A Real-Time Depth Estimation Approach for a Focused Plenoptic Camera

Ross Vasko[1(✉)], Niclas Zeller[2,3], Franz Quint[2], and Uwe Stilla[3]

[1] The Ohio State University, Columbus, USA
vasko.38@osu.edu
[2] Karlsruhe University of Applied Sciences, Karlsruhe, Germany
{niclas.zeller,franz.quint}@hs-karlsruhe.de
[3] Technische Universität München, Munich, Germany
stilla@tum.de

Abstract. This paper presents an algorithm for real-time depth estimation with a focused plenoptic camera. The described algorithm is based on pixel-wise stereo-observations in the raw image recorded by the plenoptic camera which are combined in a probabilistic depth map. Additionally, we provide efficient methods for outlier removal based on a Naive Bayes classifier as well as depth refinement using a bilateral filter. We achieve a real-time performance for our algorithm by an optimized parallel implementation.

1 Introduction

The plenoptic camera, first described more than a century ago [1,2], provides a way for photographers to capture more information from a scene, compared to a traditional camera, by placing a micro lens array (MLA) in front of the camera sensor. The MLA allows for light-field information to be captured as a 4D function and the recorded image retains information about the structure of the scene itself [3,4]. In a single image captured from a plenoptic camera, information is recorded from several different viewpoints. By finding correspondences between these view points, depths for objects in the scene are able to be estimated.

In contrast with binocular stereo systems, the plenoptic camera is able to estimate depth information from a single image. This fact combined with the small size of a plenoptic camera makes it suitable for a compact visual odometry system which is able to measure metric scale.

In this paper, we present the adaptations made to parallelize a probabilistic depth estimation algorithm [5] and a method of depth map refinement for plenoptic cameras to allow for their use in real-time visual odometry systems.

1.1 Related Work

Due to the complexity of the light-field depth estimation problem, many different types of approaches to estimate a depth map are possible. Each of these

G. Bebis et al. (Eds.): ISVC 2015, Part II, LNCS 9475, pp. 70–80, 2015.
DOI: 10.1007/978-3-319-27863-6_7

approaches will be better suited towards different applications depending on their run-times and accuracies.

Some methods seek a globally optimal solution and find a dense depth map for the image such as [6,7].

Other approaches will calculate a sparse depth map using only local constraints by considering only the textured regions of the image such as [8,9].

In visual odometry and SLAM systems, we are specifically interested in having depth map estimation algorithms with a low complexity to allow for real-time systems. The algorithm presented in [5] uses a probabilistic approach to generate a sparse depth map from a single image captured by a plenoptic camera. This algorithm's low complexity makes it feasible to complete depth calculations in real-time and move closer to a real-time visual odometry system based on a plenoptic camera. We aim towards a depth estimation algorithm that will enable a visual odometry system for plenoptic cameras.

1.2 Outline of Work

In this paper we provide an approach for adapting an existing algorithm to be able to perform in real-time. In Sect. 2 we briefly introduce the focused plentoptic camera and in Sect. 3 we describe the already existing algorithm that we proposed in previous work [5]. Section 4 continues on to describe the post-processing of the depth map to increase its reliability in visual odometry systems. Section 5 details the adapation of this algorithm to allow it to run in real-time on a GPU. Lastly, Sect. 6 shows the results of our depth estimation algorithm along with the effects of the post-processing.

2 The Focused Plenoptic Camera

Rather than only capturing light intensities like a traditional camera, the plenoptic camera is able to record the direction of the light as well. The camera model used in this research is a focused plenoptic camera constructed by Raytrix. A diagram of this camera model is shown in Fig. 1. Additionally, the Raytrix camera features micro lenses of three different focal lengths arranged in a hexagonal grid to increase the depth of field. A subsection of an example raw image captured from the Raytrix camera is shown in Fig. 2.

Figure 1 also shows how one is able to estimate the virtual depth for some image point. Given a correspondence between pixels on the image sensor, the distance b in Fig. 1 can be estimated by triangulation [12]:

$$b = \frac{d \cdot B}{p_x} \tag{1}$$

where d is the distance between the corresponding micro lens centers, B is the constant, but not precisely known, distance between the image sensor and the MLA, and p_x is the disparity between the corresponding pixels under different micro lenses. A full derivation of this formula can be found in [5]. Note that

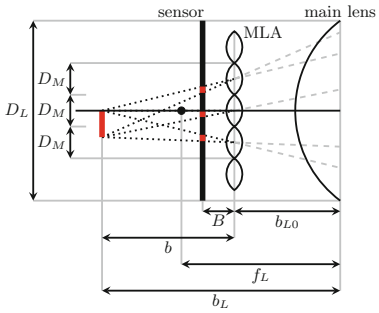


Fig. 1. Diagram of the inside of a focused plenoptic camera [10,11]. The image sensor is positioned in front of the virtual image, which is in the distance b behind the sensor.

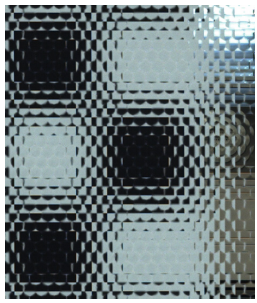


Fig. 2. Section of a raw image that was captured by a Raytrix camera. The image shows the edge of a chessboard with a portion of the background.

because we do not have a recorded value for B, we will calculate a virtual depth [9], v, relative to B instead. Notice that as the disparity between pixels increases, the virtual depth will decrease.

$$v = \frac{b}{B} = \frac{d}{p_x} \tag{2}$$

3 A Probabilistic Virtual Depth Map Estimation Algorithm

Once an image is captured from a plenoptic camera, we are able to seek a correspondence between pixels in the raw image (Fig. 2) to estimate the virtual depths for pixels in textured regions. We are able to complete this correspondence quickly by only searching regions of the image that a match could possibly be for a pixel. To address the sensor noise, we use a probabilistic approach and calculate a variance for each depth estimate as well. This section is provided as a brief overview for how the depth estimations are performed for each pixel to allow for greater insight in later sections. A full description of this algorithm can be found in [5].

The MLA of the Raytrix camera has the micro lenses arranged in a hexagonal pattern. On the MLA, we define a graph of baselines from the center of one micro lens to the center of all micro lenses to the right of it. The search for the matching pixels is done across these baselines because it is guaranteed that corresponding pixels will be on the same baseline. It is only necessary to search to the right of a microlens because it is only required that a disparity between pixel pairs is found once. Figure 3 shows the micro lens array along with several of the defined baselines. Each baseline is defined by a vector $\mathbf{e}_p$ where $\|\mathbf{e}_p\| = 1$ pixel and a distance in pixels, d. Since the micro image can be considered to be rectified, $\mathbf{e}_p$ also defines the direction of the epipolar line for each pixel under the microlens.

To begin the matching process for pixel $\mathbf{x}_R = (x_R, y_R)^T$ with intensity $I(\mathbf{x}_R)$, the baseline $\mathbf{e}_p$ with the shortest distance that has not yet been considered for the current pixel is chosen. It is then checked that the gradient of $\mathbf{x}_R$ meets the following condition, where T_H is a predefined threshold.

$$|\mathbf{g}_I(\mathbf{x}_R)^T \mathbf{e}_p| \geq T_H \tag{3}$$

This is done to verify that $\mathbf{x}_R$ has sufficient contrast in the direction of the epipolar line. The search for a corresponding pixel is then performed along the line

$$\mathbf{x}^s_R(p_x) = \mathbf{x}^s_{R0} + p_x \cdot \mathbf{e}_p \tag{4}$$

where $\mathbf{x}^s_{R0}$ is defined by

$$\mathbf{x}^s_{R0} = \mathbf{x}_R + d \cdot \mathbf{e}_p \tag{5}$$

The disparity that best matches two pixels is the p_x that minimizes the sum of squared intensity error between two 5×1 pixel patches below

$$e_{ISS}(p_x) = \sum_{k=-2}^{2} \left[I(\mathbf{x}_R + k\mathbf{e}_p) - I(\mathbf{x}^s_R(p_x) + k\mathbf{e}_p) \right]^2 \tag{6}$$

An inverse virtual depth, $z(\mathbf{x}_R)$ is then calculated from the disparity that minimizes $e_{ISS}(p_x)$ for the current pixel. Additionally, a variance σ_z^2 can be defined for the inverse virtual depth observation as given in [5]. We record and work with the inverse virtual depths because they can be considered to be Gaussian distributed [5]. We continue to advance and consider the next shortest unprocessed baseline for the current pixel until none of the baselines at a single distance were able to find pixel correspondences under a certain error threshold. The multiple pixel correspondences are incorporated into a probabilistic estimate, similar to the update step in a Kalman Filter.

If a correspondence was able to be found for pixel $\mathbf{x}_R$, we consider this pixel to be valid and assign it an inverse virtual depth estimation based on all observed disparities. If a correspondence was not found, the pixel is marked invalid. Once the correspondence is attempted for each pixel, we project the raw inverse virtual depth map to a 3D space [5].

4 Virtual Depth Map Post-processing

With the probabilistic approach, there is expected to be missing and incorrect depth values in the virtual depth map once the algorithm completes. For this depth map to be used later in visual odometry applications, it is necessary that we fill in missing depth values in regions where we can confidently assign a depth value as well as remove and smooth incorrect estimates in our depth map.

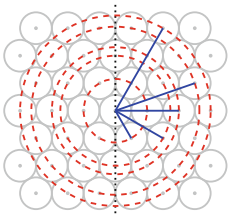


Fig. 3. The hexagonal microlens grid of the camera used with various baselines that can be searched for matches.

Fig. 4. A heatmap of the time spent on correspondence calculations for each pixel. The lighter the pixel color is, the longer is spent performing the correspondence calculations.

4.1 Removing Noise and Estimating Missing Depth Values

Due to only searching in image regions with a high gradient, we receive a sparse depth map with depth information only for the textured regions of the image. But from noise and imperfections in the image, there are incorrect depth values recorded for untextured regions as well as missing values in textured regions. We seek to remove depth values in unreliable or unstable regions and fill in data where we can confidently estimate a value.

First, we are able to remove depth pixels with a variance σ_z^2 that do not meet a previously defined threshold,

$$\sigma_z^2(\mathbf{x}_v) < T(z) = \beta \cdot z(\mathbf{x}_v)^3 \tag{7}$$

This threshold removes inaccurate depth estimates with an unexpectedly high variance for their virtual depth. A full explanation of this threshold can be found in [5].

Additionally, depth values are filled in or removed based on how much depth information is in some nearby region. If the depth calculations for a region were indeed correct for a pixel, we expect that there are other pixels with valid depth calculations in a nearby window. Due to the perspective projection, which is performed by each of the micro lenses, objects with a high virtual depth occur smaller in the micro images than objects with a low virtual depth. Additionally, virtual image points with a high virtual depth are observed by more micro lenses, as can be seen by Fig. 1. Consequently, regions with high virtual depths that are projected back from the micro images consist of more points which are spread over a larger region than regions that have small virtual depths.

Thus, we define the following relationship in Eq. (8), which defines the neighborhood for a pixel,

$$m = c \cdot v_n \tag{8}$$

where m is the width of a window around pixel p, v_n is the average virtual depth in some fixed neighborhood around p, and c is a fixed constant. Within this $m \times m$ window, we expect to find other pixels with valid recorded depth values in a high

density if p is indeed a valid depth estimation. Also, if p is the result of noise in this image, the density of valid pixels in the $m \times m$ neighborhood should be low.

Once m is calculated, we measure the density of valid pixels with recorded depth information and then a Naive Bayes classifier is used to make an estimate on whether or not the pixel can be assumed to be a part of some textured region with valid depth information:

$$P(V_k|\rho) \propto P(V_k)P(\rho|V_k) \tag{9}$$

where V_k is a label representing the validity of the current pixel and ρ represents the density information of neighboring pixels. The most probable label for the pixel's validity is then selected given the density information in the $m \times m$ window.

4.2 Depth Map Refinement

In the final depth map refinement step, we wish to smooth the depth estimates to reduce noise. To accomplish this, we use a bilateral filter [13] to remove the noise while preserving the edges. The filtered inverse virtual depth after an iteration of the bilateral filter is defined by:

$$Z^{filtered}(\mathbf{x}) = \sum_{\mathbf{p} \in N_\mathbf{x}} \frac{w_i(|Z(\mathbf{x}) - Z(\mathbf{p})|)w_d(|\mathbf{x} - \mathbf{p}|)w_v(V_\mathbf{p})Z(\mathbf{p})}{\sum_{\mathbf{p} \in N_\mathbf{x}} w_i(|Z(\mathbf{x}) - Z(\mathbf{p})|)w_d(|\mathbf{x} - \mathbf{p}|)w_v(V_\mathbf{p})} \tag{10}$$

where $\mathbf{x}$ and $\mathbf{p}$ are pixel coordinates, $N_\mathbf{x}$ is a window around $\mathbf{x}$, $Z(\mathbf{x})$ is an inverse virtual depth, and V is a variance. The weighting functions were chosen to be Geman-McClure functions. In general, $w_a(x)$ is defined to be:

$$w_a(x) = \frac{x^2}{x^2 + \sigma_a^2} \tag{11}$$

where σ_a^2 is a predefined variance for each of the weighting functions. The weighting functions are designed so that the inverse virtual depth value for each pixel is only smoothed with other inverse virtual depth values that are reliable and similar in depth, to prevent the smoothing of edges. The weighting functions include w_i for the inverse virtual depth difference, w_d for the Euclidean distance difference, and w_v for the variance.

5 Implementation on a CUDA Device

As previously described, we attempt to find a correspondence between pixels in neighboring micro images by searching through various baselines. For a single pixel in the image, the correspondence search consists of several independent operations along the baselines. These independent calculations are able to be completed much more quickly with the use of parallelization. This algorithm was adapted to run on a CUDA capable GPU so that instead of finding the

correspondences for each pixel sequentially, threads work in parallel to find correspondences for their assigned pixel.

To eliminate any communication between threads during the correspondence search, a single thread completes all of the correspondence calculations required for one pixel. The time to complete the correspondence calculations for a single pixel depends both on the contrast in the pixel's region as well as the virtual depth of the pixel. If a pixel does not have much contrast in its region the correspondence calculations will finish quickly, as correspondences will not be able to be found due to a low gradient. Also, as the virtual depth of a pixel increases, the correspondence calculations will take longer because the pixel will appear in many micro images and many correspondences will have to be calculated, based on the relationship shown in Fig. 1. Figure 4 shows a heatmap of the time taken for the correspondence calculations for each pixel in a micro image. It can be seen by Fig. 4 that it is possible for the correspondence calculations of pixels to all take significantly different times.

The varying computation times will cause warp divergence, as each thread in a warp will be spending different amounts of time on its correspondence calculations. This could lead to a performance loss because the threads that were able to finish their correspondence calculations will not be doing anything while the remaining threads in the warp finish their computations. To address this, a list of pixels to be processed is built and threads process the next available pixel to minimize the effect of correspondence calculations taking different amounts of time for different pixels. This allows for a single thread to move on to a new pixel to process instead of being required to wait for all of the warp's threads to complete their correspondence calculations.

To gain a further decrease in run-time, it is possible to skip calculating the correspondences for some of the pixels in the image. Skipping the correspondence calculations for pixels in the image decreases the number of recorded depths in the final depth map, but this decrease in density is able to be rectified by the post-processing methods described.

To make the Naive Bayes classifier computations suitable for real-time and interactive applications, a Summed Area Table (SAT) [14] is used to count the number of valid pixels in the previously described $m \times m$ neighborhoods. Rather than requiring a number of memory accesses dependent on m, an SAT allows for these computations to be done in constant time, requiring only four memory accesses for each neighborhood to be computed.

We find that the CUDA implementation allows this algorithm to be suitable for real-time systems. Evaluations for the algorithm and the effects of skipping pixels are given in the Sect. 6.

6 Results

In this section we present the performance of the parallel implementation of our algorithm along with the effects of the post-processing by displaying the results generated from test images. We wish to show that the modifications

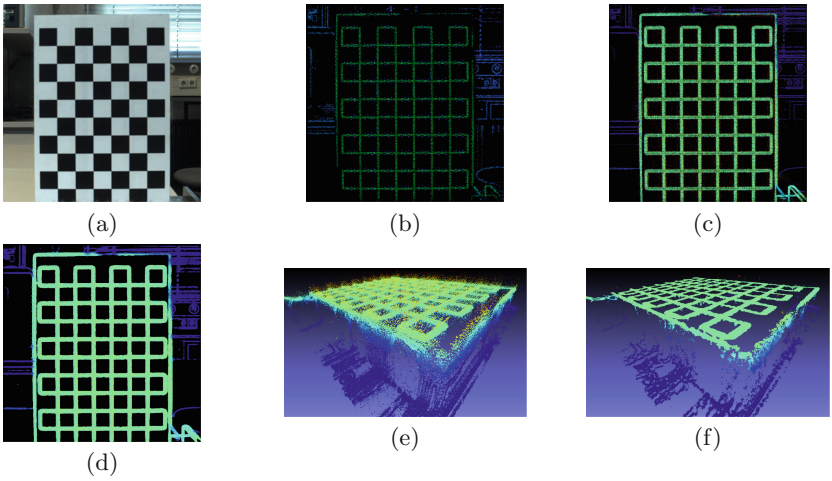
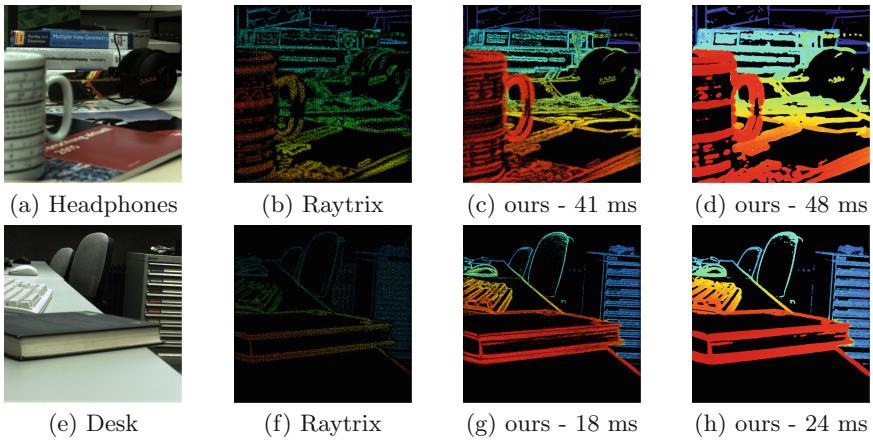


Fig. 5. (a) shows the totally focused image of the chessboard, (b) shows the results from the commercial software [9] with similar settings, (c) shows the original depth map generated by our algorithm, (d) shows the depth map with pixel validities adjusted, (e) shows a point cloud of the unfiltered depth information, (f) shows a point cloud of the filtered depth data. The output depth map has a resolution of 1024×1024.

made to this algorithm allow for it to be used with plenoptic cameras in real-time visual odometry systems. All results were computed on a NVIDIA GeForce GTX TITAN GPU. In the sample images, we compare our algorithm with software from Raytrix [9]. The Raytrix software [9] has many parameters that affect the results of the depth map. We chose the settings by finding parameters that produced results with similar pixel densities. We have chosen to not include the timings for the commercial software [9], because the timings were very sensitive to changes in the settings. Nevertheless, we found our timings to always be comparable or faster than the timings for the commercial software.

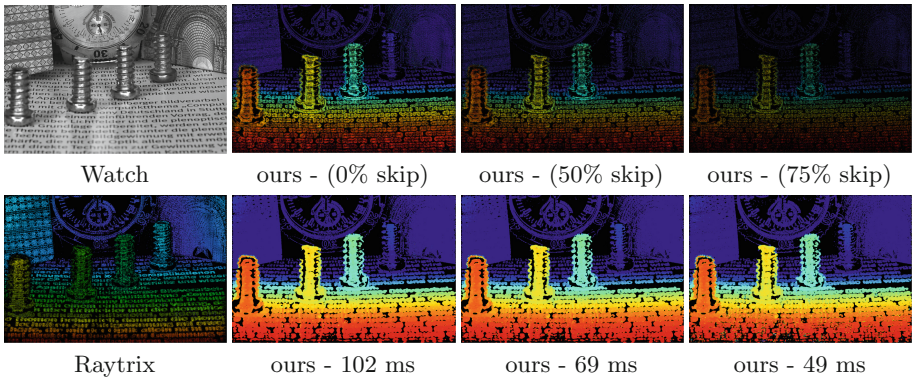
First, a walkthrough of the algorithm will be shown in detail for a sample image of a chessboard and then the results from additional images will be shown. Once the depth estimates above the previously discussed variance threshold are removed, the validity of pixels are readjusted using the Naive Bayes classifier. This resulting image is then refined using the bilateral filter. Results from this process are displayed in Fig. 5. For this chessboard image, the depth estimation takes 25 ms, the pixel validity adjustment takes approximately 3 ms, and the filtering takes approximately 4 ms. A longer period of time can be spent on the filtering to further smooth the image if required for the application. Additionally, Fig. 6 displays several filtered depth maps generated along with the total time taken to perform the depth estimations and then filter the image. In the displayed images, the black pixels are in regions of weak texture and are marked invalid with no recorded depth information.



(a) Headphones (b) Raytrix (c) ours - 41 ms (d) ours - 48 ms

(e) Desk (f) Raytrix (g) ours - 18 ms (h) ours - 24 ms

Fig. 6. (a and e) show the totally focused intensity images, (b and f) show depth maps from the commericial software, (c and g) show our original depth maps for the scene, (d and h) show the final post-processed depth maps from our algorithm. The time under the images is the total time taken to produce the depth map. Both output depth maps have resolutions of 1024×1024.



Watch ours - (0% skip) ours - (50% skip) ours - (75% skip)

Raytrix ours - 102 ms ours - 69 ms ours - 49 ms

Fig. 7. The effects of pixel skipping on the Watch image. Column 1 contains an image of the original scene along with Rayrix's depth map. Columns 2–4 contain original and post-processed images from our algorithm with a percentage of pixels skipped and the computation time. The output depth maps have resolutions of 2004×1332.

The resulting depth maps for skipping pixels, as well as post-processing these images, are shown in Fig. 7. It can be seen by this figure that by skipping pixels we are able to save a significant amount of time in our depth map calculations. Although the quality of the original depth map is decreased significantly, through post-processing, we are able to restore the depth map.

It can be seen from the results that as the complexity of the scene increases, the run-time of the algorithm also increases. The algorithm seeks pixel corre-

spondences, so if many pixel matches are found, the algorithm will take longer. Additionally, scenes with close objects take longer to calculate a depth map for because points near the camera appear in many micro lenses and many correspondences are calculated. But, the many parameters available in the algorithm make it possible to decrease the computation time and keep reasonable results, as needed for the application.

7 Conclusion

In this paper we described the modifications made to a previous algorithm to make it suitable for use in a real-time visual odometry system as well as post-processing methods to improve the image quality. We gave a method for removing and adding depth estimations using a Naive Bayes classifier. Also, a refinement method using a bilateral filter was described. We then detailed the process of parallelizing and optimizing the existing algorithm.

Our results section shows that we are able to compute and post-process many depth maps per second. The computation times can be decreased further through the refinement of parameters, if it is found that an application does not require as precise depth information. From our timings and results, we find that our algorithm is suitable for real-time visual odometry systems.

References

1. Ives, F.E.: Parallax stereogram and process of making same. US Patent 725,567. Google Patents (1903). http://www.google.com/patents/US725567
2. Lippmann, G.: Epreuves reversibles. photographies integrales. Comptes Rendus De l'Academie Des Sciences De Paris, vol. 146, pp. 446–451 (1908)
3. Adelson, E.H., Wang, J.Y.A.: Single lens stereo with a plenoptic camera. IEEE Trans. Pattern Anal. Mach. Intell. **14**, 99–106 (1992)
4. Gortler, S.J., Grzeszczuk, R., Szeliski, R., Cohen, M.F.: The lumigraph. In: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH, pp. 43–54. ACM, New York, NY, USA (1996)
5. Zeller, N., Quint, F., Stilla, U.: Establishing a probabilistic depth map from focused plenoptic cameras. In: Proceedings of International Conference on 3D Vision (3DV), pp. 91–99 (2015)
6. Wanner, S., Goldluecke, B.: Globally consistent depth labeling of 4D lightfields. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2012)
7. Wanner, S., Goldluecke, B.: Variation light field analysis fo disparity estimation and super-resolution. IEEE Trans. Pattern Anal. Mach. Intell. **36**, 606–619 (2014)
8. Bishop, T.E., Favaro, P.: Full-resolution depth map estimation from an aliased plenoptic light field. In: Kimmel, R., Klette, R., Sugimoto, A. (eds.) ACCV 2010, Part II. LNCS, vol. 6493, pp. 186–200. Springer, Heidelberg (2011)
9. Perwaß, C., Wietzke, L.: Single lens 3D-camera with extended depth-of-field. In: Proceedings of SPIE 8291, Human Vision and Electronic Imaging XVII, Burlingame, California, USA (2012)

10. Lumsdaine, A., Georgiev, T.: Full resolution lightfield rendering. Technical report, Adobe Systems, Inc. (2008)
11. Lumsdaine, A., Georgiev, T.: The focused plenoptic camera. In: Proceedings of IEEE International Conference on Computational Photography (ICCP), San Francisco, CA, pp. 1–8 (2009)
12. Zeller, N., Quint, F., Stilla, U.: Calibration and accuracy analysis of a focused plenoptic camera. ISPRS Ann. Photogrammetry Remote Sens. Spatial Inf. Sci. **II–3**, 207–212 (2014)
13. Paris, S., Kornprobst, P., Tumblin, J., Durand, F.: Bilateral Filtering: Theory and Applications. Now Publishers Inc, Hanover (2009)
14. Crow, F.C.: Summed-area tables for texture mapping. ACM SIGGRAPH Comput. Graph. **18**, 207–212 (1984)