational complexity is exponential, SIFT features are only suitable for small sets of images or special applications such as the concatenation of two image sequences recorded independently. Thus, applied to a sequence of $200$ images, the time for tracking amounted to several hours for SIFT features as opposed to below one minute using KLT tracking.

PSNR was utilized for the first time to make an automatic decision between several reconstruction methods available. It was applied to the choice of factorization method for processing a subsequence to initialize reconstruction, where paraperspective factorization, Christy and Horaud's iterative version, and perspective factorization with two different metric updates were to consider. It was shown that while the back-projection error does not always offer a clear solution, PSNR reliably rules out the failed attempts. The method is therefore to be preferred if it is necessary to obtain a successful reconstruction without additional manual trials.

Robustness was also the motivation for applying outlier detection, weights on individual features, and robust estimators to the image-wise estimation of camera parameters and 3-D points. While for two out of three test image sequences the high quality of KLT feature correspondences alone provided for reliable reconstructions, the Lorentzian estimator was shown to yield good results without exception. It thus outmatched the other methods tested which showed an improvement only in few cases, and failed entirely in reconstructing the light field for a considerable number of the parameter settings evaluated.

Increasing the global validity of a static light field is aimed at for several reasons. It reduces rendering artifacts which appear if errors accumulate while reconstructing a long sequence of images. Furthermore, it decreases the size of the model especially if, as a result, scene geometry can be represented by one globally valid 3-D mesh, called a *global proxy*. The main approaches introduced for this purpose are thus intended to increase the number of images the same feature is found in. This was achieved on one hand by creating a mesh of camera positions which provides additional neighborhood relations between images in a sequence and was used during feature

tracking and camera calibration. These approaches were named *non-sequential* or *mesh* tracking and reconstruction. On the other hand, this relationship was generated by back-projecting known 3-D points into the images and utilizing the resulting positions as initialization of feature tracking. This second approach was therefore called *feature prediction by back-projection*.

The benefit of non-sequential tracking and reconstruction is most noticeable for turntable sequences, where camera pose errors decreased by more than $50$ percent points to $17.3\%$ and $13.4\%$ for translational and rotational error respectively, and PSNR increased by about $10$ dB for both sequences examined. For hand-recorded sequences the success is not as obvious, but is revealed if PSNR is calculated for reference images which are farther apart. The same, but on a smaller scale, is true for feature prediction by back-projection. Here, the average length of a feature trail could be increased by a factor of more than three, from $20.4$ to more than $60$ images per feature and from $10.9$ to more than $30$ images for the two sequences tested, respectively. In case of non-sequential tracking trail length increased even by a factor of $23$ for one turntable sequence.

Thus, if global validity is desired, both approaches are superior to sequential processing, where non-sequential tracking and reconstruction is considerably more accurate, but requires two complete iterations if no mesh information is available a priori. Prediction by back-projection should only be used if streaming-like processing is necessary. Here, the simple source image selection scheme provides the best compromise between processing time and accuracy.

Another approach that allows for increasing global consistency was adopted from robotics and autonomous mapping of environments. By coercively closing a loop in camera movement if accumulated errors prevented a correct reconstruction, a globally correct constellation can be created. It was shown exemplarily that for large errors the method is able to halve the pose error, from, e. g., $71.3\%$ to $26.1\%$ translational error, although the improved global correctness is paid for with an increase of local errors. In general however, the application of bundle adjustment to said loops is the preferable approach.

Beside the parameters of the recording camera, a light field requires depth information in order to correctly visualize a virtual view. Originally, this information was provided as dense depth maps pertaining to each input image. Here, two alternatives were discussed, local and global proxies. Both approaches encode depth information as triangle meshes, but where the first requires a mesh for each input image, the second combines all depth information in one globally valid mesh. While global proxies were already required as part of the unstructured lumigraph, local proxies form a compromise between those and dense depth maps. They are easy to generate, use less memory and are more efficient in rendering than depth maps. In the context

of global consistency however, a new approach to generating a global proxy from 2-D feature correspondences was introduced and first results were presented. Although the constraints in its construction are still too restrictive for universal utilization, rendering is more than four times faster than with depth maps, improving from $3.3$ to $15$ frames per second for a proxy affecting the whole output image. It is thus on a par with local proxies, and quality is not reduced. Additionally, a method was presented to improve global proxies based on non-linear optimization of rendering quality using, again, an information feedback loop.

Finally, the extension of an existing light field with additional image sequences can be assigned to the same category of increasing global correctness. Two methods for finding the best image to restart feature tracking and thus integrate a new sequence were described. Thereby, the matching with SIFT features was $20$ to $30$ pixels more accurate in predicting feature positions than the rendering feedback approach based on adaptive random search or the particle filter, the divergence being only $2.0$ pixels in average in contrast to $31.6$ pixels.

Regarding the second major topic of this thesis, dynamic light fields, it was stated that the relaxation of the static-scene constraint cannot be general. It is impossible to capture a light field densely enough with only one camera if parameter space is thus increased by one dimension. Therefore, the dynamic light field models regarded here had to include new but less stringent constraints than for the static case.

Three different models and their mostly automatic reconstruction from one or several image sequences were introduced. The first one, the step-wise static light field, requires that for multiple time steps one image sequence is recorded each, in such a way that the scene remains static during each recording. The sequences are reconstructed individually in a first step. In a second step, the camera paths are concatenated and subsequently, the whole model is refined using non-sequential tracking and reconstruction. During rendering of the resulting light field, a switching between the original time steps as well as free navigation within each time step is then possible.

The second type of dynamic light field does not require an as laborious recording step as the first one, as it permits recording the dynamic scene in a single image sequence. It imposes, however, the constraint that a moving object visible in the scene has to be rigid in itself. This allows for the automatic segmentation of background and foreground on the point feature level using the subspace separation method of Kanatani et al. Thus, background and object geometry can be reconstructed individually and are then merged into a common coordinate system. Again, the light field is divided into different time steps, but now this is done by grouping views of the moving object using vector quantization. The visible time step during rendering is selected by masking out all other source views using confidence maps. Thus, the same effect as before is

achieved but requiring considerably less user interaction.

However, the final goal is to provide a continuous representation of the dynamic part of the light field. Therefore, a first implementation of an object light field was presented, which consists of one light field per object in the scene. It is thus possible to view background and foreground individually, but also together in the same context with a user-adjustable transformation between the two. The approach, however, still requires some manual post-processing of confidence maps.

Both step-wise static light fields and light fields of rigid, moving objects were demonstrated on three examples each. For the object light field, one prototype model was presented.

In the previous chapter finally, new applications of static and dynamic light fields to other areas were shown. It was demonstrated that light fields are well suited as object models in augmented reality since they form representations of real-world objects themselves. The possible discrepancy between reality and virtuality is thus avoided. Likewise, light field reconstruction was applied to the classic fields of object localization and tracking, either by directly utilizing light fields as object models or by using the methods for their reconstruction to generate the required models from hand-held camera sequences.

A different and still uncommon application area is medical imaging, particularly in endoscopic surgery. Here, light fields were used to reduce highlights which are created by an endoscope's light source on wet tissue. Additionally, an application of the step-wise static light field model was described with the visualization of different stages of an operation in [Vog06], thus monitoring and documenting its progress. Finally, an investigation of the increasing demands on the reconstruction system when using flexible instead of rigid endoscopes was introduced, and a first solution presented.

## 7.2 Future Work

The work presented offers a number of starting points for future research. A first such point is the Kanade-Lucas-Tomasi feature tracking algorithm. As it was already observed for the application to images acquired by a flexible endoscope in Section 6.3.3, the affine distortion correction which is currently applied to the feature windows may not be sufficient in all cases. If the surfaces the features are found on are mostly perpendicular to the viewing direction of the camera, an affine distortion matrix is sufficient. However, if features are on surfaces parallel to the viewing direction and the camera moves forward or backwards, such as through a corridor, the resulting distortions of the feature windows are perspective and thus poorly modeled by an affine distortion matrix.

Modeling perspective feature distortion is equivalent to estimating a homography with eight degrees of freedom for each feature. Determining these eight parameters all at once may, however, pose a considerable issue for stability of the estimation. Good initialization and limitation of the additional translation are therefore essential.

Next, regarding the subsequent reconstruction procedures, selecting a suitable factorization method using PSNR is only one possible way for modifying the process based on rendering quality. The approach could likewise be used to determine optimal parameters for tracking and reconstruction, such as the number of features to track or the maximum back-projection error. Another application would be to monitor the progress of reconstruction using PSNR and in this way to detect a failure of calibration of the current camera pose.

A major problem for globally consistent reconstruction was clearly visible in the experiments of Chapter 5. The per-image error increases even if global consistency is improved, and thus usually impairs rendering quality as well. A solution is offered by bundle adjustment, which refines all 3-D point and camera parameters at once. However, it also requires considerable time for this task. A way out is presented by efficient bundle adjustment strategies such as the one proposed in [Shu99]. It combines several images into segments and defines virtual key frames for each segment which are optimized as substitutes for the whole segment and thus reduce the overall number of parameters.

Another promising initial point for further work is the global proxy, both its construction and its optimization. The successful application of image difference as error measure for optimization was already described in Section 3.7.3. However, it may be likewise applied to the construction of the global proxy as a means for identifying nodes which were inserted wrongly. But not only the seamless integration of a new 3-D point is important for rendering quality. A set of four non-collinear points may be meshed in two different ways using Delaunay triangulation. The common, internal edge of the resulting triangles may thus be, in the worst case, perpendicular to a 3-D edge in the scene. Morris and Kanade showed in [Mor00] that by flipping these kinds of edges and calculating an error based on projecting an image onto the mesh, the correctness of the mesh can be improved. This fact has not been considered yet in neither construction nor optimization of global proxies.

The optimization of global proxies offers some additional room for refinement. Currently, full-sized images are rendered in order to calculate the error modification for each node in the mesh. However, only a fraction of the image is affected by the variation of one node, namely the adjacent triangles. By restricting rendering to these areas, a considerable speed-up could be achieved. In addition to that, the algorithm does not change the number of nodes in the mesh.

Adding new nodes may further improve the results if applied in areas with large errors.

Even more than static reconstruction, the generation of dynamic light fields offers a large number of different directions for future research. One major issue for rigid motion light fields and object light fields is the determination of scaling between reconstructions of different entities in the scene. However, the problem is not easily solved. A potential solution is offered again by an image-based approach. If scaled wrongly and viewed from a different angle, the moving object is rendered at the wrong position relative to the background. The effect would be visible in image difference if the same method as throughout this thesis of leaving out the original image is applied.

The problem of precisely segmenting the moving object in each image may be solved similarly. Rendering a light field of the object alone, image quality inside the object's boundaries is high given correct camera pose and depth reconstruction. The surrounding background on the other hand is formed by many different, wrongly placed and thus overlapping patches of the original images. Thus, the object boundary is formed by a marked difference of high and low quality expressed by image difference, and can be approximated, e. g., by an active contour model such as snakes [Kas88]. This method, of course, is bound to fail if the background is mostly homogeneous in texture.

The segmentation of foreground and background has so far only been demonstrated for one moving object in front of the static background. Nevertheless, it is theoretically possible to separate any number of objects provided that enough features are found on each one. Neither camera calibration and 3-D reconstruction nor the partitioning into time steps afterwards form an obstacle, only the number of possible combinations of visible and invisible objects increases exponentially with the number of objects. This poses a problem for visualization as for each combination, one light field is generated, leading to vast memory requirements and non-intuitive user interaction. These issues do not occur for object light fields.

In the same context of multiple objects falls the next issue to be treated, occluding objects and those leaving the field of view of the camera, or even the continued tracking if movement is discontinued. In this case, it is not possible to rely on the segmentation algorithm any longer. In order to treat these circumstances correctly, additional methods for object tracking and a more sophisticated object management have to be incorporated.

Beginning from the initial two-plane light fields, constraints on the observed scene have been removed step by step in both light field reconstruction and rendering. The automatic generation of dynamic light fields containing rigid, moving objects will not be the end of this development. Algorithms exists which are able to reconstruct the 3-D geometry of a scene despite the presence

of moving and deforming objects. One such algorithm is, e. g., outlined in [Har03]. The difficulties here are to find a stable solution for long image sequences and to define a suitable model for visualizing the resulting light field. It should be interesting to follow the development of this complex and exciting area of research in years to come, especially since new applications such as in medicine are emerging even now.

# Appendix A

# Mathematical Symbols

In this appendix all mathematical symbols and notations used in this work are explained. First some general notations are given which are applicable to different symbols.

- Scalar values are denoted by italic letters like $a$, $b$, $c$.
- Vectors are denoted by bold italic letters like $\boldsymbol{x}$
- The $i$-th element of a vector $\boldsymbol{x}$ is denoted by $x_i$.
- Matrices are denoted by capital bold italic letters $\boldsymbol{X}$.
- The element at the $i$-th row and $j$-th column of a matrix $\boldsymbol{X}$ is denoted by $x_{ij}$.
- The transposed of a vector $\boldsymbol{x}$ and a matrix $\boldsymbol{X}$ is denoted by $\boldsymbol{x}^{\mathrm{T}}$ and $\boldsymbol{X}^{\mathrm{T}}$.
- The inverse of a matrix $\boldsymbol{X}$ is denoted by $\boldsymbol{X}^{-1}$.
- The Euclidean norm of a vector is denoted by $\|\boldsymbol{x}\|$.
- The Frobenius norm of a matrix is also denoted by $\|\boldsymbol{X}\|$.
- A homogeneous vector is denoted by underlining like $\underline{\boldsymbol{x}}$.
- An estimation to a value $x$ is denoted by $\widehat{x}$. $\widehat{x}$ is also used for a computed value when compared with a ground truth/real value.
- A matrix $\boldsymbol{Y}$ composed of a matrix $\boldsymbol{X}$ and a vector $\boldsymbol{x}$ is denoted by $\boldsymbol{Y} = (\boldsymbol{X}|\boldsymbol{x}\,)$.

The following table lists the used symbols, their meaning, and the page of their first occurrence.

| | | |
|---|---|---|
| $\boldsymbol{x}_w$ | $x$-axis of world coordinate system | 5 |
| $\boldsymbol{y}_w$ | $y$-axis of world coordinate system | 5 |
| $\boldsymbol{z}_w$ | $z$-axis of world coordinate system | 5 |
| $L$ | The plenoptic function | 5 |
| $\theta$ | First viewing direction angle | 5 |
| $\phi$ | Second viewing direction angle | 5 |
| $\lambda_L$ | Wavelength parameter of the plenoptic function | 5 |
| $\tau$ | Time instance | 5 |
| $\boldsymbol{t}$ | 3-D translation vector | 5 |
| $L_\mathrm{p}$ | Two-plane light field | 6 |
| $u_\mathrm{p}$ | First coordinate axis of camera plane | 6 |
| $v_\mathrm{p}$ | Second coordinate axis of camera plane | 6 |
| $s_\mathrm{p}$ | First coordinate axis of focal plane | 6 |
| $t_\mathrm{p}$ | Second coordinate axis of focal plane | 6 |
| $M_{u,v}$ | Size of the grid on the camera plane | 6 |
| $N_{s,t}$ | Size of the grid on the focal plane | 6 |
| $x_{i,j,k,l}$ | Data point on one pair of grid points | 6 |
| $B_{i,j,k,l}$ | Basis function for each pair of grid points | 6 |
| $\tilde{L}_\mathrm{p}$ | The finite dimensional lumigraph approximating $L_\mathrm{p}$ | 6 |
| $\boldsymbol{x}_c$ | $x$-axis of camera coordinate system | 22 |
| $\boldsymbol{y}_c$ | $y$-axis of camera coordinate system | 22 |
| $\boldsymbol{z}_c$ | $z$-axis of camera coordinate system | 22 |
| $\boldsymbol{R}$ | 3-D rotation matrix | 22 |

| | | |
|---|---|---|
| $\boldsymbol{p}$ | 3-D point in world coordinates | 22 |
| $\boldsymbol{p}_c$ | 3-D point in camera coordinates | 22 |
| $\underline{\boldsymbol{p}}$ | Homogeneous 3-D point in world coordinates | 23 |
| $\underline{\boldsymbol{p}}_c$ | Homogeneous 3-D point in camera coordinates | 23 |
| $\mathbf{0}_3$ | The 3-D column null-vector | 23 |
| $\boldsymbol{x}_m$ | $x$-axis of image coordinate system | 23 |
| $\boldsymbol{y}_m$ | $y$-axis of image coordinate system | 23 |
| $\boldsymbol{q}_m$ | 2-D point in image coordinates | 23 |
| $f$ | The focal length | 23 |
| $\underline{\boldsymbol{q}}_m$ | Homogeneous 2-D point in image coordinates | 24 |
| $\boldsymbol{o}$ | Origin of image | 24 |
| $\boldsymbol{u}_s$ | Horizontal axis of the sensor coordinate system | 24 |
| $\boldsymbol{v}_s$ | Vertical axis of the sensor coordinate system | 24 |
| $d_x$ | Horizontal size of sensor elements | 24 |
| $d_y$ | Vertical size of sensor elements | 24 |
| $\mathbf{0}_2$ | The 2-D column null-vector | 25 |
| $u_{pp}$ | Horizontal coordinate of principal point | 25 |
| $v_{pp}$ | Vertical coordinate of principal point | 25 |
| $\underline{\boldsymbol{q}}$ | Homogeneous 2-D point in sensor coordinates | 25 |
| $f_x$ | Effective focal length in horizontal direction | 25 |
| $f_y$ | Effective focal length in vertical direction | 25 |
| $\beta$ | Skew of sensor coordinate axes | 25 |
| $\boldsymbol{K}$ | The calibration matrix | 26 |
| $\boldsymbol{I}_{3\times3}$ | The $3 \times 3$ identity matrix | 26 |
| $\boldsymbol{P}$ | The projection matrix | 26 |
| $\boldsymbol{p}_g$ | Center of gravity of an object | 27 |

| | | |
|---|---|---|
| $\boldsymbol{A}$ | An arbitrary affine transformation | 29 |
| $\boldsymbol{e}$ | The epipole of an image | 30 |
| $\boldsymbol{l}$ | An epipolar line | 30 |
| $\boldsymbol{q}$ | 2-D point in sensor coordinates | 30 |
| $\boldsymbol{F}$ | The fundamental matrix | 30 |
| $N$ | Number of feature and scene points | 31 |
| $F$ | Number of frames | 31 |
| $\boldsymbol{W}$ | The measurement matrix | 31 |
| $\overline{\boldsymbol{q}}$ | Centroid of image points | 32 |
| $\widetilde{\boldsymbol{q}}$ | Registered image point | 32 |
| $\widetilde{\boldsymbol{W}}$ | The registered measurement matrix | 32 |
| $\widetilde{\boldsymbol{p}}$ | Registered 3-D world point | 32 |
| $\boldsymbol{\Psi}$ | Motion matrix | 32 |
| $\boldsymbol{\Phi}$ | Shape matrix | 32 |
| $\boldsymbol{U}$ | Left matrix of singular value decomposition | 32 |
| $s$ | A singular value | 32 |
| $\boldsymbol{V}$ | Right matrix of singular value decomposition | 32 |
| $\widehat{\boldsymbol{\Psi}}$ | Estimated Motion matrix | 33 |
| $\widehat{\boldsymbol{\Phi}}$ | Estimated Shape matrix | 33 |
| $\overline{p}_{f,3}$ | Distance of object centroid to camera center along principal axis | 34 |
| $\mu$ | Projective depth of an image point | 36 |
| $\boldsymbol{W}_p$ | The measurement matrix for projective factorization | 36 |
| $x_p$ | Horizontal position of a 2-D point for paraperspective projection | 39 |
| $\boldsymbol{W}_{\text{CH}}$ | The modified measurement matrix for iterative factorization | 39 |
| $\boldsymbol{D}$ | A transformation for ambiguous reconstructions | 40 |
| $\underline{\boldsymbol{m}}_\infty$ | The plane at infinity | 41 |

| | | |
|---|---|---|
| $\check{s}$ | A scale factor | 41 |
| $\boldsymbol{C}_a$ | The absolute conic | 41 |
| $\boldsymbol{Q}_a$ | The absolute quadric | 41 |
| $\boldsymbol{E}$ | The essential matrix | 42 |
| $\boldsymbol{Q}_a^*$ | The absolute dual quadric | 43 |
| $\boldsymbol{C}_a^*$ | The absolute dual conic | 43 |
| $\overline{\boldsymbol{Q}}_a^*$ | The absolute dual quadric in its canonic form | 43 |
| $\epsilon_{\text{SC}}$ | Cost function for non-linear self-calibration | 45 |
| $\boldsymbol{\epsilon}_{\text{B}}$ | Residual vector of back-projected feature points | 45 |
| $\boldsymbol{J}$ | The Jacobian matrix | 46 |
| $\boldsymbol{f}^g$ | A gray-level image | 52 |
| $x$ | Horizontal pixel position in an image | 53 |
| $y$ | Vertical pixel position in an image | 53 |
| $\boldsymbol{g}$ | Gradient vector of an image (window) | 53 |
| $\boldsymbol{G}$ | The structure matrix | 53 |
| $r$ | Window radius | 53 |
| $\lambda$ | An eigenvalue of the structure matrix | 53 |
| $\boldsymbol{d}$ | An image displacement vector | 54 |
| $\epsilon$ | Displacement error between two feature windows | 54 |
| $\boldsymbol{d}_a$ | The affine motion field for feature tracking | 55 |
| $\boldsymbol{D}_a$ | Affine distortion matrix for feature tracking | 55 |
| $\boldsymbol{g}_a$ | Affine gradient vector of an image | 55 |
| $\alpha$ | Contrast portion of illumination compensation | 56 |
| $\kappa$ | Brightness portion of illumination compensation | 56 |
| $\boldsymbol{m}_{\text{SIFT}}$ | Image gradient magnitude for SIFT features | 57 |
| $\boldsymbol{r}_{\text{SIFT}}$ | Image gradient rotation for SIFT features | 57 |

| | | |
|---|---|---|
| $\boldsymbol{c}$ | A SIFT feature vector | 58 |
| $\mathcal{C}$ | A set of SIFT feature vectors | 58 |
| $d$ | The Euclidean distance between two vectors | 58 |
| $\epsilon_{\text{SIFT}}$ | A threshold value for the similarity of two SIFT feature vectors | 58 |
| $\nu$ | Fraction of initial number of features in mesh tracking | 59 |
| $\widehat{\sigma}$ | Threshold for outlier detection in LMedS | 61 |
| $\boldsymbol{\Delta}$ | Difference matrix for LMedS in factorization | 62 |
| $\overline{\epsilon}_{\text{bp}}$ | The average back-projection error | 62 |
| $\boldsymbol{f}$ | A color image | 63 |
| $\widehat{\boldsymbol{f}}$ | An image rendered from a light field | 63 |
| SNR | Signal-to-noise ratio | 63 |
| PSNR | Peak signal-to-noise ratio | 63 |
| $\chi$ | Projective scale factor | 66 |
| $N_t$ | Minimum number of features for triangulating a 3-D point | 68 |
| $\overline{\Delta_t}$ | Average distance between two camera positions | 70 |
| $\overline{\alpha}$ | Average angle between viewing directions of two camera poses | 70 |
| $\mu_t$ | Multiple of average camera position distance | 70 |
| $\mu_\alpha$ | Multiple of average angle between camera viewing directions | 70 |
| $\mu_i$ | Minimum index different between two frames | 70 |
| $\gamma$ | Angle between camera motion direction and neighbor | 70 |
| $\gamma_{min}$ | Threshold on the direction angle | 70 |
| $\epsilon_{\text{SSD}}$ | Sum of squared differences of back-projection errors | 71 |
| $\boldsymbol{\Sigma}_{\text{B}}$ | Covariance matrix of feature locations | 72 |
| $\epsilon_{\text{ML}}$ | Squared Mahalanobis distance of back-projection errors | 72 |
| $z$ | A depth value | 76 |
| $\mathcal{F}_{\text{GP}}$ | Set of images whose features are considered in a global proxy | 81 |

| | | |
|---|---|---|
| $\mathcal{T}$ | Face of a global proxy | 81 |
| $\boldsymbol{v}$ | Vertex of a global proxy | 81 |
| $\mathcal{T}_{\text{in}}$ | Set of intersected faces | 81 |
| $\mathcal{E}_b$ | Set of border edges of a global proxy | 82 |
| $w$ | Weight value for selecting frames for tracking of predicted features | 86 |
| $\boldsymbol{\epsilon}_L$ | Residual vector for closing loops in camera movement | 88 |
| $\boldsymbol{\epsilon}_{\text{DO}}$ | Residual vector for optimizing depth maps | 89 |
| $\boldsymbol{x}$ | A pixel position | 89 |
| $\boldsymbol{v}_J$ | Parameter vector for optimizing a global proxy | 91 |
| $\boldsymbol{\epsilon}_{\text{GP}}$ | Residual vector for optimizing global proxies | 91 |
| $a$ | An accumulator for counting common SIFT features | 93 |
| $\delta$ | The Kronecker delta function | 93 |
| $\mathcal{M}$ | A set of matching SIFT feature vector pairs | 93 |
| $\boldsymbol{H}$ | A 2-D homography | 93 |
| $\boldsymbol{\rho}$ | Camera state (pose) vector | 94 |
| $\mathcal{R}$ | Set of camera state hypotheses | 94 |
| $\varrho$ | Scalar rating for a hypothesis $\boldsymbol{\rho}$ | 94 |
| $\boldsymbol{S}$ | A $4 \times 4$ matrix for scaling projection matrices | 101 |
| $K$ | Number of objects in a dynamic scene | 104 |
| $\boldsymbol{Q}$ | The shape interaction matrix for feature segmentation | 105 |
| $\boldsymbol{\Sigma}$ | Center matrix of singular value decomposition | 105 |
| $\boldsymbol{W}^*$ | The measurement matrix in its canonical form for segmentation | 105 |
| $C$ | Cost function for sorting the shape interaction matrix | 106 |
| $\mathcal{L}$ | A set of features grouped together | 106 |
| $J$ | Residual value for merging two groups of features | 107 |
| G-AIC | The geometric AIC of a subspace | 107 |

| | | |
|---|---|---|
| $\boldsymbol{h}$ | A confidence map | 115 |
| $\mathcal{F}_{\text{bg}}$ | Set of input images used for rendering scene background | 116 |
| $\mathcal{F}_{\text{fg}}$ | Set of input images used for rendering dynamic scene foreground | 116 |
| $\epsilon_{\boldsymbol{t},\text{rel}}$ | Relative translation error between two camera poses | 123 |
| $\epsilon_{\boldsymbol{R},\text{rel}}$ | Relative rotation error between two camera poses | 123 |
| $\epsilon_{\text{shape}}$ | Normalized relative shape error for a sphere fit to reconstructed 3-D points | 123 |
| $\check{\alpha}$ | A strictness parameter for the normalized correlation coefficient | 167 |

# Appendix B

# German Title, Contents, Introduction and Summary

Der deutsche Titel dieser Dissertation lautet:

**Rekonstruktion und Modellierung statischer und dynamischer Lichtfelder**

## B.1 Inhaltsverzeichnis

## B.2 Einleitung

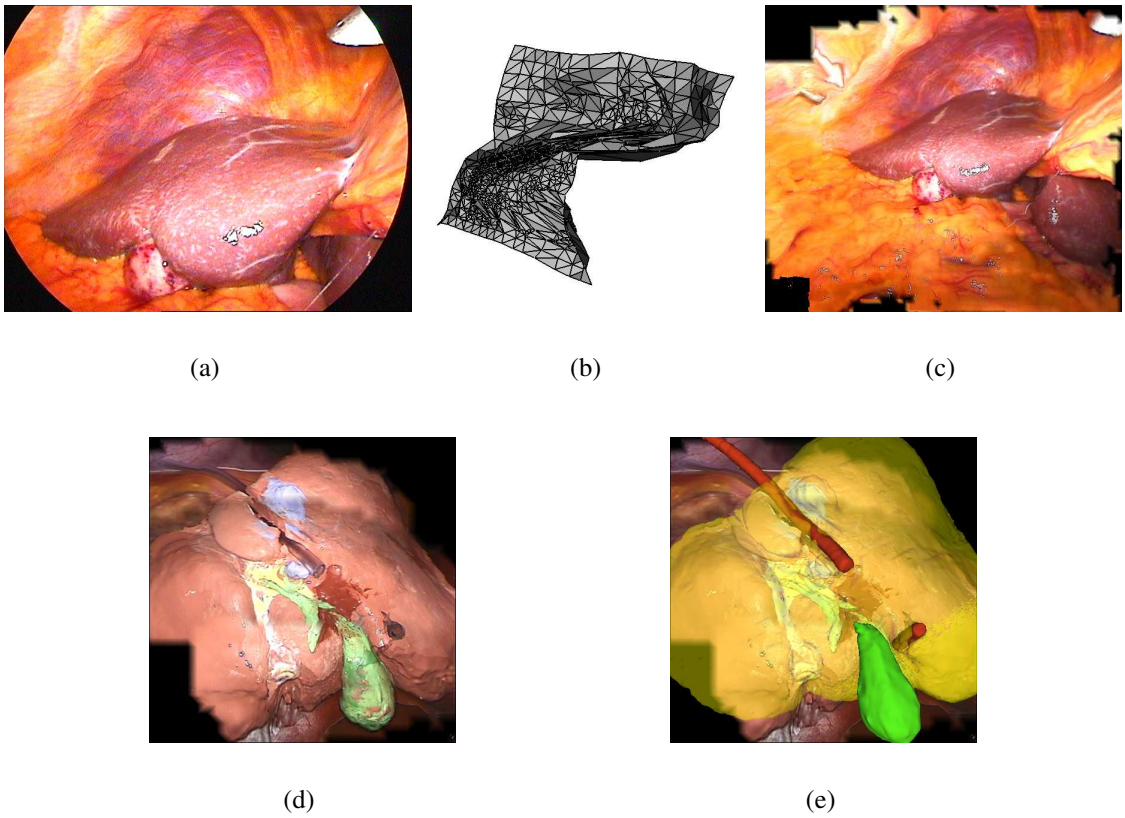
Die Modellierung virtueller Umgebungen hat ihren Weg in viele Anwendungen gefunden, nicht nur in der Unterhaltung, sondern auch in die Bereiche Technik, Medizin, Wirtschaft und Bildung. In der Unterhaltung werden diese virtuellen Umgebungen meist in Filmproduktionen und Videospielen eingesetzt. Während hier jedoch die Akzeptanz unrealistischer oder künstlicher Szenen inhärent ist, benötigen speziell medizinische Anwendungen einen hohen Grad an Realismus. Moderne Techniken der geometrischen Modellierung erlauben eine sehr realitätsnahe Rekonstruktion schwieriger Eigenschaften, jedoch fordern sie auch einige künstlerische Fähigkeiten um diese Ergebnisse zu erreichen.

Eine Alternative zur herkömmlichen geometrischen Modellierung wurde mit dem Konzept des bildbasierten Renderings (IBR) eingeführt, deren gängigste Repräsentationen der *Lumigraph* [Gor96] und das *Lichtfeld* [Lev96] sind. Durch die Verwendung von Bildern realer Umgebungen ermöglichen sie es, Modelle realer Szene und Objekte zu generieren und sie in photorealistischer Qualität zu visualisieren. Für Sammlungen von Bildern oder Bildsequenzen statischer Szenen stellt die Rekonstruktion solcher bildbasierter Modelle bereits eine herausfordernde Aufgabe dar und um so mehr noch für sich dynamisch ändernde Umgebungen. Ansätze zu deren robuster Lösung sind das Thema dieser Arbeit

### B.2.1 Lichtfelder aus Bildsequenzen

In einfachen Worten ausgedrückt benötigt ein bildbasiertes Modell wie ein Lichtfeld als Eingabedaten nur einen Satz Bilder sowie das Wissen über die Lage der Kamera (d. h. ihre Position und Orientierung) und ihre internen Parameter während der Aufnahme jedes einzelnen Bildes. Aus diesen Daten kann ein Bild des aufgenommenen Gebietes, sei es ein einzelnes Objekt oder eine große Szene, aus einem beliebigen Blickwinkel generiert werden, solange dieser den ursprünglichen Kamerapositionen halbwegs ähnlich ist. Als Beispiel kann man sich einen Verkäufer im Internet vorstellen, der ein 3D-Modell seiner Ware in seinem Onlineshop präsentieren möchte. Benutzt er ein Lichtfeld, so muss er nur einige Dutzend Bilder seines Produktes aufnehmen und ermöglicht es so seinen Kunden das Produkt aus jedem Blickwinkel zu betrachten den sie wünschen.

Ein weiteres, eher wissenschaftliches Beispiel für die Verwendung von Lichtfeldern ist die endoskopische Chirurgie [Vog06]. Hier werden die Bilder aus einem Endoskop zusammengeführt, um dem Chirurgen einen dreidimensionalen Eindruck des Operationsgebiets zu geben, in dem er nun ohne Einschränkungen durch unhandliche medizinische Geräte navigieren kann.



(a) (b) (c)

(d) (e)

Bild B.1: Lichtfelder in der endoskopischen Chirurgie: (a) Sicht des Chirurgen durch ein Endoskop, (b) Oberflächenrekonstruktion und (c) aus dem resultierenden Lichtfeld generierte Ansicht, welche die Szene aus einer größeren Entfernung zeigt. (d) Lichtfeld eines Leber-/Gallenblasenmodells mit (e) einer überlagerten CT-Aufnahme. Bilder mit freundlicher Genehmigung von F. Vogt [Vog06].

Dies wird durch Abbildung B.1 verdeutlicht, die als Beispiel das laparoskopische Entfernen einer Gallenblase (Cholecystectomie) zeigt. Durch das Endoskop steht dem Chirurgen nur eine Nahansicht mit begrenztem Blickwinkel zur Verfügung, so wie in Abbildung B.1(a). Durch die Rekonstruktion der Geometrie des Operationsgebiets aus mehreren endoskopischen Aufnahmen, wie in Abbildung B.1(b) dargestellt, kann die Szene wie in Abbildung B.1(c) von einer Überblicksperspektive aus visualisiert werden, indem die Texturinformation mehrerer Bilder kombiniert wird. Darüber hinaus können bei erfolgreicher 3D-Rekonstruktion und bildbasierter Modellierung unterschiedliche Modalitäten, wie beispielsweise Computertomographie (CT), Kernspintomographie (MRI) oder Ultraschall, eingeblendet werden und so dem Chirurgen weitere Informationen bereitstellen, die ansonsten unsichtbar wären. In Abbildung B.1(e) wurde dies für das Lichtfeld eines Modells einer Leber und Gallenblase, dargestellt in Abbildung B.1(d), in Kombination mit einer CT-Aufnahme durchgeführt.

(a) (b) (c)

Bild B.2: Drei unterschiedliche Möglichkeiten für die Bildaufnahme für Lichtfelder: (a) Verwendung eines Roboterarms der auch die Lage der Kamera liefert, (b) eines optischen Trackingsystems das die Kameraposition durch Verfolgung von optischen Markern bestimmt oder (c) einer handgeführten Kamera und Algorithmen aus dem Bereich Struktur-aus-Bewegung zur Lageschätzung. Abbildungen (a) und (b) mit freundlicher Genehmigung von F. Vogt [Vog06].

Für beide Beispiele stellt sich die Frage, woher die Kameraparameter für die einzelnen Aufnahmen genommen werden können. In [Vog06] sind zwei Optionen für das Sammeln dieser Daten ein Roboterarm wie der in Abbildung B.2(a) dargestellte und ein optisches Trackingsystem, das die Position und Orientierung des Endoskops mit zwei Kameras verfolgt, wie in Abbildung B.2(b) gezeigt. Während dies praktikable Lösungen für ein klinisches Umfeld sind, sind sie viel zu teuer für die zuerst beschriebene Anwendung. Für den Internethändler wäre es eine akzeptable Lösung, wenn er eine Videokamera aus dem Endanwendersegment wie die aus Abbildung B.2(c) benutzen könnte, um so eine Videosequenz eines Gegenstandes aufnehmen zu können, ohne auf spezialisierte Hardware zurückgreifen zu müssen, welche die Informationen über Lage und Parameter der Kamera liefert.

Im Prinzip enthalten die Bilder die komplette Information, die zum Lösen dieses Szenarios benötigt wird. Der Schlüssel dazu ist die Identifikation von *Punktkorrespondenzen* zwischen einzelnen Bildern der Bildsequenz. Das Wissen über die Projektion bestimmter Punkte der Szene in zwei Bilder – die Definition einer Punktkorrespondenz – erlaubt die Berechnung einer Abbildung für Punkte von einem Bild in ein anderes insofern, als die möglichen Positionen im zweiten Bild auf eine Gerade eingeschränkt werden. Sind die Korrespondenzen für drei Bilder bekannt, so kann die Abbildung exakt berechnet werden. Die drei relativen Lagen der Kamera können so sogar inklusive einiger interner Parameter der Kamera geschätzt werden, solange einige An-


Bild B.3: Beispiel einer Bildsequenz für dynamische Lichtfelder: ein Spielzeugfahrzeug bewegt sich in einer statischen Umgebung während die handgeführte Kamera unabhängig davon bewegt wird.

nahmen gemacht werden. Sind Korrespondenzen in weiteren Bildern gegeben so ist es möglich, diese Einschränkungen für die internen Parameter schrittweise aufzuheben.

In der praktischen Anwendung wird die Schätzung der Kameraparameter durch viele Faktoren erschwert. Die gefundenen Punktkorrespondenzen sind üblicherweise aufgrund von Bildrauschen leicht verschoben, oder beispielsweise aufgrund von Verdeckungen nicht für die gesamte Sequenz verfügbar. Daher sind robuste Verfahren notwendig die in der Lage sind, die sich ergebenden Fehler zu kompensieren und sogar sehr lange Bildsequenzen zu verarbeiten. Diese Methoden beinhalten meist die Verwendung von (wesentlich) mehr Daten als notwendig und das Bestimmen derjenigen Lösung, die diesen Daten am besten entspricht. Ausreißerdetektion und probabilistische Modellierung können die Ergebnisse weiter verbessern.

Zu diesem Zeitpunkt sind bereits alle Daten für die Modellierung eines einfachen Lichtfeldes wie das in der ursprünglichen Veröffentlichung zu Lichtfeldern [Lev96] beschriebene vorhanden. Allerdings benötigt die Parametrisierung dieser Art von Lichtfeld eine “dichte Abtastung” der Szene ausgehend von einem regulären Gitter von Blickpunkten. Im Falle einer handgeführten Kamera sind weder eine dichte Abtastung noch eine Abtastung von regelmäßigen Blickpunkten aus möglich, da die menschliche Hand schlecht für eine derartige Aufgabe geeignet ist. Jüngere Lichtfeldmodelle berücksichtigen daher diese zusätzlichen Anforderungen [Hei99a, Sch01b, Bue01]. Aufgrund der dünnen Abtastung werden durch unbekannte Oberflächengeometrie bedingte Fehler in aus dem Lichtfeld generierten Bildern sichtbar. Wissen über die Szenengeometrie reduziert diese Fehler daher entsprechend. Es existiert eine Reihe von Methoden zur Schätzung der Tiefe einer Szene aus mehreren Ansichten. *Struktur-aus-Bewegung* sowie Tiefenberechnung aus Stereobildern oder mehreren Ansichten können hier angewendet werden.

Um schließlich wieder zu dem Beispiel des Internethändlers zurückzukehren, könnte dieser

nun nicht nur ein statisches Modell eines Gegenstandes demonstrieren wollen, sondern auch die Funktionsweise eines sich bewegenden Teils oder zum Beispiel die Bewegung eines Fahrzeugs durch die Umgebung wie im Falle des Spielzeugfahrzeugs in Abbildung B.3. In medizinischen Anwendungen wie der endoskopischen Chirurgie [Vog06] kann es nötig sein, das Bewegen von Organen oder medizinischen Instrumenten zu modellieren. Ein solches *dynamisches* Lichtfeld lässt sich wiederum am einfachsten mit Hilfe von spezialisierter Hardware rekonstruieren wie beispielsweise einer Anordnung von mehreren Kameras. Wird an das Problem mit nur einer (handgeführten) Kamera herangegangen, muss eine softwarebasierte Lösung gefunden werden, und das Einsatzgebiet ist im Allgemeinen nicht so allgemeingültig wie im ersten Fall. Mit der Identifizierung unterschiedlicher Zeitschritte oder der Trennung sich unterschiedlich bewegender Objekte kann ein zusammengesetztes Lichtfeld rekonstruiert und mittels angepasster Darstellungstechniken visualisiert werden.

Diese Verarbeitungskette der Rekonstruktion statischer oder dynamischer Lichtfeldmodelle aus einer oder mehreren Bildsequenzen, die mit einer handgeführten Kamera aufgenommen wurden, ist das Hauptthema dieser Arbeit. Der darauf folgende Schritt der korrekten Visualisierung von Bildern aus einem solchen Lichtfeld heraus, mit seinen vielen verschiedenen Ansätzen, wird nur kurz als Teil des Überblicks über den Stand der Technik in der Lichtfeldmodellierung im nächsten Abschnitt abgedeckt. Er wird eher als ein Hilfsmittel zur Überwachung und Bewertung des Erfolgs der Rekonstruktion betrachtet.

### B.2.2 Stand der Technik in der Modellierung von Lichtfeldern

Die Idee des Lichtfeldes wurde von der *plenoptischen Funktion* $L(\theta, \phi, \lambda_L, \tau, \boldsymbol{t})$, vorgestellt in [Ade91], abgeleitet. Sie beschreibt das Aussehen eines sichtbaren Raumes mit Hilfe von sieben Parametern, nämlich dem Blickpunkt $\boldsymbol{t}$ des Beobachters in Weltkoordinaten, den zwei Winkeln $\theta$ und $\phi$ für die Blickrichtung und die Wellenlänge $\lambda_L$ des beobachteten Lichtstrahls zu einem bestimmten Zeitpunkt $\tau$. Dieser Aufbau ist in Abbildung B.4 für einen einzelnen Lichtstrahl gezeigt.

McMillan und Bishop [McM95] waren die ersten die ein bildbasiertes Visualisierungssystem mit Hilfe der plenoptischen Funktion beschrieben. Dort wird ein *plenoptisches Modell* aus Bildern einer Szene geformt – diese stellen "Abtastungen" der plenoptischen Funktion dar – die mit einer Kamera aufgenommen wurden, die um ihre vertikale Achse rotiert wird. Das Modell wird über eine zylindrische Parametrisierung repräsentiert, die einem zweidimensionalen Panoramabild entspricht. Durch zusätzliche Disparitätsinformation können beliebige Ansichten aus dem Inneren des Zylinders heraus berechnet werden.

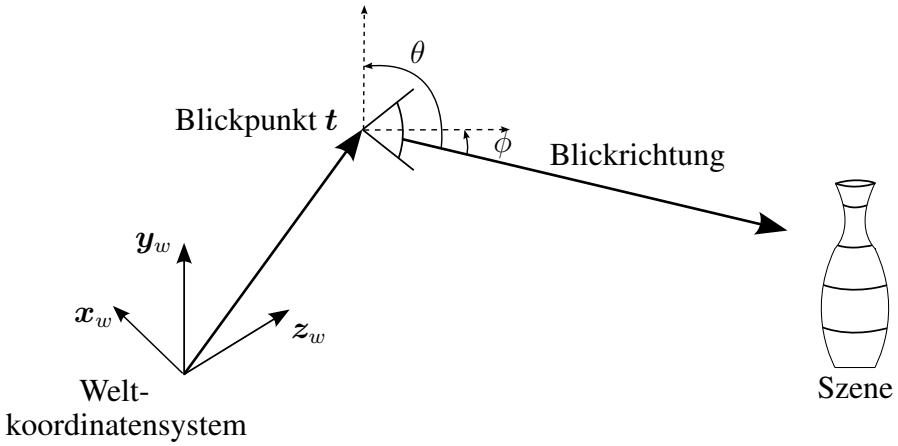



Bild B.4: Die plenoptische Funktion: ein einzelner Lichtstrahl wird von einem Beobachter mit Blickpunkt $\boldsymbol{t}$ aus der Richtung, die mit $\theta$ und $\phi$ beschrieben wird, gesehen. Wellenlänge $\lambda_L$ und Zeit $\tau$ sind nicht dargestellt.

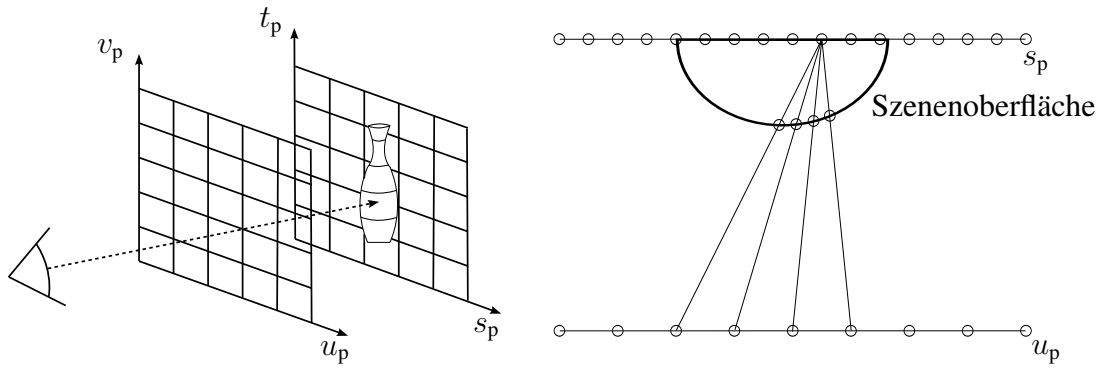
Das Lichtfeld reduziert den hochdimensionalen Raum der plenoptischen Funktion indem es einen anderen Satz an Restriktionen verwendet, der gleichwohl die Positionierung der aufnehmenden Kamera an beliebigen Orten erlaubt. Es wird angenommen, dass die beobachtete Szene über eine gewisse Zeit hinweg unverändert bleibt, und statt der Intensität für jede Wellenlänge wird nur ein Farbwert modelliert, wodurch zwei Parameter wegfallen. Zusätzlich wird die Luft zwischen dem Beobachter und der Szenenoberfläche als transparent angenommen, sodass die Intensität des Lichtstrahls, der von einem Oberflächenpunkt in eine Richtung ausgesandt wird, konstant bleibt, egal wo sich der Beobachter auf diesem Strahl befindet. Durch die Wahl einer passenden Parametrisierung wird die plenoptische Funktion so auf vier Parameter reduziert.

#### B.2.2.1 Lichtfeldmodelle und Visualisierungstechniken

Im Folgenden wird ein kurzer Überblick über die verschiedenen Parametrisierungen und Visualisierungstechniken für Lichtfelder gegeben. Die Techniken werden in zwei Hauptkategorien eingeteilt, Zwei-Ebenen-Lichtfelder und Freiform-Parametrisierungen. In einer dritten Kategorie werden weitere Typen von Lichtfeldern zusammengefasst.

**Zwei-Ebenen-Parametrisierungen**

Die Ansätze von Levoy et al. [Lev96] und Gortler et al. [Gor96] benutzen beide eine Zwei-Ebenen-Parametrisierung $L_{\mathrm{p}}(u_{\mathrm{p}}, v_{\mathrm{p}}, s_{\mathrm{p}}, t_{\mathrm{p}})$ zur Darstellung des Lichtfelds. Jeder Lichtstrahl der (reduzierten) plenoptischen Funktion wird hier durch je einen Punkt auf den Ebenen $(u_{\mathrm{p}}, v_{\mathrm{p}})$ und $(s_{\mathrm{p}}, t_{\mathrm{p}})$ beschrieben. Während in [Lev96] die Projektionszentren der Originalkameras in die




(a) (b)

Bild B.5: (a) Zwei-Ebenen-Parametrisierung des Lichtfelds: jeder Lichtstrahl wird mittels der vier Koordinaten $s_\mathrm{p}$, $t_\mathrm{p}$, $u_\mathrm{p}$ und $v_\mathrm{p}$ parametrisiert. (b) Ohne Tiefenkorrektur in der Ebene $(s_\mathrm{p}, t_\mathrm{p})$ werden nicht zusammengehörige Lichtstrahlen interpoliert.

Ebene $(u_\mathrm{p}, v_\mathrm{p})$ gelegt werden, die als *Kameraebene* bezeichnet wird, und die $(s_\mathrm{p}, t_\mathrm{p})$-Ebene die gemeinsame *Brennebene* dieser Kameras ist, wird in [Gor96] keine derartige Einschränkung gemacht. In letzterem werden die zwei Ebenen parallel zueinander gelegt, wobei eine Ebene z. B. in der Szenenobfläche und die andere näher am Beobachter liegt, wie etwa in Abbildung B.5(a) gezeigt.

Die Zwei-Ebenen-Parametrisierung vereinfacht die Erstellung synthetischer Ansichten aus dem Lichtfeld deutlich. Wie bereits vorher erwähnt ist das Lichtfeld nur dünn abgetastet wenn reale Eingabebilder verwendet werden, was bedeutet, dass es unwahrscheinlich ist, dass ein beliebiger Lichtstrahl vorher beobachtet wurde. Gortler et al. generieren eine Annäherung des idealen, kontinuierlichen Lichtfeldes $L_\mathrm{p}$ indem sie die zwei Ebenen in Gitter mit $M_{u,v} \times M_{u,v}$ und $N_{s,t} \times N_{s,t}$ Punkten unterteilen, wobei $x_{i,j,k,l}$ den Farbwert jedes Paares aus Gitterpunkten bezeichnet. $(i, j)$ bzw. $(k, l)$ indizieren die Gitterpunkte in den Gittern $(u_\mathrm{p}, v_\mathrm{p})$ bzw. $(s_\mathrm{p}, t_\mathrm{p})$. Eine Basisfunktion $B_{i,j,k,l}(u_\mathrm{p}, v_\mathrm{p}, s_\mathrm{p}, t_\mathrm{p})$ wird mit jedem Gitterpunktpaar verknüpft und dazu verwendet, den sogenannten *endlich-dimensionalen Lumigraphen* $\tilde{L}_\mathrm{p}$ zu rekonstruieren, welcher eine Annäherung des kontinuierlichen Lichtfeldes durch diskrete Werte mittels

$$\tilde{L}_\mathrm{p}(u_\mathrm{p}, v_\mathrm{p}, s_\mathrm{p}, t_\mathrm{p}) = \sum_{i=0}^{M_{u,v}} \sum_{j=0}^{M_{u,v}} \sum_{k=0}^{N_{s,t}} \sum_{l=0}^{N_{s,t}} x_{i,j,k,l} B_{i,j,k,l}(u_\mathrm{p}, v_\mathrm{p}, s_\mathrm{p}, t_\mathrm{p}) \tag{B.1}$$

darstellt. Verschiedene Basisfunktionen stehen zu Auswahl, wie etwa eine Rechteckfunktion, die den Wert $1$ um den zugehörigen Gitterpunkt herum besitzt, und ansonsten $0$, oder eine qua-

drilineare Funktion, welche die Interpolation benachbarter Gitterpunkte erlaubt und so zu einer kontinuierlichen Funktion $\tilde{L}_{\mathrm{p}}$ führt. Die Werte $x_{i,j,k,l}$ an den Gitterpunkten werden aus den Stichproben in $L_{\mathrm{p}}$, den Eingangsbildern, über die Duale der Basisfunktion berechnet. Eine Interpolation des nächsten Lichtstrahls im Lichtfeld zu einem durch $(u_{\mathrm{p}}, v_{\mathrm{p}}, s_{\mathrm{p}}, t_{\mathrm{p}})$ gegebenen, benötigten Strahl wird also auf einfache Weise aus $\tilde{L}_{\mathrm{p}}$ berechnet. Anstatt diese Interpolation aber Strahl für Strahl, also für jedes Pixel eines generierten Bildes, durchzuführen, können die Bildbereiche in jedem Gitterelement der $(u_{\mathrm{p}}, v_{\mathrm{p}})$-Ebene als Texturen gespeichert und die Interpolation auf der Grafikhardware für mehrere Pixel auf einmal durchgeführt werden.

Bis zu diesem Punkt sind die Ansätze von Levoy et al. und Gortler et al. sehr ähnlich. Sind die Positionen der aufnehmenden Kamera jedoch nicht nahe zusammen, können Tiefenänderungen in der Szene zu erheblicher Unschärfe der Ausgabebilder, den sogenannten *Geisterartefakten*, führen. Der Grund dafür ist in Abbildung B.5(b) dargestellt, nämlich dass, auch wenn Lichtstrahlen die $(s_{\mathrm{p}}, t_{\mathrm{p}})$-Ebene an demselben Punkt kreuzen, sie nicht den gleichen Punkt auf der Szenenoberfläche zeigen falls der Tiefenunterschied zu groß ist. Der Ansatz von Gortler et al. berücksichtigt Tiefeninformation und reduziert dadurch diese Artefakte.

Der Vorteil des Zwei-Ebenen-Ansatzes ist seine Effizienz in der Darstellung, der durch eine Reihe von nachfolgenden Veröffentlichungen weiter verfeinert wurde. Sloan et al. [Slo97] erhöhen die Darstellungsgeschwindigkeit zusätzlich durch eine verbesserte Auswahl der Knoten auf der $(u_{\mathrm{p}}, v_{\mathrm{p}})$-Ebene, jedoch auf Kosten der Qualität. Dasselbe Ziel wird von Schirmacher et al. [Sch00a] verfolgt, welche die Technik des sogenannten *image warpings* (“Bildverzerrung”) hinzunehmen um die Darstellung zu beschleunigen. Dieser Ansatz ist zwar nur anwendbar, falls dichte Tiefeninformation für jedes Bild zur Verfügung steht, kann dann aber die Anzahl der benötigten Stichproben deutlich reduzieren. Chai et al. [Cha00] untersuchten die Anzahl an Stichproben, die für die optimale Darstellung einer Ansicht aus einem Lichtfeld notwendig sind.

Ein wesentlicher Nachteil des Zwei-Ebenen-Ansatzes ist jedoch seine geringe Flexibilität hinsichtlich der Stichprobenbilder, da sie sich auf einem regulären Gitter mit gemeinsamer Brennebene befinden müssen. [Lev96] löst dieses Problem, indem eine spezielle Kamerahalterung verwendet wird, während in [Gor96] die Bilder verzerrt werden, um zu dem regulären Gitter zu passen, was aber zu einem Verlust an Qualität führt. Aus diesem Grund wurden neue Parametrisierungen entwickelt, die eine flexiblere Kamerapositionierung erlauben.

### Freiform-Lichtfelder

Die erste Lockerung der Zwei-Ebenen-Bedingung wurde von Heigl et al. [Hei99a] für Lichtfelder vorgeschlagen, die mit einer handgeführten Kamera aufgenommen wurden. Die Bilder der

aufnehmenden Kamera werden dabei auf die Bildebene der virtuellen Kamera abgebildet. Die beitragenden Kameras werden ausgewählt, indem ihre Kamerazentren auf die virtuelle Bildebene projiziert und die drei zum benötigten Sichtstrahl am nächsten liegenden Projektionen gewählt werden. Die virtuelle Bildebene wird in Dreiecke unterteilt und passende Bildbereiche der Originalkameras werden auf dieses abgebildet. Geometrieinformation wird mittels Tiefenkarten für jedes Bild eingebunden, wobei für jedes Dreieck eine Ebene approximiert wird.

Der Begriff *Freiform-Lichtfeld* wurde von Schirmacher et al. [Sch01b] für einen ähnlichen, aber effizienteren Ansatz als dem obigen eingeführt. Als Ausgleich für die gesteigerte Leistung müssen die Originalkameras hier jedoch eine konvexe Hülle um die Szene oder das Objekt herum bilden, wobei jedes Bild die komplette Silhouette der Szene einsehen kann.

In [Bue01] definieren Buehler et al. einen Satz aus acht Anforderungen an die Lichtfeldvisualisierung wie beispielsweise *Kontinuität* der Farbwerte, *Auflösungsempfindlichkeit* oder *unstrukturierte Eingabedaten*. Gleichzeitig werden frühere Parametrisierungen und Visualisierungstechniken wie die oben beschriebenen auf diese Anforderungen hin analysiert und verglichen. Die Autoren schlagen entsprechend einen Visualisierungsalgorithmus genannt *Unstrukturierter Lumigraph* vor, der diese Anforderungen berücksichtigt. Der Unstrukturierte Lumigraph stellt daher den allgemeinsten Freiform-Lichtfeldrenderer dar, der derzeit verfügbar ist, und wird für alle weiterführenden Verfahren und Experimente in dieser Arbeit verwendet werden.

### Weitere Repräsentationen

Analog zum Zwei-Ebenen-Lichtfeld wurde noch eine Reihe weiterer Repräsentationen eingeführt, die beispielsweise eine sphärische Parametrisierung für eine oder beide Ebenen benutzen. Diese Ansätze haben allerdings nie eine ähnlich weitverbreitete Anwendung wie die Zwei-Ebenen-Parametrisierung gefunden. Der geneigte Leser wird daher auf die kurze Zusammenfassung dieser Ansätze in [Sch01b] verwiesen.

Ein weiterer Ansatz der Freiform-Lichtfeldvisualisierung ist das *punktbasierte Rendering*. Das Lichtfeld wird hierbei mit einer Menge von 3D-Punkten repräsentiert, die gerichtete Lichtremissionsinformation enthalten. Der Vorteil davon ist, dass anders als bei der Speicherung von Dreiecksnetzen als Geometrieinformation die Verbindungsinformation zwischen den Punkten weggelassen werden kann, was zu einem kleineren Modell führt. Der Nachteil ist, dass bei Nahaufnahmen und Verdeckungen Lücken zwischen den gezeichneten Punkten auftreten können. Punktbasiertes Rendering für Lichtfelder wird in [ES03b] und [Vog05] beschrieben.

Das *Oberflächenlichtfeld*, eingeführt von Wood et al. [Woo00], ist eine Parametrisierung die sich auf ein sehr genaues Geometriemodell des dargestellten Objekts stützt. Remissionsinfor-

mation für Gitterpunkte auf der Oberfläche des Modells wird in sogenannten *Lumisphären* gespeichert, welche die Lichtstrahlen repräsentieren, die von dem Gitterpunkt aus in jede beliebige Richtung abgegeben werden. Die Stärke dieses Ansatzes liegt in der realistischen Modellierung der Reflexionseigenschaften des Objekts, sodass sogar Glanzlichter sehr gut reproduzieren werden können.

Abgesehen von den oben genannten Lichtfeldmodellen und Visualisierungstechniken wurden noch eine Reihe weitere Beiträge zu Lichtfeldrepräsentationen veröffentlicht. Ein ausführlicher Überblick über diese Ansätze findet sich in [Vog05].

#### B.2.2.2 Bildaufnahme

Wie weiter oben beschrieben wurde, besteht die minimale Eingabe für ein Lichtfeld – abgesehen von Tiefeninformation – aus einer Menge an Bildern und den zugehörigen Kameraparametern. Diese umfassen die intrinsischen Parameter wie z. B. Brennweite, Pixelseitenverhältnis und andere Parameter, welche die internen Eigenschaften der Kamera beschreiben, und extrinsische Parameter, also ihre Position und Orientierung für jedes Bild, die üblicherweise als deren *Lage* bezeichnet werden. Die Kameraparameter werden im Detail in Abschnitt 2.1.1 vorgestellt. Im Allgemeinen gibt es zwei Möglichkeiten, diese Parameter zu bestimmen, entweder über einen entsprechenden Hardwareaufbau wie einen Roboterarm, oder ausschließlich aus den Bildern selbst. Dies gilt sowohl für die intrinsischen als auch für die extrinsischen Parameter.

Bleiben die intrinsischen Parameter während des gesamten Vorgangs der Bildaufnahme konstant, so wird ihre Schätzung meist mittels eines Kalibriermusters mit genau bekannten geometrischen Eigenschaften vorgenommen. Die Parameter können so unter Verwendung eines Standardkalibrieralgorithmus wie etwa [Tsa87] sehr genau berechnet werden. Kamerakalibrierung ohne Kalibriermuster wird *Selbstkalibrierung* genannt und ist dann nützlich, wenn sich die Kameraparameter während der Bildaufnahme oder zwischen zwei Aufnahmen verändern. Der derzeitige Stand der Technik auf diesem Gebiet wird ausführlicher in Abschnitt 1.2.3 vorgestellt.

Um die Lage der Kamera für jedes Bild zu erlangen, benutzen die verschiedenen, in diesem Abschnitt zusammengefassten Ansätze eine Reihe unterschiedlicher Techniken. Für das Zwei-Ebenen-Lichtfeld aus [Lev96] muss sich die Kamera für jedes Bild an bestimmten, diskreten Gitterpunkten auf einer Ebene befinden. Um dies zu erreichen wird ein Aufbau eingesetzt, mit dessen Hilfe die Kamera auf einer Ebene bewegt werden kann, während die Kamera selbst mit einer dreh- und neigbaren Aufhängung ausgerüstet wurde, mit der das Objekt im Sichtfeld gehalten werden kann. Um einen kompletten Rundumflug aufzunehmen wurde das Objekt auf einen Drehteller gesetzt und jeweils um $90$ Grad gedreht, sodass Lichtfelder aus vier Richtungen ent-

stehen. Eine ähnliche, aber ausgefeiltere Kameraaufhängung wird in [Mat02] benutzt um spekulare und transparente Objekte aufzunehmen. Eine vertikale Reihe aus sechs Kameras wird über einem Drehteller aufgehängt, während eine zusätzliche, rotierende Reihe Lichtquellen für eine wechselnde Beleuchtung sorgt. Zusätzlich dazu liefern Monitore unter und neben dem Objekt sogenannte *environment mattes* [Zon99]. Die Darstellung derartiger Objekte erfolgt über einen punktbasierten Visualisierungsansatz [Zwi01].

Ein weiterer, noch flexibler hardware-basierter Ansatz für die Bildaufnahme ist der Einsatz eines Roboterarms. Dabei ist die Lage der Kamera auf der Spitze des Arms über die Rotationen der mechanischen Gelenke bekannt. Im medizinischen Umfeld werden Roboterarme beispielsweise zur Aufnahme von Lichtfeldern während endoskopischer Operationen eingesetzt [Vog04, Vog06].

Anstatt einer oder mehrerer beweglicher Kameras stellt die *Lichtfeldvideokamera* [Wil02, Wil05] eine Multikameramatrix aus bis zu $100$ Kameras dar, die in der Lage ist, ihre Bilder gleichzeitig aufzunehmen. Die Herausforderung bei diesem Ansatz ist die exakte Synchronisation einer derart großen Anzahl an Kameras. Obwohl als Nachteil bei diesem Aufbau die Anzahl der Bilder im resultierenden Lichtfeld auf die Anzahl der physikalischen Kameras beschränkt ist, ist sie sehr gut für die Erstellung *dynamischer* Lichtfelder geeignet, und wird daher nochmals im Detail in Abschnitt 1.2.5 behandelt. Der gleiche Ansatz, jedoch in mikroskopischem Maßstab, wurde für die *plenoptische Kamera* eingesetzt [Ng05]. Hier wurde eine Matrix aus $296 \times 296$ Mikrolinsen zwischen der Hauptlinse und dem optischen Sensor einer handgeführten Kamera eingefügt, die dann wie vielfache einzelne Kameras agieren. Abgesehen von der Möglichkeit, den Beobachter in der Makrophotographie verschieben zu können, erlaubt diese Technik das Neufokussieren des Bildes während der Nachbearbeitung.

Ein weniger aufwendiger Ansatz zu Berechnung der Lage der Kamera ist der Einsatz eines Kalibriermusters oder bestimmter Marker, die während der gesamten Bildsequenz sichtbar sind. Diese Methode wurde für die Bildaufnahme für den ersten Lumigraphen gewählt [Gor96]. Die hier visualisierten Objekte wurden vor und auf eine Oberfläche mit speziellen Markern gestellt, deren Positionen relativ zueinander bekannt waren. Auch hier wurde der Algorithmus von Tsai [Tsa87] eingesetzt, um die Lage der Kamera zu bestimmen. Ähnlich dazu wird in [Sha02] die Lage der Kamera über zwei Ebenen berechnet, die jeweils vier Marker tragen, wobei sich eine Ebene hinter dem Objekt befindet und die andere transparente Ebene davor.

Vor allem in medizinischen Anwendungen werden optische Trackingsysteme eingesetzt, die auf dem gleichen Prinzip wie Kalibrierung mit bekannten Markern basieren [Sal01]. Für Lichtfelder aus endoskopischen Bildern wie in [Vog06] wird die Kamera mit einem “Target” ausge-

stattet, das von einer Stereokamera aufgenommen wird. Dadurch kann die Position der Kamera recht genau berechnet werden. Diese Technik benötigt jedoch die Berechnung einer sogenannten Hand-Auge-Transformation zwischen dem Target und der Endoskopspitze [Tsa89, Sch04a].

Alle bisher vorgestellten Bildaufnahmetechniken umfassten zusätzliche Hardware abgesehen von der Kamera selbst. Es ist jedoch möglich, die benötigte Information ausschließlich aus der Bildsequenz zu beziehen, sodass keine zusätzliche Hardware notwendig ist. Die Techniken, die hierfür eingesetzt werden, sind Selbstkalibrierung für die intrinsischen Parameter der Kamera und Struktur-aus-Bewegung für die extrinsischen Parameter. Ansätze, die ein Lichtfeld mit Hilfe dieser Techniken aufbauen und daher nur eine einzige, handgeführte Kamera benötigen, sind in [Koc99a, Hei99a, Bue01, Hei04] beschrieben. Da die vorliegende Arbeit auf dem gleichen Konzept beruht, nämlich der Verwendung nur einer handgeführten Kamera für die Lichtfeldrekonstruktion, wird der Stand der Technik auf diesem Gebiet im nächsten Abschnitt umrissen. Trotzdem sollte ein letztes Lichtfeldaufnahmesystem, eingeführt in [Koc02], hier erwähnt werden. Es stellt eine Zwischenlösung zwischen hardware-basierten und Ein-Kamera-Methoden dar, da es einen handgeführten Multi-Kamera-Aufbau verwendet, der aus zwei oder mehr Kameras besteht, deren Lageparameter mittels Struktur-aus-Bewegung berechnet werden.

#### B.2.2.3 Schätzung von Kameraparametern

Der erste Schritt zur Schätzung von Kameraparametern aus aufgenommenen Bildern ist die Erkennung von irgendwie gearteten Merkmalen in den Bildern, und die Erstellung von Korrespondenzen zwischen den Merkmalen in unterschiedlichen Bildern. Diese Merkmale können unterschiedlicher Art sein, wobei die häufigsten Punkte oder Linien sind. Der folgende Abschnitt fasst herkömmliche Ansätze hinsichtlich Punktmerkmalen zusammen, da sie die Grundlage für diese Arbeit darstellen. Die Übersicht über die Kameraparameterschätzung selbst ist unterteilt in Ansätze, die verschiedene Faktorisierungsmethoden verwenden, Selbstkalibrierungsmethoden für die Schätzung intrinsischer Kameraparameter und weiterführende Techniken die beispielsweise lange Bildsequenzen behandeln, die nicht von den faktorisierungsbasierten Ansätzen abgedeckt werden.

**Punktmerkmalsdetektion und -verfolgung**

Die Algorithmen zur Merkmalsdetektion, die in dieser Arbeit betrachtet werden, sind im Allgemeinen intensitätsbasiert, was bedeutet, dass sie einen Gewichtungsoperator definieren der die Intensitätswerte in einem – üblicherweise rechteckigen – Fenster des Bildes betrachtet. Die bekanntesten, und immer noch am weitesten verbreiteten Gewichtungsoperatoren sind diejenigen

von Moravec [Mor77], Förstner [För87] und Harris [Har88]. Alle diese Ansätze basieren auf der Autokorrelationsfunktion, während jedoch der erste nur die Ähnlichkeit eines Fensters mit seiner Nachbarschaft betrachtet, definieren die anderen eine Matrix, die darauf hinweist, ob ein Fenster homogene Regionen, eine Kante oder, idealerweise, eine Ecke enthält. Eine Übersicht und Bewertung dieser und vieler anderer Gewichtungsoperatoren findet sich in [Sch00b]. In den vergangenen Jahren haben die sogenannten SIFT-Merkmale [Low99] eine deutliche Popularität erreicht. Diese rotations- und skalierungsinvarianten Merkmalsvektoren werden üblicherweise in der Objekterkennung und -lokalisierung eingesetzt, sind aber auch auf die Merkmalsverfolgung anwendbar. Die von Tomasi und Kanade [Tom91] explizit für die Aufgabe der Verfolgung definierten Merkmale sind inzwischen jedoch ein gängiger Standard. Hierbei sind die besten Merkmale diejenigen, die signifikante, senkrecht aufeinander stehende Gradienten in ihren Fenstern enthalten, wie beispielsweise Ecken.

Wurden interessante Punktmerkmale mit Hilfe eines der obigen Algorithmen gefunden, so ist der nächste Schritt das Wiederfinden jedes Merkmals in einem anderen Bild oder das Abgleichen von zwei Merkmalsmengen, die in zwei Bildern gefunden wurden. Letzteres wird oft von Hand durchgeführt, falls nur wenige Bilder mit wenigen Merkmalen betrachtet werden. Dies geschieht üblicherweise zu Demonstrationszwecken, wie beispielsweise in [Men99]. Diese Methode kann automatisiert werden, falls Abstandsmaße zwischen Merkmalen definiert und die ähnlichsten miteinander identifiziert werden, wie in [Deu04] für SIFT-Merkmale [Low99] geschehen. Diese Abgleichtechnik funktioniert allerdings nur dann gut, wenn genügend korrespondierende Merkmale in beiden Bildern gefunden werden.

Die Alternative dazu, das Wiederfinden eines Merkmals im folgenden Bild, das einmal detektiert wurde, ist mittlerweile weit verbreitet, besonders für Bildsequenzen und große Mengen an Bildern. Die Grundlage der meisten dieser Verfolgungstechniken ist der sogenannte Lucas-Kanade-Tracker [Luc81]. Ursprünglich für die Bildregistrierung gedacht, benutzt er eine Minimierung mittels Gradientenabstiegs, um die beste Übereinstimmung für jedes einzelne Merkmal im zweiten Bild zu finden. Der zu minimierende Fehler ist die Differenz zwischen den Intensitätswerten in einem Fenster um das Merkmal herum in beiden Bildern. Diese Methode wurde in [Tom91] aufgegriffen und von Shi und Tomasi [Shi94] erweitert, um nicht nur translatorische Bewegung, sondern auch eine affine Deformationsmatrix für das Merkmalsfenster zu schätzen.

Der Algorithmus erfuhr im Folgenden viele weitere Verbesserung und Modifikationen. Eine Methode zur Beleuchtungskompensation, die von Hager und Belhumeur [Hag98] für die Objektverfolgung eingeführt worden war, wurde in [Jin01] zusammen mit einem Verfahren zur Ausreißerdetektion dem Verfolger hinzugefügt. Das Problem des “Wegdriftens” von Merkmalen

vom Original über lange Bildsequenzen hinweg wurde in [Mat03] behandelt. Eine deutliche Beschleunigung wurde durch den sogenannten *Ansatz der inversen Verknüpfung* zur Aktualisierung der Bewegungsparameter erreicht [Bak01, Bak04]. Diese Verbesserungen wurden schließlich in dem Echtzeitverfolgungssystem von Zinßer et al. [Zin04] kombiniert, welches in der vorliegenden Arbeit eingesetzt wird.

**Faktorisierungsbasierte Struktur-aus-Bewegung**

Der faktorisierungsbasierte Ansatz zu Struktur-aus-Bewegung wurde erstmals von Tomasi und Kanade in [Tom92] vorgestellt. Sein zugrunde liegendes Prinzip ist die Zerlegung einer sogenannten Messmatrix, die alle Punktkorrespondenzen in einer Menge von Bildern enthält, in die 3D-Struktur der Szene, also die zu den Bildmerkmalen gehörenden 3D-Punkte, und die Kamerabewegung. Eine detaillierte Einführung zu diesen Methoden wird in Abschnitt 2.2 gegeben.

Im Anschluss an die erste Veröffentlichung wurden viele Verbesserungen und Variationen vorgeschlagen. Abhängig vom erreichten Typ der Rekonstruktion (siehe Abschnitt 2.3.1) können diese Methoden in affine und projektive Faktorisierungen eingeteilt werden. Die ersteren beinhalten die Originalmethode [Tom92], die ein orthographisches Projektionsmodell voraussetzt, und die schwach- und paraperspektivischen Erweiterungen aus [Poe97], die auch in [Chr96] angewendet werden, um iterativ eine euklidische Rekonstruktion aus ihnen abzuleiten. Projektive Rekonstruktionen werden über den Ansatz von Sturm und Triggs [Stu96] geschätzt, eine alternativer Algorithmus wurde von Oliensis [Oli01] aufgezeigt. Ein rekursiver Ansatz zur projektiven Rekonstruktion basierend auf einer Faktorisierung wurde von Heyden et al. [Hey99b] vorgestellt. Er ist in der Lage, mehr und mehr Bilder zu integrieren, sobald sie zur Verfügung stehen. Eine ausführlichere Übersicht über Faktorisierungsmethoden wird in [Kan98] gegeben, darunter z. B. auch Methoden, die Linien- anstatt Punktmerkmalen verwenden.

In jüngerer Zeit wurden Faktorisierungsmethoden auch in einem probabilistischen Kontext betrachtet, entweder durch Gewichtung jedes Merkmals im Hinblick auf seine Zuverlässigkeit [Aan02, Aan03], oder mit Hilfe der Modellierung jedes Merkmals durch eine Kovarianzmatrix [Ana02]. Besonders falls Informationen über die Verlsslichkeit aus der Punktverfolgung heraus zur Verfügung stehen, sind diese Methoden in der Lage, die Robustheit einer Rekonstruktion aus verrauschten Daten zu erhöhen.

**Selbstkalibrierung**

Eine gemeinsame Einschränkung der oben zusammengefassten Faktorisierungsmethoden ist, dass sie nur in der Lage sind, die Bewegung der Kamera zu erfassen, nicht aber deren intrinsi-

sche Parameter wie z. B. ihre Brennweite. Die Techniken die zu diesem Zweck eingesetzt werden heißen *Selbstkalibrierung* oder auch *Autokalibrierung*.

Die erste Lösung des Selbstkalibrierungsproblems wurde von Faugeras et al in [Fau92] unter Verwendung der *Kruppa-Gleichungen* angegeben. Diese quadratischen Gleichungen werden aus der Fundamentalmatrix nur zweier Ansichten berechnet. Obwohl der Ansatz im Folgenden verfeinert wurde, z. B. in [Zel96], wird er üblicherweise nicht angewendet, da er im Allgemeinen als zu rauschempfindlich angesehen wird und es sich zeigte, dass er in mehr Situationen fehlschlägt als andere Ansätze [Stu00].

Ein weiterer Ansatz, der ebenfalls die Eigenschaften der Fundamentalmatrix zweier Ansichten ausnutzt, ist in [Men99] beschrieben und wurde in [Fus01] untersucht. Anders als die Kruppa-Gleichungen berechnet er die intrinsischen Parameter von mindestens drei Bildern über die iterative Minimierung einer Fehlerfunktion. Da dies für verrauschte Daten leicht in einem lokalen Minimum endet, wurden mehrere Verbesserungen mittels globaler Optimierung vorgeschlagen [Rot02, Ben03].

Hauptsächlich auf einer Minimierung des Rückprojektionsfehlers der rekonstruierten 3D-Punkte und Kameraparameter mittels der Levenberg-Marquardt-Optimierung basiert der als geschichtete Selbstkalibrierung bezeichnete Ansatz von Hartley [Har93]. Die Idee dabei ist die Erweiterung einer projektiven Rekonstruktion erst auf eine quasi-affine und dann auf eine metrische, während die intrinsischen Parameter als für jedes Bild gleich angenommen werden. Eine alternative Erweiterung, die kritische Kamerabewegungen in Betracht zieht wird in [Dem98] aufgezeigt.

Triggs verwendet die sogenannte *absolute Quadrik*, um die Erweiterungsmatrix von einer projektiven auf eine metrische Rekonstruktion zu bestimmen [Tri97]. Der Vorteil dieser Methode ist, dass die zu berechnenden intrinsischen Parameter nicht konstant für alle Bilder sein müssen, wie von Pollefeys gezeigt [Pol98, Pol99, Pol04]. In [Gib02] wird die Robustheit der Schätzung über eine dem RANSAC ähnliche Technik gesteigert. Theoretische Voraussetzungen für die Anzahl der bekannten Parameter werden von Heyden und Åström in [Hey97, Hey98, Hey99a] untersucht.

Leider ist die Selbstkalibrierung weiterhin sehr empfindlich gegenüber Rauschen. Daher schlagen viele der Autoren auf diesem Gebiet die Verwendung einer Technik namens *Bündelausgleich* vor, um die Ergebnisse der Selbstkalibrierung zu verfeinern. Bündelausgleich optimiert die 3D-Punkte und Kameraparameter für eine große Anzahl an Bildern gleichzeitig und stellt daher eine langsame aber robuste Technik dar, da sie ein extrem überbestimmtes Gleichungssystem löst. Zwar wurde es ursprünglich in der Photogrammetrie entwickelt [Sla80], wurde im Rechner-

sehen aber in [Har93] eingeführt. Eine Erweiterung, genannt *verschachtelter Bündelausgleich* [Sze98], wird in den meisten Fällen verwendet.

Da Selbstkalibrierung ein wichtiges, aber schwieriges Problem darstellt, wurden noch viele weitere Beiträge zu diesem Thema veröffentlicht. Ausführlichere Zusammenfassungen als der hier vorgestellte können in [Pol99, Fus00, Har03] gefunden werden.

**Rekonstruktion langer Bildsequenzen**

Bildbasierte Modellierung mit Lichtfeldern benötigt üblicherweise nicht nur einige wenige Bilder, sondern oft große Mengen mit mehreren hundert Bildern. Die oben beschriebenen Faktorisierungsmethoden teilen den bedeutenden Nachteil, dass sie die Sichtbarkeit aller Merkmalspunkte in allen Bildern fordern. Obwohl Methoden existieren, welche die Position fehlender Merkmale unter Berücksichtigung der Einschränkung auf den von den Merkmalen aufgespannten Unterraum schätzen [Tom92, Gue02], ist es wahrscheinlich, dass diese den Fehler der Rekonstruktion über Gebühr erhöhen, falls zu viele Merkmale ersetzt werden.

Ein alternativer, oft benutzter Ansatz ist es, mit einer projektiv verzerrten Rekonstruktion der Kameraparameter zu beginnen, indem die Fundamentalmatrix nacheinander für alle Bildpaare [Har93] oder der trifokale Tensor für alle Bildtripel [Bea96] geschätzt wird, und die resultierenden Teilsequenzrekonstruktionen verschmolzen werden. Da sich die für jeden trifokalen Tensor gemachten Fehler aufsummieren, schlagen Fitzgibbon und Zisserman [Fit98] vor, den Fehler gleichmäßig auf alle Kamerapositionen zu verteilen, falls kreisförmige Kamerabewegungen gegeben sind. Der Ansatz wurde weiter durch die Verwendung eines hierarchischen Verschmelzens verbessert, um das gleiche Ziel einer gleichmäßigen Fehlerverteilung zu erreichen, sowie durch unterschiedliche Bildauswahlverfahren [Nis00, Gib02, Sai03]. Das sogenannte *Weben eines Ansichtennetzes* von Koch et al. [Koc99a] erstellt eine 3D-Topologie der Kamerapositionen und ermöglicht so das Verschmelzen von Teilsequenzen in mehrere Richtungen.

Die Einschränkung der Faktorisierungsmethoden auf kurze Bildsequenzen wird von Heigl und Niemann in [Hei99b] dadurch vermieden, dass sie sukzessive Faktorisierungen kombinieren und so rekonstruierte, überlappende Teilsequenzen verschmelzen. Die Verwendung einer Faktorisierung als Initialisierung für die Schätzung der Kameraparameter der verbleibenden Bilder durch nicht-lineare Minimierung des Rückprojektionsfehlers wird in [Hei04] beschrieben.

#### B.2.2.4 Szenentiefe

Wie vorher erläutert benötigen dünn abgetastete Lichtfelder, und damit speziell Freiform-Lichtfelder aus handgeführten Kamerasequenzen, Informationen über die Tiefe der Szene für die kor-

rekte Visualisierung. Die Berechnung von Tiefe aus mehreren Bildern ist seit etwa 30 Jahren ein sehr aktives Forschungsgebiet innerhalb des Rechnersehens. Daher wird hier nur eine kurze Einführung in das Thema mit Blick auf die Lichtfeldmodellierung gegeben.

Abgesehen vom ursprünglichen Lichtfeld von Levoy verwenden alle bisher eingeführten Visualisierungsmethoden zusätzliche Tiefeninformation. In den meisten Fällen, aber besonders für Freiform-Lichtfelder, wird die Tiefeninformation meist in einer Tiefenkarte pro Eingabebild abgelegt. Im Zwei-Ebenen-Fall, wie etwa im ursprüglichen Lumigraphen, wird Tiefeninformation oft global in das Modell integriert, während der Lichtfeldansatz von Isaksen [Isa00] eine globale Brennebene verwendet, die dynamisch an die Szenentiefe angepasst werden kann. Der Unstrukturierte Lumigraph [Bue01] hingegen verwendet ein globales 3D-Netz, um die komplette Szenengeometrie anzunähern.

Es existiert eine Vielzahl von Möglichkeiten, um Tiefenkarten pro Bild aus einer Bildsequenz heraus zu berechnen. Falls, wie bei Heigl [Hei04] geschehen, eine 3D-Rekonstruktion der Szene aus den Merkmalskorrespondenzen berechnet wird, kann die resultierende 3D-Information verwendet werden, um eine dünne Tiefenkarte für jedes Bild abzuleiten. Sind die Kameraparameter bekannt, kann die Tiefe jedoch auch aus dem optischen Fluss in Stereobildpaaren oder mehreren Bildern berechnet werden. Hauptsächlich basierend auf dem Algorithmus von Horn und Schunck [Hor81] zur Berechnung des Optischen Flusses wurden in den folgenden Jahren viele verschiedene Techniken und Verbesserungen für den Stereofall vorgeschlagen. Umfassende Übersichten sind in [Bar94, Bea95] zu finden, wenn auch die Entwicklung in diesem Bereich nicht stillsteht, wie z. B. an den skalenraumbasierten Ansätzen von Alvarez et al. [Alv00, Alv02] zu sehen ist. Die Berechnung des optischen Flusses aus mehreren Bildern wurde erst kürzlich z. B. in [Ira99, Kan04] untersucht.

Ähnlich wie Heigl benutzen Kang und Szeliski [Kan96b] eine mittels Struktur-aus-Bewegung berechnete 3D-Rekonstruktion um ein globales 3D-Szenenmodell für zylindrische Panoramabilder zu erhalten. Für beliebige Kamerapositionen wurden mehrere Algorithmen vorgeschlagen, um Stereotiefenkarten zu einem globalen Modell zu verschmelzen [Hig94, Cur96, Koc99b]. Wird nur ein einzelnes Objekt betrachtet, das zuverlässig vom Hintergrund getrennt werden kann, so kann ein sogenannter *Form-aus-Silhouette*-Ansatz angewendet werden [Pot87, Den98]. Andernfalls generiert die Technik des *Space Carvings* [Kut99] iterativ einen Voxelraum, indem die Konsistenz der Oberflächenvoxel mit allen Bildern überprüft wird. Der Nachteil, der allen diesen Ansätzen gemein ist, liegt darin, dass sie auf eine sehr genaue Kalibrierung der Kameraparameter angewiesen sind, um *konsistente* globale Tiefeninformation zu generieren. Kleine Fehler für einzelne Bilder können sich bei Verwendung aller Bilder derart aufsummieren, dass der Algo-

rithmus fehlschlägt.

#### B.2.2.5 Dynamische Lichtfelder

Im Gegensatz zu statischen Lichtfeldern wurden bisher nur wenige Beiträge zu dem Thema dynamische Lichtfelder veröffentlicht. Da sich das Problemfeld drastisch erweitert, wenn eine weitere Dimension zum Parameterraum hinzugefügt wird, wurden bisher nur Spezialfälle behandelt. Im Hinblick auf die Visualisierung von Lichtfeldern sich bewegender Objekte waren Li et al. [Li98] die ersten, die eine Lösung vorschlugen, wenn auch nur für synthetische Datensätze. Das dynamische Lichtfeld wird hier aus einer Anzahl statischer Teillichtfelder zusammengesetzt, die sich frei im Raum bewegen und rotieren können.

Ein sehr direkter Ansatz für die Visualisierung dynamischer Lichtfelder realer Szenen ist das Generieren mehrerer statischer Lichtfelder unterschiedlicher Zeitschritte. In einem Beispiels wurde dies zuerst in [Bue01] für den Unstrukturierten Lumigraphen demonstriert. Die periodische Bewegung eines Beispielobjekts wird hier von Hand in mehrere Zeitschritte unterteilt. Der Ansatz der *dynamischen Texturen* [Dor03a] zielt darauf ab, das Aussehen von Phänomenen wie Wellen und Rauch zu modellieren. In [Dor03b] schlagen die Autoren vor, einen dynamischen Lumigraphen für die Modellierung dynamischer Texturen zu verwenden, wobei dieser Ansatz nicht implementiert wurde.

Die vorher erwähnte Lichtfeldvideokamera (siehe Abschnitt 1.2.2), die mehrere synchronisierte Kameras kombiniert, vereinfacht die Herstellung dynamischer Lichtfelder deutlich. Ein statisches Lichtfeld kann so, abhängig von der Bildwiederholfrequenz der verwendeten Kamera, mehrere Male pro Sekunde generiert werden, und das Lichtfeld kann wie ein 3D-Video betrachtet werden, bei dem der Blickwinkel beliebig gewählt werden kann [Gol02]. Ähnliche Kameraaufbauten werden in [Ooi01, Sch01a, Yan02] vorgestellt.

Im Fall, dass nur eine einzige Kamera für die Erfassung eines dynamischen Lichtfeldes benutzt wird, ist ein möglicher Ansatz, zuerst die bewegten Dinge oder Regionen in der Szene zu identifizieren. Für den Spezialfall starrer, sich bewegender Objekte in einer ansonsten statischen Szene wurden mehrere unterschiedliche Ansätze vorgeschlagen. Bei eingeschränkter Bewegung kann eine 3D-Rekonstruktion gleichzeitig für alle Objekte in der Szene bestimmt werden, so etwa für lineare Objektbewegung [Avi99, Han00, Sha01, Han03] oder Objektbewegung entlang eines Konikschnitts [Sha99]. Die 3D-Bewegung kann jedoch nur bis auf einen Skalierungsfaktor bestimmt werden. Für beliebige Objektbewegung wurden mehrere ähnliche Methoden vorgeschlagen um die unterschiedlichen Objekte in der Szene voneinander zu trennen [Cos98, Kan01a, Kan03, Vid04]. Eine 3D-Rekonstruktion wird dann unabhängig voneinander

für jedes Objekt erstellt. Die in Abschnitt 4.3 vorgestellten dynamischen Lichtfelder basieren auf diesen Methoden zur Trennung von Objekten.

### B.2.3 Beitrag dieser Arbeit

Die im vorliegenden Text entwickelten Methoden basieren auf der Arbeit von Heigl [Hei04] und verwenden dieselbe Implementierungsumgebung. Die hier gemachten grundlegenden Annahmen sind daher ähnlich: Unter Verwendung nur einer handgeführten Kamera wird ein bildbasiertes Modell einer Szene aus einer aufgenommenen Bildsequenz generiert. Die Rekonstruktion der Kameraparameter jedes Bildes erfolgt mittels eines punktmerkmalsbasierter Struktur-aus-Bewegung, was voraussetzt, dass sich die Kamera relativ langsam bewegt und sich so die Blickwinkel von Bild zu Bild nur wenig verändern, was die Punktverfolgung ermöglicht. Keine Annahmen werden über das Anwendungsgebiet der aufgenommenen Bilder gemacht, was bedeutet, dass beliebige Szenen verwendbar sein sollten, seien es nun Innenraumszenen oder solche im Freien, mit nur einem sichtbaren Objekt oder einer komplexen Umgebung. Die einzige Anforderung hier ist, dass die Szenenoberfläche eine gewisse Textur oder Struktur aufweist, die eine Ermittelung von Punktmerkmalen erlaubt. Die Beleuchtung wird als konstant angenommen, da Beleuchtungsänderungen während der Visualisierung nicht modelliert werden.

Eine wichtige Einschränkung in Heigls Arbeit wird hier aufgehoben, und zwar die Konstanz der Szene. Anstatt rein statischer Szenen werden nun auch dynamische Szenen in Betracht gezogen, wenn auch keine allgemeine Bewegung zugelassen wird, sondern einige Einschränkungen weiterhin angewendet werden: die Bewegung muss schrittweise, sich wiederholend oder starr sein.

Der Beitrag dieser Arbeit kann in drei unterschiedliche Aspekte unterteilt werden. Der erste stellt Verbesserungen und neue Ansätze für die Rekonstruktion statischer Lichtfeldmodelle dar, wohingegen der zweite Aspekt die Rekonstruktion und Modellierung dynamischer Szenen für die Lichtfeldvisualisierung darstellt, inklusive der Anpassung der Visualisierungstechniken. Der dritte Aspekt ist die Anwendung von Lichtfeldmodellen auf andere Gebiete der Mustererkennung sowie der Bild- oder Videoverarbeitung.

**Statische Lichtfelder.** Die Rekonstruktion von Lichtfeldern erfordert eine große Anzahl Verarbeitungsschritte und die Verbindung vieler verschiedener Techniken. Die in [Hei04] vorgeschlagene Verarbeitungskette wird so um neue Ansätze für die Punktverfolgung [Zin04], Faktorisierung [Chr96] und Tiefenkartenschätzung [Alv02] erweitert. Zusätzlich werden Ansätze zu probabilistischen [Kan01b] und robusten [Ham86] Modellierung von unterschiedlichen Aspekten

des Rekonstruktionsprozesses untersucht. Ein wichtiger Beitrag ist die Entkopplung der linearen Verarbeitungskette und die Untersuchung unterschiedlicher Möglichkeiten zur Rückführung von Information aus späteren Verarbeitungsschritten hin zu früheren Schritten. Beispiele sind die Verwendung von Kalibrierinformation, um die Punktverfolgung zu verfeinern, oder die Verwendung der abschließenden Bildsynthese für die Verbesserung der vorhandenen Tiefenkarten. Die Erweiterung bestehender Lichtfelder mit neuen Bildern ist Teil der Entkopplung des Rekonstruktionsprozesses. Schließlich wird auch eine Methode vorgestellt, um globale Szenengeometrie zu erzeugen, welche die Leistungsfähigkeit der Visualisierung deutlich erhöht.

In Anbetracht dieser Modifikationen wird eine Möglichkeit zur Evaluation der resultierenden Verbesserung benötigt. Es wird eine Methode vorgestellt, die eine quantitative Messung der Qualität der aus einem statischen Lichtfeld erzeugten Bilder ermöglicht und so eine Möglichkeit zum Vergleich unterschiedlicher Ansätze der Lichtfeldrekonstruktion bietet.

**Dynamische Lichtfelder.** Der hauptsächliche Beitrag dieser Arbeit in Bezug auf dynamische Lichtfelder ist, dass im Gegensatz zu den in Abschnitt 1.2.5 erwähnten Ansätzen nur eine Kamera für die Aufnahme von Bildern einer Szene benutzt wird, und das Lichtfeld mittels Ansätzen der Struktur-aus-Bewegung rekonstruiert wird. Die Einschränkung auf eine Kamera macht einige Bedingungen für die Szenenbewegung erforderlich. Abhängig von der Rekonstruktionsmethode sind diese Einschränkungen, dass die Szenenbewegung entweder nur schrittweise ist oder nur starre, sich bewegende Objekte und sich wiederholende Bewegung enthält. Falls schrittweise Bewegung vorhanden ist, kann die Rekonstruktion über herkömmliche Algorithmen der Struktur-aus-Bewegung erfolgen. Für starre, bewegte Objekte werden zusätzliche Segmentierungsmethoden [Kan03] angewendet. Zusätzlich zu den Rekonstruktionsmethoden werden die vorhandenen Visualisierungsalgorithmen erweitert, um die Visualisierung der dynamischen Lichtfelder zu ermöglichen.

**Anwendungen von Lichtfeldern.** Da Lichtfelder Modelle von Szenen oder einzelnen Objekten darstellen, ist es eine logische Konsequenz, sie in verschiedenen modellgetriebenen Anwendungen der Bildverarbeitung einzusetzen. Aussehensbasierte Objektlokalisierung, -erkennung und -verfolgung sind einige der Gebiete, in denen Lichtfelder erfolgreich eingesetzt wurden. Hier werden nun Ansätze vorgestellt, die entweder den Trainingsschritt eines Erkennungs- und Lokalisierungssystems vereinfachen oder bei denen Lichtfelder direkt als Objektmodelle eingesetzt werden. Ein anderes Gebiet, auf dem Lichtfelder nützliche Werkzeuge darstellen, ist die Erweiterte Realität, die in den vergangenen Jahren ein gesteigertes Interesse hervorgerufen hat.

Aus realen Bildern rekonstruierte Lichtfelder werden hier eingesetzt, um Szenen nicht nur um synthetische, sondern um reale Objekte zu erweitern.

### B.2.4 Überblick

Im Folgenden wird ein kurzer Überblick über den Inhalt dieser Arbeit gegeben. Nach der Einführung des Problemfeldes der Lichtfeldrekonstruktion in Kapitel 1 beschreibt Kapitel 2 die theoretischen Grundlagen der Projektionsmodelle, Kamerakalibrierung und Algorithmen der Struktur-aus-Bewegung wie etwa die Faktorisierungsmethode.

In Kapitel 3 wird der Rekonstruktionsprozess eines statischen Lichtfelds aus Bildsequenzen im Detail erklärt. Nach einer Einführung in den Aufbau des Rekonstruktionssystems werden die unterschiedlichen Verarbeitungsschritte beschrieben, angefangen mit der Punktverfolgung, der Faktorisierung von Teilsequenzen und der Erweiterung auf eine komplette Bildsequenz. Das Kapitel deckt zusätzlich die Themen der probabilistischen Ansätze und der Berechnung von Szenengeometrie ab. Die Unterschiedlichen Schritte werden schließlich in neuer Reihenfolge in sogenannten Informationsrückkopplungsschleifen zusammengesetzt.

Die Erweiterung der Ansätze zur statischen Lichtfeldrekonstruktion auf dynamische Szenen wird in Kapitel 4 untersucht. Nach der Definition der zusätzlichen Anforderungen dynamischer Lichtfelder werden verschiedene Ansätze zu ihrer Rekonstruktion aus Bildsequenzen vorgestellt. Das Kapitel beschäftigt sich auch mit den zusätzlichen Anforderungen, die für die Visualisierung dynamischer Lichtfelder entstehen.

Nach den theoretischen Beschreibungen stellt Kapitel 5 eine experimentelle Auswertung der statischen sowie der dynamischen Ansätze zur Lichtfeldrekonstruktion vor. Eine Methode für die quantitative Bewertung der Visualisierungsqualität wird eingeführt.

Im Anschluss an die Experimente beschreibt das nächste Kapitel mehrere Anwendungen von Lichtfeldern in Erweiterter Realität, Objekterkennung und -verfolgung, und dem medizinischen Umfeld. Die letztere zeigt ihre Nützlichkeit besonders in der Unterstützung von Chirurgen in der minimal-invasiven Medizin.

Das letzte Kapitel stellt schließlich eine Zusammenfassung der Arbeit und einen Ausblick auf mögliche zukünftige Erweiterungen und Verbesserungen des Rekonstruktionssystems dar.

## B.3 Zusammenfassung und Ausblick

Im folgenden Kapitel werden die Hauptaspekte dieser Arbeit und ihre Schlussfolgerungen in Abschnitt B.3.1 zusammengefasst. Abschnitt B.3.2 beschreibt weiterführende Arbeiten und bietet einige Möglichkeiten, auf den Ergebnissen der unterschiedlichen, in den vorangegangenen Kapiteln vorgestellten Methoden und Anwendungen aufzubauen.

### B.3.1 Zusammenfassung

Das Prinzip der bild-basierten Visualisierung und, als eine Implementierung davon, das des Lichtfelds wurde eingeführt, um eine Möglichkeit zur Verfügung zu stellen, reale Objekte und Szenen zu modellieren und sie in photorealistischer Qualität wiederzugeben. Während frühe Veröffentlichungen zu Lichtfeldern spezialisierte, oft stationäre Kameraaufhängungen verwenden, ergab sich bald der Wunsch, kommerzielle, handgeführte Kameras verwenden zu können, um die nötigen Eingabebilder auf einfache Weise aufzunehmen. Dies erfordert die Berechnung von Lageinformation der Kamera, sowie Wissen über die Geometrie der Szene.

In dieser Arbeit wurden unterschiedliche Methoden untersucht und erweitert, um diese Daten ausschließlich aus den Eingabebildern mit Hilfe von Punktmerkmalsverfolgung und (Selbst-)Kalibrierungsmethoden zu berechnen. Zwei unterschiedliche Arten von Lichtfeldern, statische und dynamische, wurden betrachtet, wobei letztere eine Erweiterung des ersten insofern darstellt, als es die hauptsächliche Einschränkung, die statische Natur der aufgenommenen Szene, auflöst.

Im Hinblick auf statische Lichtfelder können die eingeführten Methoden wiederum in solche, welche die Robustheit der eingesetzten Rekonstruktionsmethoden erhöhen, und solche, die auf die Erstellung einer global gültigen 3D-Rekonstruktion abzielen, unterteilt werden. Ein generelles Werkzeug zur Erreichung dieser Ziele ist das Rückführen von Information über den aktuellen Stand der Rekonstruktion in einer Schleife zu früheren Stufen des Prozesses, um so die Ergebnisse iterativ zu verfeinern. Um die Qualität der Rekonstruktion zu messen, wurde das Signal-Rausch-Verhältnis (“Peak Signal-to-Noise Ratio”, PSNR) für diese Anwendung eingeführt. Es wird neben dem Rückprojektionsfehler verwendet, der üblicherweise für diesen Zweck eingesetzt wird, aber nicht unbedingt die objektive Qualität eines Lichtfelds widerspiegelt.

Hinsichtlich der Robustheit der Rekonstruktion zeigte ein Vergleich zwischen SIFT-Merkmalen und dem oft verwendeten Kanade-Lucas-Tomasi (KLT) Merkmalsverfolgungsalgorithmus, dass eine Rekonstruktion mit Hilfe der SIFT-Merkmale gleichwertig zu den auf KLT-Merkmalen basierenden ist, und sogar deutlich besser ist, wenn die Kamerapositionen weit voneinander entfernt sind. Beispielsweise führte die Erhöhung des Abstands zwischen den Kamerapositionen in

einer üblichen Bildsequenz um den Faktor vier zu einer drastischen Eröhung des Translationsfehlers gegenüber den wahren Daten von $8,42\%$ auf $751\%$ für KLT-Punktverfolgung, während er für SIFT-Merkmale unverändert bei etwa $8\%$ blieb. Da die Berechnungskomplexität jedoch exponentiell ist, ist dies nur für kleine Bildmengen oder spezielle Anwendungen, wie das Aneinanderfügen zweier unabhängig voneinander aufgenommener Bildsequenzen, möglich. Bei der Anwendung auf eine Bildsequenz mit $200$ Bildern betrug die Zeit für die Punktverfolgung daher mehrere Stunden für SIFT-Merkmale, im Vergleich zu weniger als einer Minute bei der Verwendung der KLT-Punktverfolgung.

Das PSNR wurde zum ersten Mal verwendet, um eine automatische Entscheidung zwischen mehreren unterschiedlichen Rekonstruktionsmethoden zu ermöglichen. Es wurde auf die Wahl zwischen Faktorisierungsmethoden angewendet, die eine Teilsequenz verarbeiten, um die Rekonstruktion zu initialisieren. Dabei standen die paraperspektivische Faktorisierung, Christy und Horauds iterative Version sowie die perspektivische Faktorisierung mit zwei unterschiedlichen metrischen Updates zur Auswahl. Es wurde gezeigt, dass, während der Rückprojektionsfehler nicht immer eine klare Lösung vorgibt, das PSNR zuverlässig die fehlgeschlagenen Versuche aussortiert. Das Verfahren ist daher zu bevorzugen, falls es nötig ist, eine erfolgreiche Rekonstruktion ohne weitere manuelle Versuche zu erlangen.

Robustheit war auch die Motivation zur Anwendung von Ausreißerdetektion, Gewichtung einzelner Merkmale und robuster Schätzer auf die bildweise Schätzung von Kameraparametern und 3D-Punkten. Während für zwei der drei Testbildsequenzen die hohe Qualität der KLT-Merkmalskorrespondenzen ausreichend war, um eine zuverlässige Rekonstruktion zu ermöglichen, konnte gezeigt werden, dass der Lorentz'sche Schätzer ohne Ausnahme gute Ergebnisse liefert. Er war so den anderen getesteten Methoden überlegen, die nur in wenigen Fällen eine Verbesserung zeigten, jedoch für eine bedeutende Anzahl der untersuchten Parametereinstellungen bei der Rekonstruktion des Lichtfelds vollkommen fehlschlugen.

Die globale Gültigkeit eines statischen Lichtfelds zu erhöhen, wird aus mehreren Gründen angestrebt. Es verringert Visualisierungsartefakte, die auftreten, wenn sich Fehler während der Rekonstruktion einer langen Folge von Bildern aufsummieren. Außerdem reduziert es die Größe des Modells, besonders wenn, als ein Ergebnis davon, Szenengeometrie durch ein global gültiges 3D-Netz, genannt *globaler Proxy*, repräsentiert werden kann. Die wichtigsten, für diesen Zweck eingeführten Ansätze zielen daher darauf ab, die Anzahl der Bilder zu erhöhen, in denen das gleiche Merkmal gefunden wird. Dies wurde einerseits dadurch erreicht, dass ein Netz der Kamerapositionen generiert wurde, das zusätzliche Nachbarschaftsbeziehungen zwischen Bildern in einer Sequenz zur Verfügung stellt und während der Merkmalsverfolgung und Kamerakalibrie-

rung eingesetzt wurde. Diese Ansätze wurden *nicht-sequentielle* oder *Netz*-Punktverfolgung und Rekonstruktion genannt. Auf der anderen Seite wurde diese Beziehung durch Rückprojektion bekannter 3D-Punkte in das Bild erstellt, und die sich ergebenden Positionen als Initialisierung der Merkmalsverfolgung verwendet. Dieser zweite Ansatz wurde daher als *Merkmalsvorhersage durch Rückprojektion* bezeichnet.

Der Nutzen nicht-sequentieller Punktverfolgung und Rekonstruktion ist für Drehtellersequenzen am deutlichsten, wobei die Lagefehler der Kamera um mehr als $50$ Prozentpunkte auf $17,3\%$ bzw. $13,4\%$ für Translations- und Rotationsfehler verringert wurden und sich das PSNR für beide untersuchten Sequenzen um etwa $10$ dB erhöhte. Für von Hand aufgenommene Sequenzen ist der Erfolg nicht so deutlich, zeigt sich aber, wenn das PSNR für Referenzbilder berechnet wird, die weiter voneinander entfernt sind. Das gleiche, wenn auch in kleinerem Maßstab, gilt für die Merkmalsvorhersage durch Rückprojektion. Hier konnte die durchschnittliche Länge einer Merkmalsspur um einen Faktor von mehr als drei, von $20,4$ auf mehr als $60$ Bilder pro Merkmal bzw. von $10,9$ auf mehr als $30$ Bilder für die beiden getesteten Sequenzen, erhöht werden. Für den Fall der nicht-sequentiellen Punktverfolgung erhöhte sich die Spurlänge sogar um einen Faktor von $23$ für eine der Drehtellersequenzen.

Wenn also globale Gültigkeit erwünscht ist, sind beide Ansätze der sequentiellen Verarbeitung überlegen, wobei die nicht-sequenzielle Verfolgung und Rekonstruktion deutlich genauer ist, aber auch zwei komplette Iterationen benötigt, falls Netzinformationen nicht a priori vorhanden sind. Die Vorhersage über Rückprojektion sollte nur verwendet werden, falls eine Verarbeitung als kontinuierlicher Datenstrom notwendig ist. Hier bietet das einfache Auswahlverfahren den besten Kompromiss zwischen Verarbeitungszeit und Genauigkeit.

Ein weiterer Ansatz, der eine Erhöhung der globalen Konsistenz erlaubt, wurde aus der Robotik und der autonomen Kartierung von Umgebungen übernommen. Durch das Erzwingen des Schließens von Schleifen in der Kamerabewegung, falls aufsummierte Fehler eine korrekte Rekonstruktion verhinderten, kann eine global korrekte Konstellation erzeugt werden. Es wurde beispielhaft gezeigt, dass diese Methode für große Fehler in der Lage ist, die Lagefehler zu halbieren, von beispielsweise $71,3\%$ auf $26,1\%$ Translationsfehler, obgleich die verbesserte globale Korrektheit mit einem Anwachsen der lokalen Fehler bezahlt wird. Generell jedoch ist die Anwendung des Bündelausgleichs auf die besagten Schleifen der zu bevorzugende Ansatz.

Neben den Parametern der aufnehmenden Kamera benötigt ein Lichtfeld Tiefeninformation, um die virtuellen Ansichten korrekt visualisieren zu können. Ursprünglich wurde diese Information in Form von dichten, jeweils zu einem Eingabebild passenden Tiefenkarten zur Verfügung gestellt. Hier wurden nun zwei Alternativen, lokale und globale Proxys, diskutiert. Beide Ansätze

kodieren die Tiefeninformation in Form von Dreiecksnetzen, doch während der erste ein Netz für jedes Eingabebild benötigt, vereint der zweite alle Tiefeninformationen in einem global gültigen Netz. Während globale Proxys bereits als Teil des Unstrukturierten Lumigraphen benötigt wurden, stellen die lokalen Proxys einen Kompromiss zwischen diesen und den dichten Tiefenkarten dar. Sie sind leicht zu erstellen, benötigen weniger Speicher und sind in der Visualisierung effizienter als Tiefenkarten. Im Kontext der globalen Konsistenz jedoch wurde ein neuer Ansatz zur Erstellung eines globalen Proxys aus 2D-Merkmalskorrespondenzen eingeführt, und erste Ergebnisse wurden vorgestellt. Obwohl die Einschränkungen in seiner Konstruktion noch zu streng für eine universelle Einsetzbarkeit sind, ist das Visualisieren jedoch mehr als vier mal so schnell wie für Tiefenkarten, und verbessert sich von $3,3$ auf $15$ Bilder pro Sekunde für einen Proxy der sich auf das gesamte Ausgabebild auswirkt. Er ist damit auf gleichem Niveau wie die lokalen Proxys, wobei die Qualität nicht sinkt. Zusätzlich wurde eine Methode vorgestellt, mit der sich globale Proxys basierend auf einer nicht-linearen Optimierung der Visualisierungsqualität und unter erneuter Verwendung einer Informationsrückkopplungsschleife verbessern lassen.

Die Erweiterung eines bestehenden Lichtfeldes um zusätzliche Bildsequenzen kann schließlich ebenfalls der gleichen Kategorie der Steigerung der globalen Korrektheit zugewiesen werden. Zwei Methoden zum Auffinden des besten Bildes als Ausgangspunkt einer erneuten Merkmalsverfolgung, und so zur Integration einer neuen Sequenz, wurden beschrieben. Dabei war der Abgleich über SIFT-Merkmale für die Vorhersage der Merkmalspositionen um $20$ bis $30$ Pixel genauer als die Anwendung der Rückkopplung aus der Visualisierung, basierend auf der adaptiven Zufallssuche bzw. dem Partikelfilter. Die Abweichung betrug dabei im Schnitt nur $2,0$ Pixel im Vergleich zu $31,6$ Pixeln.

Hinsichtlich des zweiten Hauptthemas dieser Arbeit, den dynamischen Lichtfeldern, wurde erklärt, dass die Lockerung der Einschränkung auf statische Szenen nicht allgemein sein kann. Es ist unmöglich, ein Lichtfeld mit nur einer Kamera dicht genug abzutasten, wenn der Parameterraum auf diese Weise um eine Dimension erhöht wurde. Daher mussten die hier betrachteten dynamischen Lichtfeldmodelle neue, aber weniger harte Einschränkungen als der statische Fall umfassen.

Drei unterschiedliche Modelle und ihre größtenteils automatische Rekonstruktion aus einer oder mehreren Bildsequenzen wurden vorgestellt. Das erste, das schrittweise statische Lichtfeld, setzt voraus, dass für mehrere Zeitschritte jeweils eine Bildsequenz aufgenommen wird, wobei die Szene während jeder Aufnahme unbewegt bleibt. Die Sequenzen werden in einem ersten Schritt einzeln rekonstruiert. In einem zweiten Schritt werden die Kamerapfade aneinander gehängt, und daraufhin wird das Modell mittels nicht-sequenzieller Punktverfolgung und

Rekonstruktion verfeinert. Während der Visualisierung des sich ergebenden Lichtfeldes ist anschließend ein Umschalten zwischen den Originalzeitschritten sowie die freie Navigation innerhalb jedes Zeitschritts möglich.

Der zweite Typ dynamischer Lichtfelder benötigt diesen aufwendigen Aufnahmeschritt des ersten Typs nicht, da er es ermöglicht, die dynamische Szene in einer einzigen Sequenz aufzunehmen. Er führt jedoch die Einschränkung ein, dass ein in der Szene sichtbares, bewegtes Objekt in sich selbst starr sein muss. Dies ermöglicht die automatische Segmentierung von Hintergrund und Vordergrund auf der Ebene der Punktmerkmale unter Verwendung der Unterraumtrennungsmethode von Kanatani et al. So können Hintergrund- und Objektgeometrie individuell rekonstruiert und anschließend in ein gemeinsames Koordinatensystem übertragen werden. Das Lichtfeld wird wiederum in unterschiedliche Zeitschritte unterteilt, wobei dies nun durch Gruppierung der Ansichten des bewegten Objekts mittels Vektorquantisierung geschieht. Der während der Visualisierung jeweils sichtbare Zeitschritt wird durch Ausmaskierung aller anderen Quellansichten unter Verwendung von Vertrauenskarten ausgewählt. So wird der gleiche Effekt wie zuvor erzielt, jedoch mit deutlich weniger Benutzerinteraktion.

Das endgültige Ziel ist jedoch die Bereitstellung einer kontinuierlichen Repräsentation des dynamischen Teils eines Lichtfeldes. Daher wurde eine erste Implementierung eines Objektlichtfelds präsentiert, das aus einem Lichtfeld pro Objekt in der Szene besteht. So ist es möglich, Hintergrund und Vordergrund unabhängig voneinander zu betrachten, aber auch zusammen im selben Kontext mit einer durch den Benutzer einstellbaren Transformation zwischen beiden. Der Ansatz benötigt jedoch weiterhin eine manuelle Nachbearbeitung der Vertrauenskarten.

Sowohl schrittweise statische Lichtfelder als auch Lichtfelder starrer, sich bewegender Objekte wurden jeweils anhand von drei Beispielen demonstriert. Für das Objektlichtfeld wurde ein Prototypmodell präsentiert.

Im vorangegangenen Kapitel wurden schließlich neue Anwendungen statischer und dynamischer Lichtfelder auf andere Gebiete gezeigt. Es wurde demonstriert, dass Lichtfelder gut als Objektmodelle in der Erweiterten Realität geeignet sind, da sie selbst eine Repräsentation realer Objekte darstellen. Die mögliche Diskrepanz zwischen Realität und Virtualität wird so vermieden. Die Lichtfeldrekonstruktion wurde weiterhin auf die klassischen Felder der Objektlokalisation und -verfolgung angewandt, entweder durch direkte Verwendung des Lichtfeldes für die Objektmodelle oder über die Anwendung der Methoden zu ihrer Rekonstruktion auf die Generierung der benötigten Modelle aus Sequenzen einer handgeführten Kamera.

Ein anderes und immer noch ungewöhnliches Anwendungsgebiet stellt die medizinische Visualisierung dar, besonders in der endoskopischen Chirurgie. Hier wurden Lichtfelder dazu

verwendet, Glanzlichter zu reduzieren, welche durch die Lichtquelle des Endoskops auf feuchtem Gewebe erzeugt werden. Zusätzlich wurde eine Anwendung des schrittweise statischen Lichtfeldmodells zur Visualisierung unterschiedlicher Abschnitte einer Operation in [Vog06] beschrieben, wodurch ihr Verlauf beobachtet und dokumentiert werden kann. Schließlich wurde eine Untersuchung der gesteigerten Ansprüche an das Rekonstruktionssystem bei Verwendung flexibler anstatt starrer Endoskope vorgestellt, und ein erstes Ergebnis demonstriert.

### B.3.2 Zukünftige Arbeit

Die vorgestellte Arbeit bietet eine Reihe von Ansatzpunkten für zukünftige Forschung. Ein erster solcher Ansatzpunkt ist der Kanade-Lucas-Tomasi-Punktverfolgungsalgorithmus. Wie bereits in Abschnitt 6.3.3 für die Anwendung auf Bilder, die über ein flexibles Endoskop aufgenommen wurden, beobachtet wurde, ist die derzeit auf die Merkmalsfenster angewendete, affine Verzerrungskorrektur nicht in allen Fällen ausreichend. Falls die Oberflächen, auf denen die Merkmale gefunden werden, hauptsächlich senkrecht zur Blickrichtung der Kamera stehen, ist eine affine Verzerrungsmatrix ausreichend. Befinden sich die Merkmale jedoch auf Oberflächen parallel zur Blickrichtung und bewegt sich die Kamera vorwärts oder rückwärts, wie beispielsweise durch einen Korridor, so ist die entstehende Verzerrung der Merkmalsfenster perspektivisch, und daher schlecht durch eine affine Verzerrungsmatrix anzunähern.

Die Modellierung perspektivischer Merkmalsverzerrung ist äquivalent zur Schätzung einer Homographie mit acht Freiheitsgraden für jedes Merkmal. Diese acht Parameter auf einmal zu bestimmen, mag jedoch ein beträchtliches Problem für die Stabilität der Schätzung darstellen. Eine gute Initialisierung und Einschränkung der zusätzlichen Translation ist daher unabdingbar.

Als nächstes ist im Hinblick auf die folgenden Rekonstruktionsverfahren die Auswahl einer geeigneten Faktorisierungsmethode unter Verwendung des PSNR nur eine Möglichkeit zur Modifikation des Verfahrens basierend auf der Visualisierungsqualität. Der Ansatz kann genauso dazu verwendet werden, die optimalen Parameter für Punktverfolgung und Rekonstruktion zu bestimmen, wie beispielsweise die Anzahl der zu verfolgenden Merkmale oder den maximalen Rückprojektionsfehler. Eine weitere Anwendung wäre die Beobachtung des Fortschritts der Rekonstruktion mit Hilfe des PSNR, um auf diese Weise ein Fehlschlagen der Kalibrierung der aktuellen Kameralage zu erkennen.

Ein hauptsächliches Problem für die global konsistente Rekonstruktion trat in den Experimenten in Kapitel 5 deutlich zum Vorschein. Der bildweise Fehler steigt an, selbst wenn die globale Konsistenz verbessert wird, wodurch im Allgemeinen auch die Visualisierungsqualität geschmälert wird. Eine Lösung bietet der Bündelausgleich, der alle 3D-Punkte und Kamerapa-

rameter gleichzeitig verfeinert. Er benötigt jedoch auch beträchtliche Zeit für diese Aufgabe. Einen Ausweg zeigen hier effiziente Bündelausgleichsstrategien wie die in [Shu99] vorgeschlagene. Mehrere Bilder werden hier zu Segmenten zusammengefasst und virtuelle Schlüsselframes für jedes Segment definiert, die als Ersatz für das gesamte Segment optimiert werden und so die Gesamtanzahl der Parameter reduzieren.

Ein weiterer, vielversprechender Ausgangspunkt für weitere Arbeiten ist der globale Proxy, sowohl hinsichtlich seiner Erstellung als auch seiner Optimierung. Die erfolgreiche Anwendung der Bilddifferenz als Fehlermaß zur Optimierung wurde bereits in Abschnitt 3.7.3 beschrieben. Sie kann jedoch genauso auf die Erstellung des globalen Proxys angewendet werden, um Knoten zu identifizieren, die falsch eingefügt wurden. Aber nicht nur die nahtlose Integration eines neuen 3D-Punktes ist für die Visualisierungsqualität von Bedeutung. Eine Menge aus vier nichtkollinearen Punkten kann auf zwei verschiedene Arten mit Hilfe der Delaunay-Triangulation vernetzt werden. Die gemeinsame, interne Kante der sich ergebenden Dreiecke kann so – im schlimmsten Fall – senkrecht zu einer 3D-Kante in der Szene sein. Morris und Kanade zeigten in [Mor00], dass durch das Umdrehen dieser Art von Kanten und die Berechnung eines Fehlers basierend auf der Projektion eines Bildes auf das Netz die Korrektheit des Netzes verbessert werden kann. Diese Tatsache wurde bisher weder in der Erstellung noch in der Optimierung der globalen Proxys berücksichtigt.

Die Optimierung globaler Proxys bietet Raum für weitere Verfeinerungen. Derzeit werden Bilder kompletter Größe gezeichnet, um die Veränderung des Fehlers für jeden Knoten des Netzes zu berechnen. Jedoch ist nur ein geringer Teil des Bildes von Änderungen eines Knotens betroffen, nämlich die benachbarten Dreiecke. Durch Einschränkung der Visualisierung auf diese Bereiche kann eine deutliche Beschleunigung erreicht werden. Darüber hinaus verändert der Algorithmus die Anzahl der Knoten im Netz nicht. Neue Knoten hinzuzufügen kann das Ergebnis weiter verbessern, falls dies in Gebieten mit großem Fehler angewendet wird.

Noch mehr als die statische Rekonstruktion bietet die Erstellung dynamischer Lichtfelder eine große Anzahl unterschiedlicher Richtungen für zukünftige Untersuchungen. Ein wichtiges Problem für Lichtfelder mit starrer Bewegung sowie Objektlichtfelder ist die Bestimmung der Skalierung zwischen Rekonstruktionen unterschiedlicher Bereiche der Szene. Dieses Problem ist jedoch nicht leicht zu lösen. Eine mögliche Lösung bietet wiederum ein bildbasierter Ansatz. Falsch skaliert und von einem anderen Winkel aus betrachtet wird das bewegte Objekt in einer falschen Position relativ zum Hintergrund dargestellt. Der Effekt wäre in Bilddifferenzen sichtbar, falls die in dieser Arbeit benutzte Methode des Weglassens des Originalbildes angewandt würde.

Das Problem der exakten Segmentierung des bewegten Objekts in jedem Bild könnte ähnlich gelöst werden. Bei der alleinigen Visualisierung eines Lichtfeldes des Objekts ist die Qualität innerhalb der Grenzen des Objekts hoch, vorausgesetzt, die Kameralagen- und Tiefenrekonstruktionen sind korrekt. Der umgebende Hintergrund wird andererseits aus vielen unterschiedlichen, falsch positionierten und sich daher überlappenden Teilen der Originalbilder zusammengesetzt. Die Objektgrenze stellt sich so als ein deutlicher Unterschied zwischen hoher und niedriger Qualität, ausgedrückt als Bilddifferenz, dar, und kann beispielsweise durch ein aktives Konturmodell wie Snakes [Kas88] angenähert werden. Diese Methode ist natürlich zum Scheitern verurteilt, falls der Hintergrund größtenteils homogen ist.

Die Segmentierung von Vorder- und Hintergrund wurde bisher nur für ein bewegtes Objekt vor einem statischen Hintergrund demonstriert. Nichtsdestotrotz ist es theoretisch möglich, eine beliebige Anzahl Objekte zu trennen, vorausgesetzt dass genügend Merkmale auf jedem gefunden werden. Weder Kamerakalibrierung und 3D-Rekonstruktion noch die nachfolgende Partitionierung in Zeitschritte stellen ein Hindernis dar, nur die Anzahl möglicher Kombinationen sichtbarer und unsichtbarer Objekte erhöht sich exponentiell mit der Anzahl der Objekte. Dies stellt ein Problem für die Visualisierung dar, da für jede Kombination ein Lichtfeld generiert wird, was zu erheblichen Speicheranforderung und wenig intuitiver Benutzerinteraktion führt. Diese Probleme treten für Objektlichtfelder nicht auf.

In denselben Kontext der mehrfachen Objekte fällt das nächste zu behandelnde Problem, nämlich verdeckte Objekte und solche, die den Sichtbereich der Kamera verlassen, oder sogar die weitere Verfolgung falls die Bewegung nicht fortgesetzt wird. In diesem Fall ist es nicht mehr möglich, sich auf den Segmentierungsalgorithmus zu verlassen. Um diese Gegebenheiten korrekt behandeln zu können, müssen zusätzliche Methoden für die Objektverfolgung und eine verbesserte Objektverwaltung integriert werden.

Angefangen bei den ersten Zwei-Ebenen-Lichtfeldern wurden Einschränkungen für die beobachtete Szene, sowohl in der Lichtfeldrekonstruktion als auch in der Visualisierung, Schritt für Schritt rückgängig gemacht. Die automatische Generierung dynamischer Lichtfelder mit starren, bewegten Objekten wird nicht das Ende dieser Entwicklung darstellen. Es existieren Algorithmen, die in der Lage sind, die 3D-Geometrie einer Szene trotz der Anwesenheit sich bewegender und deformierender Objekte zu rekonstruieren. Ein solcher Algorithmus ist z. B. in [Har03] skizziert. Die Schwierigkeiten dabei sind das Auffinden einer stabilen Lösung für lange Bildsequenzen und die Definition eines geeigneten Modells für die Visualisierung des resultierenden Lichtfeldes. Es sollte interessant sein, die Entwicklung in diesem komplexen und aufregenden Forschungsgebiet in den kommenden Jahren zu verfolgen, besonders da neue Anwendungen wie

in der Medizin gerade erst am Entstehen sind.

# Bibliography


[Aan02] Henrik Aanæs, Rune Fisker, Kalle Åström, and Jens Michael Carstensen. Robust factorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(9):1215–1225, September 2002.

[Aan03] Henrik Aanæs. *Methods for Structure from Motion*. PhD thesis, Informatics and Mathematical Modelling, Technical University of Denmark, Kgs. Lyngby, Denmark, September 2003.

[Ade91] Edward H. Adelson and James R. Bergen. *Computational Models of Visual Processing*, chapter 1 (The Plenoptic Function and the Elements of Early Vision). MIT Press, Cambridge, MA, 1991.

[Alo90] John Y. Aloimonos. Perspective approximations. *Image and Vision Computing*, 8(3):177–192, August 1990.

[Alv00] Luis Alvarez, Joachim Weickert, and Javier Sánchez. Reliable estimation of dense optical flow fields with large displacements. *International Journal of Computer Vision*, 39(1):41–56, August 2000.

[Alv02] Luis Alvarez, Rachid Deriche, Javier Sánchez, and Joachim Weickert. Dense disparity map estimation respecting image derivatives: a PDE and scale-space based approach. *Journal of Visual Communication and Image Representation*, 13(1/2):3–21, March/June 2002.

[Ana02] Padmanabhan Anandan and Michal Irani. Factorization with uncertainty. *International Journal of Computer Vision*, 49(2/3):101–116, September/October 2002.

[Avi99] Shai Avidan and Amnon Shashua. Trajectory triangulation of lines: Reconstruction of a 3D point moving along a line from a monocular image sequence. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2,

pages 2062–2066, Fort Collins, CO, USA, June 1999. IEEE Computer Society Press, Washington.

[Bak01] Simon Baker and Iain Matthews. Equivalence and efficiency of image alignment algorithms. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 1090–1097, Kauai, HI, USA, December 2001. IEEE Computer Society Press, Washington.

[Bak04] Simon Baker and Iain Matthews. Lucas-Kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, 56(3):221–255, March 2004.

[Bar94] John L. Barron, Steven S. Beauchemin, and David J. Fleet. On optical flow. In *Proceedings of the Sixth International Conference on Artificial Intelligence and Information-control Systems of Robots*, pages 3–14, Bratislava, Slovakia, September 1994. World Scientific Publishing Co., Inc., River Edge, NJ.

[Bea95] Steven S. Beauchemin and John L. Barron. The computation of optical flow. *ACM Computing Surveys (CSUR)*, 27(3):433–467, September 1995.

[Bea96] Paul A. Beardsley, Philip Torr, and Andrew Zisserman. 3D model acquisition from extended image sequences. In *Computer Vision – ECCV '96. 4th European Conference on Computer Vision*, volume 2, pages 683–695, Cambridge, UK, April 1996. Springer-Verlag, London.

[Bei92] Thaddeus Beier and Shawn Neely. Feature-based image metamorphosis. In *Proceedings of SIGGRAPH '92*, volume 26, pages 35–42, Chicago, IL, USA, July 1992. ACM Press, New York.

[Ben03] Arrigo Benedetti, Alessandro Busti, Michela Farenzena, and Andrea Fusiello. Globally convergent autocalibration. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2, pages 1426–1432, Nice, France, October 2003. IEEE Computer Society Press, Washington.

[Bim06] Oliver Bimber and Ramesh Raskar. Modern approaches to augmented reality. In *ACM SIGGRAPH 2006 Courses*, Article No. 1, Boston, MA, USA, July/August 2006. ACM Press, New York.

[Bla92] Andrew Blake and Alan Yuille, editors. *Active Vision*. MIT Press, Cambridge, MA, 1992.

[Bla96] Michael J. Black and Anand Rangarajan. On the unification of line processes, outlier rejection, and robust statistics with applications in early vision. *International Journal of Computer Vision*, 19(1):57–91, July 1996.

[Bue01] 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.

[Cha98] Douglas Chai and King N. Ngan. Locating facial region of a head-and-shoulder color image. In *Proceedings of the 3rd International Conference on Automatic Face and Gesture Recognition*, pages 124–129, Nara, Japan, April 1998. IEEE Computer Society Press, Washington.

[Cha00] Jin-Xiang Chai, Xin Tong, Shing-Chow Chan, and Heung-Yeung Shum. Plenoptic sampling. In *Proceedings of SIGGRAPH 2000*, pages 307–318, New Orleans, LA, USA, July 2000. ACM Press, New York.

[Chr94] Stéphane Christy and Radu Horaud. Euclidean shape and motion from multiple perspective views by affine iterations. Rapport de Recherche No. 2421, INRIA Rhône-Alpes, December 1994.

[Chr96] Stéphane Christy and Radu Horaud. Euclidean shape and motion from multiple perspective views by affine iterations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(11):1098–1104, November 1996.

[Cos98] João Paulo Costeira and Takeo Kanade. A multibody factorization method for independently moving objects. *International Journal of Computer Vision*, 29(3):159–179, September 1998.

[Cur96] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of SIGGRAPH '96*, pages 303–312, New Orleans, LA, USA, August 1996. ACM Press, New York.

[Dem98] David Demirdjian, Gabriella Csurka, and Radu Horaud. Autocalibration in the presence of critical motions. In J. Carter and M. Nixon, editors, *Proceedings of the Ninth British Machine Vision Conference*, volume 2, pages 751–759, Southampton, UK, September 1998. British Machine Vision Association, Malvern, UK.

[Den98] Joachim Denzler, Benno Heigl, and Heinrich Niemann. An efficient combination of 2D and 3D shape description for contour based tracking of moving objects. In H. Burkhardt and B. Neumann, editors, *Computer Vision – ECCV '98. 5th European Conference on Computer Vision*, pages 843–857, Freiburg, Germany, June 1998. Springer-Verlag, Berlin.

[Deu04] Benjamin Deutsch, Ingo Scholz, Christoph Gräßl, and Heinrich Niemann. Extending light fields using object tracking techniques. In B. Girod, M. Magnor, and H.-P. Seidel, editors, *Vision, Modeling, and Visualization 2004*, pages 109–116, Stanford, CA, USA, November 2004. Aka/IOS Press, Berlin, Amsterdam.

[Dor03a] Gianfranco Doretto, Alessandro Chiuso, Ying Nian Wu, and Stefano Soatto. Dynamic textures. *International Journal of Computer Vision*, 51(2):91–109, February 2003.

[Dor03b] Gianfranco Doretto and Stefano Soatto. Towards plenoptic dynamic textures. In *Proceedings of the 3rd International Workshop on Texture Analysis and Synthesis*, pages 25–30, Nice, France, October 2003. Heriot-Watt University, Edinburgh.

[ES03a] Jan-Friso Evers-Senne and Reinhard Koch. Image based interactive rendering with view dependent geometry. *Computer Graphics Forum*, 22(3):573–582, September 2003.

[ES03b] Jan-Friso Evers-Senne and Reinhard Koch. Image based rendering from handheld cameras using quad primitives. In *Vision, Modeling and Visualization 2003*, pages 19–26, Munich, Germany, November 2003. Aka/IOS Press, Berlin, Amsterdam.

[Fau92] Olivier D. Faugeras, Quang-Tuan Luong, and Stephen J. Maybank. Camera self-calibration: Theory and experiments. In *Computer Vision – ECCV '92. Second European Conference on Computer Vision*, pages 321–334, Santa Margherita Ligure, Italy, May 1992. Springer-Verlag, London.

[Fau01] Olivier D. Faugeras and Quang-Tuan Luong. *The Geometry of Multiple Images*. MIT Press, Cambridge, MA, 2001.

[Fec05] Ulrich Fecker and Andr'e Kaup. Statistical analysis of multi-reference block matching for dynamic light field coding. In G. Greiner, J. Hornegger, H. Niemann, and M. Stamminger, editors, *Vision, Modeling, and Visualization 2005*, pages 445–452, Erlangen, Germany, November 2005. Aka/IOS Press, Berlin, Amsterdam.

[Fec06] Ulrich Fecker, Audrey Guenegues, Ingo Scholz, and André Kaup. Depth map compression for unstructured lumigraph rendering. In J. G. Apostolopoulos and A. Said, editors, *Visual Communications and Image Processing 2006*, Proceedings of SPIE-IS&T Electronic Imaging, SPIE Vol. 6077, pages 60770B–1 – 60770B–8, San Jose, CA, USA, January 2006. SPIE and IS&T, Bellingham, WA.

[Fis81] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[Fit98] Andrew W. Fitzgibbon and Andrew Zisserman. Automatic camera recovery for closed or open image sequences. In H. Burkhardt and B. Neumann, editors, *Computer Vision – ECCV '98. 5th European Conference on Computer Vision*, volume 1, pages 311–326, Freiburg, Germany, June 1998. Springer-Verlag, Berlin.

[För87] Wolfgang Förstner and Eberhard Gülch. A fast operator for detection and precise location of distinct points, corners, and centers of circular features. In *Intercommission Conference on Fast Processing of Photogrammetric Data*, pages 281–305, Interlaken, Switzerland, 1987.

[For97] Steven Fortune. Voronoi diagrams and delaunay triangulations. In J. Goodman and J. O'Rourke, editors, *Handbook of discrete and computational geometry*, The CRC Press series on discrete mathematics and its applications, pages 377–388. CRC Press, Boca Raton, 1997.

[Fri02] Mario Fritz. 3-D Objektverfolgung mit Lichtfeldern. Studienarbeit im Fach Informatik, Lehrstuhl f"ur Mustererkennung, Martensstr. 3, 91058 Erlangen, September 2002.

[Fus00] Andrea Fusiello. Uncalibrated euclidean reconstruction: a review. *Image and Vision Computing*, 18(6-7):555–563, May 2000.

[Fus01] Andrea Fusiello. A new autocalibration algorithm: Experimental evaluation. In *Computer Analysis of Images and Patterns: 9th International Conference, CAIP 2001*, pages 717–724, Warsaw, Poland, September 2001. Springer-Verlag, Berlin.

[Ger82] Allen Gersho. On the structure of vector quantizers. *IEEE Transactions on Information Theory*, 28(2):157–166, March 1982.

[Gib02] Simon Gibson, Jonathan Cook, Toby Howard, Roger J. Hubbold, and Daniel Oram. Accurate camera calibration for off-line, video-based augmented reality. In *International Symposium on Mixed and Augmented Reality*, pages 37–46, Darmstadt, Germany, September/October 2002. IEEE Computer Society Press, Washington.

[Gol96] Gene H. Golub and Charles F. van Loan. *Matrix computations*. Johns Hopkins University Press, Baltimore, 3rd edition, 1996.

[Gol02] Bastian Goldlücke, Marcus Magnor, and Bennett Wilburn. Hardware-accelerated dynamic light field rendering. In G. Greiner, H. Niemann, T. Ertl, B. Girod, and H.-P. Seidel, editors, *Vision, Modeling, and Visualization 2002*, pages 455–461, Erlangen, Germany, November 2002. Aka/IOS Press, Berlin, Amsterdam.

[Gor96] 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.

[Grä05] Christoph Gräßl, Timo Zinßer, Ingo Scholz, and Heinrich Niemann. 3-D object tracking with the adaptive hyperplane approach using sift models for initialization. In K. Ikeuchi, editor, *Proceedings of Machine Vision Applications 2005*, pages 5–8, Tsukuba, Japan, May 2005. IAPR MVA Conference Committee.

[Grz04] Marcin Grzegorzek, Ingo Scholz, Michael Reinhold, and Heinrich Niemann. Fast training for object recognition with structure-from-motion. In V.V. Geppener, I.B. Gurevich, S.E. Ivanova, A.P. Nemirko, H. Niemann, D.V. Puzankov, Yu.O. Trusova, and Yu.I. Zhuravlev, editors, *7th International Conference on Pattern Recognition and Image Analysis 2004: New Information Technologies*, pages 231–234, St. Petersburg, Russia, 2004. SPbETU, St. Petersburg.

[Grz05] Marcin Grzegorzek, Ingo Scholz, Michael Reinhold, and Heinrich Niemann. Fast training for object recognition with structure-from-motion. *Pattern Recognition and Image Analysis: Advances in Mathematical Theory and Applications*, 15(1):183–186, January 2005.

[Grz07a] Marcin Grzegorzek. *Appearance-Based Statistical Object Recognition Including Color and Context Modeling*. Logos Verlag, Berlin, April 2007.

[Grz07b] Marcin Grzegorzek, Ingo Scholz, Michael Reinhold, and Heinrich Niemann. Fast training for object recognition with structure-from-motion. *Pattern Recognition and Image Analysis*, 17(1):87–92, March 2007.

[Gue02] Rui F. C. Guerreiro and Pedro M. Q. Aguiar. 3D structure from video streams with partially overlapping images. In *Proceedings of the International Conference on Image Processing*, volume 3, pages 897–900, Rochester, NY, USA, September 2002. IEEE Computer Society Press, Washington.

[Hag98] Gregory D. Hager and Peter N. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(10):1025–1039, October 1998.

[Ham86] Frank R. Hampel, Elvezio M. Ronchetti, Peter J. Rousseeuw, and Werner A. Stahel. *Robust Statistics*. John Wiley and Sons, New York, 1986.

[Han00] Mei Han and Takeo Kanade. Reconstruction of a scene with multiple linearly moving objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 2542–2549, Hilton Head Island, SC, USA, June 2000. IEEE Computer Society Press, Washington.

[Han03] Mei Han and Takeo Kanade. Multiple motion scene reconstruction with uncalibrated cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(7):884–894, July 2003.

[Har88] Chris Harris and Mike Stephens. A combined edge and corner detector. In *4th Alvey Vision Conference*, pages 147–151, Manchester, UK, August 1988. Butterworth Publishers, Stoneham, MA.

[Har93] 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.

[Har97] Richard I. Hartley. In defense of the eight-point algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(6):580–593, 1997.

[Har03] Richard I. Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2nd edition, 2003.

[Hei98] Benno Heigl, Dietrich Paulus, and Heinrich Niemann. Tracking points in sequences of color images. In B. Radig, H. Niemann, Y. Zhuravlev, I. Gourevitch, and I. Laptev, editors, *5th German-Russian Workshop on Pattern Recognition and Image Understanding*, pages 70–77, Herrsching, Germany, September 1998. infix, Sankt Augustin.

[Hei99a] Benno Heigl, Reinhard Koch, Marc Pollefeys, Joachim Denzler, and Luc van Gool. Plenoptic modeling and rendering from image sequences taken by a hand-held camera. In W. Förstner, J.M. Buhmann, A. Faber, and P. Faber, editors, *Mustererkennung 1999, 21. DAGM-Symposium*, pages 94–101, Bonn, Germany, September 1999. Springer-Verlag, Heidelberg.

[Hei99b] Benno Heigl and Heinrich Niemann. Camera calibration from extended image sequences for lightfield reconstruction. In *Vision, Modeling and Visualization '99*, pages 43–50, Erlangen, Germany, November 1999. infix, Sankt Augustin.

[Hei04] Benno Heigl. *Plenoptic Scene Modeling from Uncalibrated Image Sequences*. ibidem-Verlag Stuttgart, January 2004.

[Hey97] Anders Heyden and Kalle Åström. Euclidean reconstruction from image sequences with varying and unknown focal length and principal point. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 438–443, San Juan, Puerto Rico, June 1997. IEEE Computer Society Press, Washington.

[Hey98] Anders Heyden and Kalle Åström. Minimal conditions on intrinsic parameters for euclidean reconstruction. In *Proceedings of the Third Asian Conference on Computer Vision*, volume 2, pages 169–176, Hong Kong, China, January 1998. Springer-Verlag, Berlin.

[Hey99a] Anders Heyden and Kalle Åström. Flexible calibration: Minimal cases for auto-calibration. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 1, pages 350–355, Corfu, Greece, September 1999. IEEE Computer Society Press, Washington.

[Hey99b] Anders Heyden, Rikard Berthilsson, and Gunnar Sparr. An iterative factorization method for projective structure and motion from image sequences. *Image and Vision Computing*, 17(13):981–991, November 1999.

[Hig94] Kazunori Higuchi, Martial Hebert, and Katsushi Ikeuchi. Building 3D models from unregistered range images. In *Proceedings of IEEE Conference on Robotics and Automation*, volume 3, pages 2248–2253, San Diego, CA, USA, May 1994. Kluwer Academic Publishers, Norwell, MA.

[Hop06] Christian Hopfgartner. Tiefenoptimierung in Lichtfeldern mit dem Levenberg-Marquardt-Verfahren. Diplomarbeit im Fach Mathematik, Lehrstuhl für Angewandte Mathematik II, Universität Erlangen-Nürnberg, October 2006.

[Hor81] Berthold K. P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence*, 16(1-3):185–203, August 1981.

[Ira99] Michal Irani. Multi-frame optical flow estimation using subspace constraints. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 1, pages 626–633, Corfu, Greece, September 1999. IEEE Computer Society Press, Washington.

[Isa98] Michael Isard and Andrew Blake. Condensation—conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, August 1998.

[Isa00] Aaron Isaksen, Leonard McMillan, and Steven J. Gortler. Dynamically reparameterized light fields. In *Proceedings of SIGGRAPH 2000*, pages 297–306, New Orleans, LA, USA, July 2000. ACM Press, New York.

[Jäh95] Bernd Jähne. *Digital image processing*. Springer-Verlag, London, 3rd edition, 1995.

[Jin01] Hailin Jin, Paolo Favaro, and Stefano Soatto. Real-time feature tracking and outlier rejection with changes in illumination. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 684–689, Vancouver, Canada, July 2001. IEEE Computer Society Press, Washington.

[Kan96a] Kenichi Kanatani. *Statistical Optimization for Geometric Computation: Theory and Practice*, volume 18 of *Machine Intelligence and Pattern Recognition*. Elsevier, 1996.

[Kan96b] Sing Bing Kang and Richard Szeliski. 3-D scene data recovery using omnidirectional multibaseline stereo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 364–370, San Francisco, CA, USA, June 1996. IEEE Computer Society Press, Washington.

[Kan98] Takeo Kanade and Daniel D. Morris. Factorization methods for structure from motion. *Philosophical Transactions: Mathematical, Physical and Engineering Sciences*, 356(1740):1153–1173, May 1998.

[Kan01a] Kenichi Kanatani. Motion segmentation by subspace separation and model selection. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2, pages 586–591, Vancouver, Canada, July 2001. IEEE Computer Society Press, Washington.

[Kan01b] Yasushi Kanazawa and Kenichi Kanatani. Do we really have to consider covariance matrices for image features? In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2, pages 301–306, Vancouver, Canada, July 2001. IEEE Computer Society Press, Washington.

[Kan02] Kenichi Kanatani. Evaluation and selection of models for motion segmentation. In A. Heyden, G. Sparr, M. Nielsen, and P. Johansen, editors, *Computer Vision – ECCV 2002. 7th European Conference on Computer Vision*, volume 3, pages 335–349, Copenhagen, Denmark, May 2002. Springer-Verlag, Berlin.

[Kan03] Kenichi Kanatani and Yasuyuki Sugaya. Multi-stage optimization for multi-body motion segmentation. In *Proceedings of the Australia-Japan Advanced Workshop on Computer Vision*, pages 25–31, Adelaide, Australia, September 2003.

[Kan04] Sing Bing Kang and Richard Szeliski. Extracting view-dependent depth maps from a collection of images. *International Journal of Computer Vision*, 58(2):139–163, 2004.

[Kas88] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, January 1988.

[Koc99a] Reinhard Koch, Marc Pollefeys, Benno Heigl, Luc van Gool, and Heinrich Niemann. Calibration of hand-held camera sequences for plenoptic modeling. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 1, pages 585–591, Corfu, Greece, September 1999. IEEE Computer Society Press, Washington.

[Koc99b] Reinhard Koch, Marc Pollefeys, and Luc van Gool. Robust calibration and 3D geometric modeling from large collections of uncalibrated images. In W. Förstner, J.M. Buhmann, A. Faber, and P. Faber, editors, *Mustererkennung 1999, 21. DAGM-Symposium*, pages 413–420, Bonn, September 1999. Springer-Verlag, Heidelberg.

[Koc02] Reinhard Koch, Jan-Michael Frahm, Jan-Friso Evers-Senne, and Jan Wötzel. Plenoptic modeling of 3D scenes with a sensor-augmented multi-camera rig. In *Tyrrhenian International Workshop on Digital Communication*, Capri, Italy, September 2002.

[Kou95] Weidong Kou. *Digital Image Compression: Algorithms and Standards*. Kluwer Academic Publishers, Norwell, MA, 1995.

[Kut99] Kiriakos N. Kutulakos and Steven M. Seitz. A theory of shape by space carving. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 1, pages 307–314, Corfu, Greece, September 1999. IEEE Computer Society Press, Washington.

[Lev96] 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.

[Li98] Wei Li, Qi Ke, Xiaohu Huang, and Nanning Zheng. Light field rendering of dynamic scene. *Machine Graphics and Vision*, 7(3):551–563, 1998.

[Lin80] Yoseph Linde, Andrés Buzo, and Robert M. Gray. An algorithm for vector quantizer design. *IEEE Transactions on Communications*, 28(1):84–95, January 1980.

[Low99] David G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157, Corfu, Greece, September 1999. IEEE Computer Society Press, Washington.

[Low04] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.

[Lu97] Feng Lu and Evangelos E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4(4):333–349, October 1997.

[Luc81] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, pages 674–679, Vancouver, Canada, August 1981. Morgan Kaufmann Publishers, San Francisco.

[Mag00] Marcus Magnor and Bernd Girod. Data compression for light field rendering. *IEEE Transactions on Circuits and Systems for Video Technology*, 10(3):338–343, April 2000.

[Mag01] Marcus Magnor. *Geometry-Adaptive Multi-View Coding Techniques for Image-based Rendering*. Shaker Verlag, Aachen, November 2001.

[Mat02] Wojciech Matusik, Hanspeter Pfister, Remo Ziegler, Addy Ngan, and Leonard McMillan. Acquisition and rendering of transparent and refractive objects. In *Proceedings of the 13th Eurographics workshop on Rendering Techniques*, ACM International Conference Proceeding Series, pages 267–278, Pisa, Italy, 2002. Eurographics Association, Aire-la-Ville, Switzerland.

[Mat03] Iain Matthews, Takahiro Ishikawa, and Simon Baker. The template update problem. In *Proceedings of the British Machine Vision Conference*, pages 810–815, Norwich, UK, September 2003. British Machine Vision Association, Malvern, UK.

[McM95] Leonard McMillan and Gary Bishop. Plenoptic modeling: An image-based rendering system. In *Proceedings of SIGGRAPH '95*, pages 39–46, Los Angeles, CA, USA, August 1995. ACM Press, New York.

[Men99] Paulo Mendonça and Roberto Cipolla. A simple technique for self-calibration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 1500–1505, Fort Collins, CO, USA, June 1999. IEEE Computer Society Press, Washington.

[Mik05] Krystian Mikolajczk and Cordelia Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, October 2005.

[Mor77] Hans P. Moravec. Towards automatic visual obstacle avoidance. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, page 584, Cambridge, MA, USA, August 1977. Morgan Kaufmann Publishers, San Francisco.

[Mor00] 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.

[Mül02] Roger Müller. Anwendung bildbasierter Objektmodelle in der erweiterten Realit"at. Studienarbeit im Fach Informatik, Lehrstuhl f"ur Mustererkennung, Martensstr. 3, 91058 Erlangen, August 2002.

[Nel65] John A. Nelder and Roger Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.

[Ng05] Ren Ng, Marc Levoy, Mathieu Brédif, Gene Duval, Mark Horowitz, and Pat Hanrahan. Light field photography with a hand-held plenoptic camera. Technical Report CSTR 2005-02, Stanford University Computer Science, April 2005.

[Nie03] Heinrich Niemann. *Klassifikation von Mustern*. 2nd revised edition, May 2003. http://www5.informatik.uni-erlangen.de/Personen/niemann/klassifikation-von-mustern/m00links.html.

[Nie05] Heinrich Niemann and Ingo Scholz. Evaluating the quality of light fields computed from hand-held camera images. *Pattern Recognition Letters*, 26(3):239–249, February 2005. In Memoriam: Azriel Rosenfeld.

[Nis00] David Nistér. Reconstruction from uncalibrated sequences with a hierarchy of trifocal tensors. In *Computer Vision – ECCV 2000, 6th European Conference on Computer Vision*, volume 1, pages 649–663, Dublin, Ireland, June/July 2000. Springer-Verlag, London.

[Oht81] Yuichi Ohta, Kiyoshi Maenobu, and Toshiyuki Sakai. Obtaining surface orientation from texels under perspective projection. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, pages 746–751, Vancouver, Canada, August 1981. Morgan Kaufmann Publishers, San Francisco.

[Oli01] John Oliensis and Yacup Genc. Fast and accurate algorithms for projective multi-image structure from motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):546–559, June 2001.

[Ooi01] Ryutaro Ooi, Takayuki Hamamoto, Takeshi Naemura, and Kiyoharu Aizawa. Pixel independent random access image sensor for real time image-based rendering system. In *Proceedings of the International Conference on Image Processing*, volume 2, pages 193–196, Thessaloniki, Greece, October 2001. IEEE Computer Society Press, Washington.

[Poe97] Conrad J. Poelman and Takeo Kanade. A paraperspective factorization method for shape and motion recovery. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(3):206–218, March 1997.

[Pol98] Marc Pollefeys, Reinhard Koch, and Luc van Gool. Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 90–95, Bombay, India, January 1998. Narosa Publishing House, New Delhi.

[Pol99] Marc Pollefeys. *Self-Calibration and Metric 3D Reconstruction from Uncalibrated Image Sequences*. PhD thesis, Faculteit Toegepaste Wetenschappen, Katholieke Universiteit Leuven, Belgium, May 1999.

[Pol04] Marc Pollefeys, Luc van Gool, Maarten Vergauwen, Frank Verbiest, Kurt Cornelis, Jan Tops, and Reinhard Koch. Visual modeling with a hand-held camera. *International Journal of Computer Vision*, 59(3):207–232, September 2004.

[Pop06] Antonia Popp, Frank Wolfsgruber, Ingo Scholz, and Gerd Häusler. Endoskopische 3D-Rekonstruktion mit informationstheoretisch optimierter Beleuchtung. In *DGaO-Proceedings, Beiträge zur 107. Tagung der Deutschen Gesellschaft für angewandte Optik*, Weingarten, Germany, June 2006. Deutsche Gesellschaft für angewandte Optik e.V., Erlangen.

[Pot87] Michael Potmesil. Generating octree models of 3D objects from their silhouettes in a sequence of images. *Computer Vision, Graphics, and Image Processing*, 40(1):1–29, October 1987.

[Pre93] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 2nd edition, 1993.

[Reh98] Volker Rehrmann and Lutz Priese. Fast and robust segmentation of natural color scenes. In *Proceedings of the 3rd Asian Conference on Computer Vision*, volume 1, pages 598–606, Hong Kong, China, January 1998. Springer-Verlag, Berlin.

[Rot02] Gerhard Roth and Anthony Whitehead. Some improvements on two autocalibration algorithms based on the fundamental matrix. In *Proceedings of the International Conference on Pattern Recognition*, volume 2, pages 312–315, Québec, Canada, August 2002. IEEE Computer Society Press, Washington.

[Rou87] Peter J. Rousseeuw and Annick M. Leroy. *Robust Regression and Outlier Detection*. John Wiley & Sons, New York, 1987.

[Sai03] Miguel Sainz, Antonio Susin, and Nader Bagherzadeh. Camera calibration of long image sequences with the presence of occlusions. In *Proceedings of the International Conference on Image Processing*, volume 1, pages 317–320, Barcelona, Spain, September 2003. IEEE Computer Society Press, Washington.

[Sal01] Tobias Salb, Oliver Burgert, Tilo Gockel, Björn Giesler, and Rüdiger Dillmann. Comparison of tracking techniques for intraoperative presentation of medical data using a see-through head-mounted display. In *Proceedings of Medicine Meets Virtual Reality (MMVR)*, Newport Beach, CA, USA, January 2001. IOS Press, Amsterdam.

[Sch00a] Hartmut Schirmacher, Wolfgang Heidrich, and Hans-Peter Seidel. High-quality interactive lumigraph rendering through warping. In S. Fels and P. Poulin, editors, *Proceedings Graphics Interface 2000*, pages 87–94, Montreal, Canada, May 2000.

[Sch00b] Cordelia Schmid, Roger Mohr, and Christian Bauckhage. Evaluation of interest point detectors. *International Journal of Computer Vision*, 37(2):151–172, June 2000.

[Sch01a] Hartmut Schirmacher, Li Ming, and Hans-Peter Seidel. On-the-fly processing of generalized lumigraphs. *Computer Graphics Forum*, 20(3):165–174, September 2001.

[Sch01b] Hartmut Schirmacher, Christian Vogelgsang, Hans-Peter Seidel, and Günther Greiner. Efficient free form light field rendering. In T. Ertl, B. Girod, G. Greiner, H. Niemann, and H.-P. Seidel, editors, *Vision, Modeling, and Visualization 2001*, pages 249–256,528, Stuttgart, Germany, November 2001. AKA/IOS Press, Berlin, Amsterdam.

[Sch01c] Jochen Schmidt and Heinrich Niemann. Using quaternions for parametrizing 3-D rotations in unconstrained nonlinear optimization. In T. Ertl, B. Girod, G. Greiner, H. Niemann, and H.-P. Seidel, editors, *Vision, Modeling, and Visualization 2001*, pages 399–406, Stuttgart, Germany, November 2001. AKA/IOS Press, Berlin, Amsterdam.

[Sch01d] Jochen Schmidt, Ingo Scholz, and Heinrich Niemann. Placing Arbitrary Objects in a Real Scene Using a Color Cube for Pose Estimation. In B. Radig and S. Florczyk, editors, *Pattern Recognition, 23rd DAGM Symposium*, pages 421–428, Munich, Germany, September 2001. Springer-Verlag Berlin.

[Sch01e] Ingo Scholz, Jochen Schmidt, and Heinrich Niemann. Farbbildverarbeitung unter Echtzeitbedingungen in der Augmented Reality. In D. Paulus and J. Denzler, editors,

*7. Workshop Farbbildverarbeitung*, pages 59–65, Erlangen, Germany, October 2001. Universität Erlangen-Nürnberg, Institut für Informatik.

[Sch02] Ingo Scholz, Joachim Denzler, and Heinrich Niemann. Calibration of real scenes for the reconstruction of dynamic light fields. In K. Ikeuchi, editor, *Proceedings of Machine Vision Applications 2002*, pages 32–35, Nara, Japan, December 2002. IAPR MVA Organizing Committee.

[Sch04a] Jochen Schmidt, Florian Vogt, and Heinrich Niemann. Vector quantization based data selection for hand-eye calibration. In B. Girod, M. Magnor, and H.-P. Seidel, editors, *Vision, Modeling, and Visualization 2004*, pages 21–28, Stanford, CA, USA, November 2004. Aka/IOS Press, Berlin, Amsterdam.

[Sch04b] Ingo Scholz, Joachim Denzler, and Heinrich Niemann. Calibration of real scenes for the reconstruction of dynamic light fields. *IEICE Transactions on Information & Systems*, E87-D(1):42–49, January 2004.

[Sch04c] Ingo Scholz and Heinrich Niemann. Globally consistent 3-D reconstruction by utilizing loops in camera movement. In C. E. Rasmussen, H. H. Bülthoff, M. A. Giese, and B. Schölkopf, editors, *Pattern Recognition, 26th DAGM Symposium*, pages 471–479, Tübingen, Germany, August/September 2004. Springer-Verlag, Berlin.

[Sch05] Ingo Scholz, Christian Vogelgsang, Joachim Denzler, and Heinrich Niemann. Dynamic light field reconstruction and rendering for multiple moving objects. In K. Ikeuchi, editor, *Proceedings of Machine Vision Applications 2005*, pages 184–188, Tsukuba, Japan, May 2005. IAPR MVA Conference Committee.

[Sch06] Jochen Schmidt. *3-D Reconstruction and Stereo Self-Calibration for Augmented Reality*. Logos Verlag, Berlin, November 2006.

[Sha99] Amnon Shashua, Shai Avidan, and Michael Werman. Trajectory triangulation over conic sections. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 330–336, Corfu, Greece, September 1999. IEEE Computer Society Press, Washington.

[Sha01] Amnon Shashua and Anat Levin. Multi-frame infinitesimal motion model for the reconstruction of (dynamic) scenes with multiple linearly moving objects. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2, pages 592–599, Vancouver, Canada, July 2001. IEEE Computer Society Press, Washington.

[Sha02] Puneet Sharma, Angshuman Parashar, Subhashis Banerjee, and Prem Kalra. An uncalibrated lightfield acquisition system. In *Proceedings of the 3rd Indian Conference on Computer Vision, Graphics and Image Processing*, pages 25–30, Ahmadabad, India, December 2002. Allied Publishers Ltd., New Delhi.

[She96] Jonathan R. Shewchuk. Triangle: Engineering a 2D quality mesh generator and delaunay triangulator. In *Selected papers from the Workshop on Applied Computational Geometry, Towards Geometric Engineering*, pages 203–222, Philadelphia, PA, USA, May 1996. Springer-Verlag, London.

[Shi94] Jianbo Shi and Carlo Tomasi. Good features to track. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, Seattle, WA, USA, June 1994. IEEE Computer Society Press, Washington.

[Shu99] Heung-Yeung Shum, Zhengyou Zhang, and Qifa Ke. Efficient bundle adjustment with virtual key frames: A hierarchical approach to multi-frame structure from motion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 2538–2543, Fort Collins, CO, USA, June 1999. IEEE Computer Society Press, Washington.

[Sla80] Chester C. Slama, editor. *Manual of Photogrammetry*. American Society of Photogrammetry, Falls Church, VA, 4th edition, 1980.

[Sla02] Mel Slater, Anthony Steed, and Yiorgos Chrysanthou. *Computer Graphics and Virtual Environments. From Realism to Real-Time*. Addison-Wesley Publishing Co., Reading, MA, 1st edition, 2002.

[Slo97] Peter-Pike Sloan, Michael F. Cohen, and Steven J. Gortler. Time critical lumigraph rendering. In *Proceedings of the 1997 Symposium on Interactive 3D Graphics*, pages 17–23, Providence, RI, USA, April 1997. ACM Press, New York.

[Smi00] Warren J. Smith. *Modern Optical Engineering*. McGraw-Hill, New York, 3rd edition, 2000.

[Stu96] Peter Sturm and Bill Triggs. A factorization based algorithm for multi-image projective structure from motion. In *Computer Vision – ECCV '96. 4th European Conference on Computer Vision*, volume 2, pages 709–720, Cambridge, UK, April 1996. Springer-Verlag, London.

[Stu00] Peter Sturm. A case against Kruppa's equations for camera self-calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(10):1199–1204, October 2000.

[Sün05] Thomas Sünkel. Evaluation und Optimierung von Lichtfeldern mittels Ansichtenvergleichs aus der Bildsynthese. Studienarbeit im Fach Informatik, Lehrstuhl für Mustererkennung, Universität Erlangen-Nürnberg, May 2005.

[Sze97] Richard Szeliski and Sing Bing Kang. Shape ambiguities in structure from motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5):506–512, May 1997.

[Sze98] Richard Szeliski and Philip Torr. Geometrically constrained structure from motion: Points on planes. In *European Workshop on 3D Structure from Multiple Images of Large-Scale Environments (SMILE)*, pages 171–186, Freiburg, Germany, June 1998. Springer-Verlag, Berlin.

[Tom91] Carlo Tomasi and Takeo Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University, April 1991.

[Tom92] 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.

[Tör89] Aimo Törn and Antanas Žilinskas. *Global Optimization*. Springer-Verlag, Berlin, 1989.

[Tri97] Bill Triggs. Autocalibration and the absolute quadric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 609–614, San Juan, Puerto Rico, June 1997. IEEE Computer Society Press, Washington.

[Tri00] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. *Vision Algorithms: Theory and Practice: International Workshop on Vision Algorithms, Corfu, Greece, September 1999. Proceedings*, volume 1883 of *Lecture Notes in Computer Science*, chapter Bundle Adjustment – A Modern Synthesis, pages 298–372. Springer-Verlag, Berlin, 2000.

[Tru98] Emanuele Trucco and Alessandro Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice Hall, Upper Saddle River, NJ, 1998.

[Tsa87] Roger Y. Tsai. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *IEEE Journal of Robotics and Automation*, RA-3(4):323–344, August 1987.

[Tsa89] Roger Y. Tsai and Reimer K. Lenz. A new technique for fully autonomous and efficient 3D robotics hand/eye calibration. *IEEE Transactions on Robotics and Automation*, 5(3):345–358, June 1989.

[Vid04] René Vidal and Richard I. Hartley. Motion segmentation with missing data using PowerFactorization and GPCA. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 310–316, Washington, DC, USA, June/July 2004. IEEE Computer Society Press, Washington.

[Vog02a] Christian Vogelgsang, Ingo Scholz, Günther Greiner, and Heinrich Niemann. lgf3 - a versatile framework for vision and image-based rendering applications. In G. Greiner, H. Niemann, T. Ertl, B. Girod, and H.-P. Seidel, editors, *Vision, Modeling, and Visualization 2002*, pages 257–264, Erlangen, Germany, November 2002. Aka/IOS Press, Berlin, Amsterdam.

[Vog02b] Florian Vogt, Dietrich Paulus, and Heinrich Niemann. Highlight substitution in light fields. In *Proceedings of the International Conference on Image Processing*, pages 637–640, Rochester, NY, USA, September 2002. IEEE Computer Society Press, Washington.

[Vog02c] Florian Vogt, Dietrich Paulus, Ingo Scholz, Heinrich Niemann, and Christoph Schick. Glanzlichtsubstitution durch Lichtfelder. In M. Meiler, H. Handels, F. Kruggel, T. Lehmann, and D. Saupe, editors, *6. Workshop Bildverarbeitung für die Medizin*, pages 103–106, Leipzig, Germany, March 2002. Springer-Verlag, Berlin.

[Vog04] Florian Vogt, Sophie Krüger, Jochen Schmidt, Dietrich Paulus, Heinrich Niemann, Werner Hohenberger, and Christoph Schick. Light fields for minimal invasive surgery using an endoscope positioning robot. *Methods of Information in Medicine*, 43(4):403–408, 2004.

[Vog05] Christian Vogelgsang. *The lgf3 Project: A Versatile Implementation Framework for Image-Based Modeling and Rendering*, volume 38 of *Arbeitsberichte des Instituts für Informatik*. Universitä Erlangen-Nürnberg, Institut für Informatik, Erlangen, May 2005.

[Vog06] Florian Vogt. *Augmented Light Field Visualization and Real-Time Image Enhancement for Computer Assisted Endoscopic Surgery*. Der Andere Verlag, Tönning, August 2006.

[Wag01] Marc Wagner. Rekonstruktion von Oberflächen aus Punktewolken. Studienarbeit im Fach Informatik, Lehrstuhl für Graphische Datenverarbeitung, Universität Erlangen-Nürnberg, July 2001.

[Wag03] Marc Wagner, Ulf Labsik, and Günther Greiner. Repairing non-manifold triangle meshes using simulated annealing. *International Journal on Shape Modeling*, 9(2):137–153, December 2003.

[Wat92] Alan Watt and Mark Watt. *Advanced Animation and Rendering Techniques*. Addison-Wesley Publishing Co., Reading, MA, 1992.

[Wei99] Joachim Weickert, Seiji Ishikawa, and Atsushi Imiya. Linear scale-space has first been proposed in Japan. *Journal of Mathematical Imaging and Vision*, 10(3):237–252, May 1999.

[Wil02] Bennett Wilburn, Michael Smulski, Hsiao-Heng Kelin Lee, and Mark Horowitz. The light field video camera. In *Proceedings of Media Processors 2002, SPIE Electronic Imaging*, pages 29–36, San Jose, CA, USA, January 2002.

[Wil05] Bennett Wilburn, Neel Joshi, Vaibhav Vaish, Eino-Ville Talvala, Emilio Antunez, Adam Barth, Andrew Adams, Mark Horowitz, and Marc Levoy. High performance imaging using large camera arrays. *ACM Transactions on Graphics*, 24(3):765–776, July 2005.

[Win05a] Christian Winter, Stephan Rupp, Christian Münzenmayer, Klaus Spinnler, Heinz Gerhäuser, and Thomas Wittenberg. Adaptive Rasterreduktion in Aufnahmen von flexiblen Endoskopen durch spektrale Maskierung. In H.-P. Meinzer, H. Handels, A. Horsch, and T. Tolxdorff, editors, *Bildverarbeitung für die Medizin 2005*, pages 45–49, Heidelberg, Germany, March 2005. Springer-Verlag, Berlin.

[Win05b] Christian Winter, Ingo Scholz, Stephan Rupp, and Thomas Wittenberg. Reconstruction of tubes from monocular fiberscopic images – application and first results. In G. Greiner, J. Hornegger, H. Niemann, and M. Stamminger, editors, *Vision, Modeling, and Visualization 2005*, pages 57–64, Erlangen, Germany, November 2005. Aka / IOS Press, Berlin, Amsterdam.

[Wit06] Thomas Wittenberg, Christian Winter, Ingo Scholz, Stephan Rupp, Klaus Bumm, Marc Stamminger, and Christopher Nimsky. 3-D-reconstruction of the sphenoidal sinus from monocular endoscopic views: First results. In *Proceedings BMT 2006*, Biomedizinische Technik, Gem. Jahrestagung der Deutschen, Österreichischen und Schweizerischen Gesellschaften für Biomedizinische Technik, Zürich, Switzerland, September 2006. de Gruyer Verlag, Berlin.

[Woo00] Daniel N. Wood, Daniel I. Azuma, Ken Aldinger, Brian Curless, Tom Duchamp, David H. Salesin, and Werner Stuetzle. Surface light fields for 3D photography. In *Proceedings of SIGGRAPH 2000*, pages 287–296, New Orleans, LA, USA, July 2000. ACM Press, New York.

[Yan02] 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 Techniques*, pages 77–86, Pisa, Italy, June 2002. Eurographics Association, Aire-la-Ville, Switzerland.

[Zel96] Cyril Zeller and Olivier Faugeras. Camera self-calibration from video sequences: the kruppa equations revisited. Technical Report RR-2793, INRIA Sophia-Antipolis, February 1996.

[Zha95] Zhengyou Zhang. Parameter estimation techniques: A tutorial with application to conic fitting. Technical Report RR-2676, INRIA Sophia-Antipolis, October 1995.

[Zin04] Timo Zinßer, Christoph Gräßl, and Heinrich Niemann. Efficient feature tracking for long video sequences. In C. E. Rasmussen, H. H. Bülthoff, M. A. Giese, and B. Schölkopf, editors, *Pattern Recognition, 26th DAGM Symposium*, pages 326–333, Tübingen, Germany, August/September 2004. Springer-Verlag, Berlin.

[Zin05] Timo Zinßer, Jochen Schmidt, and Heinrich Niemann. Point set registration with integrated scale estimation. In R. Sadykhov, S. Ablameiko, A. Doudkin, and L. Podenok, editors, *Proceedings of the Eighth International Conference on Pattern Recognition and Image Processing*, pages 116–119, Minsk, Belarus, 2005.

[Zob02] Matthias Zobel, Mario Fritz, and Ingo Scholz. Object tracking and pose estimation using light-field object models. In G. Greiner, H. Niemann, T. Ertl, B. Girod, and H.-P. Seidel, editors, *Vision, Modeling, and Visualization 2002*, pages 371–378, Erlangen, Germany, 2002. Aka/IOS Press, Berlin, Amsterdam.

[Zon99] Douglas E. Zongker, Dawn M. Werner, Brian Curless, and David H. Salesin. Environment matting and compositing. In *Proceedings of SIGGRAPH '99*, pages 205–214, Los Angeles, CA, USA, August 1999. ACM Press, New York.

[Zwi01] Matthias Zwicker, Hanspeter Pfister, Jeroen van Baar, and Markus Gross. Surface splatting. In *Proceedings of SIGGRAPH 2001*, pages 371–378, Los Angeles, CA, USA, August 2001. ACM Press, New York.

# Index

# Curriculum Vitae

## Personal Data

| | |
|---|---|
| *Last Name, First Names* | Scholz, Ingo Paul Matthias |
| *Date of Birth* | 24 March 1975 |
| *Place of Birth* | Regensburg, Germany |
| *Nationality* | German |

## Education & Professional Experience

| | |
|---|---|
| *January 2007 – Present* | Software Developer, Giesecke & Devrient GmbH, Munich, Germany |
| *March 2001 – December 2006* | Research Assistant, Chair for Pattern Recognition, University Erlangen-Nuremberg, Erlangen, Germany |
| *January 2001 – February 2001* | Graduate Assistant, Chair for Pattern Recognition, University Erlangen-Nuremberg, Erlangen, Germany |
| *October 1994 – December 2000* | Study of Computer Science, Institute of Computer Science, University Erlangen-Nuremberg, Erlangen, Germany<br>Degree: Dipl.-Inf. Univ. |
| *September 1999 – November 1999* | Student Trainee, IBM Sevilla, Speech Recognition Group, Sevilla, Spain |

| | |
|---|---|
| *September 1997 – Mai 1998* | Studienarbeit at University of Kansas, Lawrence, KS, USA |
| *March 1997 – April 1997* | Student Trainee, Fraunhofer Institute for Integrated Circuits IIS, Erlangen, Germany |
| *November 1995 – February 1997* | Student Assistant, Chair for Pattern Recognition, University Erlangen-Nuremberg, Erlangen, Germany |
| *September 1985 – July 1994* | Gymnasium Neutraubling, Neutraubling, Germany<br>Graduation: Abitur |
| *September 1981 – August 1985* | Josef-Hofmann-Grundschule Neutraubling, Neutraubling, Germany |