NEAR REAL-TIME LIGHT FIELD RECONSTRUCTION AND RENDERING FOR ON-LINE LIGHT FIELD EVALUATION

Matthias Ziegler[1], Mojtaba Bemana[2], Joachim Keinert[1], Karol Myszkowski[2]

[1] Fraunhofer Institute for Integrated Circuits
Moving Picture Technologies
91058 Erlangen, Germany

[2] Max Planck Institute for Informatics
Computer Graphics
66123 Saarbrücken, Germany

ABSTRACT

Recently, multi-camera systems have become very popular to capture VR content. Capturing such content is complex and different types of errors are likely to appear. Unfortunately, the data processing happens off-line and reviewing the possibly erroneous result takes place a long time after the capture. In many cases, the scene needs to be captured once again.
In order to speed-up this trial-and-error procedure, real-time or near-real time processing pipelines can help. In this case study, we devise a light field processing chain for near-real time depth reconstruction and rendering. A single computer system with recent hardware processes a dataset with more than 20fps. The subsequent view synthesis runs at up to 90fps, supports head mounted displays and is integrated inside Unreal Engine. Our results show that the visual quality is close to other, state-of-the art depth reconstruction and rendering pipelines that require longer processing times.

Index Terms— Light field reconstruction, DIBR, real-time view synthesis

1. INTRODUCTION

In June 2018, Overbeck et al. [3] showed the amazing features that light field imaging can provide in virtual reality (VR) environments. The authors put a set of cameras on a rotating platform. Their demo allows a user to explore a static, pre-captured scene with full six degrees of freedom (6-Dof) on a head mounted display (HMD). The movement of the user is restricted to a small sphere (about 0.5m diameter). Besides technological challenges the large amount of carefully captured scenes (e.g. inside a space shuttle) makes their demo especially valuable.
Capturing an interesting scene is even more challenging if dynamic applications are considered. Often, capturing dynamic, live-action light field scenes is an erroneous procedure. From a technical perspective, scene, cameras and actors need to be setup such that the visual parallax is maximized while the depth reconstruction still yields satisfying results.
The ability to review a captured scene on set with no or only very short delay is a key element for such light field based VR applications. Here, reviewing a scene means that a user likes to see the captured light field dataset with the original framerate inside a real-time rendering environment possibly with a HMD as presented in [4].
Against this background, we study the problem of high-performance light field processing and rendering for light field video. The considered system comprises a single computer with a recent GPU. After image acquisition, we reconstruct per-view and per frame depth using Dąbała's method [5]. The system holds the images and depth maps in memory such that the following rendering algorithm can directly access them. In our experiment, we compare off-line and on-line depth reconstruction methods and evaluate the quality loss.

2. PREVIOUS WORK

Since the original publication by Hanrahan [6] in 1996, light-field imaging has gained more and more interest with many contributions each year that address different types of hardware setups and image processing methods. Many publications employ wide-baseline multi-camera systems [1, 2, 4, 5, 7, 8] in combination with depth-reconstruction or 3D reconstruction and matching rendering techniques. Typically, the proposed software pipelines reconstruct depth- or disparity maps in an offline process. An up-following view synthesis step may be on-line or off-line. Though many authors also address computational efficiency, no method or system addresses on set light-field reconstruction and play-back with limited hardware.
For the remaining part of this article, we focus on systems that comprise a set of individual cameras and thus are in principle capable of capturing dynamic scenes. In this section, we review related works and put the properties of our system in contrast.
The method developed by Ziegler et al. [4] in 2017 synthesizes novel views in real-time using DIBR and addresses VR applications. The underlying depth-maps are obtained in a user-steered offline process.
In 2018, Chuchvara et al. [9] presented their work on fast and accurate light field reconstruction based on superpixel segmentation. The authors report timing results ranging between 204ms and 1584ms per view on a NVIDIA Quadro M1000M GPU for the depth reconstruction. Timing results for the off-line view synthesis were not provided
Similarly, Dąbała et al. [5] presented and evaluated a very efficient multi-scale approach for depth reconstruction. The authors report between 147ms and 277ms for a complete light field dataset with 0.5MPixel per view. As before, NVS is not in focus and timing results are not given.
The method developed by Yao et al. [10] addresses real-time NVS but uses a special light-field representation format that is obtained in an off-line process. Furthermore, the rendering method does not support view reconstruction perpendicular to the camera plane. In our experiments, we learned that this is an important feature in the context of VR applications. DIBR naturally provides such functionality.

Similar to light field reconstruction, the works developed by Dou et al. [11], Collet et al. [7], De Aguilar et al [12] and Keller et al. [13] aim to reconstruct a 3D model for a subsequent rendering. Explicit 3D reconstruction or surface reconstruction can be beneficial in terms of rendering performance as modern GPU's perform this task with highest efficiency. Especially the works by Dou and Keller also target real-time applications. However, the systems comprise many computers or incorporate active depth sensing. In contrast to DIBR, surface reconstruction adds an additional processing step. This reduced computational efficiency compared to pure DIBR.

In 2017, Penner [14] presented a novel view synthesis algorithm based on a technique called "Soft 3D reconstruction". They point out that for a good view-synthesis result the quality of underlying stereo-matching methods is of minor importance. However, the computational performance of the underlying depth reconstruction step remains unclear.

Gaming engines such as Unreal Engine (UE) or Unity render a mesh or a point-cloud with very high performance since a GPU typically performs the required operations. The overall number of vertices and faces to be projected is the major factor that influences rendering performance.

3. PROPOSED METHOD

3.1. Online depth reconstruction

Our online depth reconstruction is based on former work published by Dąbała et al. [5]. We re-implemented their algorithm with CUDA and optimized the software for efficient sequence processing. Up following, we give a brief overview on the original method.

Dąbała's method works in a coarse to fine manner and processes a pre-rectified *MxN* light field frame in one run. Initially, the algorithm builds up a multi-scale image pyramid for all input images with a resolution factor 2 between each level. For each image and each level, the software also builds up Census descriptors for the following matching step. Starting with the lowest resolution level, the algorithm matches each image with up to 4 directly adjacent images. Per pixel and per stereo pair, a low number of depth candidates (i.e. 5) is evaluated. This is a key element of Dabala's method. In addition to the matching costs, the method computes a confidence value that estimates the reliability of a depth candidate. Subsequently, a merging, consolidation and filtering step combines and refines the individual stereo pairs yielding one depth map for each input image. The next level uses this depth map as a guide and refines the input disparity map. The algorithm ends when the original resolution has been reached.

3.2. VR view synthesis model

As outlined, we aim to review a dynamic light field dataset in a VR environment. Therefore, our system relies on DIBR to generate a novel view from RGBD input data. However, depth maps only provide an implicit 3D model. Especially in a VR environment, it is extremely important that the light field data and the surrounding CG environment behave consistently. The parallax of the light field and the CG environment need to match. In order to solve this issue, our rendering model uses an intermediate screen to transfer the light field into the virtual environment.

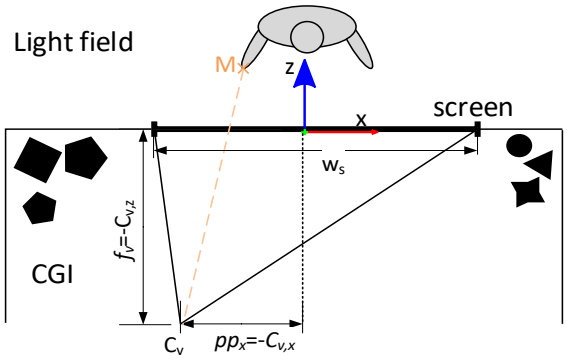

Figure 1: Our DIBR algorithm uses a screen element to integrate light fields into a virtual environment based on the observer position C_V, the screen position and the reconstructed per pixel disparity.

Figure 1 represents the top view of a level inside a gaming framework such as UE. The player, depicted by the camera C_v can move freely through the scene. The intermediate screen behaves like a window that separates CG domain and light field domain.

In case of classical 3D projection, a point *M* belonging to the actor can be projected into the views of the spectator and rays intersect with the screen at distinct points. Equivalently, corresponding points that individually emerge as rays from these intersection points appear as being situated behind the screen. Our proposed method therefore directly computes two individual images for the left and right view associated with the position of the observer C_V. The rendering engine then maps those images onto the screen and then further in the spectator's view.

We implemented this rendering method as plug-in for UE version 4.20. The core shader programs are written in High Level Shading Language (HLSL). Within the rendering process of each frame, our plug-in computes individual images for the left and right eye. The rendering of an image itself consists of forward warping, filtering, backward- warping, view blending and inpainting.

The implementation uses a dual-buffer system for dynamic light-field import and rendering. A first buffer is located in host memory and a second buffer is located in GPU memory. At the current stage, the system works with uncompressed image data. In terms of performance, we can render light-fields with about 15MP in total. Then, rendering one view is below 5ms yielding more than 90 frames in VR as required for HMD systems.

4. EXPERIMENTAL SETUP

In our experimental setup, we evaluate the performance of our selected method in terms of quality and performance, both for static scenes as well as for dynamic scenes. Given a sparsely sampled input light field, we firstly reconstruct the per view depth as described. Afterwards, we render intermediate view positions using our real-time DIBR plug-in for UE.

Table 1: SSIM scores on 3 static datasets. The comparison is performed on a 3x3 subset per scene.

SSIM score	Dąbała et al.	Chuchvara et al.	Ours
Truck	0.95	0.975	0.971
Bracelet	0.98	0.974	0.956
Jelly Beans	n.a.	0.982	0.972

Table 2: PSNR scores for 3 static datasets. The scores show average PSNR results computed from all views that are not contained in the input light field.

PSNR score [db]	Yao et al.	Ours (5x5 subset)	Ours (3x3 subset)
Amethyst	31	37.46	34.50
Truck	n.a.	38.63	36.75
Chess	32.9	36.15	33.70

4.1. Quantitative evaluation on static scenes

Here, we follow the evaluation procedure as performed by Chuchvara et al. [9]. The authors evaluate their method on scenes taken from the Stanford Lightfield Archive [1]. As input, we also use a regular 3x3 subset and reconstruct every second image in the middle row. Table 1 shows the obtained SSIM scores in comparison with previous work from Dąbała et al. and Chuchvara et al. On average, our results are slightly lower than the results reported by Chuchvara. Given the very limited amount of time (see section 4.4), this low degradation might be acceptable for a preview or play-back with low delay. In addition, Table 2 lists PSNR results for Yao's method and our method. Here, we compare Yao's results with a 3x3 and a 5x5 subset of the original 17x17 input data. Our results show that we achieve slightly higher quality as Yao given only on a 3x3 subset. On a 5x5 subset, our method is clearly better.
In addition to this numerical results, Figure 2 shows rendering details for four selected datasets. For the first three datasets, a 3x3 subset has been used as input data.

4.2. Quantitative evaluation on dynamic scenes

In this case, we compare quantitative image quality on video datasets as provided by Dąbała et al. [5] and also by Sabater et al. [2]. Instead of skipping views during depth reconstruction and view synthesis, we compute depth maps for all input views. For evaluation, we discard depth for a specific view and reconstruct this view from surrounding camera positions. In our opinion, this comes close to practical applications but allows for fair evaluation. For the following evaluation, we discarded and reconstructed the second camera in the first row on the dataset Painter.
Figure 3 shows the results for frames 0 to 60. The curves compare results of DERS [15] and our tested combination of depth reconstruction and view synthesis on full resolution images. Here, this combination loses about 0.03 points on average compared to DERS. In addition to full resolution reconstruction, we evaluate also on half resolution using downscaled input images. For evaluation, we upscale the result and compare to the full resolution reference image. In order to quantify this result, Figure 3 shows two additional curves. These curves evaluate the resolution loss as introduced by a low pass filter by a factor of 3 and 4, respectively. Our results range between these curves. This indicates that the analyzed light field processing loses more than a factor 3, but less than a factor 4 in terms of true resolution.
In our opinion, when reviewed on a recent HMD with relative low angular resolution, the quality and resolution of our tested method satisfies the requirements in these environments. According to our experience, a proper integration of the rendered content is more important than having the highest possible resolution.

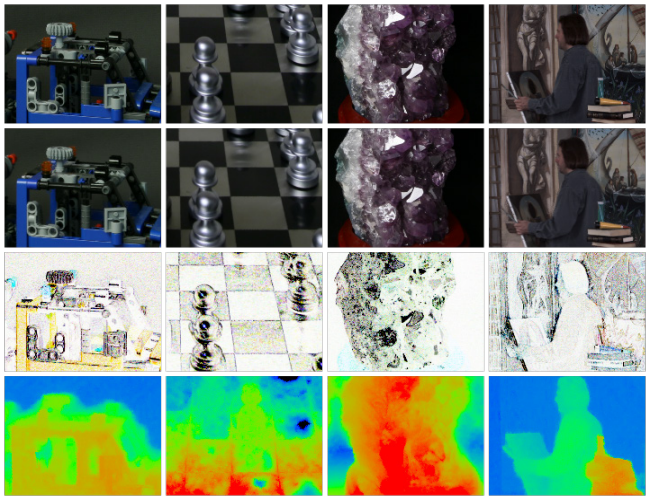

Figure 2: Rendering results on Truck, Chess, Amethyst [1] and Painter [2]. The first and second rows show the reference image and the reconstructed image. The third row highlights the differences. The last row shows a nearby input disparity map.

4.3. Depth reconstruction performance

At the current stage, our system reads pre-captured input data, streams the data into GPU memory and reconstructs depth for all views. Up following, the software transfers the data back and stores the data. Memory transfers are executed asynchronously so that they do not block computational GPU resources.
Table 3 gives timing details for the 4x4 dataset *Painter* from a system perspective and Table 4 lists the timing results for the individual processing steps on the GPU. The processing of this dataset including image reading from a RAM drive and image writing on a RAM drive takes 13.8 seconds, corresponding to 23 fps or 43.5ms per frame. This is a bit faster than the sum of all elements in Table 3 (46.7ms) and is explainable by the parallel

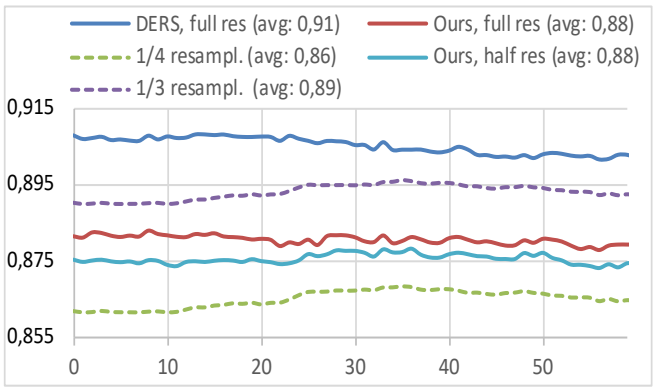

Figure 3: SSIM scores for the dataset Painter for frames 0 to 60.

Table 3: Timing results from system perspective in ms.

Image reading	5.65
CPU to GPU memory transfer	3.91
Depth reconstruction	18.11
GPU to CPU memory transfer	2.83
Image saving	16.31

execution in our implementation. Overall, the performance comes close to the requirements for real-time applications that start with about 25 fps. With the latest generation of graphics cards or several graphics cards, the performance could be increased even further.

The method proposed by Chuchvara requires 5.3s for a 3x3 light field (588ms per view) on a GPU with 1017 GFLOPS (NVIDIA QUADRO M1000M). The NVIDIA 1080Ti used in our experiments provides 11340 GFLOPS.

If Chuchvara's method scales linearly with the GPU's capabilities, the method would still require about 474ms for a complete light field. Hence, our method is computationally more efficient (36.97ms for a complete light field on a comparable dataset).

5. CONCLUSION

In this work, we analyzed near real-time depth reconstruction and real-time view synthesis. Our aim was to enable on set light-field play-back in VR with parallax. Our results show that we achieve near real-time performance for the depth reconstruction and real-time view synthesis on a single PC with a single GPU. The system processes a 4x4 light field video with 317 frames in 13.8 seconds. When reviewed on an HMD or on a display, the result shows only very little visual artifacts. The multi-scale approach of the underlying depth-reconstruction is advantageous since its low-pass characteristic hides possible view-synthesis artifacts.

In future, we'd like to integrate the temporal domain in our algorithm. Here, further improvements in terms of performance and quality can be expected. If possible, we would like to integrate image acquisition, depth processing and rendering in a single software. The results as presented indicate that such a software could be beneficial.

The Fraunhofer and the Max Planck cooperation program within the framework of the German pact for research and innovation (PFI) supported this work.

6. REFERENCES

[1] A. Adams, "The (New) Stanford Light Field Archive," 2008.

[2] N. Sabater, G. Boisson, B. Vandame, P. Kerbiriou, F. Babon, M. Hog, R. Gendrot, T. Langlois, O. Bureller, and A. Schubert, "Dataset and Pipeline for Multi-view Light-Field Video," in Computer Vision and Pattern Recognition Workshops (CVPRW), 2017 IEEE Conference on, 2017, pp. 1743-1753.

Table 4: Depth reconstruction timing results for two light-field datasets and specific processing steps in ms.

Light field views	3x3	4x4
Image resolution	1920 x 1080	960 x 540
Creating Mip-Map	3.99	1.80
Matching	7.05	2.90
Consolidation	3.91	2.32
Smoothening	17.65	8.33
Up sampling	4.37	2.07
Total	36.97	17.42

[3] R. S. Overbeck, D. Erickson, D. Evangelakos, and P. Debevec, "Welcome to light fields," in ACM SIGGRAPH 2018 Virtual, Augmented, and Mixed Reality, Vancouver, British Columbia, Canada, 2018, pp. 32.

[4] M. Ziegler, J. Keinert, N. Holzer, T. Wolf, T. Jaschke, R. op het Veld, F. S. Zakeri, and S. Foessel, "Immersive virtual reality for live-action video using camera arrays," in IBC Conference, Amsterdam, Netherlands, 2017, pp. 1-8.

[5] Ł. Dąbała, M. Ziegler, P. Didyk, F. Zilly, J. Keinert, K. Myszkowski, H. P. Seidel, P. Rokita, and T. Ritschel, "Efficient Multi-image Correspondences for On-line Light Field Video Processing," in Computer Graphics Forum, 2016, pp. 401-410.

[6] M. Levoy, and P. Hanrahan, "Light field rendering." pp. 31-42.

[7] A. Collet, M. Chuang, P. Sweeney, D. Gillett, D. Evseev, D. Calabrese, H. Hoppe, A. Kirk, and S. Sullivan, "High-quality streamable free-viewpoint video," *ACM Transactions on Graphics (TOG),* vol. 34, no. 4, pp. 69, 2015.

[8] B. Wilburn, N. Joshi, V. Vaish, E.-V. Talvala, E. Antunez, A. Barth, A. Adams, M. Horowitz, and M. Levoy, "High performance imaging using large camera arrays," *ACM Transactions on Graphics (TOG),* vol. 24, no. 3, pp. 765-776, 2005.

[9] A. Chuchvara, A. Barsi, and A. Gotchev, "Fast and Accurate Depth Estimation from Sparse Light Fields," *arXiv preprint arXiv:1812.06856*, 2018.

[10] L. Yao, Y. Liu, and W. Xu, "Real-time virtual view synthesis using light field," *EURASIP Journal on Image and Video Processing,* vol. 2016, no. 1, pp. 25, 2016.

[11] M. Dou, S. Khamis, Y. Degtyarev, P. Davidson, S. R. Fanello, A. Kowdle, S. O. Escolano, C. Rhemann, D. Kim, and J. Taylor, "Fusion4d: Real-time performance capture of challenging scenes," *ACM Transactions on Graphics (TOG),* vol. 35, no. 4, pp. 114, 2016.

[12] E. De Aguiar, C. Stoll, C. Theobalt, N. Ahmed, H.-P. Seidel, and S. Thrun, "Performance capture from sparse multi-view video," *ACM Transactions on Graphics (TOG),* vol. 27, no. 3, pp. 98, 2008.

[13] M. Keller, D. Lefloch, M. Lambers, S. Izadi, T. Weyrich, and A. Kolb, "Real-time 3d reconstruction in dynamic scenes using point-based fusion," in 3D Vision-3DV 2013, 2013 International Conference on, 2013, pp. 1-8.

[14] L. Z. Eric Penner, "Soft 3d reconstruction for view synthesis," *ACM Transactions on Graphics (Proc. SIGGRAPH Asia),* vol. 36, 2017.

[15] K. Wegner, and O. Stankiewicz, "DERS Software Manual," Sapporo, 2014.