

Creating Realistic Ground Truth Data for the Evaluation of Calibration Methods for Plenoptic and Conventional Cameras

Tim Michels, Arne Petersen and Reinhard Koch
 Department of Computer Science, Kiel University, Germany
 {tmi, arne.petersen, rk}@informatik.uni-kiel.de

Abstract

Camera calibration methods usually consist of capturing images of known calibration patterns and using the detected correspondences to optimize the parameters of the assumed camera model. A meaningful evaluation of these methods relies on the availability of realistic synthetic data. In previous works concerned with conventional cameras the synthetic data was mainly created by rendering perfect images with a pinhole camera and subsequently adding distortions and aberrations to the renderings and correspondences according to the assumed camera model. This method can bias the evaluation since not every camera perfectly complies with an assumed model. Furthermore, in the field of plenoptic camera calibration there is no synthetic ground truth data available at all. We address these problems by proposing a method based on backward ray tracing to create realistic ground truth data that can be used for an unbiased evaluation of calibration methods for both types of cameras.

1. Introduction

The most commonly used camera calibration procedure consists of three steps: i) capturing images of calibration patterns, ii) detection of the patterns, *i.e.* points of interest in the images belonging to the calibration pattern, and iii) using the correspondences to optimize the parameters of the assumed mathematical camera model. In order to evaluate single parts of this pipeline, synthetic calibration pattern renderings with known correspondences can be beneficial in two ways. Firstly, the quality of the pattern detection method can be assessed by comparing the detector results on the renderings to the ground truth positions, and secondly, the optimization as well as the camera model can be evaluated using the ground truth correspondences without depending on a possibly biased pattern detector. However, the validity of such an evaluation depends on the quality of the ground truth data, *i.e.* its ability to reflect real data.

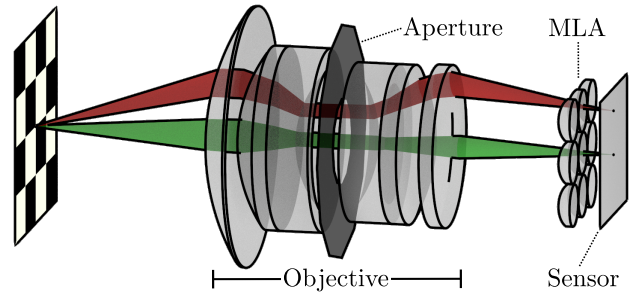


Figure 1. Schematics of a plenoptic camera as introduced by Adelson and Wang [4] and Lumsdaine and Georgiev [5] based on the ideas of Lippmann [6]. The red and green cones indicate the areas visible from the corresponding pixels.

For conventional cameras the synthetic image generation is usually done by first rendering the calibration pattern from the perspective of a simple pinhole camera model so that the correspondences are easy to calculate. Afterwards the images and correspondences are then distorted according to the assumed camera model (see *e.g.* [1][2][3]). This procedure poses the problem, that the generated data is not reflecting a real camera, but a virtual camera perfectly complying with the assumed distortion model. Accordingly, in a comparative evaluation of different calibration algorithms those methods assuming the exact same camera model have an advantage. Another problem is posed by the modeling of de-focus and image degradation effects like vignetting. In previous works these are either not modeled at all or simulated by adding Gaussian noise and blur to the perfectly distorted images. In neither of these cases the resulting images are directly comparable to real data resulting from a significantly more complex image formation process.

In the case of plenoptic cameras this imaging process is even more complicated since an additional microlens array (MLA) is placed between the main lens and the image sensor (compare Fig. 1). This renders the standard pipeline of creating perfect images and adding distortions and degradation afterwards infeasible since distortions of the main lens affect the position and angle of a ray hitting the MLA and

therefore have to be applied before the light rays enter the microlenses.

We propose a pipeline to generate realistic renderings of calibration patterns with ground truth correspondences, *i.e.* the positions of the calibration patterns' points of interest in the form of 2D pixel coordinates as well as 3D world coordinates. The key idea in generating these ground truth positions is to use ray tracing not only to render a realistic image I of a calibration pattern, but also to render a *position image* J whose pixels store positional information about the scene points hit by the rays traced from the respective pixel. Since the pose of the calibration pattern and thereby the 3D positions of its points of interest are exactly known, the search for the ground truth positions within the rendering I is reduced to simply finding the pixel positions with the correct positional information in the rendering J . Since a straight forward implementation of this idea is computationally expensive, we also propose a second method in which a ray in the scene space is calculated for every pixel, which is then intersected with the desired calibration pattern model to find the ground truth point positions. In summary, our contributions are:

- An extension of the plenoptic camera model of [7] to include multiple microlens types
- Two methods for calculating the ground truth positions of the points of interest in the rendered images
- Publicly available implementations of the simulation and ground truth creation methods¹

Note, that while the descriptions throughout this work are focused on plenoptic cameras, the whole pipeline is directly applicable to conventional cameras. We simply choose to describe the method for plenoptic cameras since these present a more complex case and the research in this area is in greater need of ground truth data as the standard approach of distorting perfect renderings is not applicable here.

2. Related Work

Camera simulation in computer graphics: The idea of using more realistic, physically-based lens models instead of a perfect pinhole camera for synthesizing images via ray tracing has first been explored by Potmesil and Chakravarty [8] and was later refined by Kolb *et al.* [9] and Wu *et al.* [10]. These models were further extended by Wu *et al.* [11] regarding the use of spectral ray tracing to simulate certain wave optics effects.

In contrast to these advanced methods for conventional cameras, the simulation of plenoptic cameras is a less explored area. This type of camera has only gained interest during the past decade due to the emergence of the first prototypes by Ng *et al.* [12] and the commercial realizations

by Lytro (no longer existing) and Raytrix [13]. Despite becoming a more active field of research, the simulation of plenoptic cameras in most publications concerned with using synthetic images is rather rudimentary. Fleischmann *et al.* [14] render images without any main lens and Zhang *et al.* [15] as well as Liang *et al.* [16] use a simplified thin main lens model. Accordingly, the synthesized images do not show the distortion and image degradation effects present in real data. Further works by Liu *et al.* [17] and Li *et al.* [18] based on ray splitting require an unrealistic large distance between the camera and the scene objects as well as simple scene materials. More recently, Michels *et al.* [7] proposed to fully model a plenoptic camera's components and presented an implementation for Blender [19]. Despite wave optic effects and multiple microlens types not being simulated in this approach, we decided to base our method on it due to its availability and extensibility.

Evaluation of pattern detectors: While previous works on the calibration of plenoptic cameras use either manually labeled data [20] or evaluate the detection and calibration as a combined system relying on precise real life measurements [21][22], approaches for conventional cameras have been evaluated with synthetic data for a broad variety of different patterns over the past decades. Luccese and Mitra [3] use projective warping and Gaussian blur on checkerboard images and Zhang [1] renders square pattern images assuming a pinhole camera with non-zero skew and also applies Gaussian blur. Heikkila [2] employs ray tracing and additional Gaussian noise as well as blur, but uses exactly the same camera model for rendering the point patterns as for the calibration. Ha *et al.* [23] render sharp single triangle pattern corners for a perspective camera and add different levels of Gaussian noise and blur afterwards.

In summary, there is no previous work for the calibration of plenoptic cameras featuring synthetic data and the work dealing with conventional cameras uses simplified or ideal models to generate synthetic data.

3. Organization

Sections 4.1 to 4.3 describe the extended camera model for ray tracing and our general approaches for creating the ground truth positions. Subsequently, some insights regarding the usefulness of the direct approach via forward ray tracing are provided in section 4.4. Finally, in section 5.1 the realization for Blender is explained and the remaining sections are devoted to the evaluation of our approach.

4. Method

The ray tracing approach presented in [7] is capable of producing realistic images for plenoptic cameras with one microlens type and by deactivating the MLA it can also be used to simulate conventional cameras within the bounds of

¹<https://gitlab.com/ungetym/plenoptic-ground-truth-creator>

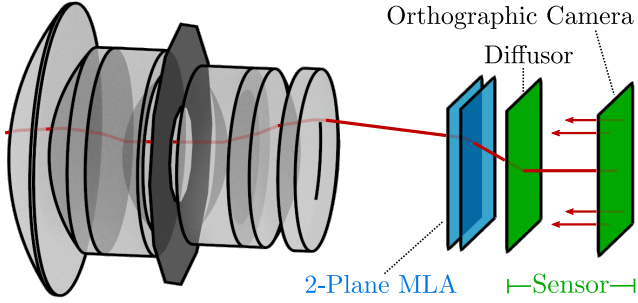


Figure 2. Schematics of the plenoptic camera model for ray tracing. While the objective’s lenses are fully modeled, the MLA is approximated by two planes with recalculated normals and the sensor is simulated by a combination of an orthographic camera and a diffusor plane.

ray tracing, *i.e.* without wave optical effects. Nevertheless, instead of directly using this simulation for our positional rendering approach, we first extend the camera model in order to also be able represent multifocus plenoptic cameras as distributed by Raytrix [24].

4.1. Simulation of Plenoptic Cameras

The basic setup of the camera model is given in Fig. 2. As described in [7], the objective’s lenses are explicitly modeled and the refraction at their surfaces is smoothed by recalculating the surface normals in order to avoid image artifacts resulting from the polygonal surface structure. The sensor is modeled by combining an orthographic camera with a diffusor plane which randomly refracts the rays traced from the orthographic camera within a specified angle distribution. This simulates a real pixel’s field of view (FOV) and its response to light rays with different angles of incidence. Accordingly, the diffusor plane can be thought of as the location of the sensor.

The last component, the MLA, is designed as a simple two plane model by exploiting the lensmaker’s equation for a thin lens with index of refraction (IOR) η and focal length f , given by

$$\frac{1}{f} \approx (\eta - 1) \left(\frac{1}{R_1} - \frac{1}{R_2} \right), \quad (1)$$

where R_1 and R_2 describe the front and back surface curvature radii. For a flat back surface, given by $R_2 = \infty$, it follows $f \approx R_1/(\eta - 1)$ and since a large radius R_1 leads to the front surface locally nearly being a plane, the microlenses can be constructed by using a two plane model with large IOR η and recalculated surface normals. We extend this part of the model to feature differently focused microlenses on the same MLA as shown in Fig. 3 by setting different values for R_1 depending on the coordinates of a microlens in the hexagonal grid. Since to our knowledge the Raytrix cameras are the only commercially available multi

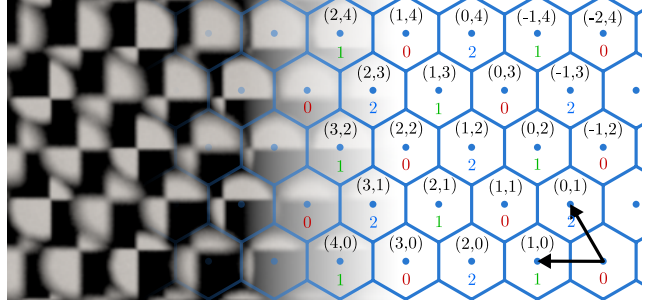


Figure 3. Rendering of a checkerboard with three differently focused microlens types overlaid with the hexagonal MLA layout. The black tuples are the coordinates of the center points with respect to the visualized base and the colored numbers indicate the lens type.

focus plenoptic cameras, we describe the extension for a setup with three microlens types as used by Raytrix. This model, however, can easily be modified to feature different configurations.

For the three microlens setup the type $t \in \{0, 1, 2\}$ of a microlens with center coordinates (i, j) in the hexagonal grid is given by $t = (i - j) \% 3$ as visualized in Fig. 3. In order to match the setup of a Raytrix camera, the three focal lengths R_1^t have to be chosen carefully with two restrictions in mind. First, all focal lengths should be larger than the distance between the MLA and the sensor plane since the MLA in Raytrix cameras is placed between the main lens and the virtual image of the scene, thus the microlenses collect converging light rays (compare Fig. 1). And second, the depth of field (DoF) of the three lens types should slightly overlap to create a connected combined DoF without focus gaps [24].

The described model can now be used to render realistic images of calibration patterns (or arbitrary scenes) for various plenoptic as well as conventional camera setups, where the camera type can be switched by (de)activating the MLA and modifying the parameters and positioning of the sensor and MLA. Note, that for the sake of simplicity, the illustrations in the remaining sections will contain the schematics of a real plenoptic camera instead of the model described here.

4.2. Rendering Positional Information

Since the pattern position and orientation are exactly known for the rendering, we can assume to have a set $\{p_k\}_{k=1, \dots, n} \subset \mathbb{R}^3$ of locations of the n relevant pattern points, *e.g.* the $n = 28$ corners of a 4×7 checkerboard. In order to use this information for finding the ground truth pixel positions of the calibration pattern points in the images rendered with the previously described setup, the same model is used to render positional information via ray tracing. In the usual backward ray tracing pipeline, for every pixel (i, j) a set of rays $R^{(i, j)}$ is traced through the

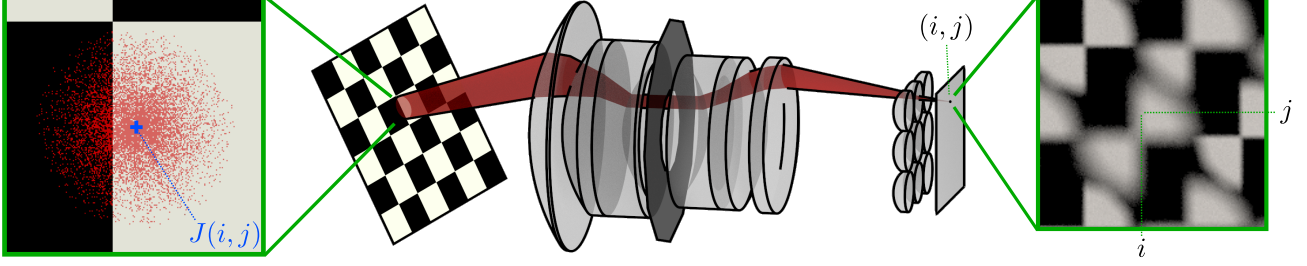


Figure 4. Positional rendering visualized: The right image shows a section of the rendering I containing the pixel (i, j) and the schematics in the middle visualize the bundle of rays traced from (i, j) and its intersection with the calibration pattern object. On the left the set of scene points hit by the rays, $\{p(r) : r \in R_{hit}^{(i,j)}\}$, is shown in red. Despite the calibration pattern not being in focus of the microlens, the pixel’s positional information, $J(i, j)$, can be calculated as the mean of the set of points, shown in blue.

camera into the scene and the colors of the scene points hit by the rays are accumulated which will be shortly formalized in the following. Let $p(r) \in \mathbb{R}^3$ denote the first scene point hit by the ray $r \in R_{hit}^{(i,j)}$ and split the set of rays into two disjoint sets $R_{hit}^{(i,j)} = R_{blocked}^{(i,j)} \cup R_{hit}^{(i,j)}$ with $R_{blocked}^{(i,j)}$ containing the rays not leaving the camera due to being blocked by the aperture or camera housing and $R_{hit}^{(i,j)}$ denoting the set of rays intersecting scene objects, whereby every ray leaving the camera is assumed to be in $R_{hit}^{(i,j)}$. The color of a pixel (i, j) in the calibration pattern rendering $I : \{0, \dots, width\} \times \{0, \dots, height\} \rightarrow \{0, \dots, 255\}^3$ is then given by

$$\begin{aligned} I(i, j) &= \frac{1}{|R_{hit}^{(i,j)}|} \sum_{r \in R_{hit}^{(i,j)}} c(p(r)) \\ &= \frac{1}{|R_{hit}^{(i,j)}|} \sum_{r \in R_{hit}^{(i,j)}} c(p(r)), \end{aligned} \quad (2)$$

where $c(p(r))$ describes the color of the 3D point hit by ray r and rays in $R_{blocked}^{(i,j)}$ are not assumed to add non-zero color information. We would like to remark, that the color $c(p(r))$ can be the result of further ray tracing calculations depending on the scene objects’ reflectivity and transmission properties. However, for the task at hand only the first scene point hit by a ray, $p(r)$, is considered.

For the positional rendering the same procedure is used, but instead of the color values $c(p(r))$ the positions $p(r)$ are accumulated and averaged, *i.e.* the value of the positional rendering $J : \{0, \dots, width\} \times \{0, \dots, height\} \rightarrow \mathbb{R}^3$ at pixel position (i, j) is given by

$$J(i, j) = \frac{1}{|R_{hit}^{(i,j)}|} \sum_{r \in R_{hit}^{(i,j)}} p(r), \quad (3)$$

as visualized in Fig. 4. Note, that the pixel value is normalized by $|R_{hit}^{(i,j)}|$ instead of $|R_{hit}^{(i,j)}|$ as in Equation 2 since we are interested in the average scene point hit by the rays unbiased by vignetting, *i.e.* the amount of blocked rays.

Despite knowing the average scene position $J(i, j)$ a camera pixel (i, j) is seeing, the known 3D calibration point positions $\{p_k\}$ can most likely not directly be found in the positional rendering due to J maximally containing $width \times height$ 3D positions of the continuous calibration pattern plane. The naive solution to this problem is searching for pixels at which the value of J is close to a position $\{p_k\}$, *i.e.* for every p_k we search for

$$(\hat{i}, \hat{j}) = \arg \min_{(i,j)} \|p_k - J(i, j)\| \quad (4)$$

and accept the solution (\hat{i}, \hat{j}) , if the distance for this position is within a certain threshold, $\|p_k - J(\hat{i}, \hat{j})\| < \lambda$ for some $\lambda > 0$. This procedure, however, does not work for plenoptic images since these can contain a scene point multiple times in different microlens images as shown in Fig. 1 and Fig. 3. Fortunately, the MLA configuration is known and therefore the image J can be splitted into microlens images J_1, \dots, J_m , each containing only the rendered information for exactly one microlens. In these images the search can then independently be performed.

This naive solution for finding the ground truth pixel positions has the obvious drawback of a limited accuracy. The solution is only accurate within $\pm 0.5px$ and if the number of samples, *i.e.* rays per pixel, is not sufficient, the found pixel (\hat{i}, \hat{j}) might even be an outlier due to $J(\hat{i}, \hat{j})$ containing a wrong position. This accuracy problem will be tackled in the following by rendering J with a higher resolution than I , filtering out unreliable results and finally using interpolation near the filtered pixel positions. First, one can observe, that for a sufficiently large number of samples per pixel, the values of small neighborhoods in J form an equidistant grid on the calibration pattern plane as visualized in Fig. 5. This observation is used as a constraint for filtering the point candidates.

Assume J was rendered with a resolution of $K \cdot width \times K \cdot height$, $K \in \mathbb{N}$ and let (\hat{i}, \hat{j}) a pixel such that some corner position p_k is located in the polygon given by $J(\hat{i}, \hat{j})$, $J(\hat{i} + 1, \hat{j})$, $J(\hat{i}, \hat{j} + 1)$ and $J(\hat{i} + 1, \hat{j} + 1)$ as shown in Fig. 5 and without loss of generality let $J(\hat{i}, \hat{j})$ be the closest of the

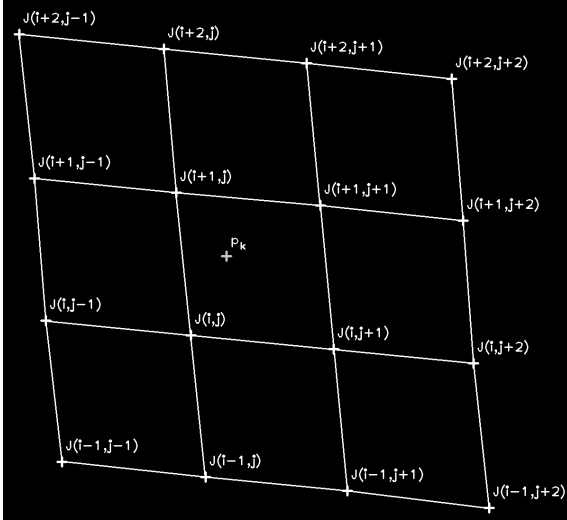
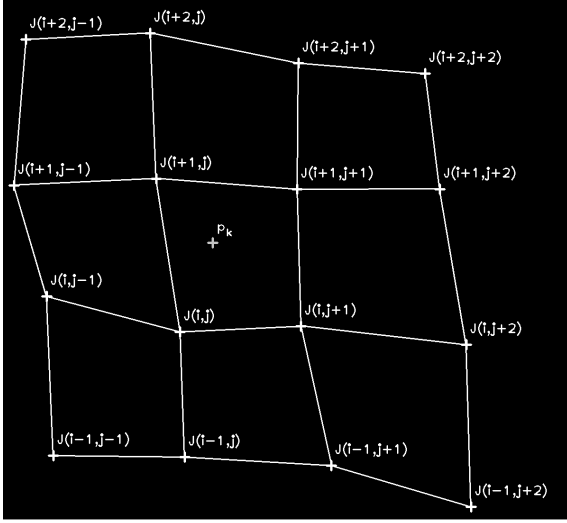


Figure 5. The values of J in a neighborhood of (i, j) approximately form a grid on the calibration pattern plane. This visualization shows the effect that the number of rays has on the grid structure. The positional images J used here were calculated with 64^2 (top) and 256^2 (bottom) samples per pixel.

four corners to p_k . In order to allow the interpolation of the pixel position within these coordinates, we first check, if the neighborhood $N := \{(\hat{i} + a, \hat{j} + b) : -1 \leq a \leq 2, -1 \leq b \leq 2\}$ approximately forms an equidistant grid. To this end, the average distances between the values of vertical and horizontal neighbors,

$$d_{vert} = \frac{1}{12} \sum_{(i,j) \in N|_{b < 2}} \|J(i, j) - J(i, j+1)\| \text{ and } \quad (5)$$

$$d_{horiz} = \frac{1}{12} \sum_{(i,j) \in N|_{a < 2}} \|J(i, j) - J(i+1, j)\|, \quad (6)$$

are calculated and then used to define the first constraint

$$\left| 1 - \frac{\|J(i, j) - J(i, j+1)\|}{d_{vert}} \right| < \lambda_d \quad (7)$$

for all $(i, j) \in N|_{b < 2}$ and a threshold $\lambda_d \in (0, 1)$ and analogously

$$\left| 1 - \frac{\|J(i, j) - J(i+1, j)\|}{d_{horiz}} \right| < \lambda_d \quad (8)$$

for all $(i, j) \in N|_{a < 2}$. In this length constraint λ_d describes the maximal relative deviation which simply enforces that the lengths of horizontal and vertical lines in the grid do not deviate too much from the respective average. A similar constraint is calculated for the angles of grid connections, *i.e.* the average angle

$$\alpha = \frac{1}{9} \sum_{(i,j) \in N|_{a < 2, b < 2}} \alpha_{(i,j)} \quad (9)$$

with $\alpha_{(i,j)} := \angle(J(i, j), J(i+1, j), J(i, j+1))$ is calculated and the respective constraint is formulated as

$$\forall (i, j) \in N|_{a < 2, b < 2} : |\alpha_{(i,j)} - \alpha| < \lambda_\alpha \quad (10)$$

for a threshold $\lambda_\alpha \in (0, \pi)$. If both constraints hold for the neighborhood of (\hat{i}, \hat{j}) , the ground truth pixel position (\tilde{i}, \tilde{j}) in I is interpolated via

$$(\tilde{i}, \tilde{j}) = \frac{1}{K} ((\hat{i}, \hat{j}) + s \cdot (1, 0) + t \cdot (0, 1)) \quad (11)$$

where s and t are the solution of the linear equation

$$p_k = J(\hat{i}, \hat{j}) + s(J(\hat{i}+1, \hat{j}) - J(\hat{i}, \hat{j})) + t(J(\hat{i}, \hat{j}+1) - J(\hat{i}, \hat{j})) \quad (12)$$

and $\frac{1}{K}$ is used to rescale the pixel position to the size of I . Note, that the solution (s, t) exists and does not require numerical approximations since all values of J as well as the point p_k are located on the same plane.

4.3. Calculating Pixel Rays

A major disadvantage of the approach described in the previous section is, that it requires one additional image to be rendered for every calibration pattern position. Especially for a large resolution scaling factor K and large sample numbers this is inefficient considering that the camera setup usually does not change during the creation of one dataset and therefore the exact same rays are traced through the camera into the scene for every positional rendering. To circumvent this redundancy, we propose the rendering of only two positional images per camera setup - the first one, J_{near} , for a plane located at the start of the cameras DoF and another one, J_{far} , for a plane at the DoF's end. For

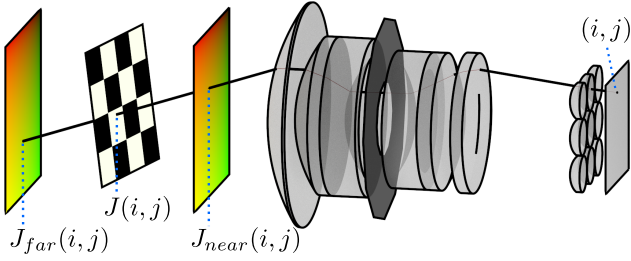


Figure 6. Two plane approach: The colored planes are used to create positional renderings J_{near} and J_{far} and the positional information $J(i, j)$ for a pixel (i, j) is then given by the intersection of the calibration pattern object and the ray defined by $J_{near}(i, j)$ and $J_{far}(i, j)$.

every pixel (i, j) the 3D points $J_{near}(i, j)$ and $J_{far}(i, j)$ define a ray in the scene space (compare Fig. 6) similar to the often used two-plane parametrization of the plenoptic function. Given the rendering of a calibration pattern I as before, the corresponding positional image J , as defined in the previous chapter, can be calculated by intersecting the pattern plane and the pixel rays, *i.e.*

$$J(i, j) = J_{near}(i, j) + t \cdot (J_{far}(i, j) - J_{near}(i, j)) \quad (13)$$

$$\text{with } t = \frac{\langle q - J_{near}(i, j), n \rangle}{\langle J_{far}(i, j) - J_{near}(i, j), n \rangle} \quad (14)$$

where q is an arbitrary point of the calibration pattern plane and n denotes its normal.

This method for calculating the positional image J requires only two positional renderings per camera setup instead of one rendering per calibration pattern image I .

4.4. How-Not-To: Forward Ray Tracing

Instead of rendering whole positional images using computationally expensive backward ray tracing with large numbers of samples, one might wonder why we do not simply use forward ray tracing, *i.e.* tracing rays from the known positions $\{p_k\}$ to the sensor. This idea is appealing since the number of required rays would be heavily reduced. However, this approach only works for scene points that are either in focus or create a perfect circle of confusion on the sensor. In the former case, the rays all hit exactly one single pixel on the sensor (for a plenoptic camera with one microlens type, they might hit unique pixels in different microlens images) and for the latter one can simply take the center of the circle of confusion as the ground truth position. By treating the sensor as a continuous plane instead of discretizing it into pixels during the ray tracing, even sub-pixel accuracy could be reached. However, the shape of the area of confusion can vary heavily depending on the optical system used for the imaging and it is unclear, which point could be regarded as ground truth position for arbitrary shapes which in addition can be split over multiple

microlens images.

Nevertheless, the forward ray tracing could be used to determine the areas of the sensor which should be rendered via backward ray tracing. After rendering a calibration pattern image I , the positions $\{p_k\}$ could be traced to the continuous sensor plane and after choosing the resolution of J , these sensor areas could be discretized into a set of pixels which are subsequently used for the positional rendering.

5. Evaluation

5.1. Realization in Blender

In order to evaluate our approach, first the model of [7] for Blender 2.79c was extended by modifying the MLA materials to support up to three configurable microlens types as described in section 4.1. With this setup, a calibration pattern rendering I can easily be created. A corresponding positional image J , however can not directly be rendered since the Cycles renderer does not provide the functionality to only accumulate rays hitting the scene. However, giving the calibration pattern plane a material that emits positional information, *i.e.* $c(p(r)) = p(r)$, and everything else a purely black material results in a rendering \hat{J} with

$$\hat{J}(i, j) = \frac{1}{|R^{(i, j)}|} \left(\sum_{r \in R_{hit}^{(i, j)}} p(r) + \underbrace{\sum_{r \in R_{blocked}^{(i, j)}} p(r)}_{=0} \right) \quad (15)$$

which differs from J (see Equation 3) only by the factor $|R^{(i, j)}_{hit}|/|R^{(i, j)}|$ describing the ratio of rays hitting the calibration pattern. This factor can be calculated by rendering an additional single channel image J_{white} for which the calibration pattern plane emits a purely white material and everything else remains black, *i.e.* $c(p(r)) = 1.0$ if $r \in R_{hit}^{(i, j)}$ and $c(p(r)) = 0.0$ otherwise. The resulting image J_{white} then contains the desired factor,

$$J_{white}(i, j) = \frac{1}{|R^{(i, j)}|} \left(\underbrace{\sum_{r \in R_{hit}^{(i, j)}} 1}_{=|R^{(i, j)}_{hit}|} + \underbrace{\sum_{r \in R_{blocked}^{(i, j)}} 0}_{=0} \right) \quad (16)$$

and accordingly $J(i, j)$ can be calculated by dividing the three channels of $\hat{J}(i, j)$ by $J_{white}(i, j)$.

Unfortunately, this procedure requires a lot of redundant ray tracing since the same rays are traced into the scene for \hat{J} as for J_{white} . Fortunately, a set point $p \in \mathbb{R}^3$ on a plane parameterized by $a + u \cdot b + v \cdot c$ with $a, b, c \in \mathbb{R}^3$ is uniquely determined by the parameters (u_p, v_p) with $a + u_p \cdot b + v_p \cdot c = p$. Thus the rendering of UV coordinates suffices to reconstruct the corresponding 3D point on the plane. Consequently only two channels are needed to save the positional information and the third channel of \hat{J}

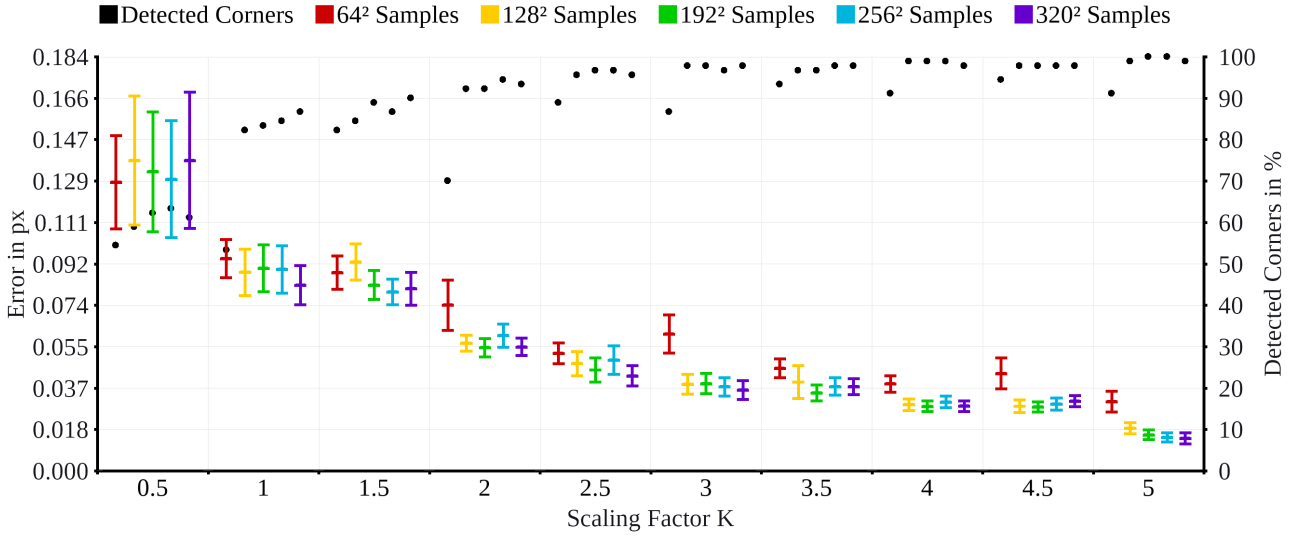


Figure 7. Method 1: Accuracy of interpolated corners for different numbers of samples and different scalings of J . For every combination of a scaling factor and a number of samples the resulting corners were compared to the reference solution rendered with $K = 10$ and 102400 samples. The average difference and standard error of mean (SEM) in terms of pixels are shown by the colored bars. Furthermore, the black dots show the ratio of detected corners corresponding to the respective colored bars.

can be used to store the ray proportion J_{white} . Analogous to the previous normalization, the positional image in terms of UV coordinates, J_{UV} is given by dividing the first two channels of \hat{J} by its third channel. The search for ground truth positions can then be performed by transferring $\{p_k\}$ into UV coordinates and searching these in J_{UV} .

For the second method the reparametrization of the planes is not necessary. Placing the two planes parallel to the worlds coordinate axes results in the points of the same plane having one fixed coordinate. This fixed coordinate can be saved in a small configuration file instead of the image channels of J_{near} and J_{far} , thus the freed channel in these images can again be used to save the ray proportion J_{white} . The final positional rendering J_{near} and J_{far} are then calculated by dividing the two positional channels by the ray proportion channel and subsequently replacing the latter by the externally saved fixed coordinate.

5.2. Number of Samples and Resolution

In order to assess the dependency of the resulting accuracy on the number of samples and the positional image resolution, we used a plenoptic camera setup with a double Gaussian 100mm objective, an MLA-to-sensor distance of 1.7mm, an MLA-to-main lens distance of 123.3mm and focal lengths of 1.9mm, 2.1mm and 2.3mm for the microlenses. The MLA as well as sensor have a size of $21.73mm \times 21.73mm$ and the MLA contains approximately 100×115 microlenses whereby the larger number of microlenses in the vertical is a result of the hexagonal ordering of the lenses. Furthermore, the thresholds for the constraints given in section 4.2 have been empirically chosen

as $\lambda_d = 0.15$ and $\lambda_\alpha = 10^\circ$. We would like to remark, that further tests confirmed, that the general conclusions of the following evaluation also hold for different threshold values as these mainly regulate the number of positional outliers. With this setup we rendered multiple images showing a checkerboard located at different depths with varying angles. For these images we applied our first approach for different combinations of sample numbers and resolutions. The results are presented in Fig. 7 where the mean and SEM of the differences between the determined pixel positions and the reference positions is shown. These results show, that increasing the scaling factor K significantly decreases the error and variance while the number of detected corners significantly improves. In contrast to this observation, the number of samples seems to have only a limited impact in the tested range. The results rendered with 320^2 samples show an average improvement of $0.0028 px$ for the pixel positions and 1.6% for the ratio of detected corners, compared to the results produced with 128^2 samples. Only positional images J rendered with significantly fewer samples seem to suffer from an imprecision caused by the low sample rate as the results produced with 64^2 samples suggest.

5.3. Comparison of the two Methods

The same combinations of sample numbers and resolutions shown in Fig. 7 were also used to evaluate the differences between the two proposed approaches. As Fig. 8 shows, the convergence behavior of the two plane approach with respect to the reference from the previous section is identical to that of the first method and the means of both methods only deviate from each other by less than less

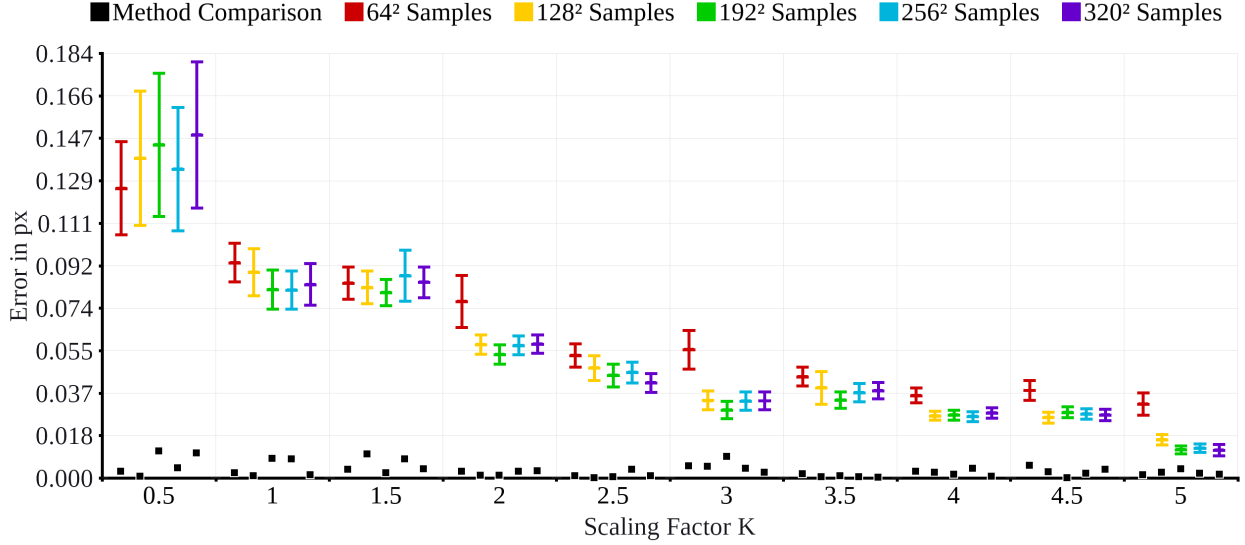


Figure 8. Method 2: Accuracy of corners calculated via the two plane method. The resulting corners were compared to the same reference solution as in Fig. 7. The average difference and SEM in terms of pixels are shown by the colored bars and the black squares show the absolute difference between the means of both methods for the respective combination of K and number of samples.

than 0.016 *px*. The remaining fluctuations between the two methods, which show no clear winner in regards of accuracy, are a result of two aspects. Firstly, the two plane method has the disadvantage of additional intersection calculations which can introduce further precision errors, especially since the ray tracing is usually done on GPUs with only single precision. And second, small positional errors can have different effects in both methods. While this error is located directly on the calibration pattern plane for the first method, the impact of a positional error in the two plane method varies with the pose of the calibration pattern since the error is located on the near or far plane.

However, since the difference in accuracy is negligible for a sufficiently large number of samples, the two plane method is recommended due to the significantly smaller render time. In our experiments we used two Nvidia Titan X and with that setup, the rendering of an image I or J with a resolution of $width \times height$ pixel and n samples per pixel took approximately $t \approx width \cdot height \cdot n \cdot 10^{-11}$ minutes. Furthermore, the time needed for searching the positions $\{p_k\}$, including the calculation of J from J_{near} and J_{far} in the second method, is in both cases by several orders of magnitude smaller than the rendering time. Accordingly, the two plane method is significantly faster for every dataset consisting of more than two images per camera setup.

5.4. Conclusion and Limitations

The proposed methods are able to produce highly accurate, realistic data for the evaluation of calibration methods and a wide range of cameras. However, they are limited regarding the geometry of the calibration object.

Throughout this work, it is assumed that the calibration pattern is placed on a plane, which excludes calibration objects, like checkerboard cubes, featuring a three dimensional structure. If the scene points $\{p(r)\}$ that are hit by the rays $r \in R^{(i,j)}$ traced from the pixel (i, j) are not located on a common plane, the calculation of $J(i, j)$ via simple averaging as described in Equation 3 is incorrect. To a limited extent this problem might be avoidable by using the two plane method. However, it remains to be analyzed whether the intersection of a more complex scene and the mean ray of a pixel can be used in the same manner as in this work.

Another problem of geometrical nature results from the proposed method for filtering outliers, where it is assumed, that a small neighborhood of pixels is projected to a grid on the calibration pattern plane as shown in Fig. 5. While this assumption holds true for most common camera setups, it is theoretically possible for an optical system to contain high frequency distortions which deform the grids even in the smallest neighborhoods. In this case it is recommended to skip the filter and simply render J or the two proxy planes with a significantly larger resolution such that the solution of the naive approach given by Equation 4 is sufficiently accurate.

6. Acknowledgment

This work was supported by the German Research Foundation, DFG, No. K02044/8-1 and the EU Horizon 2020 program under the Marie Skłodowska-Curie grant agreement No 676401.

References

- [1] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, 2000. 1, 2
- [2] J. Heikkilä, "Geometric camera calibration using circular control points," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 10, pp. 1066–1077, 2000. 1, 2
- [3] L. Lucchese and S. K. Mitra, "Using saddle points for sub-pixel feature detection in camera calibration targets," in *Asia-Pacific Conference on Circuits and Systems*, vol. 2. IEEE, 2002, pp. 191–195. 1, 2
- [4] E. H. Adelson and J. Y. Wang, "Single lens stereo with a plenoptic camera," *IEEE transactions on pattern analysis and machine intelligence*, vol. 14, no. 2, pp. 99–106, 1992. 1
- [5] A. Lumsdaine and T. Georgiev, "The focused plenoptic camera," in *Computational Photography (ICCP), 2009 IEEE International Conference on*. IEEE, 2009, pp. 1–8. 1
- [6] G. Lippmann, "Epreuves reversibles, photographies integrales," *Academie des sciences*, 446451, 1908. 1
- [7] T. Michels, A. Petersen, L. Palmieri, and R. Koch, "Simulation of plenoptic cameras," in *2018-3DTV-Conference: The True Vision-Capture, Transmission and Display of 3D Video (3DTV-CON)*. IEEE, 2018, pp. 1–4. 2, 3, 6
- [8] M. Potmesil and I. Chakravarty, "A lens and aperture camera model for synthetic image generation," *ACM SIGGRAPH Computer Graphics*, vol. 15, no. 3, pp. 297–305, 1981. 2
- [9] C. Kolb, D. Mitchell, P. Hanrahan *et al.*, "A realistic camera model for computer graphics," in *SIGGRAPH*, vol. 95, 1995, pp. 317–324. 2
- [10] J. Wu, C. Zheng, X. Hu, Y. Wang, and L. Zhang, "Realistic rendering of bokeh effect based on optical aberrations," *The Visual Computer*, vol. 26, no. 6-8, pp. 555–563, 2010. 2
- [11] J. Wu, C. Zheng, X. Hu, and F. Xu, "Rendering realistic spectral bokeh due to lens stops and aberrations," *The Visual Computer*, vol. 29, no. 1, pp. 41–52, 2013. 2
- [12] R. Ng, M. Levoy, M. Brédif, G. Duval, M. Horowitz, and P. Hanrahan, "Light field photography with a hand-held plenoptic camera," *Computer Science Technical Report CSTR*, vol. 2, no. 11, pp. 1–11, 2005. 2
- [13] Raytrix GmbH, accessed 07.06.2019. [Online]. Available: <https://www.raytrix.de/> 2
- [14] O. Fleischmann and R. Koch, "Lens-based depth estimation for multi-focus plenoptic cameras," in *German Conference on Pattern Recognition*. Springer, 2014, pp. 410–420. 2
- [15] R. Zhang, P. Liu, D. Liu, and G. Su, "Reconstruction of refocusing and all-in-focus images based on forward simulation model of plenoptic camera," *Optics Communications*, vol. 357, pp. 1–6, 2015. 2
- [16] C.-K. Liang and R. Ramamoorthi, "A light transport framework for lenslet light field cameras," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 2, p. 16, 2015. 2
- [17] B. Liu, Y. Yuan, S. Li, Y. Shuai, and H.-P. Tan, "Simulation of light-field camera imaging based on ray splitting monte carlo method," *Optics communications*, vol. 355, pp. 15–26, 2015. 2
- [18] T.-J. Li, S. Li, Y. Yuan, Y.-D. Liu, C.-L. Xu, Y. Shuai, and H.-P. Tan, "Multi-focused microlens array optimization and light field imaging study based on monte carlo method," *Optics express*, vol. 25, no. 7, pp. 8274–8287, 2017. 2
- [19] Blender Foundation, Blender 2.79c, accessed 07.06.2019. [Online]. Available: <https://www.blender.org/> 2
- [20] S. Nousias, F. Chadebecq, J. Pichat, P. Keane, S. Ourselin, and C. Bergeles, "Corner-based geometric calibration of multi-focus plenoptic cameras," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 957–965. 2
- [21] O. Johannsen, C. Heinze, B. Goldluecke, and C. Perwaß, "On the calibration of focused plenoptic cameras," in *Time-of-Flight and Depth Imaging. Sensors, Algorithms, and Applications*. Springer, 2013, pp. 302–317. 2
- [22] Y. Bok, H.-G. Jeon, and I. S. Kweon, "Geometric calibration of micro-lens-based light field cameras using line features," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 2, pp. 287–300, 2017. 2
- [23] H. Ha, M. Perdoch, H. Alismail, I. So Kweon, and Y. Sheikh, "Deltile grids for geometric camera calibration," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 5344–5352. 2
- [24] C. Perwass and L. Wietzke, "Single lens 3d-camera with extended depth-of-field," in *Human Vision and Electronic Imaging XVII*, vol. 8291. International Society for Optics and Photonics, 2012, p. 829108. 3